

# WebFOCUS

Creating Reports With WebFOCUS  
Language

Release 8.2 Version 03

May 25, 2018

Active Technologies, EDA, EDA/SQL, FIDEL, FOCUS, Information Builders, the Information Builders logo, iWay, iWay Software, Parlay, PC/FOCUS, RStat, Table Talk, Web390, WebFOCUS, WebFOCUS Active Technologies, and WebFOCUS Magnify are registered trademarks, and DataMigrator and Hyperstage are trademarks of Information Builders, Inc.

Adobe, the Adobe logo, Acrobat, Adobe Reader, Flash, Adobe Flash Builder, Flex, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Due to the nature of this material, this document refers to numerous hardware and software products by their trademarks. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2018, by Information Builders, Inc. and iWay Software. All rights reserved. Patent Pending. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

# Contents

---

<b>Preface .....</b>	<b>27</b>
Conventions .....	30
Related Publications .....	30
Customer Support .....	30
Information You Should Have .....	31
User Feedback .....	32
Information Builders Consulting and Training .....	32
<b>1. Creating Reports Overview .....</b>	<b>33</b>
Requirements for Creating a Report .....	33
Report Types .....	34
Developing Your Report Request .....	36
Starting a Report Request. ....	38
Completing a Report Request. ....	38
Creating a Report Example. ....	39
Customizing a Report .....	41
Selecting a Report Output Destination .....	42
<b>2. Displaying Report Data .....</b>	<b>43</b>
Using Display Commands in a Request .....	43
Displaying Individual Values .....	45
Displaying All Fields. ....	46
Displaying All Fields in a Segment. ....	48
Displaying the Structure and Retrieval Order of a Multi-Path Data Source. ....	49
Adding Values .....	54
Counting Values .....	56
Counting Segment Instances. ....	57
Expanding Byte Precision for COUNT and LIST .....	58
Maximum Number of Display Fields Supported in a Request .....	59
Manipulating Display Fields With Prefix Operators .....	60
Prefix Operator Basics. ....	60
Averaging Values of a Field. ....	63
Averaging the Sum of Squared Fields. ....	64

Calculating Maximum and Minimum Field Values.....	65
Calculating Median and Mode Values for a Field.....	65
Calculating Column and Row Percentages.....	66
Producing a Direct Percent of a Count.....	68
Aggregating and Listing Unique Values.....	69
Retrieving First and Last Records.....	72
Summing and Counting Values.....	74
Ranking Sort Field Values With RNK.....	76
Rolling Up Calculations on Summary Rows.....	78
Using Report-Level Prefix Operators.....	84
Displaying Pop-up Field Descriptions for Column Titles .....	87
<b>3. Sorting Tabular Reports .....</b>	<b>91</b>
Sorting Tabular Reports Overview .....	91
Sorting Rows .....	92
Using Multiple Vertical (BY) Sort Fields.....	94
Displaying a Row for Data Excluded by a Sort Phrase.....	95
Sorting Columns .....	98
Controlling Display of an ACROSS Title for a Single Field.....	100
Positioning ACROSS Titles on Report Output.....	103
Using Multiple Horizontal (ACROSS) Sort Fields.....	110
Collapsing PRINT With ACROSS.....	110
Hiding Null Columns in ACROSS Groups.....	113
Hiding ACROSS Groups and Columns Within BY Page Breaks.....	114
Generating Summary Lines and Hiding Null ACROSS Columns.....	122
Using Column Styling and Hiding Null ACROSS Columns.....	126
Hiding Null ACROSS Columns in an FML Request.....	132
Controlling Display of Sort Field Values .....	136
Reformatting Sort Fields .....	139
Manipulating Display Field Values in a Sort Group .....	141
Creating a Matrix Report .....	143
Controlling Collation Sequence .....	144
Specifying the Sort Order .....	152



Specifying Your Own Sort Order.....	154
Selecting and Assigning Column Titles to ACROSS Values.....	158
Ranking Sort Field Values .....	159
DENSE and SPARSE Ranking.....	161
Grouping Numeric Data Into Ranges .....	166
Grouping Numeric Data Into Tiles.....	170
Restricting Sort Field Values by Highest/Lowest Rank .....	175
Sorting and Aggregating Report Columns .....	176
Hiding Sort Values .....	179
Sort Performance Considerations .....	181
Sorting With Multiple Display Commands .....	182
Controlling Formatting of Reports With Multiple Display Commands.....	183
Improving Efficiency With External Sorts .....	190
Providing an Estimate of Input Records or Report Size for Sorting.....	192
Sort Work Files and Return Codes.....	192
Mainframe External Sort Utilities and Message Options.....	194
Diagnosing External Sort Errors.....	195
Aggregation by External Sort (Mainframe Environments Only).....	197
Changing Retrieval Order With Aggregation .....	199
Creating a HOLD File With an External Sort (Mainframe Environments Only) .....	200
Hierarchical Reporting: BY HIERARCHY .....	201
<b>4. Selecting Records for Your Report .....</b>	<b>219</b>
Selecting Records Overview .....	219
Choosing a Filtering Method .....	220
Selections Based on Individual Values .....	220
Controlling Record Selection in Multi-path Data Sources.....	223
Selection Based on Aggregate Values .....	228
Applying Selection Criteria to the Internal Matrix Prior to COMPUTE Processing .....	230
Using Compound Expressions for Record Selection .....	237
Using Operators in Record Selection Tests .....	238
Types of Record Selection Tests .....	243
Range Tests With FROM and TO.....	243

Range Tests With GE and LE or GT and LT.....	245
Missing Data Tests.....	246
Character String Screening With CONTAINS and OMITs.....	247
Screening on Masked Fields.....	248
Using an Escape Character for LIKE.....	253
Qualifying Parent Segments Using INCLUDES and EXCLUDES.....	256
Selections Based on Group Key Values.....	257
Setting Limits on the Number of Records Read.....	258
Selecting Records Using IF Phrases.....	259
Reading Selection Values From a File.....	260
Assigning Screening Conditions to a File.....	265
Preserving Filters Across Joins.....	271
VSAM Record Selection Efficiencies.....	274
Reporting From Files With Alternate Indexes.....	274
<b>5. Creating Temporary Fields .....</b>	<b>277</b>
What Is a Temporary Field? .....	277
Defining a Virtual Field .....	280
Defining Multiple Virtual Fields.....	287
Displaying Virtual Fields.....	288
Clearing a Virtual Field.....	289
Establishing a Segment Location for a Virtual Field.....	290
Defining Virtual Fields Using a Multi-Path Data Source.....	291
Increasing the Speed of Calculations in Virtual Fields.....	291
Preserving Virtual Fields Using DEFINE FILE SAVE and RETURN.....	292
Applying Dynamically Formatted Virtual Fields to Report Columns.....	293
Passing Function Calls Directly to a Relational Engine Using SQL.Function Syntax.....	296
Creating a Calculated Value .....	298
Using Positional Column Referencing With Calculated Values.....	302
Using ACROSS With Calculated Values.....	303
Sorting Calculated Values.....	304
Screening on Calculated Values.....	304
Assigning Column Reference Numbers .....	304

Using Column Notation in a Report Request. ....	305
Using FORECAST in a COMPUTE Command .....	314
Calculating Trends and Predicting Values With FORECAST. ....	315
FORECAST Processing. ....	316
FORECAST_MOVAVE: Using a Simple Moving Average. ....	317
FORECAST_EXPAVE: Using Single Exponential Smoothing. ....	322
FORECAST_DOUBLEXP: Using Double Exponential Smoothing. ....	326
FORECAST_SEASONAL: Using Triple Exponential Smoothing. ....	328
FORECAST_LINEAR: Using a Linear Regression Equation. ....	332
Distinguishing Data Rows From Predicted Rows. ....	336
Calculating Trends and Predicting Values With FORECAST .....	338
FORECAST Processing. ....	339
Using a Simple Moving Average. ....	343
Using Single Exponential Smoothing. ....	346
Using Double Exponential Smoothing. ....	349
Using Triple Exponential Smoothing. ....	350
Using a Linear Regression Equation. ....	353
FORECAST Reporting Techniques. ....	356
Calculating Trends and Predicting Values With Multivariate REGRESS .....	359
Using Text Fields in DEFINE and COMPUTE .....	362
Creating Temporary Fields Independent of a Master File .....	363
<b>6. Including Totals and Subtotals .....</b>	<b>369</b>
Calculating Row and Column Totals .....	369
Producing Row Totals for Horizontal (ACROSS) Sort Field Values. ....	376
Including Section Totals and a Grand Total .....	377
Including Subtotals .....	379
Recalculating Values for Subtotal Rows .....	385
Summarizing Alphanumeric Columns .....	388
Manipulating Summary Values With Prefix Operators .....	390
Controlling Summary Line Processing. ....	399
Using Prefix Operators With Calculated Values. ....	404
Using Multiple SUB-TOTAL or SUMMARIZE Commands With Prefix Operators. ....	407

Combinations of Summary Commands .....	409
Producing Summary Columns for Horizontal Sort Fields .....	415
Performing Calculations at Sort Field Breaks .....	423
Suppressing Grand Totals .....	427
Conditionally Displaying Summary Lines and Text .....	429
<b>7. Using Expressions .....</b>	<b>431</b>
Using Expressions in Commands and Phrases .....	431
Types of Expressions .....	432
Expressions and Field Formats.....	434
Creating a Numeric Expression .....	434
Order of Evaluation.....	437
Evaluating Numeric Expressions With Native-Mode Arithmetic.....	438
Using Identical Operand Formats With Native-Mode Arithmetic.....	439
Using Different Operand Formats With Native-Mode Arithmetic.....	439
Creating a Date Expression .....	440
Formats for Date Values.....	441
Performing Calculations on Dates.....	443
Cross-Century Dates With DEFINE and COMPUTE.....	444
Returned Field Format Selection.....	444
Using a Date Constant in an Expression.....	444
Extracting a Date Component.....	445
Combining Fields With Different Formats in an Expression.....	445
Creating a Date-Time Expression .....	446
Specifying a Date-Time Value.....	452
Manipulating Date-Time Values.....	456
Creating a Character Expression .....	458
Embedding a Quotation Mark in a Quote-Delimited Literal String.....	459
Concatenating Character Strings.....	460
Creating a Variable Length Character Expression .....	461
Using Concatenation With AnV Fields.....	461
Using the EDIT Function With AnV Fields.....	462
Using CONTAINS and OMITS With AnV Fields.....	462

Using LIKE With AnV Fields. ....	462
Using the EQ, NE, LT, GT, LE, and GE Operators With AnV Fields. ....	463
Using the DECODE Function With AnV Fields. ....	464
Using the Assignment Operator With AnV Fields. ....	464
Creating a Logical Expression ....	465
Creating a Conditional Expression ....	467
<b>8. Saving and Reusing Your Report Output .....</b>	<b>471</b>
Saving Your Report Output .....	472
Naming and Storing Report Output Files. ....	472
Creating a HOLD File .....	473
Holding Report Output in FOCUS Format .....	479
Controlling Attributes in HOLD Master Files .....	484
Controlling Field Names in a HOLD Master File. ....	485
Controlling Fields in a HOLD Master File. ....	490
Controlling the TITLE and ACCEPT Attributes in the HOLD Master File. ....	495
Keyed Retrieval From HOLD Files .....	496
Saving and Retrieving HOLD Files .....	498
Using DBMS Temporary Tables as HOLD Files .....	500
Column Names in the HOLD File. ....	505
Primary Keys and Indexes in the HOLD File. ....	505
Creating SAVE and SAVB Files .....	506
Creating a PCHOLD File .....	509
Choosing Output File Formats .....	511
Using Text Fields in Output Files .....	535
Creating a Delimited Sequential File .....	537
Saving Report Output in INTERNAL Format .....	546
Creating A Subquery or Sequential File With HOLD FORMAT SQL_SCRIPT .....	549
Creating a Structured HOLD File .....	551
Specifying MIME Types for WebFOCUS Reports .....	561
<b>9. Choosing a Display Format .....</b>	<b>565</b>
Report Display Formats .....	566
Preserving Leading and Internal Blanks in Report Output .....	569

Using Web Display Format: HTML .....	571
Using Print Display Formats: PDF, PS .....	574
Using PDF Display Format. ....	575
Displaying Watermarks in PDF Output. ....	576
Features Supported. ....	580
Limits. ....	580
Usage Notes. ....	580
Using PostScript (PS) Display Format. ....	580
WebFOCUS Font Support. ....	583
How WebFOCUS Uses Type 1 Fonts. ....	584
Adding PostScript Type 1 Fonts for PS and PDF Formats. ....	585
Embedding TrueType Fonts Into WebFOCUS PDF Reports Generated in Windows. .	594
Creating PDF Files on z/OS for Use With UNIX Systems. ....	600
Using Word Processing Display Formats: DOC, WP .....	602
Saving Report Output in Excel XLSX Format .....	603
Overview of EXL07/XLSX Format. ....	604
Building the .xlsx Workbook File. ....	605
Opening XLSX Report Output. ....	607
Formatting Values Within Cells in XLSX Report Output. ....	609
Displaying Formatted Numeric Values in XLSX Report Output. ....	610
Using Numeric Formats in Report Headings and Footings. ....	612
Using Numeric Format Punctuation in Headings and Footings. ....	612
Passing Dates to XLSX Report Output. ....	615
Passing Dates Without a Day Component. ....	616
Passing Date Components for Use in Excel Formulas. ....	617
Passing Quarter Formats. ....	618
Passing Date Components Defined as Translated Text. ....	619
Passing Date-Time to XLSX. ....	620
Generating Native Excel Formulas in XLSX Report Output. ....	621
Understanding Formula Versus Value. ....	621
Using XLSX FORMULA With Prefix Operators. ....	629
NODATA With Formulas. ....	632
Controlling Column Width and Wrapping in XLSX Report Output. ....	633

Preserving Leading and Internal Blanks in Report Output. ....	636
Support for Drill Downs With XLSX Report Output. ....	639
Redirection and Excel Drill-Down Reports. ....	640
Excel Page Settings. ....	641
Adding an Image to a Report. ....	642
Inserting Images Into Excel XLSX Reports. ....	642
Inserting Images Into XLSX Workbook Headers and Footers. ....	652
Creating Excel XLSX Worksheets Using Templates. ....	657
Creating Excel Table of Contents Reports. ....	659
Naming XLSX Worksheets With Case Sensitive Data. ....	661
Overcoming the Excel 2007/2010 Row Limit Using Overflow Worksheets. ....	662
Excel Compound Reports Using XLSX. ....	668
Using XLSX FORMULA With Compound Reports. ....	679
FORMAT XLSX Limitations. ....	681
Using PowerPoint PPT Display Format. ....	681
Using PowerPoint PPT Templates. ....	681
Saving Report Output in PPTX Format. ....	683
Building the .pptx Presentation File. ....	683
Opening PPTX Report Output. ....	685
Opening PPTX Report Output in Microsoft PowerPoint 2000/2003. ....	686
Viewing PowerPoint Presentations in the Browser vs. the PowerPoint Application. .	687
Grouping Tables and Components in a PowerPoint Slide. ....	687
Date and Page/Slide Number. ....	693
Text Formatting Markup Tags for a Text Object. ....	693
Display Unordered Lists With Bullets, Discs, Squares, and Circles. ....	704
Inserting Images In Various Elements of PowerPoint PPTX Reports. ....	706
Displaying PPTX Charts in PNG Image Format. ....	714
Drill Down From Microsoft PowerPoint. ....	720
PowerPoint PPTX Presentations Using Templates. ....	723
PowerPoint PPTX Compound Syntax. ....	730
Coordinated Compound Layout Reports. ....	735
Templates for Compound Reports. ....	740
Adding Images to a Compound Request. ....	741

Template Masters and Slide Layouts.....	745
Identifying Slide Master Attributes in PowerPoint.....	745
WebFOCUS Pivot Support for XLSX.....	756
ReportCaster Distribution and ReportCaster Bursting.....	761
PPTX Limitations.....	761
Related Information.....	762
<b>10. Linking a Report to Other Resources .....</b>	<b>763</b>
Linking Using StyleSheets .....	763
Linking to Another Report .....	764
Linking to a URL .....	769
Defining a Hyperlink Color.....	774
Linking to a JavaScript Function .....	776
Linking to a Maintain Data Procedure .....	779
Multi-Drill Feature With Cascading Menus and User-Defined Styling .....	785
Accessibility Support.....	785
Creating Multiple Drill-Down Links.....	786
Global Menu Styling .....	786
Menu Items Styling .....	788
Drill-Down Action Options.....	789
Summary of Drill-Down Links.....	790
Sample Drill Menu Stylesheet Code.....	791
Applying Conditional Styling.....	796
Creating Parameters .....	797
Linking With Conditions .....	806
Linking From a Graphic Image .....	809
Specifying a Base URL .....	813
Specifying a Target Frame .....	814
Creating a Compound Report .....	816
Creating a Compound Layout Report With Document Syntax.....	818
Generating a Table of Contents With BY Field Entries for PDF Compound Layout Reports .....	863
Table of Contents Features.....	864
Creating a Compound PDF or PS Report.....	870



Creating a Compound Excel Report Using EXL2K. ....	879
Creating a PDF Compound Report With Drill Through Links .....	890
Sample Drill Through PDF Compound Reports. ....	895
<b>11. Navigating Within an HTML Report .....</b>	<b>905</b>
Navigating Sort Groups From a Table of Contents .....	905
Adding the HTML Table of Contents Tree Control to Reports .....	907
Navigation Behavior in a Multi-Level TOC. ....	913
Controlling the Display of Sorted Data With Accordion Reports .....	925
Requirements for Accordion Reports. ....	927
Creating an Accordion By Row Report. ....	928
Accordion By Row Tooltips. ....	942
Accordion By Row With NOPRINT. ....	947
Differences Between Reformatted and Redefined BY Fields. ....	950
Creating an Accordion By Column Report. ....	953
Navigating a Multi-Page Report With the WebFOCUS Viewer .....	955
Using the WebFOCUS Viewer Search Option. ....	958
Linking Report Pages .....	958
<b>12. Bursting Reports Into Multiple HTML Files .....</b>	<b>965</b>
Bursting Reports Overview .....	965
<b>13. Handling Records With Missing Field Values .....</b>	<b>971</b>
Irrelevant Report Data .....	971
Missing Field Values .....	972
MISSING Attribute in the Master File. ....	974
MISSING Attribute in a DEFINE or COMPUTE Command. ....	975
Testing for Missing Values in IF-THEN-ELSE Expressions. ....	981
Testing for a Segment With a Missing Field Value. ....	984
Preserving Missing Data Values in an Output File. ....	987
Propagating Missing Values to Reformatted Fields in a Request. ....	990
Handling a Missing Segment Instance .....	992
Including Missing Instances in Reports With the ALL. Prefix. ....	995
Including Missing Instances in Reports With the SET ALL Parameter. ....	995
Testing for Missing Instances in FOCUS Data Sources. ....	1002

Setting the NODATA Character String .....	1002
<b>14. Joining Data Sources .....</b>	<b>1005</b>
Types of Joins .....	1006
Unique and Non-Unique Joined Structures.....	1008
Recursive Joined Structures.....	1012
How the JOIN Command Works .....	1016
Creating an Equijoin .....	1018
Joining From a Virtual Field to a Real Field Using an Equijoin.....	1027
Join Modes in an Equijoin.....	1031
Data Formats of Shared Fields.....	1032
Joining Fields With Different Numeric Data Types.....	1033
Using a Conditional Join .....	1034
Full Outer Joins for Relational Data Sources .....	1038
Reporting Against a Multi-Fact Cluster Synonym .....	1046
Adding a New Fact To Multi-Fact Synonyms: JOIN AS_ROOT.....	1047
Generating Outer Joins of Cluster Synonym Contexts.....	1052
Joining From a Multi-Fact Synonym.....	1056
Invoking Context Analysis for a Star Schema With a Fan Trap .....	1060
Adding DBA Restrictions to the Join Condition: SET DBAJJOIN .....	1061
Preserving Virtual Fields During Join Parsing .....	1064
Preserving Virtual Fields Using KEEPDEFINES.....	1065
Preserving Virtual Fields Using DEFINE FILE SAVE and RETURN.....	1068
Screening Segments With Conditional JOIN Expressions.....	1070
Parsing WHERE Criteria in a Join.....	1070
Displaying Joined Structures .....	1070
Clearing Joined Structures .....	1072
Clearing a Conditional Join.....	1073
<b>15. Merging Data Sources .....</b>	<b>1075</b>
Merging Data .....	1075
Types of MATCH Processing .....	1077
MATCH Processing With Common High-Order Sort Fields .....	1086
Fine-Tuning MATCH Processing .....	1090

Universal Concatenation .....	1093
Field Name and Format Matching.....	1096
Merging Concatenated Data Sources .....	1098
Using Sort Fields in MATCH Requests.....	1100
Cartesian Product .....	1104
<b>16. Formatting Reports: An Overview .....</b>	<b>1107</b>
What Kinds of Formatting Can I Do? .....	1107
How to Specify Formatting in a Report .....	1110
How to Choose a Type of Style Sheet.....	1113
Standard and Legacy Formatting .....	1114
Techniques for Quick and Easy Formatting .....	1114
Navigating From a Report to Other Resources .....	1115
<b>17. Creating and Managing a WebFOCUS StyleSheet .....</b>	<b>1117</b>
Creating a WebFOCUS StyleSheet .....	1117
Creating a WebFOCUS StyleSheet Within a Report Request.....	1118
Creating and Applying a WebFOCUS StyleSheet File.....	1120
General WebFOCUS StyleSheet Syntax .....	1122
Improving WebFOCUS StyleSheet Readability.....	1123
Adding a Comment to a WebFOCUS StyleSheet.....	1124
Reusing WebFOCUS StyleSheet Declarations With Macros .....	1124
Defining a WebFOCUS StyleSheet Macro.....	1124
Applying a WebFOCUS StyleSheet Macro.....	1125
WebFOCUS StyleSheet Attribute Inheritance .....	1127
Creating Reports With the ENWarm StyleSheet .....	1131
Report Styling.....	1132
Data, Report, and Title Styling.....	1132
Headings and Footings Styling.....	1133
Subheading and Subfooting Styling.....	1134
Across Styling.....	1135
Subtotal and Column Total Styling.....	1136
Active Reports.....	1136
Pagination, Menu, and Hover Text Styling in WebFOCUS Active Reports.....	1137

- Usage Notes for ENWarm.sty .....1138
- 18. Controlling Report Formatting ..... 1139**
  - Generating an Internal Cascading Style Sheet for HTML Reports ..... 1140
  - Selecting a Unit of Measurement .....1141
  - Conditionally Formatting, Displaying, and Linking in a StyleSheet ..... 1142
    - Applying Sequential Conditional Formatting.....1143
  - Including Summary Lines, Underlines, Skipped Lines, and Page Breaks .....1156
  - Conditionally Including Summary Lines, Underlines, Skipped Lines, and Page Breaks ..... 1158
  - Controlling the Display of Empty Reports ..... 1163
  - Formatting a Report Using Only StyleSheet Defaults ..... 1165
- 19. Identifying a Report Component in a WebFOCUS StyleSheet ..... 1167**
  - Identifying an Entire Report, Column, or Row .....1167
  - Identifying Tags for SUBTOTAL and GRANDTOTAL Lines .....1176
  - Identifying Data ..... 1179
    - Identifying Totals and Subtotals..... 1185
  - Identifying a Heading, Footing, Title, or FML Free Text ..... 1191
    - Identifying a Column or Row Title..... 1191
    - Identifying a Heading or Footing..... 1195
  - Identifying a Page Number, Underline, or Skipped Line ..... 1206
- 20. Using an External Cascading Style Sheet .....1211**
  - What Is a Cascading Style Sheet? ..... 1211
    - What Are Cascading Style Sheet Rules and Classes?.....1212
  - Why Use an External Cascading Style Sheet? ..... 1213
  - Formatting a Report With an External Cascading Style Sheet ..... 1214
  - Working With an External Cascading Style Sheet .....1220
    - Choosing an External Cascading Style Sheet..... 1221
    - External Cascading Style Sheet Location.....1221
    - Using Several External Cascading Style Sheets..... 1221
    - Editing an External Cascading Style Sheet..... 1221
    - Choosing a Cascading Style Sheet Rule..... 1222
    - Naming a Cascading Style Sheet Class..... 1223
  - Applying External Cascading Style Sheet Formatting .....1224

Combining an External CSS With Other Formatting Methods .....	1226
Combining an External CSS With a WebFOCUS StyleSheet.....	1227
Linking to an External Cascading Style Sheet .....	1228
Using the CSSURL Attribute and Parameter.....	1228
Inheritance and External Cascading Style Sheets .....	1231
Using External Cascading Style Sheets With Non-HTML Reports .....	1234
Requirements for Using an External Cascading Style Sheet .....	1239
FAQ About Using External Cascading Style Sheets .....	1241
Troubleshooting External Cascading Style Sheets .....	1245
<b>21. Laying Out the Report Page .....</b>	<b>1249</b>
Selecting Page Size, Orientation, and Color .....	1249
Setting Page Margins .....	1255
Positioning a Report Component .....	1258
Arranging Columns on a Page .....	1264
Determining Column Width.....	1265
Controlling Column Spacing.....	1270
Changing Column Order.....	1271
Stacking Columns.....	1273
Alignment of Fields in Reports Using OVER in PDF Report Output.....	1276
Positioning a Column.....	1283
Suppressing Column Display .....	1287
Inserting a Page Break .....	1293
Preventing an Undesirable Split.....	1299
Inserting Page Numbers .....	1304
Inserting the Total Page Count.....	1306
Displaying the Total Page Count Within a Sort Group.....	1308
Assigning Any Page Number to the First Page.....	1311
Controlling the Display of Page Numbers.....	1313
Setting the Number of Data Rows For Each Page in an AHTML Report Request.....	1315
Adding Grids and Borders .....	1318
Defining Borders Around Boxes With PPTX and PDF Formats .....	1349
Displaying Superscripts On Data, Heading, and Footing Lines .....	1351

Adding Underlines and Skipped Lines .....	1354
Removing Blank Lines From a Report .....	1370
Adding an Image to a Report .....	1376
Associating Bar Graphs With Report Data .....	1413
Controlling Bar Graph Scaling in Horizontal (ACROSS) Sort Fields.....	1420
Applying Scaling to Data Visualization Bar Graphs.....	1422
Working With Mailing Labels and Multi-Pane Pages .....	1424
<b>22. Using Headings, Footings, Titles, and Labels .....</b>	<b>1431</b>
Creating Headings and Footings .....	1432
Limits for Headings and Footings.....	1433
Extending Heading and Footing Code to Multiple Lines in a Report Request.....	1434
Creating a Custom Report or Worksheet Title.....	1436
Creating a Report Heading or Footing.....	1438
Creating a Page Heading or Footing.....	1445
Freezing HTML Headings, Footings, and Column Titles.....	1453
Creating a Sort Heading or Footing.....	1456
Including an Element in a Heading or Footing .....	1469
Including a Field Value in a Heading or Footing.....	1470
Including a Text Field in a Heading or Footing.....	1477
Including a Page Number in a Heading or Footing.....	1479
Including a Dialogue Manager Variable in a Heading or Footing.....	1479
Including an Image in a Heading or Footing.....	1481
Displaying Syntax Components in Heading Objects .....	1482
Repeating Headings and Footings on Panels in PDF Report Output .....	1484
Customizing a Column Title .....	1501
Customizing a Column Title in a Master File.....	1507
Distinguishing Between Duplicate Field Names.....	1507
Controlling Column Title Underlining Using a SET Command .....	1508
Controlling Column Title Underlining Using a StyleSheet Attribute .....	1510
Creating Labels to Identify Data .....	1513
Creating a Label for a Row or Column Total.....	1513
Creating a Label for a Subtotal and a Grand Total.....	1515

Creating a Label for a Row in a Financial Report. ....	1520
Formatting a Heading, Footing, Title, or Label .....	1520
Applying Font Attributes to a Heading, Footing, Title, or Label .....	1522
Adding Borders and Grid Lines .....	1526
Justifying a Heading, Footing, Title, or Label .....	1529
Justifying a Heading or Footing. ....	1530
Justifying a Column Title. ....	1537
Justifying a Label for a Row or Column Total. ....	1542
Justifying a Label for a Subtotal or Grand Total. ....	1544
Choosing an Alignment Method for Heading and Footing Elements .....	1546
Aligning a Heading or Footing Element in an HTML, XLSX, EXL2K, PDF, PPTX, or DHTML Report .....	1548
Aligning a Heading or Footing Element Across Columns in an HTML or PDF Report .....	1566
Aligning Content in a Multi-Line Heading or Footing .....	1573
Aligning Decimals in a Multi-Line Heading or Footing. ....	1578
Combining Column and Line Formatting in Headings and Footings. ....	1579
Positioning Headings, Footings, or Items Within Them .....	1584
Using PRINTPLUS. ....	1591
Using Spot Markers to Refine Positioning. ....	1593
Controlling the Vertical Positioning of a Heading or Footing .....	1598
Placing a Report Heading or Footing on Its Own Page .....	1605
<b>23. Formatting Report Data .....</b>	<b>1611</b>
Specifying Font Format in a Report .....	1611
Specifying Fonts for Reports. ....	1617
Specifying Background Color in a Report .....	1619
Alternating Background Color By Wrapped Line .....	1623
Specifying Data Format in a Report .....	1626
Changing the Format of Values in a Report Column. ....	1627
Controlling Missing Values for a Reformatted Field. ....	1629
Using Commas vs. Decimals (Continental Decimal Notation). ....	1631
Setting Characters to Represent Null and Missing Values. ....	1631
Using Conditional Grid Formatting in a Field. ....	1632

Positioning Data in a Report .....	1633
Controlling Wrapping of Report Data.....	1633
Justifying Report Columns.....	1648
Field-Based Reformatting.....	1650
Displaying Multi-Line An and AnV Fields.....	1653
<b>24. Creating a Graph .....</b>	<b>1657</b>
Content Analysis: Determining Graphing Objectives .....	1657
The GRAPH Command .....	1658
Similarities Between GRAPH and TABLE.....	1659
Differences Between GRAPH and TABLE.....	1659
Creating an HTML5 Graph .....	1661
Selecting a Graph Type .....	1663
Graph Types.....	1663
Selecting Scales.....	1664
Determining Graph Styles With Display Commands and Sort Phrases.....	1665
Determining Graph Styles Using LOOKGRAPH.....	1672
Selecting Values for the X and Y Axes .....	1683
Hiding the Display of a Y-Axis Field.....	1684
Interpolating X and Y Axis Values Using Linear Regression.....	1684
Creating Multiple Graphs .....	1685
Merging Multiple Graphs.....	1686
Merging Multiple OLAP Graphs.....	1688
Displaying Multiple Graphs in Columns.....	1690
Plotting Dates in Graphs .....	1691
Basic Date Support for X and Y Axes.....	1692
Formatting Dates for Y-Axis Values.....	1694
Refining the Data Set For Your Graph .....	1694
Displaying Missing Data Values in a Graph .....	1695
Applying Conditional Styling to a Graph .....	1700
Linking Graphs to Other Resources .....	1703
Creating Parameters.....	1710
Adding Labels to a Graph .....	1710



Adding Vertical (Y-axis) and Horizontal (X-axis) Labels to a Graph.....	1712
Applying Custom Styling to a Graph .....	1712
Setting the Graph Height and Width.....	1712
Customizing Graphs Using SET Parameters.....	1713
Setting Fixed Scales for the X-Axis.....	1718
Setting Fixed Scales for the Y-Axis.....	1719
Customizing Graphs Using the Graph API and HTML5 JSON Properties.....	1719
Saving a Graph as an Image File .....	1722
Saving a Graph as an Image File Using GRAPHSEVURL.....	1722
Printing a Graph .....	1727
<b>25. Creating Financial Reports With Financial Modeling Language (FML) .....</b>	<b>1729</b>
Reporting With FML .....	1729
Creating Rows From Data .....	1732
Creating Rows From Multiple Records.....	1735
Using the BY Phrase in FML Requests.....	1740
Combining BY and FOR Phrases in an FML Request.....	1741
Supplying Data Directly in a Request .....	1741
Performing Inter-Row Calculations .....	1743
Referring to Rows in Calculations .....	1744
Referring to Columns in Calculations .....	1747
Referring to Column Numbers in Calculations.....	1748
Referring to Contiguous Columns in Calculations.....	1749
Referring to Column Addresses in Calculations.....	1750
Referring to Relative Column Addresses in Calculations.....	1751
Applying Relative Column Addressing in a RECAP Expression.....	1752
Controlling the Creation of Column Reference Numbers.....	1752
Referring to Column Values in Calculations.....	1753
Referring to Rows and Columns in Calculations .....	1754
Referring to Cells in Calculations .....	1755
Using Functions in RECAP Calculations .....	1757
Inserting Rows of Free Text .....	1759
Adding a Column to an FML Report .....	1761

Creating a Recursive Model .....	1763
Reporting Dynamically From a Hierarchy .....	1764
Requirements for FML Hierarchies.....	1765
Displaying an FML Hierarchy.....	1767
Consolidating an FML Hierarchy.....	1770
Loading a Hierarchy Manually.....	1778
Customizing a Row Title .....	1781
Formatting an FML Report .....	1782
Indenting Row Titles in an FML Hierarchy.....	1799
Suppressing the Display of Rows .....	1802
Suppressing Rows With No Data.....	1803
Saving and Retrieving Intermediate Report Results .....	1804
Posting Data.....	1804
Creating HOLD Files From FML Reports .....	1807
<b>26. Creating a Free-Form Report .....</b>	<b>1809</b>
Creating a Free-Form Report .....	1809
Designing a Free-Form Report .....	1813
Incorporating Text in a Free-Form Report.....	1814
Incorporating Data Fields in a Free-Form Report.....	1814
Incorporating Graphic Characters in a Free-Form Report.....	1815
Laying Out a Free-Form Report.....	1815
Sorting and Selecting Records in a Free-Form Report.....	1816
<b>27. Using SQL to Create Reports .....</b>	<b>1817</b>
Supported and Unsupported SQL Statements .....	1817
Using SQL Translator Commands .....	1820
The SQL SELECT Statement.....	1822
SQL Joins.....	1823
SQL CREATE TABLE and INSERT INTO Commands.....	1826
SQL CREATE VIEW and DROP VIEW Commands.....	1827
Cartesian Product Style Answer Sets.....	1829
Continental Decimal Notation (CDN).....	1829
Specifying Field Names in SQL Requests.....	1829

SQL UNION, INTERSECT, and EXCEPT Operators.....	1830
Numeric Constants, Literals, Expressions, and Functions.....	1830
SQL Translator Support for Date, Time, and Timestamp Fields .....	1830
Extracting Date-Time Components Using the SQL Translator.....	1832
Index Optimized Retrieval .....	1835
Optimized Joins.....	1835
TABLEF Optimization .....	1836
SQL INSERT, UPDATE, and DELETE Commands .....	1836
<b>28. Improving Report Processing .....</b>	<b>1839</b>
Rotating a Data Structure for Enhanced Retrieval .....	1839
Optimizing Retrieval Speed for FOCUS Data Sources .....	1842
Automatic Indexed Retrieval .....	1842
Data Retrieval Using TABLEF .....	1845
Compiling Expressions .....	1846
Compiling Expressions Using the DEFINES Parameter.....	1846
<b>A. Master Files and Diagrams .....</b>	<b>1847</b>
EMPLOYEE Data Source .....	1847
EMPLOYEE Master File.....	1849
EMPLOYEE Structure Diagram.....	1850
JOBFILE Data Source .....	1850
JOBFILE Master File.....	1851
JOBFILE Structure Diagram.....	1851
EDUCFILE Data Source .....	1852
EDUCFILE Master File.....	1852
EDUCFILE Structure Diagram.....	1853
SALES Data Source .....	1853
SALES Master File.....	1854
SALES Structure Diagram.....	1855
CAR Data Source .....	1855
CAR Master File.....	1857
CAR Structure Diagram.....	1858
LEDGER Data Source .....	1859

LEDGER Master File.....	1859
LEDGER Structure Diagram.....	1859
FINANCE Data Source .....	1859
FINANCE Master File.....	1859
FINANCE Structure Diagram.....	1860
REGION Data Source .....	1860
REGION Master File.....	1860
REGION Structure Diagram.....	1860
EMPDATA Data Source .....	1861
EMPDATA Master File.....	1861
EMPDATA Structure Diagram.....	1861
TRAINING Data Source .....	1861
TRAINING Master File.....	1862
TRAINING Structure Diagram.....	1862
COURSE Data Source .....	1862
COURSE Master File.....	1862
COURSE Structure Diagram.....	1863
JOBHIST Data Source .....	1863
JOBHIST Master File.....	1863
JOBHIST Structure Diagram.....	1863
JOBLIST Data Source .....	1863
JOBLIST Master File.....	1864
JOBLIST Structure Diagram.....	1864
LOCATOR Data Source .....	1864
LOCATOR Master File.....	1864
LOCATOR Structure Diagram.....	1865
PERSINFO Data Source .....	1865
PERSINFO Master File.....	1865
PERSINFO Structure Diagram.....	1865
SALHIST Data Source .....	1866
SALHIST Master File.....	1866
SALHIST Structure Diagram.....	1866
VIDEOTRK, MOVIES, and ITEMS Data Sources .....	1866

VIDEOTRK Master File.....	1867
VIDEOTRK Structure Diagram.....	1868
MOVIES Master File.....	1869
MOVIES Structure Diagram.....	1869
ITEMS Master File.....	1869
ITEMS Structure Diagram.....	1870
VIDEOTR2 Data Source .....	1870
VIDEOTR2 Master File.....	1870
VIDEOTR2 Structure Diagram.....	1871
Gotham Grinds Data Sources .....	1871
GGDEMOG Master File.....	1872
GGDEMOG Structure Diagram.....	1873
GGORDER Master File.....	1873
GGORDER Structure Diagram.....	1874
GGPRODS Master File.....	1874
GGPRODS Structure Diagram.....	1875
GGSALES Master File.....	1875
GGSALES Structure Diagram.....	1876
GGSTORES Master File.....	1876
GGSTORES Structure Diagram.....	1876
Century Corp Data Sources .....	1877
CENTCOMP Master File.....	1878
CENTCOMP Structure Diagram.....	1878
CENTFIN Master File.....	1879
CENTFIN Structure Diagram.....	1879
CENTHR Master File.....	1880
CENTHR Structure Diagram.....	1882
CENTINV Master File.....	1883
CENTINV Structure Diagram.....	1883
CENTORD Master File.....	1884
CENTORD Structure Diagram.....	1885
CENTQA Master File.....	1886
CENTQA Structure Diagram.....	1887

CENTGL Master File.....	1887
CENTGL Structure Diagram.....	1888
CENTSYSF Master File.....	1888
CENTSYSF Structure Diagram.....	1888
CENTSTMT Master File.....	1889
CENTSTMT Structure Diagram.....	1890
CENTGLL Master File.....	1890
CENTGLL Structure Diagram.....	1891
<b>B. Error Messages .....</b>	<b>1893</b>
Displaying Messages .....	1893
<b>C. Table Syntax Summary and Limits .....</b>	<b>1895</b>
TABLE Syntax Summary .....	1896
Hierarchical Reporting Syntax Summary.....	1897
TABLEF Syntax Summary .....	1898
MATCH Syntax Summary .....	1899
FOR Syntax Summary .....	1900
TABLE Limits .....	1901
<b>D. Referring to Fields in a Report Request .....</b>	<b>1903</b>
Referring to an Individual Field .....	1903
Referring to Fields Using Qualified Field Names .....	1904
Referring to All of the Fields in a Segment .....	1905
Displaying a List of Field Names .....	1906
Listing Field Names, Aliases, and Format Information.....	1906

# Preface

---

This content describes how to use WebFOCUS to create and style tabular reports, generate a wide variety of graphs, prepare data for those reports and graphs, and enable links to provide additional information. It is intended for application developers and other information technology professionals who use WebFOCUS to create and deploy reporting applications on the Internet or corporate intranets.

---

## How This Manual Is Organized

This manual includes the following chapters:

Chapter/Appendix		Contents
1	Creating Reports Overview	Describes the tools available for creating and formatting reports.
2	Displaying Report Data	Describes ways to retrieve field values from a data source and display them.
3	Sorting Tabular Reports	Describes how to display report information grouped in a particular order by sorting.
4	Selecting Records for Your Report	Describes how to use and specify selection criteria to display only the field values that meet your needs.
5	Creating Temporary Fields	Describes how to use the DEFINE and COMPUTE commands to create temporary fields.
6	Including Totals and Subtotals	Describes how to use subtotals and grand totals to summarize numeric information and aid in interpreting detailed information in a report.
7	Using Expressions	Describes how to combine operators, field names, and constants in an expression to derive new values.
8	Saving and Reusing Your Report Output	Describes how to save report data to files for reuse in different respects.
9	Choosing a Display Format	Describes the display formats available for viewing reports on the screen, including PDF, EXL2K, and XLSX.

<b>Chapter/Appendix</b>		<b>Contents</b>
10	Linking a Report to Other Resources	Describes how to use StyleSheets to link reports to other reports, URLs, Maintain Data procedures, and JavaScript functions.
11	Navigating Within an HTML Report	Describes navigation in an HTML report including dynamic table of contents (TOC), the WebFOCUS Viewer, and linking report pages.
12	Bursting Reports Into Multiple HTML Files	Describes how to provide more efficient retrieval of reports by bursting, or separating, a single report into multiple HTML files.
13	Handling Records With Missing Field Values	Describes how missing data affects report results and how to treat and represent it.
14	Joining Data Sources	Describes how to join two or more related data sources to create a larger integrated data structure from which you can report.
15	Merging Data Sources	Describes how to merge and concatenate two or more data sources into a new permanent data source.
16	Formatting Reports: An Overview	Provides a brief overview of the formatting options available in WebFOCUS.
17	Creating and Managing a WebFOCUS StyleSheet	Describes how to create a StyleSheet and reviews general StyleSheet syntax.
18	Controlling Report Formatting	Describes how you can control how reports are formatted including generating internal cascading style sheets, conditional formatting, setting measurement units, and controlling the display of empty reports.
19	Identifying a Report Component in a WebFOCUS StyleSheet	Describes how to identify a report component using StyleSheet declarations.
20	Using an External Cascading Style Sheet	Describes how you can increase your formatting options by using external cascading style sheets.



Chapter/Appendix		Contents
21	Laying Out the Report Page	Describes basic report page layout including page size, orientation, page numbers, margins, images, grids, borders, page-breaks, and mailing labels.
22	Using Headings, Footings, Titles, and Labels	Describes how to add and format headings, footings, titles, and labels to add context to your report.
23	Formatting Report Data	Describes formatting and positioning text in a report including font style, size, and color.
24	Creating a Graph	Describes the GRAPH facility, which you can use to display data in graph format instead of tabular format.
25	Creating Financial Reports With Financial Modeling Language (FML)	Describes the Financial Modeling Language (FML) used to create and present financially oriented data, using inter-row calculations.
26	Creating a Free-Form Report	Describes how to present data in an unrestricted (non-tabular) format.
27	Using SQL to Create Reports	Describes how to use SQL to retrieve and analyze FOCUS and DBMS data.
28	Improving Report Processing	Describes methods of increasing data retrieval and reporting efficiency.
A	Master Files and Diagrams	Contains Master Files and diagrams of sample data sources used in the documentation examples.
B	Error Messages	Describes how to obtain information about error messages.
C	Table Syntax Summary and Limits	Summarizes TABLE commands and options.
D	Referring to Fields in a Report Request	Describes methods for referring to fields in a request including by alias, using long and qualified field names, and referring to all fields in a segment.

## Conventions

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

Convention	Description
<code>THIS TYPEFACE</code> or <code>this typeface</code>	Denotes syntax that you must enter exactly as shown.
<i>this typeface</i>	Represents a placeholder (or variable), a cross-reference, or an important term.
<u>underscore</u>	Indicates a default setting.
Key + Key	Indicates keys that you must press simultaneously.
{ }	Indicates two or three choices. Type one of them, not the braces.
[ ]	Indicates a group of optional parameters. None is required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis (...).
. . . . . .	Indicates that there are (or could be) intervening or additional commands.

## Related Publications

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

## Customer Support

Do you have any questions about this product?

Join the Focal Point community. Focal Point is our online developer center and more than a message board. It is an interactive network of more than 3,000 developers from almost every profession and industry, collaborating on solutions and sharing tips and techniques. 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 <http://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 effectively, be prepared to provide the following information when you call:

- ☐ Your six-digit site code (xxxx.xx).
- ☐ Your WebFOCUS configuration:
  - ☐ The front-end software you are using, including vendor and release.
  - ☐ The communications protocol (for example, TCP/IP or HLLAPI), including vendor and release.
  - ☐ The software release.
  - ☐ Your server version and release. You can find this information using the Version option in the Web Console.
- ☐ The stored procedure (preferably with line numbers) or SQL statements being used in server access.

- ☐ 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. You can 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? Information Builders Education Department offers a wide variety of training courses for this and other Information Builders products.

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

## Creating Reports Overview

---

WebFOCUS is a complete information control system with comprehensive features for retrieving and analyzing data that enables you to create reports quickly and easily. It provides facilities for creating highly complex reports, but its strength lies in the simplicity of the request language. You can begin with simple queries and progress to complex reports as you learn about additional facilities.

WebFOCUS serves the needs of both end users with no formal training in data processing, and data processing professionals who need powerful tools for developing complete applications. A variety of tools are available that enable you to create reports and charts even if you do not know HTML or WebFOCUS reporting language commands and syntax.

### In this chapter:

- ☐ [Requirements for Creating a Report](#)
  - ☐ [Report Types](#)
  - ☐ [Developing Your Report Request](#)
  - ☐ [Customizing a Report](#)
  - ☐ [Selecting a Report Output Destination](#)
- 

## Requirements for Creating a Report

To create a report, only two things are required:

- ☐ **Data.** You need data from which to report. If the data is protected by an underlying security system, you may need permission to report from the data source. In addition, the server must be able to locate the data source. For more information on data source locations, see the *Developing Reporting Applications* manual.

You can report from many different types of data sources (with variations for different operating environments), including the following:

- ☐ Relational data sources, such as DB2, Teradata, Oracle, and Sybase.
- ☐ Hierarchical data sources, such as IMS and FOCUS.

- ☐ Indexed data sources, such as ISAM and VSAM.
- ☐ Network data sources, such as CA-IDMS.
- ☐ Sequential data sources, both fixed-format and delimited format.
- ☐ Multi-dimensional data sources, such as SAP BW and Essbase.
- ☐ XML data sources.

For a complete list, see the *Describing Data With WebFOCUS Language* manual.

- ☐ **A data description.** You need a Master File, which describes the data source from which you are reporting. The Master File is a map of the segments in the data source and all of the fields in each segment. For some types of data sources, the Master File is supplemented by an Access File. For more information on Master Files and Access Files, see the *Describing Data With WebFOCUS Language* manual.

By looking at the Master File, you can determine what fields are in the data source, what they are named, and how they are formatted. You can also determine how the segments in the data source relate to each other. Although you can create a very simple report without this information, knowing the structure of the data source enables you to generate creative and sophisticated reports.

You can supplement the information in the Master File by generating a picture of the data source structure (that is, of how the data source segments relate to each other). Use the following command:

```
CHECK FILE filename PICTURE RETRIEVE
```

In the picture, segments are shown in the order in which they are retrieved. Four fields of each segment appear. For details see [Displaying Report Data](#) on page 43.

## Report Types

With WebFOCUS, you can create the following basic report types using graphical tools:

- ☐ **Tabular reports.** Displays information in rows and columns. This is the basic report type, incorporating the fundamental reporting concepts. Most of the other report formats build on these concepts. You can display these reports in formats such as HTML, Excel®, and PDF.

- ❑ **Financial reports.** Specifically designed to handle the task of creating, calculating, and presenting financially oriented data, such as balance sheets, consolidations, and budgets. You can build these reports with the Report canvas. The Matrix Report canvas enables you to define the content of the report on a row-by-row basis. This organization provides a number of advantages. You can:
  - ❑ Identify and display a title for each row of the report.
  - ❑ Perform row-based calculations and include the results at any point on the report.
  - ❑ Include the same record in multiple categories.
  - ❑ Include many types of formatting enhancements on a cell-by-cell basis.
  - ❑ Save individual rows and row titles in extract files.
- ❑ **Free-form reports.** Presents detailed information about a single record in a form-like context that is often used with letters and forms. If your goal is to present a detailed picture of one record per report page, you can use free-form reports to:
  - ❑ Position headers, footers, free text, and fields precisely on a page.
  - ❑ Customize your headers and footers by including fields as display variables.
  - ❑ Incorporate prefix operators in your headers and footers to perform calculations on the aggregated values of a single field.
  - ❑ Use vertical (BY) sorting to put one or more report records on each page.For details about free-form reports, see [Creating a Free-Form Report](#) on page 1809.
- ❑ **Graphs**, which can present the same kinds of information as tabular reports, but in a wide variety of two-dimensional and three-dimensional graph types.  
For details, see [Creating a Graph](#) on page 1657.
- ❑ **SQL requests**, which retrieve information using the SQL reporting language, and can directly incorporate WebFOCUS formatting commands. For details, see [Using SQL to Create Reports](#) on page 1817.

❑ **Drill Through reports.** Allow users to create a PDF document that contains a summary report plus a detail report, where the detail report contains all the detail data for designated fields in the summary report. Clicking a Drill Through hyperlink navigates internally in the PDF file and no additional reports are run. Drill Through reports are static. You can save the PDF file to disk or distribute it using ReportCaster. When opened with Acrobat® Reader, it retains its full Drill Through functionality. For more information about the Drill Through feature, see [Creating a PDF Compound Report With Drill Through Links](#) on page 890. For more information about Compound Reports, see [Creating a Compound Report](#) on page 816.

❑ **Excel Compound reports.** Provides a way to generate multiple worksheet reports using the EXL2K output format. By default, each of the component reports from the compound report is placed in a new Excel worksheet. If the NOBREAK keyword is used, the next report follows the current report on the same worksheet.

For more information, see [Creating a Compound Excel Report Using EXL2K](#) on page 879.

❑ **Excel Table of Contents reports.** Provides a way to generate a multiple worksheet report where a separate worksheet is generated for each value of the first BY field in the WebFOCUS report.

For more information, see [Choosing a Display Format](#) on page 565.

## Developing Your Report Request

The only requirement for reporting is identifying a data source. Beyond that, the structure of a report request is very flexible and you only need to include the report elements you want. For example, you only need to include sorting instructions if you want your report to be sorted, or selection criteria if you want to report on a subset of your data.

A report request begins with the TABLE FILE command and ends with the END command. The commands and phrases between the beginning and end of a request define the contents and format of a report. These parts of the request are optional; you only need to include the commands and phrases that produce the report functions you want.

The following are the most frequently used options for structuring a report request.

❑ **Specifying fields and columns.** Each column in your report represents a field. You can specify which fields you want to display, which fields you want to use to sort the report, which fields you want to use to select records, and which data source fields you want to use in creating temporary fields. Therefore, specifying the fields you want in a report is fundamentally tied to how you want to use those fields in your report.



- ❑ **Displaying data.** You can display data in your report by listing all the records for a field (detailed presentation), or by totaling the records for a field (summary presentation). You can also perform calculations and other operations on fields, such as finding the highest value of a field or calculating the average sum of squares of all the values of a field, and present the results of the operation in your report.
- ❑ **Sorting a report column.** Sorting a report enables you to organize column information. WebFOCUS displays the sort field, which is the field that controls the sorting order, at the left of the report if you are sorting vertically, or at the top, if you are sorting horizontally. Sort fields appear when their values change. You can also choose not to display sort fields.  
  
You can sort information vertically, down a column, or horizontally, across a row. You can also combine vertical sorting and horizontal sorting to create a simple matrix.
- ❑ **Selecting records.** When you generate a report, you may not want to include every record. Selecting records enables you to define a subset of the data source based on your criteria and then report on that subset. Your selection criteria can be as simple or complex as you wish.
- ❑ **Showing subtotals and totals.** You can display column and row totals, grand totals, and section subtotals in your report.
- ❑ **Customizing the presentation.** A successful report depends upon the information presented and how it is presented. A report that identifies related groups of information and draws attention to important facts will be more effective than one that simply shows columns of data. For example, you can:
  - ❑ Give column titles more meaningful names.
  - ❑ Control the display of columns in your report.
  - ❑ Create headings and footings for different levels of the report (including each sort group, each page, and the entire report), and dynamically control the display of headings and footings based on conditions you set.
  - ❑ Add fonts, colors, grids, and images.
  - ❑ Highlight a group of related information and separate it from other groups by inserting blank lines, underlines, and page breaks.
- ❑ **Creating temporary fields.** When you create a report, you are not limited to the fields that already exist in the data source. You can create temporary fields, deriving their values from real data source fields, and include them in your report.

For details see, [Creating Temporary Fields](#) on page 277.

- ❑ **Joining data sources.** You can join two or more data sources to create a larger integrated data structure from which you can report in a single request.

For details, see [Joining Data Sources](#) on page 1005.

- ❑ **Storing and reusing the results.** You can store your report data as a data source against which you can make additional queries. This is especially helpful for creating a subset of your data source and for generating two-step reports. You can also format the new data source for use by other data processing tools, such as spreadsheets and word processors.

For details, see [Saving and Reusing Your Report Output](#) on page 471.

You can run the request as an ad hoc query or save it as a procedure. Saving a report request as a procedure enables you to run or edit it at any time.

## Starting a Report Request

A report request begins with the designation of a data source. You can then specify the details of your report request. A data source can be specified in the following ways:

- ❑ The `TABLE FILE filename` command sets the data source for a single request.
- ❑ The `FILE SET` parameter sets a data source for all requests within a procedure.

For details on the `FILE SET` parameter, see the *Developing Reporting Applications* manual.

### **Syntax:** How to Begin a Report Request

To begin a report request, use the command

```
TABLE FILE filename
```

where:

```
filename
```

Is the data source for the report.

## Completing a Report Request

To complete a report request, use the `END` or `RUN` command. These commands must be typed on a line by themselves. To discontinue a report request without executing it, enter the `QUIT` command.

If you plan to issue consecutive report requests against the same data source during one session, you have the option of using the RUN command. RUN keeps the TABLE facility and the data source active for the duration of the TABLE session. After viewing one report you do not need to repeat the TABLE command to produce another report. You terminate the TABLE session by issuing the END command after the last request.

## Creating a Report Example

The example in this topic is a simple report request that illustrates some of the basic functions of WebFOCUS. However, there are many more functions not shown here that you can find information on throughout this documentation.

### *Example:* Creating a Simple Report

The following annotated example illustrates some of the basic functions of WebFOCUS. The numbered explanation in this example corresponds with the code in this request. This request can be generated by typing the commands into a text editor.

```

1. JOIN PIN IN EMPDATA TO ALL PIN IN TRAINING AS J1
2. DEFINE FILE EMPDATA
   YEAR/YY=COURSESTART;
3. END

4. TABLE FILE EMPDATA
5. HEADING CENTER
   "Education Cost vs. Salary"
6. SUM EXPENSES AS 'Education, Cost' SALARY AS 'Current, Salary'
7. AND COMPUTE PERCENT/D8.2=EXPENSES/SALARY * 100; AS 'Percent'
8. BY DIV
   BY DEPT
9. WHERE YEAR EQ 1991
10. ON TABLE SUMMARIZE
11. ON TABLE SET STYLE *
    TYPE=HEADING, STYLE=BOLD, COLOR=BLUE,$
    TYPE=REPORT, FONT=TIMES, SIZE=8,$
    TYPE=REPORT, GRID=OFF,$
    ENDSTYLE
12. END

```

The output is:

Education Cost vs. Salary				
		<u>Education</u>	<u>Current</u>	
DIV	DEPT	<u>Cost</u>	<u>Salary</u>	<u>Percent</u>
CE	ADMIN SERVICES	1,250.00	\$25,400.00	4.92
	PROGRAMMING & DVLPMT	1,580.00	\$40,900.00	3.86
CORP NE	ACCOUNTING	2,050.00	\$62,500.00	3.28
	CUSTOMER SUPPORT	1,800.00	\$19,300.00	9.33
	MARKETING	3,100.00	\$32,300.00	9.60
	SALES	1,800.00	\$43,600.00	4.13
SE	CONSULTING	3,350.00	\$35,900.00	9.33
WE	MARKETING	9,850.00	\$102,200.00	9.64
	PROGRAMMING & DVLPMT	5,310.00	\$42,900.00	12.38
	SALES	4,500.00	\$100,500.00	4.48
TOTAL		34,590.00	\$505,500.00	6.84

The request processes in the following way:

1. The JOIN command joins the EMPDATA and TRAINING data sources, allowing the request to access information from both data sources as if it were a single structure.
2. The DEFINE command creates a virtual field which extracts the year from the COURSESTART field in the TRAINING data source.
3. The END command ends the DEFINE command.
4. The TABLE command begins the report request.
5. The HEADING command adds the heading, Education Cost vs. Salary to the report output.
6. The SUM command adds the values within both the EXPENSES field and the SALARY field. The AS phrase changes the name of the column headings.
7. The COMPUTE command creates a calculated value using the values that have been aggregated in the SUM command and sorted with the BY command.
8. The BY phrase sorts the data in the report by the DIV field, and then by the DEPT field.
9. The WHERE command includes only the data that falls in the year 1991.
10. The ON TABLE SUMMARIZE command adds all values in both the EXPENSES and SALARY columns, and recalculates the Percent column.
11. The StyleSheet information formats the report heading and content.
12. The END command ends the report request.

## Customizing a Report

A successful report depends on the information presented and how it is presented. A report that identifies related groups of information and draws attention to important facts will be more effective than one that simply shows columns of data.

When you have selected the data that is going to be included in your report and how you want it to appear, you can then continue developing your report with custom formatting. There are many things you can add to your request in order to make your report more effective. You can:

- ☐ Add titles, headings, and footings. You can also change column titles with the AS phrase, and create headings and footings for different levels of the report (including each sort group, each page, and the entire report).
- ☐ Change the format of a field and the justification of a column title. For details, see [Formatting Report Data](#) on page 1611.
- ☐ Determine the width of a report column. For details, see [Formatting Report Data](#) on page 1611.
- ☐ Dynamically control the display of subtotals, headings, and footings based on conditions you define. For details, see [Controlling Report Formatting](#) on page 1139.
- ☐ Highlight a group of related information and separate it from other groups by inserting blank lines or underlines between each group.
- ☐ Emphasize data using color to highlight certain values in your report based on conditions you define. For details, see [Formatting Report Data](#) on page 1611.
- ☐ Format your report using external cascading style sheets. For details, see [Using an External Cascading Style Sheet](#) on page 1211.
- ☐ Add drill-down capability to your report. This adds extra value by linking your report to other reports or URLs that provide more detail. For details, see [Linking a Report to Other Resources](#) on page 763.

## Selecting a Report Output Destination

After you create a report, you can send it:

- ❑ **To the screen.** When you run a report, the default output destination is the screen. Reports run in WebFOCUS usually appear in a web browser, or in a helper application (such as, Adobe® Reader® or Microsoft® Excel), within a web browser.

You can also view reports outside of the browser in a standalone helper application such as Adobe Reader or Microsoft Excel. In Windows Folder Options, File Types (Advanced Settings), uncheck the *Browse in same window* option for the file type; such as .pdf or .xls. When the *Browse in same window option* is not selected, the browser window created by WebFOCUS is blank because the report output is displayed in the helper application window.

Note that if you selected the *Save Report* check box in the configuration pane of the WebFOCUS Administration Console (under Redirection Settings), you will be prompted whether to save or open the output file. If the procedure contained a PCHOLD command that specified an AS name for the output file, the name is retained if you choose to save the file. If no AS name was specified, a random filename is generated.

If the output is produced as a result of a GRAPH request, the returned HTML file contains a link to the actual graph output, which is stored as a temporary image file (for example, as a JPEG, GIF, or SVG file). The image file will eventually expire and be deleted from the server. For information about saving the graph output, see [Creating a Graph](#) on page 1657.

- ❑ **To a file.** You can store the results of your report for reuse using the HOLD, SAVE, or SAVB commands. For details see [Saving and Reusing Your Report Output](#) on page 471.
- ❑ **To a printer.** If you wish to print a report using a format such as PDF or HTML, first display the report using the desired format, and then print the report from the display application (for example, from Adobe Reader or from the web browser).

You cannot send a report directly to a printer in WebFOCUS.

## Displaying Report Data

---

Reporting, at the simplest level, retrieves field values from a data source and displays those values. There are three ways to do this:

- ☐ List each field value (PRINT and LIST commands).
- ☐ Add all the values and display the sum (SUM command).
- ☐ Count all the values and display the quantity (COUNT command).

**In this chapter:**

- ☐ [Using Display Commands in a Request](#)
  - ☐ [Displaying Individual Values](#)
  - ☐ [Adding Values](#)
  - ☐ [Counting Values](#)
  - ☐ [Expanding Byte Precision for COUNT and LIST](#)
  - ☐ [Maximum Number of Display Fields Supported in a Request](#)
  - ☐ [Manipulating Display Fields With Prefix Operators](#)
  - ☐ [Displaying Pop-up Field Descriptions for Column Titles](#)
- 

### Using Display Commands in a Request

The four display commands (PRINT, LIST, SUM, and COUNT) are also known as verbs. These commands are flexible; you can report from several fields using a single command, and include several different display commands in a single report request.

**Syntax:**      **How to Use Display Commands in a Request**

*display* [THE] [SEG.] *fieldname1* [AND] [THE] *fieldname2* ...

or

*display* \*

where:

*display*

Is the PRINT, LIST, SUM, or COUNT command. WRITE and ADD are synonyms of SUM and can be substituted for it.

SEG.

Displays all fields in a segment (a group of related fields in a Master File). The field name you specify can be any field in the segment.

*fieldname*

Is the name of the field to be displayed in the report.

The maximum number of display fields your report can contain is determined by a combination of factors. For details, see [Maximum Number of Display Fields Supported in a Request](#) on page 59.

The fields appear in the report in the same order in which they are specified in the report request. For example, the report column for *fieldname1* appears first, followed by the report column for *fieldname2*.

The field to be displayed is also known as the display field.

AND

Is optional and is used to enhance readability. It can be used between any two field names, and does not affect the report.

THE

Is optional and is used to enhance readability. It can be used before any field name, and does not affect the report.

\*

Applies the display command to every field in the left path of the data source.

**Note:** The SEG. and \* options do not display virtual fields. To print virtual fields, explicitly reference them in the PRINT statement (PRINT \* virtual field name). This is true even if the virtual field name redefines a real field.



## Displaying Individual Values

The display commands LIST and PRINT list the individual values of the fields you specify in your report request. LIST numbers the items in the report. PRINT does not number the items.

You can easily display all of the fields in the data source by specifying an asterisk (\*) wildcard instead of a specific field name, as described in [Displaying All Fields](#) on page 46.

For all PRINT and LIST requests, the number of records retrieved and the number of lines displayed are the same. In addition, there is no order to the report rows. The PRINT and LIST commands display all the values of the selected fields found in the data source in the order in which they are accessed. The order in which data is displayed may be affected by the AUTOPATH setting. For more information, see [Optimizing Retrieval Speed for FOCUS Data Sources](#) on page 1842, and the documentation on SET parameters in the *Developing Reporting Applications* manual.

In general, when using PRINT or LIST, the order of the values displayed in the report depends on whether or not the field is a key field, as described in the *Describing Data With WebFOCUS Language* manual.

Alternatively, you can sort the values using the BY or ACROSS sort phrases. When LIST is used in a request that includes a sort phrase, the list counter is reset to 1 every time the value in the outermost sort field changes. For more information on sorting, see [Sorting Tabular Reports](#) on page 91.

PRINT \* or PRINT SEG.\* prints only the real fields in the Master File. To print virtual fields, explicitly reference them in the PRINT statement (PRINT \* virtual field name). This is true even if the virtual field name is a re-defines of a real field.

For PRINT and LIST syntax, see [Using Display Commands in a Request](#) on page 43.

### **Example:** Displaying Individual Field Values

To display the values of individual fields, use the PRINT command. The following request displays the values of two fields, LAST\_NAME and FIRST\_NAME, for all employees.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME AND FIRST_NAME
END
```

The following shows the report output.

LAST_NAME	FIRST_NAME
-----	-----

STEVENS	ALFRED
SMITH	MARY
JONES	DIANE
SMITH	RICHARD
BANNING	JOHN
IRVING	JOAN
ROMANS	ANTHONY
MCCOY	JOHN
BLACKWOOD	ROSEMARIE
MCKNIGHT	ROGER
GREENSPAN	MARY
CROSS	BARBARA

**Example: Listing Records**

To number the records in a report, use the LIST command.

```
TABLE FILE EMPLOYEE
LIST LAST_NAME AND FIRST_NAME
END
```

The following shows the report output.

LIST	LAST_NAME	FIRST_NAME
----	-----	-----
1	STEVENS	ALFRED
2	SMITH	MARY
3	JONES	DIANE
4	SMITH	RICHARD
5	BANNING	JOHN
6	IRVING	JOAN
7	ROMANS	ANTHONY
8	MCCOY	JOHN
9	BLACKWOOD	ROSEMARIE
10	MCKNIGHT	ROGER
11	GREENSPAN	MARY
12	CROSS	BARBARA

**Displaying All Fields**

You can easily display all of the fields in the left path of the data source by specifying an asterisk (\*) wildcard instead of a specific field name. For additional information about Master File structures and segment paths, including left paths and short paths, see the *Describing Data With WebFOCUS Language* manual.

**Example: Displaying All Fields**

The following request produces a report displaying all of the fields in the EDUCFILE data source.

```
TABLE FILE EDUCFILE
LIST *
END
```

The following shows the report output.

LIST	COURSE_CODE	COURSE_NAME	DATE_ATTEND	EMP_ID
----	-----	-----	-----	-----
1	101	FILE DESCRPT & MAINT	83/01/04	212289111
2	101	FILE DESCRPT & MAINT	82/05/25	117593129
3	101	FILE DESCRPT & MAINT	82/05/25	071382660
4	101	FILE DESCRPT & MAINT	81/11/15	451123478
5	101	FILE DESCRPT & MAINT	81/11/15	112847612
6	102	BASIC REPORT PREP NON- PROG	82/07/12	326179357
7	103	BASIC REPORT PREP NON- PROG	83/01/05	212289111
8	103	BASIC REPORT PREP NON- PROG	82/05/26	117593129
9	103	BASIC REPORT PREP NON- PROG	81/11/16	112847612
10	104	FILE DESC & MAINT NON- PROG	82/07/14	326179357
11	106	TIMESHARING WORKSHOP	82/07/15	326179357
12	202	WHAT'S NEW IN FOCUS	82/10/28	326179357
13	301	DECISION SUPPORT WORKSHOP	82/09/03	326179357
14	107	BASIC REPORT PREP DP MGRS	82/08/02	818692173
15	302	HOST LANGUAGE INTERFACE	82/10/21	818692173
16	108	BASIC RPT NON-DP MGRS	82/10/10	315548712
17	108	BASIC RPT NON-DP MGRS	82/08/24	119265415
18	201	ADVANCED TECHNIQUES	82/07/26	117593129

## Displaying All Fields in a Segment

You can easily display all fields in a segment by adding the prefix "SEG." to any field in the desired segment.

### **Syntax:** How to Display All Fields in a Segment

*seg.anyfield*

where:

*anyfield*

Is any field that is in the desired segment.

### **Example:** Displaying All Fields in a Segment

The following request produces a report displaying all of the fields in the segment that contains the QTY\_IN\_STOCK field.

```
TABLE FILE CENTINV
PRINT SEG.QTY_IN_STOCK
BY  PRODNAME NOPRINT
END
```

The following shows the report output.

<u>Product</u> <u>Number:</u>	<u>Product</u> <u>Name:</u>	<u>Quantity</u> <u>In Stock:</u>	<u>Our</u> <u>Price:</u>	<u>Cost:</u>
1024	110 VHS-C Camcorder 20 X	4000	349.00	249.00
1022	120 VHS-C Camcorder 40 X	2300	399.00	259.00
1020	150 8MM Camcorder 20 X	5961	319.00	240.00
1004	2 Hd VCR LCD Menu	43068	179.00	129.00
1018	250 8MM Camcorder 40 X	60073	399.00	320.00
1016	330DX Digital Camera 1024K P	12707	279.00	199.00
1014	340SX Digital Camera 65K P	990	249.00	199.00
1012	650DL Digital Camcorder 150 X	2972	899.00	710.00
1010	750SL Digital Camcorder 300 X	10758	999.00	750.00
1028	AR2 35MM Camera 8 X	11499	109.00	79.00
1026	AR3 35MM Camera 10 X	12444	129.00	95.00
1006	Combo Player - 4 Hd VCR + DVD	13527	399.00	289.00
1008	DVD Upgrade Unit for Cent. VCR	199	199.00	139.00
1030	QX Portable CD Player	22000	169.00	99.00
1032	R5 Micro Digital Tape Recorder	1990	89.00	69.00
1036	ZC Digital PDA - Standard	33000	299.00	249.00
1034	ZT Digital PDA - Commercial	21000	499.00	349.00

## Displaying the Structure and Retrieval Order of a Multi-Path Data Source

When using display commands, it is important to understand the structure of the data source and the relationship between segments, since these factors affect your results. You can use the CHECK command PICTURE option to display a diagram of the data source structure defined by the Master File.

You can also display the retrieval order of a data source using the CHECK command PICTURE RETRIEVE option. It should be noted that retrieval is controlled by the minimum referenced subtree. For more information, see *Understanding the Efficiency of the Minimum Referenced Subtree* in the *Describing a Group of Fields* chapter in the *Describing Data With WebFOCUS Language* manual.

### ***Example:***    **Displaying the Structure of a Multi-Path Data Source**

To display the structure diagram of the CENTORD data source, which is joined to the CENTINV and CENTCOMP data sources, issue the following command:

```
CHECK FILE CENTORD PICTURE
```

The following shows the structure diagram output.

```

NUMBER OF ERRORS=      0
NUMBER OF SEGMENTS=    4  ( REAL=      2  VIRTUAL=      2 )
NUMBER OF FIELDS=     23  INDEXES=    4  FILES=      3
NUMBER OF DEFINES=      8
TOTAL LENGTH OF ALL FIELDS= 139

```

SECTION 01

STRUCTURE OF FOCUS      FILE CENTORD    ON 07/18/03 AT 11.06.34

```

01          OINFO
          S1
*****
*ORDER_NUM  **I
*STORE_CODE **I
*PLANT      **I
*ORDER_DATE **
*           **
*****
          I
          +-----+
          I                      I
          I STOSEG              I PINFO
02      I KU                    03      I S1
.....
:STORE_CODE :K      *PROD_NUM  **I
:STORENAME  :      *QUANTITY   **
:STATE      :      *LINEPRICE  **
:           :      *           **
:           :      *           **
:.....:      *****
JOINED  CENTCOMPFO*****
          I
          I
          I
          I INVSEG
          04      I KU
.....
:PROD_NUM   :K
:PRODNAME   :
:QTY_IN_STOCK:
:PRICE      :
:           :
:.....:
          JOINED  CENTIN

```

### Example: Displaying the Retrieval Order of a Multi-Path Data Source

To display the retrieval order of the EMPLOYEE data source, which is joined to the JOBFILE and EDUCFILE data sources, issue the following command:

CHECK FILE EMPLOYEE PICTURE RETRIEVE

The following shows the command output that adds the numbers that display at the top left of each segment, indicating the retrieval order of the segments. A unique segment such as FUNDTRAN is treated as a logical addition to the parent segment for retrieval. FUNDTRAN and SECSEG are unique segments, and are therefore treated as part of their parents.



The following shows the retrieval order:

```

check file employee picture retrieve
NUMBER OF ERRORS= 0
NUMBER OF SEGMENTS= 11 ( REAL= 6 VIRTUAL= 5 )
NUMBER OF FIELDS= 34 INDEXES= 0 FILES= 3
TOTAL LENGTH OF ALL FIELDS= 365
SECTION 01
RETRIEVAL VIEW OF FOCUS FILE EMPLOYEE ON 12/29/93 AT 14.42.18

EMP INFO
01 S1
*****
*EMP_ID **
*LAST_NAME **
*FIRST_NAME **
*HIRE_DATE **
*
*****
I
I
I
I
I FUNDTRAM
02 I U
*****
*BANK_NAME *
*BANK_CODE *
*BANK_ACCT *
*EFFECT_DATE *
*
*****
I
I
-----
I I I I
I PAYINFO I ADDRESS I SALINFO I ATTNDSEG
03 I SH1 07 I S1 08 I SH1 10 I KM
*****
*DAT_INC ** *TYPE ** *PAY_DATE ** :DATE_ATTEND :
*PCT_INC ** *ADDRESS_LN1 ** *GROSS ** :EMP_ID :K
*SALARY ** *ADDRESS_LN2 ** * ** :
*JOBCODE ** *ADDRESS_LN3 ** * ** :
* ** * ** :
*****
*****
I I I I EDUCFILE
I I I I
I I I I
I JOBSEG I DEDUCT I COURSESEG
04 I KU 09 I S1 11 I KLU
*****
:JOBCODE :K *DED_CODE ** :COURSE_CODE :
:JOB_DESC ** *DED_AMT ** :COURSE_NAME :
: ** * ** :
: ** * ** :
*****
I JOBFILE ***** EDUCFILE
I
I
I
I SECSEG
05 I KLU
:SEC_CLEAR :
:
:
:
I JOBFILE
I
I
I
I SKILLSEG
06 I KL
:SKILLS :
:SKILL_DESC :
:
:
:

```

### **Example:**    **Displaying Fields From a Multi-Path Data Source**

The following request produces a report displaying all of the fields on the left path of the EMPLOYEE data source.

```
TABLE FILE EMPLOYEE
PRINT *
END
```

The following shows a list of the output fields the previous request produces. Due to the size of the report, only the fields for which all instances will be printed are listed here. In the report, these fields would be displayed from left to right, starting with EMP\_ID.

```
EMP_ID
LAST_NAME
FIRST_NAME
HIRE_DATE
DEPARTMENT
CURR_SAL
CURR_JOBCODE
ED_HRS
BANK_NAME
BANK_CODE
BANK_ACCT
EFFECT_DATE
DAT_INC
PCT_INC
SALARY
JOBCODE
JOBDESC
JOB_DESC
SEC_CLEAR
SKILLS
SKILL_DESC
```

Each field in this list appears in segments on the left path of the EMPLOYEE data source. To view the retrieval order structure of the EMPLOYEE data source, see [Displaying the Retrieval Order of a Multi-Path Data Source](#) on page 51.

**Tip:** In some environments, the following warning is displayed whenever you use PRINT \* with a multi-path data source, to remind you that PRINT \* only displays the left path:

```
(FOC757) WARNING. YOU REQUESTED PRINT * OR COUNT * FOR A MULTI-PATH FILE
```

## Adding Values

SUM, WRITE, and ADD sum the values of a numeric field. The three commands are synonyms; they can be used interchangeably, and every reference to SUM in this documentation also refers to WRITE and ADD.

When you use SUM, multiple records are read from the data source, but only one summary line is produced. If you use SUM with a non-numeric field—such as an alphanumeric, text, or date field—SUM does not add the values. Instead, by default, it displays the last value retrieved from the data source. You can change this to the first value, minimum value, or maximum value using the SUMPREFIX parameter.

For SUM, WRITE, and ADD syntax, see [Using Display Commands in a Request](#) on page 43.

### **Example:** Adding Values

This request adds all the values of the field CURR\_SAL:

```
TABLE FILE EMPLOYEE
SUM CURR_SAL
END
```

The following shows the output of the request.

```
      CURR_SAL
      -----
$222,284.00
```

### **Example:** Adding Non-Numeric Values

This request attempts to add non-numeric fields. Any request for aggregation on non-numeric data returns the last record retrieved from the data source.

```
TABLE FILE EMPLOYEE
SUM LAST_NAME AND FIRST_NAME
END
```

The following shows the output of the request.

```
      LAST_NAME      FIRST_NAME
      -----      -
      CROSS          BARBARA
```

Note that any request for aggregation on all date format fields also returns the last record retrieved from the data source.

**Tip:** You can set the SUMPREFIX parameter to FST, MIN, MAX, or LST to control the sort order. For details, see [Sorting Tabular Reports](#) on page 91.

## Counting Values

The COUNT command counts the number of instances that exist for a specified field. The COUNT command is particularly useful combined with the BY phrase, which is discussed in [Sorting Tabular Reports](#) on page 91.

COUNT counts the instances of data contained in a report, not the data values.

For COUNT syntax, see [Using Display Commands in a Request](#) on page 43.

By default, a COUNT field is a five-digit integer. You can reformat it using the COMPUTE command, and change its field length using the SET COUNTWIDTH parameter. For details about the COMPUTE command, see [Creating Temporary Fields](#) on page 277. For information about SET COUNTWIDTH, see the *Developing Reporting Applications* manual.

When COUNT is used in a request, the word COUNT is appended to the default column title, unless the column title is changed with an AS phrase.

### **Example:** Counting Values

To determine how many employees are in the EMPLOYEE data source, you can count the instances of EMP\_ID, the employee identification number.

```
TABLE FILE EMPLOYEE
COUNT EMP_ID
END
```

The following shows the output of the request.

```
EMP_ID
COUNT
-----
      12
```

### **Example:** Counting Values With a Sort Phrase

To count the instances of EMP\_ID for each department, use this request:

```
TABLE FILE EMPLOYEE
COUNT EMP_ID
BY DEPARTMENT
END
```

The following shows the output of the request indicating that of the 12 EMP\_IDs in the data source, six are from the MIS department and six are from the PRODUCTION department:

DEPARTMENT	EMP_ID COUNT
MIS	6
PRODUCTION	6

### **Example:** Counting Instances of Data

The following example counts the instances of data in the LAST\_NAME, DEPARTMENT, and JOBCODE fields in the EMPLOYEE data source.

```
TABLE FILE EMPLOYEE
COUNT LAST_NAME AND DEPARTMENT AND JOBCODE
END
```

The following shows the output of the request.

LAST_NAME COUNT	DEPARTMENT COUNT	JOBCODE COUNT
12	12	19

The EMPLOYEE data source contains data on 12 employees, with one instance for each LAST\_NAME. While there are only two values for DEPARTMENT, there are 12 instances of the DEPARTMENT field because each employee works for one of the two departments. Similarly, there are 19 instances of the JOBCODE field because employees can have more than one job code during their employment.

### Counting Segment Instances

You can easily count the instances of the lowest segment in the left path of a data source by specifying an asterisk (\*) wildcard instead of a specific field name. In a single-segment data source, this effectively counts all instances in the data source.

COUNT \* accomplishes this by counting the values of the first field in the segment. Instances with a missing value in the first field are not counted (when SET MISSING=ON).

Segment instances in short paths are not counted by COUNT \*, regardless of the value of the ALL parameter of the SET command.

For more information about missing values, short paths, and the SET ALL parameter, see [Handling Records With Missing Field Values](#) on page 971.

### **Example:** Counting Segments From a Multi-Path Data Source

The following request counts the number of instances of the SKILLSEG segment of the EMPLOYEE data source.

```
TABLE FILE EMPLOYEE
COUNT *
END
```

The following shows the output of the request.

```
COUNT *
COUNT
-----
      19
```

COUNT \* counts the number of instances of the SKILLSEG segment, which is the lowest segment in the left path of the EMPLOYEE data source structure (that is, the EMPLOYEE data source joined to the JOBFIL and EDUCFIL data sources). You can see a picture of the path structure in [Displaying the Structure and Retrieval Order of a Multi-Path Data Source](#) on page 49.

**Tip:** In some environments, the following warning is displayed if you use COUNT \* with a multi-path data source (such as EMPLOYEE in the above example):

```
(FOC757) WARNING. YOU REQUESTED PRINT * OR COUNT * FOR A MULTI-PATH FILE
```

## Expanding Byte Precision for COUNT and LIST

By default, the number of characters that display for counter values retrieved using the COUNT and LIST commands is five. You can increase the number of characters to nine.

For example, if the number of records retrieved for a field exceeds 99,999 (5 bytes), asterisks appear in the report to indicate an overflow condition. You can increase the display to allow as large a count as 999,999,999 (9 bytes) using SET COUNTWIDTH.

**Note:** You can change the overflow character by issuing the SET OVERFLOWCHAR command.

**Syntax:**      **How to Set the Precision for COUNT and LIST**

```
SET COUNTWIDTH = {OFF|ON}
```

where:

**OFF**

Displays five characters (bytes) for COUNT and LIST counter values. Asterisks are displayed if the number of records retrieved for a field exceeds five characters. OFF is the default.

**ON**

Displays up to nine characters (bytes) for COUNT and LIST counter values. Asterisks are displayed if the value exceeds nine characters.

**Example:**      **Setting Precision for COUNT and LIST**

The following example shows the COUNT command with SET COUNTWIDTH = OFF:

```
TABLE FILE filename
COUNT Fldxx
BY Fldyy
END
```

	Fldxx
<u>Fldyy</u>	<u>COUNT</u>
value	*****

The following example shows the COUNT command with SET COUNTWIDTH = ON:

```
TABLE FILE filename
COUNT Fldxx
BY Fldyy
END
```

	Fldxx
<u>Fldyy</u>	<u>COUNT</u>
value	999999999

**Note:** This feature affects the width of a report when COUNTWIDTH is set to ON. Calculating the width of a report now requires an additional four display positions for each COUNT or LIST column.

**Maximum Number of Display Fields Supported in a Request**

There is no limit to the number of verb objects in a TABLE or MATCH request.

However, an error can occur under the following conditions:

- ❑ The report output format has a limit to the number of columns supported. For example, Excel, FOCUS, and XFOCUS formats have limits on the number of columns.
- ❑ The operating system has a maximum record length that cannot fit all of the columns.
- ❑ The amount of memory needed to store the output is not available.

If the combined length of the display fields in the data area exceeds the maximum capacity, an error message displays. To correct the problem, adjust the number or lengths of the fields in the request.

## Manipulating Display Fields With Prefix Operators

You can use prefix operators to perform calculations directly on the values of fields.

**Note:** Unless you change a column or ACROSS title with an AS phrase, the prefix operator is automatically added to the title. Without an AS phrase, the column title is constructed using the prefix operator and either the field name or the TITLE attribute in the Master File (if there is one):

- ❑ If there is no TITLE attribute, the field name is used.
- ❑ If there is a TITLE attribute in the Master File, the choice between using the field name or the TITLE attribute depends on the value of the TITLES parameter:
  - ❑ If SET TITLES = ON, the TITLE attribute is used.
  - ❑ If SET TITLES = OFF or NOPREFIX, the field name is used.

For a list of prefix operators and their functions, see [Functions You Can Perform With Prefix Operators](#) on page 62.

## Prefix Operator Basics

This topic describes basic syntax and notes for using prefix operators.



**Syntax: How to Use Prefix Operators**

Each prefix operator is applied to a single field, and affects only that field.

```
{SUM|COUNT} prefix.fieldname AS 'coltitle'
```

```
{PRINT|COMPUTE} RNK.byfield
```

where:

*prefix*

Is any prefix operator.

*fieldname*

Is the name of the field to be displayed in the report.

*'coltitle'*

Is the column title for the report column, enclosed in single quotation marks.

*byfield*

Is the name of a vertical sort field to be ranked in the report.

**Reference: Usage Notes for Prefix Operators**

- ☐ Because PRINT and LIST display individual field values, not an aggregate value, they are not used with prefix operators, except TOT.
- ☐ To sort by the results of a prefix command, use the phrase BY TOTAL to aggregate and sort numeric columns simultaneously. For details, see [Sorting Tabular Reports](#) on page 91.
- ☐ The WITHIN phrase is very useful when using prefixes.
- ☐ You can use the results of prefix operators in COMPUTE commands.
- ☐ With the exception of CNT. and PCT.CNT., resulting values have the same format as the field against which the prefix operation was performed.
- ☐ Text fields can only be used with the FST., LST., and CNT. prefix operators.
- ☐ PCT., TOT., PCT.CNT., RNK., and RPCT. are *not* supported with TABLEF and should not be used with TABLEF.

**Reference: Functions You Can Perform With Prefix Operators**

The following table lists prefix operators and describes the function of each.

Prefix	Function
ASQ.	Computes the average sum of squares for standard deviation in statistical analysis.
AVE.	Computes the average value of the field.
CNT.	Counts the number of occurrences of the field. The data type of the result is always Integer.
AVE.DST.	Averages the distinct values within a field.
CNT.DST.	Counts the number of distinct values within a field.
SUM.DST.	Sums the distinct values within a field.
CT.	Produces a cumulative total of the specified field. This operator only applies when used in subfootings. For details, see <a href="#">Using Headings, Footings, Titles, and Labels</a> on page 1431.
DST.	Determines the total number of distinct values in a single pass of a data source.
FST.	Generates the first physical instance of the field. Can be used with numeric or text fields.
LST.	Generates the last physical instance of the field. Can be used with numeric or text fields.
MAX.	Generates the maximum value of the field.
MDE.	Computes the mode of the field values.
MDN.	Computes the median of the field values.
MIN.	Generates the minimum value of the field.

Prefix	Function
<a href="#">PCT.</a>	Computes a field percentage based on the total values for the field. The PCT operator can be used with detail as well as summary fields.
<a href="#">PCT.CNT.</a>	Computes a field percentage based on the number of instances found. The format of the result is always F6.2 and cannot be reformatted.
<a href="#">RNK.</a>	Ranks the instances of a BY sort field in the request. Can be used in PRINT commands, COMPUTE commands, and IF or WHERE TOTAL tests.
<a href="#">ROLL.</a>	Recalculates values on summary lines using the aggregated values from lower level summary lines.
<a href="#">RPCT.</a>	Computes a field percentage based on the total values for the field across a row.
<a href="#">ST.</a>	Produces a subtotal value of the specified field at a sort break in the report. This operator only applies when used in subfootings. For details, see <a href="#">Using Headings, Footings, Titles, and Labels</a> on page 1431.
<a href="#">SUM.</a>	Sums the field values.
<a href="#">TOT.</a>	Totals the field values for use in a heading (includes footings, subheads, and subfoots).

## Averaging Values of a Field

The AVE. prefix computes the average value of a particular field. The computation is performed at the lowest sort level of the display command. It is computed as the sum of the field values within a sort group divided by the number of records in that sort group. If the request does not include a sort phrase, AVE. calculates the average for the entire report.

## Example: Averaging Values of a Field

This request calculates the average number of education hours spent in each department.

```
TABLE FILE EMPLOYEE
SUM AVE.ED_HRS BY DEPARTMENT
END
```

The following shows the output of the request.

DEPARTMENT	AVE ED_HRS
-----	-----
MIS	38.50
PRODUCTION	20.00

## Averaging the Sum of Squared Fields

The ASQ. prefix computes the average sum of squares, which is a component of the standard deviation in statistical analysis (shown as a formula in the following image).

$$\left( \sum_{i=1}^n x_i^2 \right)$$

**Note:** If the field format is integer and you get a large set of numbers, the ASQ. result may exceed the limit of the I4 field, which is 2,147,483,647. The display of any number larger than this will generate a negative number or an incorrect positive number. For this reason, we recommend that you do not use Integer fields if this result could occur.

## Example: Averaging the Sum of Squared Fields

This request calculates the sum and the sum of squared fields for the DELIVER\_AMT field.

```
TABLE FILE SALES
SUM DELIVER_AMT AND ASQ.DELIVER_AMT
BY CITY
END
```

The following shows the output of the request.

CITY	DELIVER_AMT	ASQ DELIVER_AMT
----	-----	-----
NEW YORK	300	980
NEWARK	60	900
STAMFORD	430	3637
UNIONDALE	80	1600

## Calculating Maximum and Minimum Field Values

The prefixes MAX. and MIN. produce the maximum and minimum values, respectively, within a sort group. If the request does not include a sort phrase, MAX. and MIN. produce the maximum and minimum values for the entire report.

### *Example:* Calculating Maximum and Minimum Field Values

This report request calculates the maximum and minimum values of SALARY.

```
TABLE FILE EMPLOYEE
SUM MAX.SALARY AND MIN.SALARY
END
```

The following shows the output of the request.

MAX SALARY -----	MIN SALARY -----
\$29,700.00	\$8,650.00

## Calculating Median and Mode Values for a Field

You can use the MDN. (median) and MDE. (mode) prefix operators, in conjunction with an aggregation display command (SUM, WRITE) and a numeric or smart date field, to calculate the statistical median and mode of the values in the field.

These calculations are not supported in a DEFINE command, in WHERE or IF expressions, or in a summary command. If used in a multi-verb request, they must be used at the lowest level of aggregation.

The median is the middle value (50th percentile). If there is an even number of values, the median is the average of the middle two values. The mode is the value that occurs most frequently within the set of values. If no value occurs more frequently than the others, MDE. returns the lowest value.

**Example:    Calculating the Median and Mode**

The following request against the EMPLOYEE data source displays the current salaries and calculates the average (mean), median, and mode within each department.

```
TABLE FILE EMPLOYEE
SUM CURR_SAL AS 'INDIVIDUAL,SALARIES'
AVE.CURR_SAL WITHIN DEPARTMENT AS 'DEPARTMENT,AVERAGE'
MDN.CURR_SAL WITHIN DEPARTMENT AS 'DEPARTMENT,MEDIAN'
MDE.CURR_SAL WITHIN DEPARTMENT AS 'DEPARTMENT,MODE'
BY DEPARTMENT
BY CURR_SAL NOPRINT
BY LAST_NAME NOPRINT BY FIRST_NAME NOPRINT
ON TABLE SET PAGE NOPAGE
END
```

Both departments have an even number of employees. For the MIS department, the two middle values are the same, making that value (\$18,480.00) both the median and the mode. For the PRODUCTION department, the median is the average of the two middle values (\$16,100.00 and \$21,120.00) and, since there are no duplicate values, the mode is the lowest value (\$9,500.00).

DEPARTMENT	INDIVIDUAL SALARIES	DEPARTMENT AVERAGE	DEPARTMENT MEDIAN	DEPARTMENT MODE
-----	-----	-----	-----	-----
MIS	\$9,000.00	\$18,000.33	\$18,480.00	\$18,480.00
	\$13,200.00	\$18,000.33	\$18,480.00	\$18,480.00
	\$18,480.00	\$18,000.33	\$18,480.00	\$18,480.00
	\$18,480.00	\$18,000.33	\$18,480.00	\$18,480.00
	\$21,780.00	\$18,000.33	\$18,480.00	\$18,480.00
	\$27,062.00	\$18,000.33	\$18,480.00	\$18,480.00
PRODUCTION	\$9,500.00	\$19,047.00	\$18,610.00	\$9,500.00
	\$11,000.00	\$19,047.00	\$18,610.00	\$9,500.00
	\$16,100.00	\$19,047.00	\$18,610.00	\$9,500.00
	\$21,120.00	\$19,047.00	\$18,610.00	\$9,500.00
	\$26,862.00	\$19,047.00	\$18,610.00	\$9,500.00
	\$29,700.00	\$19,047.00	\$18,610.00	\$9,500.00

**Calculating Column and Row Percentages**

For each individual value in a column, PCT. calculates what percentage that field makes up of the column total value. You can control how values are distributed down the column by sorting the column using the BY phrase. The new column of percentages has the same format as the original field.

You can also determine percentages for row values. For each individual value in a row that has been sorted using the ACROSS phrase, the RPCT. operator calculates what percentage it makes up for the total value of the row. The percentage values have the same format as the original field.

**Example: Calculating Column Percentages**

To calculate each employee share of education hours, issue the following request:

```
TABLE FILE EMPLOYEE
SUM ED_HRS PCT.ED_HRS BY LAST_NAME
ON TABLE COLUMN-TOTAL
END
```

The output is:

LAST_NAME	ED_HRS	PCT ED_HRS
-----	-----	-----
BANNING	.00	.00
BLACKWOOD	75.00	21.37
CROSS	45.00	12.82
GREENSPAN	25.00	7.12
IRVING	30.00	8.55
JONES	50.00	14.25
MCCOY	.00	.00
MCKNIGHT	50.00	14.25
ROMANS	5.00	1.42
SMITH	46.00	13.11
STEVENS	25.00	7.12
TOTAL	351.00	100.00

Since PCT. and RPCT. take the same format as the field, the column may not always total exactly 100 because of the nature of floating-point arithmetic.

**Example: Calculating Row Percentages**

The following request calculates the total units sold for each product (UNIT\_SOLD column), and the percentage that total makes up in relation to the sum of all products sold (RPCT.UNIT\_SOLD column) in each city.

```
TABLE FILE SALES
SUM UNIT_SOLD RPCT.UNIT_SOLD ROW-TOTAL
BY PROD_CODE
ACROSS CITY WHERE
CITY EQ 'NEW YORK' OR 'STAMFORD'
END
```

The output is:

PROD_CODE	CITY		STAMFORD		TOTAL	
	NEW YORK					
	UNIT_SOLD	RPCT	UNIT_SOLD	RPCT	UNIT_SOLD	RPCT
B10	30	33	60	66	90	99
B12	.	.	40	100	40	100
B17	20	40	29	59	49	99
B20	15	100	.	.	15	100
C13	.	.	25	100	25	100
C17	12	100	.	.	12	100
C7	.	.	45	100	45	100
D12	20	42	27	57	47	99
E1	30	100	.	.	30	100
E2	.	.	80	100	80	100
E3	35	33	70	66	105	99

Because UNIT\_SOLD has an integer format, the columns created by RPCT. also have integer (I) formats. Therefore, individual percentages may be truncated and the total percentage may be less than 100%. If you require precise totals, redefine the field with a format that declares decimal places (D, F).

## Producing a Direct Percent of a Count

When counting occurrences in a file, a common reporting need is determining the relative percentages of each row's count within the total number of instances. You can do this, for columns only, with the following syntax:

```
PCT.CNT.fieldname
```

The format is a decimal value of six digits with two decimal places (F6.2).

### **Example:** Producing a Direct Percent of a Count

This request illustrates the relative percentage of the values in the EMP\_ID field for each department.

```
TABLE FILE EMPLOYEE
SUM PCT.CNT.EMP_ID
BY DEPARTMENT
END
```



The output is:

	PCT.CNT
DEPARTMENT -----	EMP_ID -----
MIS	50.00
PRODUCTION	50.00

## Aggregating and Listing Unique Values

The distinct prefix operator (DST.) may be used to aggregate and list unique values of any data source field. Similar in function to the SQL COUNT, SUM, and AVG(DISTINCT col) column functions, it permits you to determine the total number of distinct values in a single pass of the data source.

The DST. operator can be used with the SUM, PRINT or COUNT commands, and also in conjunction with the aggregate prefix operators SUM., CNT., and AVE. Multiple DST. operators are supported in TABLE and TABLEF requests. They are supported in requests that use the BY, ACROSS, and FOR phrases.

Note that in a request using the PRINT command and multiple DST operators, you should issue the command SET PRINTDST=NEW. For more information, see the *Developing Reporting Applications* manual.

### **Syntax:** How to Use the Distinct Operator

*command* DST.*fieldname*

or

SUM [*operator*].DST.*fieldname*

where:

*command*

Is SUM, PRINT, or COUNT.

DST.

Indicates the distinct operator.

*fieldname*

Indicates the display-field object or field name.

*operator*

Indicates SUM., CNT., or AVE.

### **Example:** Using the Distinct Operator

The procedure requesting a count of unique ED\_HRS values is either:

```
TABLE FILE EMPLOYEE
SUM CNT.DST.ED_HRS
END
```

or

```
TABLE FILE EMPLOYEE
COUNT DST.ED_HRS
END
```

The output is:

```
COUNT
DISTINCT
ED_HRS
-----
          9
```

Notice that the count includes records for both employees with the last name SMITH, but excludes the second records for values 50.00, 25.00, and .0, resulting in nine unique ED\_HRS values.

### **Example:** Counting Distinct Field Values With Multiple Display Commands

The following request against the GGSALES data source counts the total number of records by region, then the number of records, distinct categories, and distinct products by region and by state. The DST or CNT.DST operator can be used only with the last display command:

```
TABLE FILE GGSALES
COUNT CATEGORY AS 'TOTAL,COUNT'
      BY REGION
SUM CNT.CATEGORY AS 'STATE,COUNT'
      CNT.DST.CATEGORY      CNT.DST.PRODUCT
      BY REGION
      BY ST
END
```

The output is:

Region	TOTAL COUNT	State	STATE COUNT	COUNT DISTINCT CATEGORY	COUNT DISTINCT PRODUCT
Midwest	1085	IL	362	3	9
		MO	361	3	9
		TX	362	3	9
Northeast	1084	CT	361	3	10
		MA	360	3	10
		NY	363	3	10
Southeast	1082	FL	361	3	10
		GA	361	3	10
		TN	360	3	10
West	1080	CA	721	3	10
		WA	359	3	10

### Reference: Distinct Operator Limitations

- ❑ If you reformat a column created using COUNT DST. or the CNT.DST operator, you must reformat it to an integer (I) data type. If you specify another data type, the following error occurs:

(FOC950) INVALID REFORMAT OPTION WITH COUNT OR CNT.

- ❑ The following error occurs if you use the prefix operators CNT., SUM., and AVE. with any other display command:

(FOC1853) CNT/SUM/AVE.DST CAN ONLY BE USED WITH AGGREGATION VERBS

- ❑ The following error occurs if you use DST. in a MATCH command:

(FOC1854) THE DST OPERATOR IS ONLY SUPPORTED IN TABLE REQUESTS

- ❑ The following error occurs if you reformat a BY field (when used with the PRINT command, the DST.fieldname becomes a BY field):

(FOC1862) REFORMAT DST.FIELD IS NOT SUPPORTED WITH PRINT

- ❑ The following error occurs if you use the DST. operator with NOSPLIT:

(FOC1864) THE DST OPERATOR IS NOT SUPPORTED WITH NOSPLIT

- ❑ The following error occurs if you use a multi-verb request, SUM DST.fieldname BY field PRINT fld BY fld (a verb object operator used with the SUM command must be at the lowest level of aggregation):

(FOC1867) DST OPERATOR MUST BE AT THE LOWEST LEVEL OF AGGREGATION

- ❑ The DST. operator may not be used as part of a HEADING or a FOOTING.

Retrieving First and Last Records

FST. is a prefix that displays the first retrieved record selected for a given field. LST. displays the last retrieved record selected for a given field.

When using the FST. and LST. prefix operators, it is important to understand how your data source is structured.

- ❑ If the record is in a segment with values organized from lowest to highest (segment type S1), the first logical record that the FST. prefix operator retrieves is the lowest value in the set of values. The LST. prefix operator would, therefore, retrieve the highest value in the set of values.
- ❑ If the record is in a segment with values organized from highest to lowest (segment type SH1), the first logical record that the FST. prefix operator retrieves is the highest value in the set of values. The LST. prefix operator would, therefore, retrieve the lowest value in the set of values.

For more information on segment types and file design, see the *Describing Data With WebFOCUS Language* manual. If you wish to reorganize the data in the data source or restructure the data source while reporting, see [Improving Report Processing](#) on page 1839.

Example: Retrieving the First Record

The following request retrieves the first logical record in the EMP\_ID field:

```
TABLE FILE EMPLOYEE
SUM FST.EMP_ID
END
```

The output is:

```
FST
EMP_ID
-----
071382660
```

**Example: Segment Types and Retrieving Records**

The EMPLOYEE data source contains the DEDUCT segment, which orders the fields DED\_CODE and DED\_AMT from lowest value to highest value (segment type of S1). The DED\_CODE field indicates the type of deduction, such as CITY, STATE, FED, and FICA. The following request retrieves the first logical record for DED\_CODE for each employee:

```
TABLE FILE EMPLOYEE
SUM FST.DED_CODE
BY EMP_ID
END
```

The output is:

EMP_ID	FST DED_CODE
-----	-----
071382660	CITY
112847612	CITY
117593129	CITY
119265415	CITY
119329144	CITY
123764317	CITY
126724188	CITY
219984371	CITY
326179357	CITY
451123478	CITY
543729165	CITY
818692173	CITY

Note, however, the command SUM LST.DED\_CODE would have retrieved the last logical record for DED\_CODE for each employee.

If the record is in a segment with values organized from highest to lowest (segment type SH1), the first logical record that the FST. prefix operator retrieves is the highest value in the set of values. The LST. prefix operator would therefore retrieve the lowest value in the set of values.

For example, the EMPLOYEE data source contains the PAYINFO segment, which orders the fields JOBCODE, SALARY, PCT\_INC, and DAT\_INC from highest value to lowest value (segment type SH1). The following request retrieves the first logical record for SALARY for each employee:

```
TABLEF FILE EMPLOYEE
SUM FST.SALARY
BY EMP_ID
END
```

The output is:

EMP_ID	FST SALARY
-----	-----
071382660	\$11,000.00
112847612	\$13,200.00
117593129	\$18,480.00
119265415	\$9,500.00
119329144	\$29,700.00
123764317	\$26,862.00
126724188	\$21,120.00
219984371	\$18,480.00
326179357	\$21,780.00
451123478	\$16,100.00
543729165	\$9,000.00
818692173	\$27,062.00

However, the command `SUM LST.SALARY` would have retrieved the last logical record for SALARY for each employee.

### Summing and Counting Values

You can count occurrences and summarize values with one display command using the prefix operators `CNT.`, `SUM.`, and `TOT.` Just like the `COUNT` command, `CNT.` counts the occurrences of the field it prefixes. Just like the `SUM` command, `SUM.` sums the values of the field it prefixes. `TOT.` sums the values of the field it prefixes when used in a heading (including footings, subheads, and subfoots).

#### *Example:* Counting Values With CNT

The following request counts the occurrences of `PRODUCT_ID`, and sums the value of `UNIT_PRICE`.

```
TABLE FILE GGPRODS
SUM CNT.PRODUCT_ID AND UNIT_PRICE
END
```

The output is:

Product Code	Unit Price
COUNT	
-----	-----
10	660.00

**Example: Summing Values With SUM**

The following request counts the occurrences of PRODUCT\_ID, and sums the value of UNIT\_PRICE.

```
TABLE FILE GGPRODS
COUNT PRODUCT_ID AND SUM.UNIT_PRICE
END
```

The output is:

Product Code	Unit Price
-----	-----
10	660.00

**Example: Summing Values With TOT**

The following request uses the TOT prefix operator to show the total of current salaries for all employees.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME
BY DEPARTMENT
ON TABLE SUBFOOT
"Total salaries equal: <TOT.CURR_SAL"
END
```

The output is:

DEPARTMENT	LAST_NAME	
-----	-----	
MIS	SMITH	
	JONES	
	MCCOY	
	BLACKWOOD	
	GREENSPAN	
	CROSS	
PRODUCTION	STEVENS	
	SMITH	
	BANNING	
	IRVING	
	ROMANS	
	MCKNIGHT	
Total salaries equal:		\$222,284.00

## Ranking Sort Field Values With RNK.

RANKED BY *fieldname*, when used in a sort phrase in a TABLE request, not only sorts the data by the specified field, but assigns a RANK value to the instances. The RNK. prefix operator also calculates the rank while allowing the RANK value to be printed anywhere on the page. You use this operator by specifying RNK.*fieldname*, where *fieldname* is a BY field in the request.

The ranking process occurs after selecting and sorting records. Therefore, the RNK. operator cannot be used in a WHERE or IF selection test or in a virtual (DEFINE) field. However, RNK.*fieldname* can be used in a WHERE TOTAL or IF TOTAL test or in a calculated (COMPUTE) value. You can change the default column title for the rank field using an AS phrase.

You can apply the RNK. operator to multiple sort fields, in which case the rank for each BY field is calculated within its higher level BY field.

### **Syntax:** How to Calculate Ranks Using the RNK. Prefix Operator

In a PRINT command, COMPUTE expression, or IF/WHERE TOTAL expression :

*RNK.field ...*

where:

*field*

Is a vertical (BY) sort field in the request.

### **Example:** Ranking Within Sort Groups

The following request ranks years of service within department and ranks salary within years of service and department. Note that years of service depends on the value of TODAY. The output for this example was valid when run in September, 2006:

```
DEFINE FILE EMPDATA
  TODAY/YYMD = &YYMD;
  YRS_SERVICE/I9 = DATEDIF(HIREDATE,TODAY,'Y');
END
TABLE FILE EMPDATA
PRINT SALARY
  RNK.YRS_SERVICE AS 'RANKING,BY,SERVICE'
  RNK.SALARY AS 'SALARY,RANK'
  BY DEPT
  BY HIGHEST YRS_SERVICE
  BY HIGHEST SALARY NOPRINT
WHERE DEPT EQ 'MARKETING' OR 'SALES'
ON TABLE SET PAGE NOPAGE
END
```



The output is:

DEPT	YRS_SERVICE	SALARY	RANKING BY SERVICE	SALARY RANK
----	-----	-----	-----	-----
MARKETING	17	\$55,500.00	1	1
		\$55,500.00	1	1
	16	\$62,500.00	2	1
		\$62,500.00	2	1
		\$62,500.00	2	1
		\$58,800.00	2	2
		\$52,000.00	2	3
		\$35,200.00	2	4
		\$32,300.00	2	5
	15	\$50,500.00	3	1
		\$43,400.00	3	2
SALES	17	\$115,000.00	1	1
		\$54,100.00	1	2
	16	\$70,000.00	2	1
		\$43,000.00	2	2
	15	\$43,600.00	3	1
		\$39,000.00	3	2
	15	\$30,500.00	3	3

### **Example:** Using RNK. in a WHERE TOTAL Test

The following request displays only those rows in the highest two salary ranks within the years of service category. Note that years of service depends on the value of TODAY. The output for this example was valid when run in September, 2006:

```

DEFINE FILE EMPDATA
  TODAY/YYMD = &YYMD;
  YRS_SERVICE/I9 = DATEDIF(HIREDATE,TODAY,'Y');
END
TABLE FILE EMPDATA
PRINT LASTNAME FIRSTNAME RNK.SALARY
BY HIGHEST YRS_SERVICE BY HIGHEST SALARY
WHERE TOTAL RNK.SALARY LE 2
END

```

The output is:

YRS_SERVICE	SALARY	LASTNAME	FIRSTNAME	RANK SALARY
-----	-----	-----	-----	-----
17	\$115,000.00	LASTRA	KAREN	1
	\$80,500.00	NOZAWA	JIM	2
16	\$83,000.00	SANCHEZ	EVELYN	1
	\$70,000.00	CASSANOVA	LOIS	2
15	\$62,500.00	HIRSCHMAN	ROSE	1
		WANG	JOHN	1
	\$50,500.00	LEWIS	CASSANDRA	2

**Example:**    **Using RNK. in a COMPUTE Command**

The following request sets a flag to Y for records in which the salary rank within department is less than or equal to 5 and the rank of years of service within salary and department is less than or equal to 6. Otherwise, the flag has the value N. Note that the years of service depends on the value of TODAY. The output for this example was valid when run in September, 2006:

```
DEFINE FILE EMPDATA
  TODAY/YYMD = &YYMD;
  YRS_SERVICE/I9 = DATEDIF(HIREDATE,TODAY,'Y');
END
TABLE FILE EMPDATA
PRINT RNK.SALARY RNK.YRS_SERVICE
COMPUTE FLAG/A1 = IF RNK.SALARY LE 5  AND RNK.YRS_SERVICE LE 6
      THEN 'Y' ELSE 'N';
BY DEPT BY SALARY BY YRS_SERVICE
WHERE DEPT EQ 'MARKETING' OR 'SALES'
ON TABLE SET PAGE NOPAGE
END
```

The output is:

DEPT			RANK		FLAG
	SALARY	YRS_SERVICE	SALARY	YRS_SERVICE	
MARKETING	\$32,300.00	16	1	1	Y
	\$35,200.00	16	2	1	Y
	\$43,400.00	15	3	1	Y
	\$50,500.00	15	4	1	Y
	\$52,000.00	16	5	1	Y
	\$55,500.00	17	6	1	N
			6	1	N
	\$58,800.00	16	7	1	N
	\$62,500.00	16	8	1	N
			8	1	N
SALES	\$30,500.00	15	1	1	Y
	\$39,000.00	15	2	1	Y
	\$43,000.00	16	3	1	Y
	\$43,600.00	15	4	1	Y
	\$54,100.00	17	5	1	Y
	\$70,000.00	16	6	1	N
	\$115,000.00	17	7	1	N

**Rolling Up Calculations on Summary Rows**

Using SUMMARIZE and RECOMPUTE, you can recalculate values at sort field breaks, but these calculations use the detail data to calculate the value for the summary line.

Using the ROLL. operator in conjunction with another prefix operator on a summary line recalculates the sort break values using the values from summary lines generated for the lower level sort break.

The operator combinations supported are:

- ❑ ROLL.SUM. (same as ROLL.). Alphanumeric fields are supported with SUM. This returns either the first, minimum, maximum, or last value according to the SUMPREFIX parameter.
- ❑ ROLL.AVE.
- ❑ ROLL.MAX. (supported with alphanumeric fields as well as numeric fields)
- ❑ ROLL.MIN. (supported with alphanumeric fields as well as numeric fields)
- ❑ ROLL.FST. (supported with alphanumeric fields as well as numeric fields)
- ❑ ROLL.LST. (supported with alphanumeric fields as well as numeric fields)
- ❑ ROLL.CNT.
- ❑ ROLL.ASQ.

ROLL.*prefix* on a summary line indicates that the prefix operation will be performed on the summary values from the next lowest level of summary command.

If the ROLL. operator is used without another prefix operator, it is treated as a SUM. Therefore, if the summary command for the lowest BY field specifies AVE., and the next higher specifies ROLL., the result will be the sum of the averages. To get the average of the averages, you would use ROLL.AVE at the higher level.

**Note:** With SUMMARIZE and SUB-TOTAL, the same calculations are propagated to all higher level sort breaks.

### **Syntax:** How to Roll Up Summary Values

```
BY field {SUMMARIZE|SUBTOTAL|SUB-TOTAL|RECOMPUTE} [ROLL.][prefix1.]
[field1 field2 ...[*]] [ROLL.][prefix2.] [fieldn ...]
```

Or:

```
BY field
```

```
ON field {SUMMARIZE|SUBTOTAL|SUB-TOTAL|RECOMPUTE} ROLL.[prefix.]
[field1 field2 ...[*]]
```

where:

ROLL.

Indicates that the summary values should be calculated using the summary values from the next lowest level summary command.

*field*

Is a BY field in the request.

*prefix1, prefix2*

Are prefix operators to use for the summary values. It can be one of the following operators: SUM. (the default operator if none is specified), AVE., MAX., MIN., FST., LST., CNT., ASQ.

*field1 field2 fieldn*

Are fields to be summarized.

\*

Indicates that all fields, numeric and alphanumeric, should be included on the summary lines. You can either use the asterisk to display all columns or reference the specific columns you want to display.

### **Example:** Rolling Up an Average Calculation

The following request against the GGSALES data source contains two sort fields, REGION and ST. The summary command for REGION applies the AVE. operator to the sum of the units value for each state.

```
TABLE FILE GGSALES
  SUM UNITS AS 'Inventory '
    BY REGION
  BY ST
  ON REGION SUBTOTAL      AVE.  AS 'Average'
  WHERE DATE GE 19971001
  WHERE REGION EQ 'West' OR 'Northeast'
  ON TABLE SET PAGE NOPAGE
END
```

On the output, the UNITS values for each state are averaged to calculate the subtotal for each region. The UNITS values for each state are also used to calculate the average for the grand total row.

Region	State	Inventory
-----	-----	-----
Northeast	CT	37234
	MA	35720
	NY	36248
Average Northeast		
		36400
West	CA	75553
	WA	40969
Average West		
		58261
TOTAL		
		45144

The following version of the request adds a summary command for the grand total line that includes the ROLL. operator:

```
TABLE FILE GGSALES
  SUM UNITS AS 'Inventory '
  BY REGION
  BY ST
  ON REGION SUBTOTAL AVE. AS 'Average'
  WHERE DATE GE 19971001
  WHERE REGION EQ 'West' OR 'Northeast'
  ON TABLE SUBTOTAL ROLL.AVE. AS ROLL.AVE
  ON TABLE SET PAGE NOPAGE
  END
```

On the output, the UNITS values for each state are averaged to calculate the subtotal for each region, and those region subtotal values are used to calculate the average for the grand total row:

Region	State	Inventory
-----	-----	-----
Northeast	CT	37234
	MA	35720
	NY	36248
Average Northeast		
		36400
West	CA	75553
	WA	40969
Average West		
		58261
ROLL.AVE		47330

### ***Example:*** Propagating Rollups to Higher Level Sort Breaks

The following request against the GGSALES data source has three BY fields. The SUBTOTAL command for the PRODUCT sort field specifies AVE., and the SUMMARIZE command for the higher level sort field, REGION, specifies ROLL.AVE.

```
TABLE FILE GGSALES
SUM UNITS
BY REGION
BY PRODUCT
BY HIGHEST DATE
WHERE DATE GE 19971001
      WHERE REGION EQ 'Midwest' OR 'Northeast'
      WHERE PRODUCT LIKE 'C%'
      ON PRODUCT SUBTOTAL AVE.
      ON REGION SUMMARIZE ROLL.AVE. AS ROLL.AVE
ON TABLE SET PAGE NOPAGE
END
```

On the output, the detail rows for each date are used to calculate the average for each product. Because of the ROLL.AVE. at the region level, the averages for each product are used to calculate the averages for each region, and the region averages are used to calculate the average for the grand total line:

Region	Product	Date	Unit Sales
-----	-----	----	-----
Midwest	Coffee Grinder	1997/12/01	4648
		1997/11/01	3144
		1997/10/01	1597
*TOTAL PRODUCT Coffee Grinder			3129
	Coffee Pot	1997/12/01	1769
		1997/11/01	1462
		1997/10/01	2346
*TOTAL PRODUCT Coffee Pot			1859
	Croissant	1997/12/01	7436
		1997/11/01	5528
		1997/10/01	6060
*TOTAL PRODUCT Croissant			6341
ROLL.AVE Midwest			3776
Northeast	Capuccino	1997/12/01	1188
		1997/11/01	2282
		1997/10/01	3675
*TOTAL PRODUCT Capuccino			2381
	Coffee Grinder	1997/12/01	1536
		1997/11/01	1399
		1997/10/01	1315
*TOTAL PRODUCT Coffee Grinder			1416
	Coffee Pot	1997/12/01	1442
		1997/11/01	2129
		1997/10/01	2082
*TOTAL PRODUCT Coffee Pot			1884
	Croissant	1997/12/01	4291
		1997/11/01	6978
		1997/10/01	4741
*TOTAL PRODUCT Croissant			5336
ROLL.AVE Northeast			2754
TOTAL			3265

**Reference:** Usage Notes for ROLL.

- ❑ ROLL.*prefix* on a summary line indicates that the prefix operation will be performed on the summary values from the next lowest level of summary command.
- ❑ If no summary command was issued at the level below the ROLL., and no other operator was used in conjunction with the ROLL., a SUM. will be calculated. If the lower level had no summary command and ROLL. was used with another prefix operator (for example, ROLL.AVE.), the specified prefix operator will be used. For example, ROLL.AVE. will become AVE.
- ❑ CNT. *prefix* shows the number of data lines displayed, which is not affected by MULTILINES.
- ❑ ROLL.CNT. *prefix* shows the number of summary lines displayed, which is affected by MULTILINES.

## Using Report-Level Prefix Operators

Report level prefix operators are available for headings, footings, subheadings, subfootings, verb objects, and calculated values (COMPUTES) that calculate the average, maximum, minimum, and count for the entire report. They are based on the TOT. operator, which calculates total values to include in a heading.

These operators cannot be referenced in WHERE or WHERE TOTAL tests. However, they can be used in a COMPUTE command to generate a calculated value that can be used in a WHERE TOTAL test.

**Syntax:** How to Calculate Report-Level Average, Maximum, Minimum, and Count Values

*operator.field*

where:

*operator*

Can be one of the following prefix operators.

- ❑ **TOTAVE.** Calculates the average value of the field for the entire table.
- ❑ **TOTMAX.** Calculates the maximum value of the field for the entire table.
- ❑ **TOTMIN.** Calculates the minimum value of the field for the entire table.
- ❑ **TOTCNT.** Calculates the count of the field instances for the entire table.



*field*

Is a verb object or calculated value in the request.

**Example: Using Prefix Operators in a Heading**

The following request uses prefix operators in the heading.

```
TABLE FILE WF_RETAIL_LITE
HEADING
"Heading Calculations:"
"Total:          <TOT.COGS_US"
"Count:          <TOTCNT.COGS_US"
"Average:        <TOTAVE.COGS_US"
"Minimum:        <TOTMIN.COGS_US"
"Maximum:        <TOTMAX.COGS_US"
SUM COGS_US CNT.COGS_US AS Count AVE.COGS_US AS Average
MIN.COGS_US AS Minimum MAX.COGS_US AS Maximum
BY BUSINESS_REGION AS Region
BY PRODUCT_CATEGORY AS Category
WHERE BUSINESS_REGION NE 'Oceania'
ON TABLE SUBTOTAL COGS_US CNT.COGS_US AS Total
ON TABLE SET PAGE NOPAGE
ON TABLE SET SHOWBLANKS ON
ON TABLE SET STYLE *
type=report,grid=off, size=11,$
ENDSTYLE
END
```

The output is shown in the following image.

Heading Calculations:

Total: \$2,940,745.00

Count: 9961

Average: \$295.23

Minimum: \$16.00

Maximum: \$9,750.00

<u>Region</u>	<u>Category</u>	<u>Cost of Goods</u>	<u>Count</u>	<u>Average</u>	<u>Minimum</u>	<u>Maximum</u>
EMEA	Accessories	\$143,987.00	630	\$228.55	\$16.00	\$2,850.00
	Camcorder	\$187,000.00	611	\$306.06	\$60.00	\$8,610.00
	Computers	\$47,616.00	266	\$179.01	\$81.00	\$668.00
	Media Player	\$328,401.00	956	\$343.52	\$36.00	\$1,640.00
	Stereo Systems	\$361,756.00	1412	\$256.20	\$48.00	\$2,295.00
	Televisions	\$100,443.00	120	\$837.02	\$275.00	\$6,500.00
	Video Production	\$78,722.00	276	\$285.22	\$78.00	\$1,880.00
North America	Accessories	\$171,306.00	743	\$230.56	\$16.00	\$2,040.00
	Camcorder	\$242,967.00	692	\$351.11	\$60.00	\$6,000.00
	Computers	\$53,329.00	284	\$187.78	\$81.00	\$668.00
	Media Player	\$386,681.00	1118	\$345.87	\$36.00	\$1,600.00
	Stereo Systems	\$412,036.00	1636	\$251.86	\$48.00	\$2,700.00
	Televisions	\$101,212.00	121	\$836.46	\$275.00	\$6,500.00
	Video Production	\$89,489.00	325	\$275.35	\$78.00	\$1,410.00
South America	Accessories	\$25,723.00	103	\$249.74	\$16.00	\$1,470.00
	Camcorder	\$22,686.00	110	\$206.24	\$60.00	\$1,000.00
	Computers	\$7,845.00	47	\$166.91	\$81.00	\$501.00
	Media Player	\$60,183.00	161	\$373.81	\$36.00	\$1,480.00
	Stereo Systems	\$81,823.00	284	\$288.11	\$48.00	\$2,700.00
	Televisions	\$25,800.00	25	\$1,032.00	\$275.00	\$9,750.00
	Video Production	\$11,740.00	41	\$286.34	\$78.00	\$1,410.00
Total		\$2,940,745.00	9961			

### Reference: Usage Notes for Report-Level Prefix Operators

- ❑ These operators can be used on a field in a heading or footing without being referenced in a display command in the request.

- ❑ They work in a heading or footing for real or virtual (DEFINE) fields. They work in a display command field list on real fields, virtual (DEFINE) fields, and calculated (COMPUTE) values that are calculated prior to their use in the request.
- ❑ They can be used in subheadings and subfootings to reference the total value for the entire report.

## Displaying Pop-up Field Descriptions for Column Titles

You can have pop-up field descriptions display in an HTML report when the mouse pointer is positioned over column titles. Field description text displays in a pop-up box near the column title using the default font for the report. Pop-up text appears for report column titles including titles created with ACROSS phrases and stacked column titles created with OVER phrases.

The pop-up text displayed for a column title is defined by the Description attribute in the Master File for the corresponding field. If a column title has no Description entry in the Master File, then no pop-up box is generated when your mouse is positioned over the title.

For more information about the Description attribute, see *Null or MISSING Values: MISSING* in the *Describing Data With WebFOCUS Language* manual.

### **Syntax:** How to Use the POPUPDESC Command

```
SET POPUPDESC = {ON|OFF}
```

where:

ON

Enables pop-up field descriptions when your mouse pointer is positioned over column titles.

OFF

Disables pop-up field descriptions when your mouse pointer is positioned over column titles. OFF is the default value.

### **Example:** Using the POPUPDESC Command

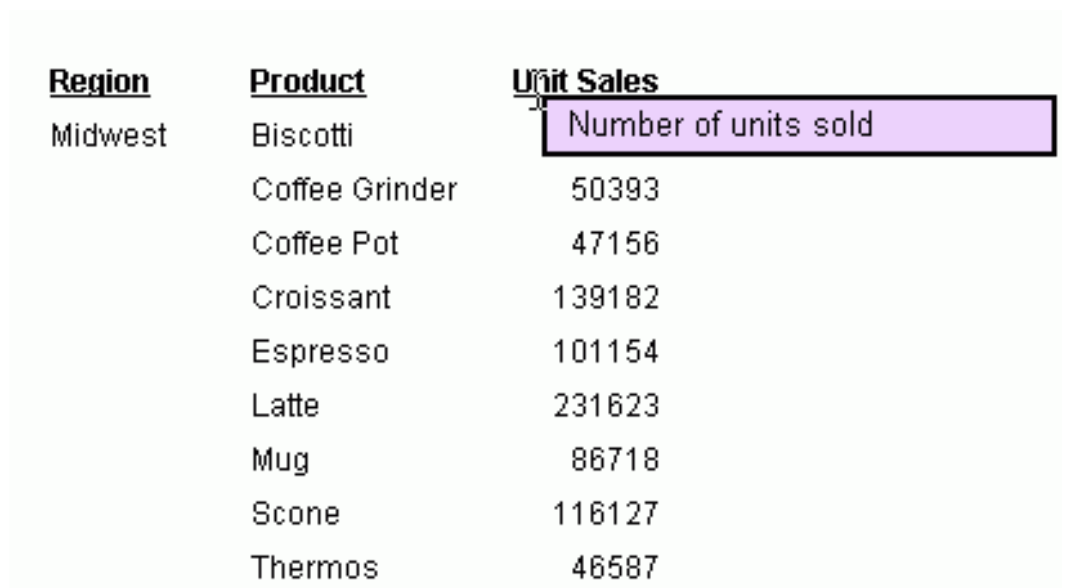
The Master File referenced by the report contains the following:

```
FIELD=UNITS, ALIAS=E10, FORMAT=I08, TITLE='Unit Sales',
DESC='Number of units sold', $
```

The code used to create the report is:

```
TABLE FILE GGSALES
SUM UNITS
BY REGION
BY PRODUCT
WHERE REGION EQ 'Midwest'
ON TABLE SET POPUPDESC ON
END
```

The following image shows the report output and the pop-up field description text that displays when your mouse pointer is positioned over the Unit Sales column title.



<u>Region</u>	<u>Product</u>	<u>Unit Sales</u>
Midwest	Biscotti	Number of units sold
	Coffee Grinder	50393
	Coffee Pot	47156
	Croissant	139182
	Espresso	101154
	Latte	231623
	Mug	86718
	Scone	116127
	Thermos	46587

**Reference: Distributing Reports With Pop-up Field Descriptions Using ReportCaster**

Distributing an HTML report containing pop-up field descriptions with ReportCaster requires the use of JavaScript components located on the WebFOCUS Client. To access these components from a report distributed by ReportCaster, the scheduled procedure must contain the SET FOCHTMLURL command, which must be set to an absolute URL, instead of the default value. For example,

```
SET FOCHTMLURL = http://hostname[:port]/ibi_apps/ibi_html
```

where:

*hostname[:port]*

Is the host name and optional port number (specified only if you are not using the default port number) where the WebFOCUS Web application is deployed.

*ibi\_apps/ibi\_html*

ibi\_apps is the site-customized web server alias pointing to the WEBFOCUS81/ibi\_apps directory (where ibi\_apps is the default value). ibi\_html is a directory within the path to the JavaScript files that are required to be accessible.

For more information about coding reports for use with ReportCaster, see the *Tips and Techniques for Coding a ReportCaster Report* appendix in the *ReportCaster* manual.





# Chapter 3

## Sorting Tabular Reports

---

Sorting enables you to group or organize report information vertically and horizontally, in rows and columns, and specify a desired sequence of data items in the report.

Any field in the data source can be the sort field. If you wish, you can select several sort fields, nesting one within another. Sort fields appear only when their values change.

### In this chapter:

- |  |   |
|--|---|
| <input type="checkbox"/> <a href="#">Sorting Tabular Reports Overview</a>                  | <input type="checkbox"/> <a href="#">Ranking Sort Field Values</a>                            |
| <input type="checkbox"/> <a href="#">Sorting Rows</a>                                      | <input type="checkbox"/> <a href="#">Grouping Numeric Data Into Ranges</a>                    |
| <input type="checkbox"/> <a href="#">Sorting Columns</a>                                   | <input type="checkbox"/> <a href="#">Restricting Sort Field Values by Highest/Lowest Rank</a> |
| <input type="checkbox"/> <a href="#">Controlling Display of Sort Field Values</a>          | <input type="checkbox"/> <a href="#">Sorting and Aggregating Report Columns</a>               |
| <input type="checkbox"/> <a href="#">Reformatting Sort Fields</a>                          | <input type="checkbox"/> <a href="#">Hiding Sort Values</a>                                   |
| <input type="checkbox"/> <a href="#">Manipulating Display Field Values in a Sort Group</a> | <input type="checkbox"/> <a href="#">Sort Performance Considerations</a>                      |
| <input type="checkbox"/> <a href="#">Creating a Matrix Report</a>                          | <input type="checkbox"/> <a href="#">Sorting With Multiple Display Commands</a>               |
| <input type="checkbox"/> <a href="#">Controlling Collation Sequence</a>                    | <input type="checkbox"/> <a href="#">Improving Efficiency With External Sorts</a>             |
| <input type="checkbox"/> <a href="#">Specifying the Sort Order</a>                         | <input type="checkbox"/> <a href="#">Hierarchical Reporting: BY HIERARCHY</a>                 |
- 

### Sorting Tabular Reports Overview

You sort a report using vertical (BY) and horizontal (ACROSS) phrases:

- ☐ BY displays the sort field values vertically, creating rows. Vertical sort fields are displayed in the left-most columns of the report.
- ☐ ACROSS displays the sort field values horizontally, creating columns. Horizontal sort fields are displayed across the top of the report.
- ☐ BY and ACROSS phrases used in the same report create rows and columns, producing a grid or matrix.

A request can include up to 128 sort phrases consisting of any combination of BY and ACROSS phrases.

Additional sorting options include:

- ☐ Sorting from low to high values or from high to low values, and defining your own sorting sequence.
- ☐ Sorting based on case-sensitive or case-insensitive collation sequence.
- ☐ Leaving the value of the sort field out of the report.
- ☐ Grouping numeric data into tiles such as percentiles or deciles.
- ☐ Aggregating and sorting numeric columns simultaneously.
- ☐ Grouping numeric data into ranges.
- ☐ Ranking data, and selecting data based on rank.

### **Reference:** **Sorting and Displaying Data**

There are two ways that you can sort information, depending on the type of display command you use:

- ☐ You can sort and display individual values of a field using the PRINT or LIST command.
- ☐ You can group and aggregate information. For example, you can show the number of field occurrences per sort value using the COUNT command, or summing the field values using the SUM command.

When you use the display commands PRINT and LIST, the report may generate several rows per sort value; specifically, one row for each occurrence of the display field. When you use the commands SUM and COUNT, the report generates one row for each unique set of sort values. For related information, see [Sorting With Multiple Display Commands](#) on page 182.

For details on all display commands, see [Displaying Report Data](#) on page 43.

## Sorting Rows

You can sort report information vertically using the BY phrase. This creates rows in your report. You can include up to 128 sort phrases (BY phrases plus ACROSS phrases) per report request (127 if using PRINT or LIST display commands).

Sort fields appear when their value changes. However, you can display every sort value using the BYDISPLAY parameter. For an example, see [Displaying All Vertical \(BY\) Sort Field Values](#).



**Syntax:**      **How to Sort by Rows**

```
BY {HIGHEST|LOWEST} [n] sortfield [AS 'text']
```

where:

HIGHEST

Sorts in descending order.

LOWEST

Sorts in ascending order. LOWEST is the default value.

*n*

Specifies that only *n* sort field values are included in the report.

*sortfield*

Is the name of the sort field.

*text*

Is the column heading to use for the sort field column on the report output.

**Reference:**      **Usage Notes for Sorting Rows**

- ☐ When using the display command LIST with a BY phrase, the LIST counter is reset to 1 each time the major sort value changes.
- ☐ The default sort sequence is low-to-high, with the following variations for different operating systems. In z/OS the sequence is a-z, A-Z, 0-9 for alphanumeric fields; 0-9 for numeric fields. In UNIX and Windows the sequence is 0-9, A-Z, a-z for alphanumeric fields; 0-9 for numeric. You can specify other sorting sequences, as described in [Specifying the Sort Order](#) on page 152.
- ☐ You cannot use text fields as sort fields. Text fields are those described in the Master File with a FORMAT value of TX.
- ☐ You can use a temporary field created by a DEFINE command, or by the DEFINE attribute in a Master File, as a sort field. In order to use a temporary field created by a COMPUTE command as a sort field, you must use the BY TOTAL phrase instead of the BY phrase.
- ☐ If you specify several sort fields when reporting from a multi-path data source, all the sort fields must be in the same path.
- ☐ Sort phrases cannot contain format information for fields.

- ❑ Each sort field value appears only once in the report. For example, if there are six employees in the MIS department, a request that declares

```
PRINT LAST_NAME BY DEPARTMENT
```

prints MIS once, followed by six employee names. You can populate every vertical sort column cell with a value, even if the value is repeating, using the SET BYDISPLAY parameter. For details, see *Displaying All Vertical (BY) Sort Field Values*.

### **Example:**    **Sorting Rows With BY**

The following illustrates how to display all employee IDs by department.

```
TABLE FILE EMPLOYEE
PRINT EMP_ID
BY DEPARTMENT
END
```

The output displays a row for each EMP\_ID in each department:

<u>DEPARTMENT</u>	<u>EMP_ID</u>
MIS	112847612
	117593129
	219984371
	326179357
	543729165
	818692173
PRODUCTION	071382660
	119265415
	119329144
	123764317
	126724188
	451123478

### **Using Multiple Vertical (BY) Sort Fields**

You can organize information in a report by using more than one sort field. When you specify several sort fields, the sequence of the BY phrases determines the sort order. The first BY phrase sets the major sort break, the second BY phrase sets the second sort break, and so on. Each successive sort is nested within the previous one.

**Example: Sorting With Multiple Vertical (BY) Sort Fields**

The following request uses multiple vertical (BY) sort fields.

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL
BY DEPARTMENT BY LAST_NAME
WHERE CURR_SAL GT 21500
END
```

The output is:

DEPARTMENT	LAST_NAME	CURR_SAL
-----	-----	-----
MIS	BLACKWOOD	\$21,780.00
	CROSS	\$27,062.00
PRODUCTION	BANNING	\$29,700.00
	IRVING	\$26,862.00

**Displaying a Row for Data Excluded by a Sort Phrase**

In a sort phrase, you can restrict the number of sort values displayed. With the PLUS OTHERS phrase, you can aggregate all other values to a separate group and display this group as an additional report row.

**Syntax: How to Display Data Excluded by a Sort Phrase**

```
[RANKED] BY {HIGHEST|LOWEST|TOP|BOTTOM}  n srtfield [AS 'text']
           [PLUS OTHERS AS 'othertext']
           [IN-GROUPS-OF m1 [TOP n2]]
           [IN-RANGES-OF m3 [TOP n4]]
```

where:

**LOWEST**

Sorts in ascending order, beginning with the lowest value and continuing to the highest value (a-z, A-Z, 0-9 for alphanumeric fields; 0-9 for numeric fields). BOTTOM is a synonym for LOWEST.

**HIGHEST**

Sorts in descending order, beginning with the highest value and continuing to the lowest value. TOP is a synonym for HIGHEST.

**n**

Specifies that only *n* sort field values are included in the report.

**srtfield**

Is the name of the sort field.

### *text*

Is the text to be used as the column heading for the sort field values.

### *othertext*

Is the text to be used as the row title for the "others" grouping. This AS phrase must be the AS phrase immediately following the PLUS OTHERS phrase.

### *m1*

Is the incremental value between sort field groups.

### *n2*

Is an optional number that defines the highest group label to be included in the report.

### *m3*

Is an integer greater than zero indicating the range by which sort field values are grouped.

### *n4*

Is an optional number that defines the highest range label to be included in the report. The range is extended to include all data values higher than this value.

## **Reference: Usage Notes for PLUS OTHERS**

- ☐ Alphanumeric group keys are not supported.
- ☐ Only one PLUS OTHERS phrase is supported in a request.
- ☐ In a request with multiple display commands, the BY field that has the PLUS OTHERS phrase does not have to be the last BY field in the request.
- ☐ The BY ROWS OVER, TILES, ACROSS, and BY TOTAL phrases are not supported with PLUS OTHERS.
- ☐ PLUS OTHERS is not supported in a MATCH FILE request. However, MORE in a TABLE request is supported.
- ☐ HOLD is supported for formats PDF, PS, HTML, DOC, and WP.

**Example: Displaying a Row Representing Sort Field Values Excluded by a Sort Phrase**

The following request displays the top two ED\_HRS values and aggregates the values not included in a row labeled Others:

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL LAST_NAME
  BY HIGHEST 2 ED_HRS
  PLUS OTHERS AS 'Others'
END
```

The output is:

ED_HRS	CURR_SAL	LAST_NAME
-----	-----	-----
75.00	\$21,780.00	BLACKWOOD
50.00	\$18,480.00	JONES
	\$16,100.00	MCKNIGHT
Others	\$165,924.00	

**Example: Displaying a Row Representing Data Not Included in Any Sort Field Grouping**

The following request sorts by highest 2 ED\_HRS and groups the sort field values by increments of 25 ED\_HRS. Values that fall below the lowest group label are included in the Others category. All values above the top group label are included in the top group:

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL LAST_NAME
  BY HIGHEST 2 ED_HRS
  PLUS OTHERS AS 'Others'
IN-GROUPS-OF 25 TOP 50
END
```

The output is:

ED_HRS	CURR_SAL	LAST_NAME
-----	-----	-----
50.00	\$18,480.00	JONES
	\$21,780.00	BLACKWOOD
	\$16,100.00	MCKNIGHT
25.00	\$11,000.00	STEVENS
	\$13,200.00	SMITH
	\$26,862.00	IRVING
	\$9,000.00	GREENSPAN
	\$27,062.00	CROSS
Others	\$78,800.00	

If the BY HIGHEST phrase is changed to BY LOWEST, all values above the top grouping (50 ED\_HRS and above) are included in the Others category:

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL LAST_NAME
  BY LOWEST 2 ED_HRS
  PLUS OTHERS AS 'Others'
IN-GROUPS-OF 25 TOP 50
END
```

The output is:

ED_HRS	CURR_SAL	LAST_NAME
-----	-----	-----
.00	\$9,500.00	SMITH
	\$29,700.00	BANNING
	\$21,120.00	ROMANS
	\$18,480.00	MCCOY
25.00	\$11,000.00	STEVENS
	\$13,200.00	SMITH
	\$26,862.00	IRVING
	\$9,000.00	GREENSPAN
	\$27,062.00	CROSS
Others	\$56,360.00	

Sorting Columns

You can sort report information horizontally using the ACROSS phrase. This creates columns in your report. The total number of ACROSS columns is equal to the total number of ACROSS sort field values multiplied by the total number of display fields.

A request can include up to 128 sort phrases consisting of any combination of BY and ACROSS phrases.

The maximum number of display fields your report can contain is determined by a combination of factors. In general, if a horizontal (ACROSS) sort field contains many data values, you may exceed the allowed width for reports, or create a report that is difficult to read. For details, see [Displaying Report Data](#) on page 43.

You can produce column totals or summaries for ACROSS sort field values using ACROSS-TOTAL, SUBTOTAL, SUB-TOTAL, RECOMPUTE, and SUMMARIZE. For details, see [Including Totals and Subtotals](#) on page 369.

**Syntax: How to Sort Columns**

`ACROSS sortfield`

where:

`sortfield`

Is the name of the sort field.

**Reference: Usage Notes for Sorting Columns**

- ❑ You cannot use text fields as sort fields. Text fields are those described in the Master File with a FORMAT value of TX.
- ❑ You can use a temporary field created by a DEFINE command, or by the DEFINE attribute in a Master File, as a sort field. However, you cannot use a temporary field created by a COMPUTE command as a sort field. You can accomplish this indirectly by first creating a HOLD file that includes the field, and then reporting from the HOLD file. HOLD files are described in [Saving and Reusing Your Report Output](#) on page 471.
- ❑ For an ACROSS phrase, the SET SPACES parameter controls the distance between ACROSS sets. For more information, see [Laying Out the Report Page](#) on page 1249.
- ❑ Sort phrases cannot contain format information for fields.
- ❑ If you specify several sort fields when reporting from a multipath data source, all the sort fields must be in the same path.
- ❑ In styled output formats (PDF, HTML, DHTML, PPT, PPTX, and XLSX), the width of ACROSS titles and ACROSS values above the data columns is defined as the largest width of all data columns, and associated column titles, within the ACROSS groups. To change the size of the ACROSS groups, apply SQUEEZE, WRAP, or WIDTH definitions to the data columns within each group.
- ❑ Each sort field value is displayed only once in the report. For example, if there are six employees in the MIS department, a report that declares

```
PRINT LAST_NAME ACROSS DEPARTMENT
```

prints MIS once, followed by six employee names.

**Example:**     **Sorting Columns With ACROSS**

The following illustrates how to show the total salary outlay for each department. This request is sorted horizontally with an ACROSS phrase.

```
TABLE FILE EMPLOYEE
SUM CURR_SAL ACROSS DEPARTMENT
END
```

The output is:

DEPARTMENT	
MIS	PRODUCTION
-----	
\$108,002.00	\$114,282.00

Notice that the horizontal sort displays a column for each sort field (department).

**Controlling Display of an ACROSS Title for a Single Field**

Using the SET ACRSVRBTITL command, you can control the display of an ACROSS column title in an ACROSS group. The behavior of the title is determined by the number of verb columns in the ACROSS group. The field count is affected by the following features, which add internal matrix columns to the report:

- ☐ Fields in a heading or footing.
- ☐ Fields whose display is suppressed with the NOPRINT phrase.
- ☐ Reformatted fields (which are normally counted twice).
- ☐ A COMPUTE command referencing multiple fields.

**Syntax:**     **How to Control Display of an ACROSS Title for a Single Field**

```
SET ACRSVRBTITL = {HIDEONE|ON|OFF}
ON TABLE SET ACRSVRBTITL {HIDEONE|ON|OFF}
```

where:

HIDEONE

Suppresses the title when there is only one display field, or there is only one display field and the request contains one or more of the features that add internal matrix columns to the report. This value is the default.

ON

Always displays the title even if there is only one display field.



**OFF**

Suppresses the title when there is only one display field. Displays the title when there is only one display field and the request contains one or more of the features that add internal matrix columns to the report. This is legacy behavior.

**Example: Hiding an ACROSS Title With ACRSVRBTTTL**

The following request against the GGSALES data source has a display field in the heading:

```
SET ACRSVRBTTTL=HIDEONE
TABLE FILE GGSALES
HEADING
"Sales Report for <CATEGORY with ACRSVRBTTTL=HIDEONE"
" "
SUM DOLLARS AS Sales
BY CATEGORY
ACROSS REGION
WHERE CATEGORY EQ 'Food'
ON TABLE SET PAGE NOPAGE
ON TABLE SET ACROSSTITLE SIDE
ON TABLE SET ACROSSLINE SKIP
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, FONT='ARIAL', SIZE=9, SQUEEZE=ON,$
TYPE=TITLE,JUSTIFY=LEFT,BACKCOLOR=RGB(102 102 102),COLOR=RGB(255 255
255),STYLE=BOLD,$
TYPE=HEADING, SIZE=11, STYLE=BOLD,JUSTIFY=CENTER, $
TYPE=ACROSSTITLE,STYLE=BOLD,$
TYPE=ACROSSVALUE,BACKCOLOR=RGB(218 225 232),$
END
```

Using the default value for ACRSVRBTTTL, HIDEONE, suppresses the ACROSS title *Sales*, even though the heading displays a field value that adds a report column to the internal matrix.

The report output is shown in the following image:

**Sales Report for Food with ACRSVRBTTTL=HIDEONE**

	Region	Midwest	Northeast	Southeast	West
Category					
Food		4338271	4379994	4308731	4202337

If you change the SUM command to the following:

```
SUM DOLLARS/D12CM
```

the field in the heading and the reformatted dollar sales values add report columns to the internal matrix, but the ACROSS title *Sales* is still suppressed.

The report output is shown in the following image:

**Sales Report for Food with ACRSVRBTITL=HIDEONE**

Region	Midwest	Northeast	Southeast	West
Category				
Food	\$4,338,271	\$4,379,994	\$4,308,731	\$4,202,337

Using the ACRSVRBTITL value ON, without reformatting the dollar sales column, does not suppress the ACROSS title Sales because the heading displays a field value that adds a report column to the internal matrix.

The report output is shown in the following image:

**Sales Report for Food with ACRSVRBTITL=ON**

Region	Midwest	Northeast	Southeast	West
Category	Sales	Sales	Sales	Sales
Food	4338271	4379994	4308731	4202337

If you change the SUM command to the following:

`SUM DOLLARS/D12CMC`

the field in the heading and the reformatted dollar sales values add report columns to the internal matrix, so the ACROSS title Sales is not suppressed.

The report output is shown in the following image:

**Sales Report for Food with ACRSVRBTITL=ON**

Region	Midwest	Northeast	Southeast	West
Category	Sales	Sales	Sales	Sales
Food	\$4,338,271	\$4,379,994	\$4,308,731	\$4,202,337

With the setting ACRSVRBTITL=OFF, the field in the heading adds a report column to the internal matrix, and the ACROSS title Sales is not suppressed.

The report output is shown in the following image:

## Sales Report for Food with ACRSVRBTTITL=OFF

Region	Midwest	Northeast	Southeast	West
Category	Sales	Sales	Sales	Sales
Food	4338271	4379994	4308731	4202337

If you change the SUM command to the following:

```
SUM DOLLARS/D12CM
```

the field in the heading and the reformatted dollar sales values add report columns to the internal matrix, and the ACROSS title *Sales* is not suppressed.

The report output is shown in the following image:

## Sales Report for Food with ACRSVRBTTITL=OFF

Region	Midwest	Northeast	Southeast	West
Category	Sales	Sales	Sales	Sales
Food	\$4,338,271	\$4,379,994	\$4,308,731	\$4,202,337

### Positioning ACROSS Titles on Report Output

In a report that uses the ACROSS sort phrase to sort values horizontally across the page, by default, two lines are generated on the report output for the ACROSS columns. The first line displays the name of the sort field (ACROSS title), and the second line displays the values for that sort field (ACROSS value). The ACROSS field name is left justified above the first ACROSS value.

If you want to display both the ACROSS title and the ACROSS values on one line in the PDF, HTML, EXL2K, or XLSX report output, you can issue the SET ACROSSTITLE = SIDE command. This command places ACROSS titles to the left of the ACROSS values. By default, the titles are right justified in the space above the BY field titles. You can change the justification of the ACROSS title by adding the JUSTIFY attribute to the StyleSheet declaration for the ACROSSTITLE component. If there are no BY fields, the heading line that is created by default to display the ACROSS title will not be generated.

This feature is designed for use in requests that have both ACROSS fields and BY fields. For requests with ACROSS fields but no BY fields, the set command is ignored, and the ACROSS titles are not moved.

Note that for certain output formats, you can control whether column titles are underlined using the SET TITLELINE command. SET ACROSSTITLE is a synonym for SET TITLELINE. For information, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.

### **Syntax:**      **How to Control the Position of ACROSS Field Names**

```
SET ACROSSTITLE = {ABOVE|SIDE}
```

where:

[ABOVE](#)

Displays ACROSS titles above their ACROSS values. ABOVE is the default value.

[SIDE](#)

Displays ACROSS titles to the left of their ACROSS values, above the BY columns.

### **Reference:**      **Usage Notes for SET ACROSSTITLE**

- ☐ When the ACROSS value wraps, the ACROSS title aligns with the top line of the wrapped ACROSS values.
- ☐ The ACROSS title spans the width of the BY columns. If the ACROSS title value is larger than the width of the BY columns on the current page, the value is truncated. The first panel may have more BY fields than subsequent panels, if SET BYPANEL is set to a value smaller than the total number of BY fields.
- ☐ This setting will not create a new column within the report for the title placement.
  - ☐ If the request does not have any BY fields, the ACROSS title is not moved.
  - ☐ With BYPANEL=OFF, the ACROSS title is not displayed on subsequent panels.
- ☐ WRAP is not supported for ACROSSTITLE with SET ACROSSTITLE=SIDE.

**Example: Placing the ACROSS Title on the Same Line as the ACROSS Values**

The following example against the GGSALES data source has two ACROSS sort fields, CATEGORY and PRODUCT. SET ACROSSTITLE=SIDE moves the ACROSS title to the left of the ACROSS values. With BYPANEL=ON the ACROSS titles are repeated in the same location on each subsequent panel.

```

SET ACROSSTITLE=SIDE
SET BYPANEL=ON
TABLE FILE GGSALES
SUM
    DOLLARS/I8M AS ''
BY REGION
BY ST
BY CITY
ACROSS CATEGORY
ACROSS PRODUCT
WHERE PRODUCT NE 'Capuccino';
ON TABLE SET PAGE-NUM ON
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
    UNITS=IN,
    SQUEEZE=ON,
    ORIENTATION=PORTRAIT,
$
TYPE=REPORT,
    FONT='ARIAL',
    SIZE=10,
    BORDER=LIGHT,
$
TYPE=ACROSSVALUE,
    WRAP=ON,
$
ENDSTYLE
END

```

The ACROSS title *Category* displays to the left of the ACROSS values *Coffee*, *Food*, and *Gifts*. The ACROSS title *Product* displays to the left of the ACROSS values *Espresso*, *Latte*, *Biscotti*, and so on. The ACROSS titles are right-justified above the space occupied by the BY field names *Region*, *State*, and *City*. Notice that the ACROSS value *Croissant* wraps onto a second line, and the ACROSS title is aligned with the top line. The following shows panel 1:

PAGE 1.1

Category			Coffee		Food			Gifts
Product			Espresso	Latte	Biscotti	Croissant	Scone	Coffee Grinder
Region	State	City						
Midwest	IL	Chicago	\$420,439	\$978,340	\$378,412	\$549,366	\$595,069	\$233,292
	MO	St. Louis	\$419,143	\$966,981	\$368,077	\$613,871	\$481,953	\$181,570
	TX	Houston	\$455,365	\$938,245	\$345,238	\$587,887	\$418,398	\$204,292
Northeast	CT	New Haven	\$279,373	\$926,052	\$589,355	\$551,489	\$283,874	\$169,908
	MA	Boston	\$248,356	\$917,737	\$570,391	\$497,234	\$332,486	\$177,940
	NY	New York	\$322,378	\$928,026	\$642,259	\$622,095	\$290,811	\$161,352
Southeast	FL	Orlando	\$256,539	\$889,887	\$511,597	\$602,076	\$311,836	\$217,204
	GA	Atlanta	\$317,389	\$907,365	\$555,231	\$661,806	\$273,420	\$217,254
	TN	Memphis	\$279,644	\$820,584	\$438,889	\$638,477	\$315,399	\$171,319
West	CA	Los Angeles	\$267,809	\$809,647	\$266,030	\$800,084	\$315,584	\$214,557
		San Francisco	\$338,270	\$935,862	\$269,518	\$824,457	\$292,839	\$187,123
	WA	Seattle	\$301,538	\$924,896	\$328,320	\$801,060	\$304,445	\$201,756

The following shows panel 2:

PAGE 1.2

Category			Gifts		
Product			Coffee Pot	Mug	Thermos
Region	State	City			
Midwest	IL	Chicago	\$204,828	\$376,754	\$187,901
	MO	St. Louis	\$190,153	\$343,852	\$195,686
	TX	Houston	\$204,897	\$366,337	\$194,319
Northeast	CT	New Haven	\$208,209	\$392,967	\$221,827
	MA	Boston	\$184,119	\$401,944	\$203,435
	NY	New York	\$198,452	\$349,300	\$178,836
Southeast	FL	Orlando	\$212,057	\$409,466	\$195,526
	GA	Atlanta	\$232,552	\$355,447	\$227,482
	TN	Memphis	\$200,694	\$337,790	\$209,449
West	CA	Los Angeles	\$202,285	\$381,926	\$207,613
		San Francisco	\$197,845	\$379,399	\$165,115
	WA	Seattle	\$213,494	\$427,339	\$198,640

**Example: ACROSS Title Spacing**

The following example against the GGSales data source has two BY fields and two ACROSS fields. This example does not set borders on and does not enable wrapping of the ACROSS values. SET ACROSSTITLE=SIDE moves the ACROSS title to the left of the ACROSS values. The SET BYPANEL=1 command repeats only the first BY field on the second panel. To prevent the ACROSS titles from being truncated to fit above the BY field on the second panel, the first BY field has an AS name that is longer than the default name:

```
SET ACROSSTITLE=SIDE
SET BYPANEL=1
TABLE FILE GGSales
SUM
    DOLLARS/I8M AS ''
BY ST AS 'State Code'
BY CITY
ACROSS CATEGORY AS 'Categories'
ACROSS PRODUCT AS 'Products'
WHERE PRODUCT NE 'Capuccino';
ON TABLE SET PAGE-NUM ON
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
    UNITS=IN,
    SQUEEZE=ON,
    ORIENTATION=PORTRAIT,
$
TYPE=REPORT,
    FONT='ARIAL',
    SIZE=10,
$
ENDSTYLE
END
```

The first panel follows:

PAGE 1.1

State Code	Categories Products	Coffee Espresso	Latte	Food Biscotti	Croissant	Scone
	City					
CA	Los Angeles	\$267,809	\$809,647	\$266,030	\$800,084	\$315,584
	San Francisco	\$338,270	\$935,862	\$269,518	\$824,457	\$292,839
CT	New Haven	\$279,373	\$926,052	\$589,355	\$551,489	\$283,874
FL	Orlando	\$256,539	\$889,887	\$511,597	\$602,076	\$311,836
GA	Atlanta	\$317,389	\$907,365	\$555,231	\$661,806	\$273,420
IL	Chicago	\$420,439	\$978,340	\$378,412	\$549,366	\$595,069
MA	Boston	\$248,356	\$917,737	\$570,391	\$497,234	\$332,486
MO	St. Louis	\$419,143	\$966,981	\$368,077	\$613,871	\$481,953
NY	New York	\$322,378	\$928,026	\$642,259	\$622,095	\$290,811
TN	Memphis	\$279,644	\$820,584	\$438,889	\$638,477	\$315,399
TX	Houston	\$455,365	\$938,245	\$345,238	\$587,887	\$418,398
WA	Seattle	\$301,538	\$924,896	\$328,320	\$801,060	\$304,445

Because of the SET BYPANEL=1 command, the space available above the BY fields on the second panel is smaller than the space on the initial panel. The AS name *State Code* adds space for the ACROSS titles, so the titles are not truncated on the second panel:

PAGE 1.2

State Code	Categories Products	Gifts Coffee Grinder	Coffee Pot	Mug	Thermos
CA	\$214,557	\$202,285	\$381,926	\$207,613	
	\$187,123	\$197,845	\$379,399	\$165,115	
CT	\$169,908	\$208,209	\$392,967	\$221,827	
FL	\$217,204	\$212,057	\$409,466	\$195,526	
GA	\$217,254	\$232,552	\$355,447	\$227,482	
IL	\$233,292	\$204,828	\$376,754	\$187,901	
MA	\$177,940	\$184,119	\$401,944	\$203,435	
MO	\$181,570	\$190,153	\$343,852	\$195,686	
NY	\$161,352	\$198,452	\$349,300	\$178,836	
TN	\$171,319	\$200,694	\$337,790	\$209,449	
TX	\$204,292	\$204,897	\$366,337	\$194,319	
WA	\$201,756	\$213,494	\$427,339	\$198,640	



**Example: Specifying Background Color for ACROSS Values With ACROSSTITLE=SIDE**

The following request against the GGSALES data source places the ACROSS titles next to the ACROSS values and sets matching styling of font color and backcolor for the ACROSSTITLES, ACROSSVALUES, and column titles to white text on grey background color.

```
SET ACROSSTITLE=SIDE
TABLE FILE GGSALES
SUM DOLLARS/I8M AS ''
BY REGION
BY ST
BY CITY
ACROSS CATEGORY
ACROSS PRODUCT
WHERE CATEGORY EQ 'Coffee' OR 'Food';
ON TABLE SET PAGE-NUM NOPAGE
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
SQUEEZE=ON,UNITS=IN,ORIENTATION=PORTRAIT,$
TYPE=REPORT,Font='ARIAL',SIZE=10,BORDER=LIGHT,$
TYPE=ACROSSTITLE,COLOR=WHITE, BACKCOLOR=GREY,$
TYPE=ACROSSVALUE,COLOR=WHITE, BACKCOLOR=GREY,$
TYPE=TITLE,COLOR=WHITE, BACKCOLOR=GREY,$
ENDSTYLE
END
```

The output has a grey background color and white text for the ACROSS titles, ACROSS values, and column titles.

Category			Coffee			Food		
Product			Capuccino	Espresso	Latte	Biscotti	Croissant	Scone
Region	State	City						
Midwest	IL	Chicago	.	\$420,439	\$978,340	\$378,412	\$549,366	\$595,069
	MO	St. Louis	.	\$419,143	\$966,981	\$368,077	\$613,871	\$481,953
	TX	Houston	.	\$455,365	\$938,245	\$345,238	\$587,887	\$418,398
Northeast	CT	New Haven	\$158,995	\$279,373	\$926,052	\$589,355	\$551,489	\$283,874
	MA	Boston	\$174,344	\$248,356	\$917,737	\$570,391	\$497,234	\$332,486
	NY	New York	\$208,756	\$322,378	\$928,026	\$642,259	\$622,095	\$290,811
Southeast	FL	Orlando	\$317,027	\$256,539	\$889,887	\$511,597	\$602,076	\$311,836
	GA	Atlanta	\$352,161	\$317,389	\$907,365	\$555,231	\$661,806	\$273,420
	TN	Memphis	\$274,812	\$279,644	\$820,584	\$438,889	\$638,477	\$315,399
West	CA	Los Angeles	\$306,468	\$267,809	\$809,647	\$266,030	\$800,084	\$315,584
		San Francisco	\$279,830	\$338,270	\$935,862	\$269,518	\$824,457	\$292,839
	WA	Seattle	\$309,197	\$301,538	\$924,896	\$328,320	\$801,060	\$304,445

Using Multiple Horizontal (ACROSS) Sort Fields

You can sort a report using more than one sort field. When several sort fields are used, the ACROSS phrase order determines the sorting order. The first ACROSS phrase sets the first sort break, the second ACROSS phrase sets the second sort break, and so on. Each successive sort is nested within the previous one.

*Example:*     **Sorting With Multiple Horizontal (ACROSS) Phrases**

The following request sorts the sum of current salaries, first by department and then by job code.

```
TABLE FILE EMPLOYEE
SUM CURR_SAL
ACROSS DEPARTMENT ACROSS CURR_JOBCODE
WHERE CURR_SAL GT 21500
END
```

The output is:

DEPARTMENT				
	MIS			
PRODUCTION				
CURR_JOBCODE				
A17	A17	B04	A15	
-----				
	\$27,062.00	\$21,780.00	\$26,862.00	\$29,700.00

Collapsing PRINT With ACROSS

The PRINT command generates a report that has a single line for each record retrieved from the data source after screening out those that fail IF or WHERE tests. When PRINT is used in conjunction with an ACROSS phrase, many of the generated columns may be empty. Those columns display the missing data symbol.

To avoid printing such a sparse report, you can use the SET ACROSSPRT command to compress the lines in the report. The number of lines is reduced within each sort group by swapping non-missing values from lower lines with missing values from higher lines, and then eliminating any lines whose columns all have missing values.

Because data may be moved to different report lines, row-based calculations such as ROW-TOTAL and ACROSS-TOTAL in a compressed report are different from those in a non-compressed report. Column calculations are not affected by compressing the report lines.

*Syntax:*     **How to Compress Report Lines**

```
SET ACROSSPRT = {NORMAL | COMPRESS}
```

```
ON TABLE SET ACROSSPRT{NORMAL|COMPRESS}
```

where:

NORMAL

Does not compress report lines. NORMAL is the default value.

COMPRESS

Compresses report lines by promoting data values up to replace missing values within a sort group.

**Reference:** Usage Notes for SET ACROSSPRT

- ☐ Compression applies only to ACROSS fields, including ACROSS ... COLUMNS. It has no effect on BY fields.
- ☐ The only data values that are subject to compression are true missing values. If the value of the stored data is either 0 or blank and the metadata indicates that MISSING is ON, that value is not subject to compression.

**Example:** Compressing Report Output With SET ACROSSPRT

The following request against the GGSALES data source prints unit sales by product across region:

```
TABLE FILE GGSALES
PRINT UNITS/I5
BY PRODUCT
ACROSS REGION
WHERE DATE FROM '19971201' TO '19971231';
WHERE PRODUCT EQ 'Capuccino' OR 'Espresso';
ON TABLE SET ACROSSPRT NORMAL
ON TABLE SET PAGE NOPAGE
END
```

Each line of the report represents one sale in one region, so at most one column in each row has a non-missing value when ACROSSPRT is set to NORMAL.

Product	Region			
	Midwest	Northeast	Southeast	West
	Unit Sales	Unit Sales	Unit Sales	Unit Sales
-----				
Capuccino	.	936	.	.
	.	116	.	.
	.	136	.	.
	.	.	1616	.
	.	.	1118	.
	.	.	774	.
	.	.	.	1696
	.	.	.	1519
	.	.	.	836
	.	.	.	.
Espresso	1333	.	.	.
	280	.	.	.
	139	.	.	.
	.	1363	.	.
	.	634	.	.
	.	406	.	.
	.	.	1028	.
	.	.	1014	.
	.	.	885	.
	.	.	.	1782
	.	.	.	1399
	.	.	.	551

Setting ACROSSPRT to COMPRESS promotes non-missing values up to replace missing values within the same BY group and then eliminates lines consisting of all missing values.

```
TABLE FILE GGSales
PRINT UNITS/I5
BY PRODUCT
ACROSS REGION
WHERE DATE FROM '19971201' TO '19971231';
WHERE PRODUCT EQ 'Capuccino' OR 'Espresso';
ON TABLE SET ACROSSPRT COMPRESS
ON TABLE SET PAGE NOPAGE
END
```

The output is:

Product	Region		Northeast		Southeast		West	
	Midwest	Unit Sales	Unit	Sales	Unit	Sales	Unit	Sales
Capuccino	.		936		1616		1696	
	.		116		1118		1519	
	.		136		774		836	
Espresso	1333		1363		1028		1782	
	280		634		1014		1399	
	139		406		885		551	

## Hiding Null Columns in ACROSS Groups

Report requests that use the ACROSS sort phrase generate a group of columns (one for each display field in the request) under each value of the ACROSS field. In many cases, some of these columns have only missing or null values. You can use the HIDENULLACRS parameter to hide the display of ACROSS groups containing only null columns in styled output formats. If there is a BY field with a PAGE-BREAK option, columns are hidden on each page of output generated by that PAGE-BREAK option. If the request contains no BY page breaks, ACROSS groups that are missing for the entire report are hidden.

Hiding null ACROSS columns is supported for all styled output formats except for the EXL2K PIVOT and EXL2K FORMULA options. It is not supported for Active Technologies.

### **Syntax:** How to Hide Null ACROSS Columns

```
SET HIDENULLACRS = {ON|OFF}
```

```
ON TABLE SET HIDENULLACRS {ON|OFF}
```

where:

**ON**

Hides columns with missing data in ACROSS groups within a BY-generated page break.

**OFF**

Does not hide columns. OFF is the default value.

### **Reference:** Usage Notes for Hiding Null Columns Within ACROSS Groups

- ❑ Aligning items in headings with the associated data columns (HEADALIGN) is not supported for ACROSS reports.

- ❑ Hiding ACROSS columns will not affect items placed in heading elements with spot markers or explicit positioning. This means that after ACROSS group columns are hidden, items may align with the ACROSS columns differently than expected.

### **Reference: Features Not Supported For Hiding Null ACROSS Columns**

- ❑ Active Technologies.
- ❑ EXL2K FORMULA.
- ❑ EXL2K PIVOT.
- ❑ OVER.
- ❑ HIDENULLACRS is only supported with page breaks specified in ON *byfieldname* PAGE-BREAK phrases or BY *fieldname* PAGE-BREAK phrases. It is not supported with:
  - ❑ BY *field* ROWS *value* OVER.
  - ❑ FML FOR fields (FOR *fieldvalue* OVER PAGE-BREAK).

### **Hiding ACROSS Groups and Columns Within BY Page Breaks**

Hiding null columns is most useful when a BY sort field has the PAGE-BREAK option, either on the BY phrase itself or in an ON phrase. The change in value of the BY field determines when a page break is generated for that BY field. The change in BY field value defines the limits within which the ACROSS columns will be hidden, even if the BY field value spans multiple physical pages.

There is no way to specify a particular BY field with this setting, so if the request has multiple BY fields with page breaks, the setting applies to all of them. If there are no BY fields with page breaks, an ACROSS column must be missing for the entire report in order to be hidden.

The entire ACROSS group will be hidden either when the ACROSS value is missing or when all of the rows for all of the display columns under that ACROSS value contain null or missing values within the given BY field value.

The set of pages generated for a BY field value with a page break will be hidden if all ACROSS groups within that BY field value are hidden.

When columns are removed from a page or a panel, the existing columns are resituated to fill the missing space.

**Example: Hiding Null ACROSS Groups**

The following request against the GGSALES data source has a page break on the BY field named REGION and an ACROSS phrase on the CITY field. The display fields in each ACROSS group are UNITS and DOLLARS:

```
SET HIDENULLACRS=OFF
TABLE FILE GGSALES
SUM UNITS DOLLARS
BY REGION PAGE-BREAK
BY ST
ACROSS CITY
WHERE CITY LE 'Memphis'
ON TABLE SET HTMLCSS ON
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET STYLE *
TYPE=REPORT, FONT=ARIAL, SIZE=9,$
ENDSTYLE
END
```

With SET HIDENULLACRS=OFF, all columns display:

PAGE 1													
		City											
		Atlanta		Boston		Chicago		Houston		Los Angeles		Memphis	
Region	State	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales
Midwest	IL	.	.	.	.	307581	3924401	.	.	.	.	.	.
	TX	.	.	.	.	.	.	299737	3714978	.	.	.	.
PAGE 2													
		City											
		Atlanta		Boston		Chicago		Houston		Los Angeles		Memphis	
Region	State	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales
Northeast	MA	.	.	301909	3707986	.	.	.	.	.	.	.	.
PAGE 3													
		City											
		Atlanta		Boston		Chicago		Houston		Los Angeles		Memphis	
Region	State	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales
Southeast	GA	330283	4100107	.	.	.	.	.	.	.	.	.	.
	TN	.	.	.	.	.	.	.	.	.	.	294647	3687057
PAGE 4													
		City											
		Atlanta		Boston		Chicago		Houston		Los Angeles		Memphis	
Region	State	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales
West	CA	.	.	.	.	.	.	.	.	298070	3772003	.	.

Running the request with SET HIDE NULLACRS=ON eliminates the ACROSS groups for cities with missing data within each region. For example, the Midwest region has no columns for Atlanta or Boston:

PAGE 1					
		City			
		Chicago		Houston	
Region	State	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales
Midwest	IL	307581	3924401	.	.
	TX	.	.	299737	3714978
PAGE 2					
		City			
		Boston			
Region	State	Unit Sales	Dollar Sales		
Northeast	MA	301909	3707986		
PAGE 3					
		City			
		Atlanta		Memphis	
Region	State	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales
Southeast	GA	330283	4100107	.	.
	TN	.	.	294647	3687057
PAGE 4					
		City			
		Los Angeles			
Region	State	Unit Sales	Dollar Sales		
West	CA	298070	3772003		



**Example: Hiding Columns Within ACROSS Groups**

In the following request against the GGSALES data source, REGION is a BY field with a PAGE-BREAK, and PRODUCT is the ACROSS field. The DEFINE command creates a field named SHOWDOLLARS that has missing values for the Espresso column within the ACROSS group Coffee:

```

SET HIDENULLACRS=OFF
SET BYPANEL=2
DEFINE FILE GGSALES
SHOWDOLLARS/I8M MISSING ON = IF (PRODUCT EQ 'Espresso') THEN MISSING ELSE
DOLLARS;
END
TABLE FILE GGSALES
HEADING
"Page <TABPAGENO "
SUM SHOWDOLLARS AS ''
BY REGION
BY ST
BY CITY
ACROSS PRODUCT
WHERE REGION EQ 'Midwest' OR 'Northeast'
WHERE CATEGORY EQ 'Coffee';
ON REGION PAGE-BREAK
ON TABLE SET PAGE-NUM ON
ON TABLE NOTOTAL
ON TABLE SET HTMLCSS ON
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET STYLE *
    UNITS=IN,
    SQUEEZE=ON,
    ORIENTATION=PORTRAIT,
$
TYPE=REPORT,
    GRID=OFF,
    FONT='ARIAL',
    SIZE=9,
$
ENDSTYLE
END

```

Running the request with SET HIDE NULLACRS=OFF displays the Espresso column and any other column containing missing values within the Coffee group:

Page 1

Region	State	City	Product	
			Capuccino	Espresso Latte
Midwest	IL	Chicago	.	\$978,340
	MO	St. Louis	.	\$966,981
	TX	Houston	.	\$938,245

Page 2

Region	State	City	Product	
			Capuccino	Espresso Latte
Northeast	CT	New Haven	\$158,995	\$926,052
	MA	Boston	\$174,344	\$917,737
	NY	New York	\$208,756	\$928,026

Running the request with SET HIDE NULLACRS=ON hides columns with missing data within each region. On page 1 (Midwest), both the Capuccino and Espresso columns are hidden, while on page 2 (Northeast), only the Espresso column is hidden:

## Page 1

Region	State	City	Product
			Latte
Midwest	IL	Chicago	\$978,340
	MO	St. Louis	\$966,981
	TX	Houston	\$938,245

## Page 2

Region	State	City	Product	
			Capuccino	Latte
Northeast	CT	New Haven	\$158,995	\$926,052
	MA	Boston	\$174,344	\$917,737
	NY	New York	\$208,756	\$928,026

**Example:**     **Hiding Null Columns With Multiple ACROSS Fields**

The following request against the GGSALES data source has two ACROSS fields, PRODUCT and CATEGORY. The BY field with the page break is REGION. The DEFINE command creates a field named SHOWDOLLARS that has missing values for the Espresso column within the ACROSS group Coffee and for the entire ACROSS group Gifts.

```

SET HIDE NULLACRS=OFF
DEFINE FILE GGSALES
SHOWDOLLARS/I8M MISSING ON = IF (PRODUCT EQ 'Espresso' OR
    CATEGORY EQ 'Gifts') THEN MISSING ELSE DOLLARS;
END
TABLE FILE GGSALES
SUM SHOWDOLLARS AS ''
BY REGION
BY ST
BY CITY
ACROSS CATEGORY
ACROSS PRODUCT
WHERE REGION EQ 'Midwest' OR 'Northeast'
ON REGION PAGE-BREAK
HEADING
"Page <TABPAGE NO />TABLASTPAGE "
ON TABLE SET PAGE-NUM OFF
ON TABLE SET BYPANEL ON
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
    UNITS=IN,
    PAGESIZE='Letter',
    SQUEEZE=ON,
    ORIENTATION=PORTRAIT,
$
TYPE=REPORT,
    HEADPANEL=ON,
    GRID=OFF,
    FONT='ARIAL',
    SIZE=8,
$
ENDSTYLE
END
    
```

Running the request with SET HIDE NULLACRS=OFF displays all of the columns:

Page 1 /2

			Category									
			Coffee		Food			Gifts				
			Product									
			Capuccino	Espresso	Latte	Biscotti	Croissant	Scone	Coffee Grinder	Coffee Pot	Mug	Thermos
Region	State	City										
Midwest	IL	Chicago	.	.	\$978,340	\$378,412	\$549,366	\$595,069	.	.	.	.
	MO	St. Louis	.	.	\$966,981	\$368,077	\$613,871	\$481,953	.	.	.	.
	TX	Houston	.	.	\$938,245	\$345,238	\$587,887	\$418,398	.	.	.	.

Page 2 /2

			Category									
			Coffee		Food			Gifts				
			Product									
			Capuccino	Espresso	Latte	Biscotti	Croissant	Scone	Coffee Grinder	Coffee Pot	Mug	Thermos
Region	State	City										
Northeast	CT	New Haven	\$158,995	.	\$926,052	\$589,355	\$551,489	\$283,874	.	.	.	.
	MA	Boston	\$174,344	.	\$917,737	\$570,391	\$497,234	\$332,486	.	.	.	.
	NY	New York	\$208,756	.	\$928,026	\$642,259	\$622,095	\$290,811	.	.	.	.

Running the request with SET HIDE NULLACRS=ON hides the Espresso product and the entire Gifts category within each region. On page 1 (Midwest), the Gifts group and the Espresso and Capuccino columns are hidden, while on page 2 (Northeast), the Gifts group and the Espresso column are hidden:

Page 1 /2

Region	State	City	Category			
			Coffee	Food		
			Product			
			Latte	Biscotti	Croissant	Scone
Midwest	IL	Chicago	\$978,340	\$378,412	\$549,366	\$595,069
	MO	St. Louis	\$966,981	\$368,077	\$613,871	\$481,953
	TX	Houston	\$938,245	\$345,238	\$587,887	\$418,398

Page 2 /2

Region	State	City	Category				
			Coffee	Food			
			Product				
			Capuccino	Latte	Biscotti	Croissant	Scone
Northeast	CT	New Haven	\$158,995	\$926,052	\$589,355	\$551,489	\$283,874
	MA	Boston	\$174,344	\$917,737	\$570,391	\$497,234	\$332,486
	NY	New York	\$208,756	\$928,026	\$642,259	\$622,095	\$290,811

Generating Summary Lines and Hiding Null ACROSS Columns

If an entire ACROSS group is hidden, so are the totals generated for the associated BY field value. If any of the columns for the ACROSS value contain non-missing data, the ACROSS group will display with the non-missing columns.

Summary elements remain tied to their ACROSS group columns. If an ACROSS group is hidden, the associated summary value will be hidden, and subsequent values will realign with their ACROSS columns.

Summary lines generated at BY field breaks display at the end of the final page for that BY field value. All ACROSS groups that contain any non-null data within the entire BY value (even if they were hidden on some pages within the BY value) will display on the summary lines so that associated summary values can be displayed.

Grand totals can contain ACROSS columns that have been hidden on some pages within a BY field value. Therefore, they are always placed on a new page and presented for all ACROSS groups and columns that displayed on any page within the report, regardless of what was hidden on other pages.

Summary lines defined for BY fields outside of the innermost BY page break may also contain ACROSS columns that have been hidden for some of the internal BY fields. For this reason, these summary lines will always present all available ACROSS columns and will be presented on a new page.

All totals calculated in columns (ACROSSTOTAL, ROWTOTAL) will be hidden if all of the column totals are missing.

**Example: Generating Column Totals and Hiding Null ACROSS Columns**

In the following request against the GGSales data source, REGION is a BY field with a PAGE-BREAK, and PRODUCT is the ACROSS field. The DEFINE command creates a field named SHOWDOLLARS that has missing values for the Espresso column within the ACROSS group Coffee. Column totals are generated at the end of the report:

```

SET HIDE NULLACRS=ON
DEFINE FILE GGSales
SHOWDOLLARS/I8M MISSING ON = IF (PRODUCT EQ 'Espresso') THEN MISSING ELSE
DOLLARS;
END
TABLE FILE GGSales
SUM SHOWDOLLARS AS ''
BY REGION
BY ST
BY CITY
ACROSS PRODUCT
ON REGION PAGE-BREAK
HEADING
"Page <TABPAGENO /<TABLASTPAGE "
WHERE CATEGORY EQ 'Coffee';
ON TABLE SET PAGE-NUM OFF
ON TABLE SET BYPANEL ON
ON TABLE COLUMN-TOTAL AS 'TOTAL'
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
    UNITS=IN,
    PAGESIZE='Letter',
    SQUEEZE=ON,
    ORIENTATION=PORTRAIT,
$
TYPE=REPORT,
    HEADPANEL=ON,
    GRID=OFF,
    FONT='ARIAL',
    SIZE=9,
$
ENDSTYLE
END

```

Running the request hides the null columns within each REGION page break and generates a separate page for the column totals.



The following shows pages one through three. On page 1, the Espresso and Capuccino columns are hidden. On pages 2 and 3, the Espresso column is hidden:

Page 1 /5

Region	State	City	Product
			Latte
Midwest	IL	Chicago	\$978,340
	MO	St. Louis	\$966,981
	TX	Houston	\$938,245

Page 2 /5

Region	State	City	Product	
			Capuccino	Latte
Northeast	CT	New Haven	\$158,995	\$926,052
	MA	Boston	\$174,344	\$917,737
	NY	New York	\$208,756	\$928,026

Page 3 /5

Region	State	City	Product	
			Capuccino	Latte
Southeast	FL	Orlando	\$317,027	\$889,887
	GA	Atlanta	\$352,161	\$907,365
	TN	Memphis	\$274,812	\$820,584

The following shows pages four and five. On page 4, the Espresso column is hidden. Page 5 is the totals page. The Espresso column is hidden since it was hidden on every detail page. However, Capuccino is not hidden since it appeared on some pages:

Page 4 /5

Region	State	City	Product	
			Capuccino	Latte
West	CA	Los Angeles	\$306,468	\$809,647
		San Francisco	\$279,830	\$935,862
	WA	Seattle	\$309,197	\$924,896

Page 5 /5

Region	State	City	Product	
			Capuccino	Latte
TOTAL			\$2,381,590	\$10,943,622

Using Column Styling and Hiding Null ACROSS Columns

Column styling remains attached to the original column, regardless of whether the column remains in the same place on the report output because of hiding null columns. In particular:

- ☐ BORDERS and BACKCOLOR will readjust to fit the resulting panel or page layout after the columns are hidden.
- ☐ Styling specified for a designated column will remain attached to the designated column and be unaffected by the hidden columns. For example, if the third ACROSS column is defined with conditional styling, and the second ACROSS column is hidden, the formatting will remain on the column that was initially third, even though it becomes the second column on the output.

For information about styling columns, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

**Example: Using Column Styling and Hiding Null ACROSS Columns**

In the following request against the GGSales data source, REGION is a BY field with a PAGE-BREAK and PRODUCT is the ACROSS field. The DEFINE command creates a field named SHOWDOLLARS that has missing values for the Capuccino column in the Midwest region, the Thermos column in the Northeast region, the Scone column in the Southeast region, and the entire West region. Column totals, row totals, and a subtotal for each region are generated.

Some of the columns are assigned background colors:

- ❑ Column C5 has BACKCOLOR=WHEAT. C5 is the fifth column counting display fields from left to right, but not counting BY fields or ROW-TOTAL fields. Column C5 corresponds to the *Croissant* column in the *Coffee* group.
- ❑ Column P5 has BACKCOLOR=THISTLE. P5 is the fifth column counting display fields, BY fields, and ROW-TOTAL fields, but not NOPRINT fields. Column P5 corresponds to the *Espresso* column in the *Coffee* group.
- ❑ Column N7 has BACKCOLOR=MEDIUM GOLDENROD. N7 is the seventh column counting display fields, BY fields, ROW-TOTAL fields, and NOPRINT fields. Column N7 corresponds to the *Biscotti* column in the *Food* group.
- ❑ Column B3 has BACKCOLOR=GOLDENROD. B3 is the third BY field, counting all BY fields, even if not printed. Column B3 corresponds to the *CITY* sort field.
- ❑ Column SHOWDOLLARS(6) has BACKCOLOR=SILVER. SHOWDOLLARS(6) is the sixth occurrence of the SHOWDOLLARS field and corresponds to the *Scone* column in the *Food* group.

The request follows:

```

SET HIDE NULLACRS=OFF
DEFINE FILE GGSALES
SHOWDOLLARS/I8M MISSING ON =
IF ((PRODUCT EQ 'Capuccino' AND REGION EQ 'Midwest') OR
(PRODUCT EQ 'Coffee Grinder' AND REGION EQ 'Northeast') OR
(PRODUCT EQ 'Scone' AND REGION EQ 'Southeast') OR
(REGION EQ 'West')) THEN MISSING ELSE DOLLARS;
END
TABLE FILE GGSALES
SUM SHOWDOLLARS AS ''
BY REGION
BY ST
BY CITY
ACROSS CATEGORY
ACROSS PRODUCT
ON REGION SUBTOTAL AS '*TOTAL'
ON REGION PAGE-BREAK
HEADING
" Page <TABPAGENO "HEADING
" Capuccino Missing in Coffee Group "
WHEN REGION EQ 'Midwest';
HEADING
" Coffee Grinder Missing in Gifts Group "
WHEN REGION EQ 'Northeast';
HEADING
" Scone Missing in Food Group "
WHEN REGION EQ 'Southeast';
WHERE CATEGORY EQ 'Coffee' OR 'Food'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET BYPANEL ON
ON TABLE ROW-TOTAL AS 'TOTAL'
ON TABLE COLUMN-TOTAL AS 'TOTAL'
ON TABLE SET HTMLCSS ON
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET STYLE *
UNITS=IN,PAGESIZE='Letter',SQUEEZE=ON,ORIENTATION=PORTRAIT,$
TYPE=REPORT,HEADPANEL=ON,GRID=OFF,FONT='ARIAL',SIZE=6,$
TYPE=HEADING, style=bold, size=8,$
TYPE=DATA, COLUMN = C5, BACKCOLOR=WHEAT,$
TYPE=DATA, COLUMN = P5, BACKCOLOR=THISTLE,$
TYPE=DATA, COLUMN = N7, BACKCOLOR=MEDIUM GOLDENROD,$
TYPE=DATA, COLUMN = B3, BACKCOLOR=GOLDENROD,$
TYPE=DATA, COLUMN = SHOWDOLLARS(6), BACKCOLOR=silver,$
ENDSTYLE
END

```

Running the report with SET HIDE NULLACRS=OFF shows all columns. A page is generated for the West region and subtotals are calculated, even though all of the values are missing:

### Page 1

#### Capuccino Missing in Coffee Group

			Category						
			Coffee	Food					
			Product						
Region	State	City	Capuccino	Espresso	Latte	Biscotti	Croissant	Scone	TOTAL
Midwest	IL	Chicago	-	\$420,439	\$978,340	\$378,412	\$549,366	\$595,069	\$2,921,626
	MO	St. Louis	-	\$419,143	\$966,981	\$368,077	\$613,871	\$481,953	\$2,850,025
	TX	Houston	-	\$455,365	\$938,245	\$345,238	\$587,887	\$418,398	\$2,745,133
TOTAL Midwest			\$0	\$1,294,947	\$2,883,566	\$1,091,727	\$1,751,124	\$1,495,420	\$8,516,784

#### Coffee Grinder Missing in Gifts Group

			Category						
			Coffee	Food					
			Product						
Region	State	City	Capuccino	Espresso	Latte	Biscotti	Croissant	Scone	TOTAL
Northeast	CT	New Haven	\$158,995	\$279,373	\$926,052	\$589,355	\$551,489	\$283,874	\$2,789,138
	MA	Boston	\$174,344	\$248,356	\$917,737	\$570,391	\$497,234	\$332,486	\$2,740,548
	NY	New York	\$208,756	\$322,378	\$928,026	\$642,259	\$622,095	\$290,811	\$3,014,325
TOTAL Northeast			\$542,095	\$850,107	\$2,771,815	\$1,802,005	\$1,670,818	\$907,171	\$8,544,011

### Scone Missing in Food Group

			Category						
			Coffee	Food					
			Product						
Region	State	City	Capuccino	Espresso	Latte	Biscotti	Croissant	Scone	TOTAL
Southeast	FL	Orlando	\$317,027	\$256,539	\$889,887	\$511,597	\$602,076	-	\$2,577,126
	GA	Atlanta	\$352,161	\$317,389	\$907,365	\$555,231	\$661,806	-	\$2,793,952
	TN	Memphis	\$274,612	\$279,644	\$820,584	\$438,889	\$638,477	-	\$2,452,406
	*TOTAL Southeast		\$944,000	\$853,572	\$2,617,836	\$1,505,717	\$1,902,359	\$0	\$7,823,484

Region	State	City	Category	Food					TOTAL
			Coffee						
			Product						
			Capuccino	Espresso	Latte	Biscotti	Croissant	Scone	
West	CA	Los Angeles	-	-	-	-	-	-	\$0
		San Francisco	-	-	-	-	-	-	\$0
	WA	Seattle	-	-	-	-	-	-	\$0
*TOTAL West			\$0	\$0	\$0	\$0	\$0	\$0	\$0

Region	State	City	Category						TOTAL
			Coffee	Food					
			Product						
			Capuccino	Espresso	Latte	Biscotti	Croissant	Scone	
TOTAL			\$1,486,095	\$2,998,626	\$8,273,217	\$4,399,449	\$5,324,301	\$2,402,591	\$24,884,279

Running the report with SET HIDE NULLACRS=ON, shows:

- ❑ On page 1, the Capuccino column is hidden and, therefore, the Espresso column is no longer P5 on the report, but it still has BACKCOLOR=THISTLE. Similarly, the Biscotti column has MEDIUM, GOLDENROD, the Croissant column has WHEAT, and the Scone column has SILVER.
- ❑ The subtotals for each region are calculated only for columns that display for that region.
- ❑ No page is generated for the West region since all of its values are missing.
- ❑ Every column is represented on the page with the grand totals.

The output is:

#### Page 1

##### Capuccino Missing in Coffee Group

			Category					
			Coffee	Food				
			Product					
Region	State	City	Espresso	Latte	Biscotti	Croissant	Scone	TOTAL
Midwest	IL	Chicago	\$420,439	\$278,340	\$378,412	\$549,366	\$395,069	\$2,921,626
	MO	St. Louis	\$419,143	\$366,981	\$368,077	\$613,871	\$481,963	\$2,850,025
	TX	Houston	\$455,365	\$338,245	\$345,238	\$587,887	\$418,398	\$2,745,133
	TOTAL Midwest		\$1,294,947	\$2,883,566	\$1,091,727	\$1,751,124	\$1,495,430	\$8,516,784

##### Coffee Grinder Missing in Gifts Group

Region	State	City	Category						TOTAL
			Coffee	Food					
			Product						
			Capuccino	Espresso	Latte	Biscotti	Croissant	Scone	
Northeast	CT	New Haven	\$168,996	\$279,373	\$206,062	\$589,366	\$551,489	\$283,874	\$2,789,138
	MA	Boston	\$174,344	\$248,366	\$917,737	\$570,391	\$497,234	\$332,486	\$2,740,548
	NY	New York	\$208,796	\$322,378	\$208,006	\$642,259	\$622,096	\$290,811	\$3,014,325
TOTAL Northeast			\$552,096	\$850,107	\$2,771,815	\$1,802,005	\$1,670,818	\$907,171	\$8,544,011

Scone Missing in Food Group									
Region	State	City	Category						
			Coffee					Food	
			Product						
			Cappuccino	Espresso	Latte	Biscotti	Croissant	TOTAL	
Southeast	FL	Orlando	\$317,027	\$256,539	\$889,887	\$511,597	\$602,076	\$2,577,126	
	GA	Atlanta	\$352,161	\$317,389	\$407,365	\$555,231	\$661,806	\$2,793,952	
	TN	Memphis	\$274,812	\$279,644	\$820,584	\$438,889	\$538,477	\$2,452,406	
TOTAL Southeast			\$944,000	\$853,572	\$2,617,836	\$1,505,717	\$1,902,359	\$7,823,484	
Region	State	City	Category						
			Coffee					Food	
			Product						
			Cappuccino	Espresso	Latte	Biscotti	Croissant	Score	TOTAL
TOTAL			\$1,486,095	\$2,998,626	\$8,273,217	\$4,399,449	\$5,324,301	\$2,402,591	\$24,884,279

Hiding Null ACROSS Columns in an FML Request

An FML request always has a FOR field that defines the order of specific rows. The FOR field cannot be used to trigger hiding of null ACROSS columns. However, the request can also have a BY field with a PAGE-BREAK option and this can be used to hide null ACROSS columns.



**Example: Hiding Null ACROSS Columns in an FML Request**

The following FML request against the GGSALES data source has a BY field named REGION with the PAGE-BREAK option and an ACROSS field named QTR. The FOR field is PRODUCT. The DEFINE command creates the QTR field and contains missing values for Q4 in the Midwest region, Q2 in the Northeast region, and for all quarters in the Southeast region.

```

SET HIDENULLACRS=ON
DEFINE FILE GGSALES
QTR/Q=DATE;
SHOWDOLLARS/D12CM MISSING ON =
    IF REGION EQ 'Midwest' AND QTR EQ 'Q4' THEN MISSING
    ELSE IF REGION EQ 'Northeast' AND QTR EQ 'Q2' THEN MISSING
    ELSE IF REGION EQ 'Southeast' THEN MISSING
    ELSE DOLLARS;
END
TABLE FILE GGSALES
SUM SHOWDOLLARS
BY REGION
ACROSS QTR
FOR PRODUCT
'Biscotti' AS 'Biscotti' LABEL R1 OVER
'Capuccino' AS 'Capuccino' LABEL R2 OVER
'Latte' AS 'Latte' LABEL R3 OVER
'Mug' AS 'Mug' LABEL R4 OVER
'Coffee Pot' AS 'Coffee Pot' LABEL R5 OVER
RECAP R6/D12.2=R1+R2+R3+R4+R5;
AS ''
ON REGION PAGE-BREAK
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLCSS ON

ON TABLE SET STYLE *
    UNITS=IN,
    SQUEEZE=ON,
    ORIENTATION=PORTRAIT,$
TYPE=REPORT,
    GRID=OFF,
    FONT='ARIAL',
    SIZE=9,$
TYPE=TITLE,
    STYLE=BOLD,$
TYPE=ACROSSTITLE,
    STYLE=BOLD,$
ENDSTYLE
END

```

Running the request with SET HIDE NULLACRS=OFF generates all columns and a page for all regions, including the Southeast regions where all values are missing:

		QTR			
		Q1	Q2	Q3	Q4
Region					
Midwest	Biscotti	\$205,440	\$252,072	\$275,976	.
	Capuccino	.	.	.	.
	Latte	\$630,883	\$704,089	\$776,434	.
	Mug	\$261,825	\$264,003	\$261,527	.
	Coffee Pot	\$128,026	\$169,533	\$164,565	.
		1,226,174.00	1,389,697.00	1,478,502.00	.

		QTR			
		Q1	Q2	Q3	Q4
Region					
Northeast	Biscotti	\$393,362	.	\$512,375	\$453,794
	Capuccino	\$62,070	.	\$175,864	\$185,712
	Latte	\$791,332	.	\$652,872	\$628,026
	Mug	\$265,133	.	\$307,917	\$308,783
	Coffee Pot	\$130,517	.	\$145,670	\$163,594
		1,642,414.00	.	1,794,698.00	1,739,909.00

		QTR			
		Q1	Q2	Q3	Q4
Region					
Southeast	Biscotti	.	.	.	.
	Capuccino	.	.	.	.
	Latte	.	.	.	.
	Mug	.	.	.	.
	Coffee Pot	.	.	.	.
		.	.	.	.

Running the request with SET HIDE NULLACRS=ON hides column Q4 for the Midwest region, Q2 for the Northeast region, and the entire page for the Southeast region:

		QTR		
		Q1	Q2	Q3
Region				
Midwest	Biscotti	\$205,440	\$252,072	\$275,976
	Capuccino	.	.	.
	Latte	\$630,883	\$704,089	\$776,434
	Mug	\$261,825	\$264,003	\$261,527
	Coffee Pot	\$128,026	\$169,533	\$164,565
		1,226,174.00	1,389,697.00	1,478,502.00

		QTR		
		Q1	Q3	Q4
Region				
Northeast	Biscotti	\$393,362	\$512,375	\$453,794
	Capuccino	\$62,070	\$175,864	\$185,712
	Latte	\$791,332	\$652,872	\$628,026
	Mug	\$265,133	\$307,917	\$308,783
	Coffee Pot	\$130,517	\$145,670	\$163,594
		1,642,414.00	1,794,698.00	1,739,909.00

		QTR			
		Q1	Q2	Q3	Q4
Region					
West	Biscotti	\$229,781	\$224,070	\$230,649	\$179,368
	Capuccino	\$205,057	\$194,149	\$237,778	\$258,511
	Latte	\$727,734	\$664,618	\$619,832	\$658,221
	Mug	\$296,839	\$281,809	\$313,692	\$296,324
	Coffee Pot	\$132,354	\$159,799	\$153,072	\$168,399
		1,591,765.00	1,524,445.00	1,555,023.00	1,560,823.00

## Controlling Display of Sort Field Values

By default, a sort field value displays only on the first row or column of the set of detail rows or columns generated for that sort field value. You can control this behavior using the BYDISPLAY parameter. BYDISPLAY is supported for all output formats and can control display of ACROSS values as well as BY values.

This feature enables you to avoid specifying the sort field twice, once as a display field and once for sorting (with the NOPRINT option). For example:

```
PRINT FIRST_NAME LAST_NAME  
BY FIRST_NAME NOPRINT
```

### *Syntax:* How to Control Display of Sort Field Values

```
SET BYDISPLAY = {OFF|ON|BY|ACROSS|ALL}
```

```
ON TABLE SET BYDISPLAY {OFF|ON|BY|ACROSS|ALL}
```

where:

#### OFF

Displays a sort field value only on the first line or column of the report output for the sort group and on the first line or column of a page. OFF is the default value.

#### ON or BY

Displays the relevant BY field value on every line of report output produced. BY is a synonym for ON.

#### ACROSS

Displays the relevant ACROSS field value on every column of report output produced.

#### ALL

Displays the relevant BY field value on every line of report output and the relevant ACROSS field value on every column of report output.

**Example: Controlling Display of Sort Field Values on Report Output**

The following request generates a report on which sort field values only display when they change (BYDISPLAY OFF).

```
-SET &BYDISP = OFF;
SET BYDISPLAY = &BYDISP
TABLE FILE WF_RETAIL_LITE
HEADING CENTER
" BYDISPLAY = &BYDISP"
" "
SUM QUANTITY_SOLD DAYSDELAYED
BY PRODUCT_CATEGORY
BY PRODUCT_SUBCATEG
ACROSS BUSINESS_REGION
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

The output is shown in the following image.

BYDISPLAY = OFF									
		Customer,Business,Region							
Product Category	Product Subcategory	EMEA		North America		Oceania		South America	
		Quantity Sold	Days Delayed	Quantity Sold	Days Delayed	Quantity Sold	Days Delayed	Quantity Sold	Days Delayed
Accessories	Charger	187	55	220	84	.	.	26	25
	Headphones	403	137	470	180	2	0	56	18
	Universal Remote Controls	287	102	325	132	7	7	51	23
Camcorder	Handheld	466	200	527	209	4	1	94	20
	Professional	23	4	35	17	.	.	.	.
	Standard	344	157	404	130	1	0	58	37
Computers	Smartphone	384	162	411	151	5	2	65	27
Media Player	Blu Ray	1,206	495	1,395	518	15	11	203	78
	DVD Players	32	14	48	17	.	.	13	0
	Streaming	88	30	124	53	3	0	13	2
Stereo Systems	Home Theater Systems	741	239	847	349	4	5	138	59
	Receivers	266	90	282	109	.	.	68	21
	Speaker Kits	451	172	476	199	.	.	95	33
	iPod Docking Station	529	186	673	272	7	0	102	62
Televisions	Flat Panel TV	159	67	166	62	1	0	36	15
Video Production	Video Editing	381	139	445	150	3	1	58	27

Changing BYDISPLAY to ON or BY displays BY field values on every row, as shown in the following image.

BYDISPLAY = BY									
Product Category	Product Subcategory	Customer,Business,Region							
		EMEA		North America		Oceania		South America	
		Quantity Sold	Days Delayed	Quantity Sold	Days Delayed	Quantity Sold	Days Delayed	Quantity Sold	Days Delayed
Accessories	Charger	187	55	220	84	.	.	26	25
Accessories	Headphones	403	137	470	180	2	0	56	18
Accessories	Universal Remote Controls	287	102	325	132	7	7	51	23
Camcorder	Handheld	466	200	527	209	4	1	94	20
Camcorder	Professional	23	4	35	17	.	.	.	.
Camcorder	Standard	344	157	404	130	1	0	58	37
Computers	Smartphone	384	162	411	151	5	2	65	27
Media Player	Blu Ray	1,206	495	1,395	518	15	11	203	78
Media Player	DVD Players	32	14	48	17	.	.	13	0
Media Player	Streaming	88	30	124	53	3	0	13	2
Stereo Systems	Home Theater Systems	741	239	847	349	4	5	138	59
Stereo Systems	Receivers	266	90	282	109	.	.	68	21
Stereo Systems	Speaker Kits	451	172	476	199	.	.	95	33
Stereo Systems	iPod Docking Station	529	186	673	272	7	0	102	62
Televisions	Flat Panel TV	159	67	166	62	1	0	36	15
Video Production	Video Editing	381	139	445	150	3	1	58	27

Changing BYDISPLAY to ACROSS displays ACROSS field values over every column, as shown in the following image.

BYDISPLAY = ACROSS									
Product Category	Product Subcategory	Customer,Business,Region							
		EMEA	EMEA	North America	North America	Oceania	Oceania	South America	South America
		Quantity Sold	Days Delayed	Quantity Sold	Days Delayed	Quantity Sold	Days Delayed	Quantity Sold	Days Delayed
Accessories	Charger	187	55	220	84	.	.	26	25
	Headphones	403	137	470	180	2	0	56	18
	Universal Remote Controls	287	102	325	132	7	7	51	23
Camcorder	Handheld	466	200	527	209	4	1	94	20
	Professional	23	4	35	17	.	.	.	.
	Standard	344	157	404	130	1	0	58	37
Computers	Smartphone	384	162	411	151	5	2	65	27
Media Player	Blu Ray	1,206	495	1,395	518	15	11	203	78
	DVD Players	32	14	48	17	.	.	13	0
	Streaming	88	30	124	53	3	0	13	2
Stereo Systems	Home Theater Systems	741	239	847	349	4	5	138	59
	Receivers	266	90	282	109	.	.	68	21
	Speaker Kits	451	172	476	199	.	.	95	33
	iPod Docking Station	529	186	673	272	7	0	102	62
Televisions	Flat Panel TV	159	67	166	62	1	0	36	15
Video Production	Video Editing	381	139	445	150	3	1	58	27

Changing BYDISPLAY to ALL displays BY field values on every row and ACROSS field values over every column, as shown in the following image.

		BYDISPLAY = ALL							
		Customer,Business,Region							
Product Category	Product Subcategory	EMEA	EMEA	North America	North America	Oceania	Oceania	South America	South America
		Quantity Sold	Days Delayed	Quantity Sold	Days Delayed	Quantity Sold	Days Delayed	Quantity Sold	Days Delayed
Accessories	Charger	187	55	220	84	.	.	26	25
Accessories	Headphones	403	137	470	180	2	0	56	18
Accessories	Universal Remote Controls	287	102	325	132	7	7	51	23
Camcorder	Handheld	466	200	527	209	4	1	94	20
Camcorder	Professional	23	4	35	17	.	.	.	.
Camcorder	Standard	344	157	404	130	1	0	58	37
Computers	Smartphone	384	162	411	151	5	2	65	27
Media Player	Blu Ray	1,206	495	1,395	518	15	11	203	78
Media Player	DVD Players	32	14	48	17	.	.	13	0
Media Player	Streaming	88	30	124	53	3	0	13	2
Stereo Systems	Home Theater Systems	741	239	847	349	4	5	138	59
Stereo Systems	Receivers	266	90	282	109	.	.	68	21
Stereo Systems	Speaker Kits	451	172	476	199	.	.	95	33
Stereo Systems	iPod Docking Station	529	186	673	272	7	0	102	62
Televisions	Flat Panel TV	159	67	166	62	1	0	36	15
Video Production	Video Editing	381	139	445	150	3	1	58	27

## Reformatting Sort Fields

When displaying a vertical (BY) sort column or horizontal (ACROSS) sort row on report output, you can reformat the sort field values by specifying the new format in the sort phrase. The reformatting affects only the sort field value as displayed on the sort row or column. That is, if the field used as a sort field is referenced in a heading, subheading, footing, subfooting, or summary line, it displays with its original format.

### **Syntax:** How to Reformat a Sort Field

```
{BY [TOTAL]|ACROSS [TOTAL]} sortfield/fmt ...
```

where:

*sortfield*

Is the sort field.

*fmt*

Is the new display format.

### **Reference:** Usage Notes for Reformatting Sort Fields

- ☐ Reformatting is not supported in an ON phrase.
- ☐ Reformatting is only applied to the row or column generated by the sort phrase, not to subheading, subfooting, or summary rows that reference the sort field.

- ❑ Field-based reformatting (the format is stored in a field) is not supported for sort fields. For information on field-based reformatting, see [Field-Based Reformatting](#) on page 1650.
- ❑ BY field reformatting is propagated to HOLD files. ACROSS field reformatting *is not* propagated to HOLD files.
- ❑ The following features issue a warning, ignore the reformatting, and proceed with the request:
  - ❑ IN-GROUPS/RANGES-OF
  - ❑ ROWS
  - ❑ COLUMNS
  - ❑ FOR field/reformat
- ❑ All decisions are made based on an original field, therefore any option on a reformatted sort field should be placed on an original sort field break point. As a consequence, whenever the sort field value should appear (for example, in SUBTOTALS) an original field value displays.

### **Example:** Reformatting Sort Fields

The following request against the GGSALES data source includes the following reformatted sort fields:

- ❑ BY CATEGORY, with CATEGORY reformatted as A3.
- ❑ BY PRODUCT, with PRODUCT reformatted as A4.
- ❑ ACROSS REGION, with region reformatted as A6.

```
TABLE FILE GGSALES
SUM UNIT
BY CATEGORY/A3
BY PRODUCT/A4
ACROSS REGION/A6
ON CATEGORY SUBTOTAL
ON CATEGORY SUBHEAD
"CATEGORY IS <CATEGORY "
" "
ON TABLE SET PAGE NOPAGE
END
```



On the output, the reformatting displays on the BY and ACROSS rows but is not propagated to the subheading and subtotal rows:

		REGION			
		Midwes	Northe	Southe	West
CATEGORY	PRODUCT				
-----					
CATEGORY IS Coffee					
Cof	Capu	.	44785	73264	72831
	Espr	101154	68127	68030	71675
	Latt	231623	226243	211063	213920
*TOTAL CATEGORY Coffee		332777	339155	352357	358426
CATEGORY IS Food					
Foo	Bisc	90413	149793	119594	70569
	Croi	139881	137394	156456	197022
	Scon	116127	70732	73779	72776
*TOTAL CATEGORY Food		346421	357919	349829	340367
CATEGORY IS Gifts					
Gif	Coff	54002	40977	50556	48081
	Coff	47156	46185	49922	47432
	Mug	86718	91497	88474	93881
	Ther	46587	48870	48976	45648
*TOTAL CATEGORY Gifts		234463	227529	237928	235042
TOTAL		913661	924603	940114	933835

## Manipulating Display Field Values in a Sort Group

You can use the WITHIN phrase to manipulate a display field values as they are aggregated within a sort group. This technique can be used with a prefix operator to perform calculations on a specific aggregate field rather than a report column. In contrast, the SUM and COUNT commands aggregate an entire column.

The WITHIN phrase requires a BY phrase and/or an ACROSS phrase. A maximum of two WITHIN phrases can be used per display field. If one WITHIN phrase is used, it must act on a BY phrase. If two WITHIN phrases are used, the first must act on a BY phrase and the second on an ACROSS phrase.

You can also use WITHIN TABLE, which allows you to return the original value within a request command. The WITHIN TABLE command can also be used when an ACROSS phrase is needed without a BY phrase. Otherwise, a single WITHIN phrase requires a BY phrase.

### **Syntax:** How to Use WITHIN to Manipulate Display Fields

```
{SUM|COUNT} display_field WITHIN by_sort_field [WITHIN across_sort_field]  
BY by_sort_field [ACROSS across_sort_field]
```

where:

*display\_field*

Is the object of a SUM or COUNT display command.

*by\_sort\_field*

Is the object of a BY phrase.

*across\_sort\_field*

Is the object of an ACROSS phrase.

### **Example:** Summing Values Within Sort Groups

The following report shows the units sold and the percent of units sold for each product within store and within the table:

```
TABLE FILE SALES  
SUM UNIT_SOLD AS 'UNITS'  
AND PCT.UNIT_SOLD AS 'PCT,SOLD,WITHIN,TABLE'  
AND PCT.UNIT_SOLD WITHIN STORE_CODE AS 'PCT,SOLD,WITHIN,STORE'  
BY STORE_CODE SKIP-LINE BY PROD_CODE  
END
```

The output is:

STORE_CODE	PROD_CODE	UNITS	PCT SOLD WITHIN	PCT SOLD WITHIN
			TABLE	STORE
14B	B10	60	9	15
	B12	40	6	10
	B17	29	4	7
	C13	25	3	6
	C7	45	6	11
	D12	27	4	7
	E2	80	12	21
	E3	70	10	18
14Z	B10	30	4	18
	B17	20	3	12
	B20	15	2	9
	C17	12	1	7
	D12	20	3	12
	E1	30	4	18
	E3	35	5	21
	B20	25	3	38
77F	C7	40	6	61
	B10	13	2	30
K1	B12	29	4	69

## Creating a Matrix Report

You can create a matrix report by sorting both rows and columns. When you include both BY and ACROSS phrases in a report request, information is sorted vertically and horizontally, turning the report into a matrix of information that you read like a grid. A matrix report can have multiple BY and ACROSS sort fields.

### *Example:* Creating a Simple Matrix

The following request displays total salary outlay across departments and by job codes, creating a matrix report.

```
TABLE FILE EMPLOYEE
SUM CURR_SAL
ACROSS DEPARTMENT
BY CURR_JOBCODE
END
```

The output is:

CURR_JOBCODE	DEPARTMENT	
	MIS	PRODUCTION
A01	.	\$9,500.00
A07	\$9,000.00	\$11,000.00
A15	.	\$26,862.00
A17	\$27,062.00	\$29,700.00
B02	\$18,480.00	\$16,100.00
B03	\$18,480.00	.
B04	\$21,780.00	\$21,120.00
B14	\$13,200.00	.

**Example: Creating a Matrix With Several Sort Fields**

The following request uses several BY and ACROSS sort fields to create a matrix report.

```
TABLE FILE EMPLOYEE
SUM CURR_SAL
ACROSS DEPARTMENT ACROSS LAST_NAME
BY CURR_JOBCODE BY ED_HRS
WHERE DEPARTMENT EQ 'MIS'
WHERE CURR_SAL GT 21500
END
```

The output is:

CURR_JOBCODE	ED_HRS	DEPARTMENT	
		MIS	CROSS
		LAST_NAME	
		BLACKWOOD	
A17	45.00	.	\$27,062.00
B04	75.00	\$21,780.00	.

**Controlling Collation Sequence**

Collation is defined as a set of rules that apply to the ordering and matching of all language elements that involve comparison of two values. A wide variety of elements are affected by this feature. Among these features are sorting, aggregation, WHERE conditions, and StyleSheets. By default, items are sorted based on their binary values. The COLLATION settings SRV\_CI and SRV\_CS, case-insensitive and case-sensitive collation, implement collation based on the LANGUAGE setting. Case-insensitive collation means that all WHERE clauses and sorts ignore the case of the elements being compared. COLLATION is a session level setting (it is not supported in an ON TABLE phrase and should be set in the edasprof server profile).

The collation setting applies only to alphanumeric values.

**Syntax: How to Establish Binary or Case-Insensitive Collation Sequence**

Add the following command to the server edasprof.prf profile:

```
SET COLLATION = {BINARY|SRV_CI|SRV_CS|CODEPAGE}
```

where:

**BINARY**

Bases the collation sequence on binary values.

**SRV\_CI**

Bases collation sequence on the LANGUAGE setting, and is case-insensitive.

**SRV\_CS**

Bases collation sequence on the LANGUAGE setting, and is case-sensitive.

**CODEPAGE**

Bases collation sequence on the code page in effect, and is case-sensitive. CODEPAGE is the default value.

In most cases, CODEPAGE is the same as BINARY. The only differences are for Danish, Finnish, German, Norwegian, and Swedish in an EBCDIC environment.

**Reference: Usage Notes for SET COLLATION**

- ❑ SUFFIX=FIX or SUFFIX=FOCUS/XFOCUS HOLD files created in one mode may not be usable as targets for a JOIN in another mode if the join field is on alphanumeric data with mixed-cases.
- ❑ FIXRETRIEVE is supported only for binary data, so setting COLLATION to anything other than BINARY will turn FIXRETRIEVE OFF, which may affect join performance.

**Rules for Sorting and Aggregation**

- ❑ Records with the same characters in the same order, but with variations in case, are considered to be identical. If multiple input records have these variations, the value used is from the first such record.
- ❑ In a detail level report, the sort value is the same for each output record. That value will be the one for the input record that had the lowest value (collated first).

- ❑ When the MIN (or the MAX) value for two or more alphanumeric display fields having a given instance of the sort field values is the same by case-insensitive collation, but the two values vary by case in some positions, the one retained is the last one in the input file (highest input record number) when SUMPREF=LST and the first (lowest record number) when SUMPREF=FST.

### **Example:** Using Binary and Case-Insensitive Collation Sequence for Sorting

The following request creates a FOCUS data source named COLLATE that has some records with product names that differ only by the case of one letter:

```
CREATE FILE COLLATE
-RUN
MODIFY FILE COLLATE
FIXFORM PROD_NUM/C4 PRODNAME/C30 QTY_IN_STOCK/C7 PRICE/C12 COST/C12
CHECK OFF
DATA
10042 Hd VCR LCD Menu          43068      179.00      129.00
10052 HD VCR LCD Menu          43068      179.00      129.00
1006Combo Player - 4 HD VCR + DVD 13527      399.00      289.00
1007Combo Player - 4 Hd VCR + DVD 13527      399.00      289.00
1008DVD Upgrade Unit for Cent. VCR   199      199.00      139.00
1010750SL Digital Camcorder 300 X  10758      999.00      750.00
1012650DL Digital Camcorder 150 X   2972      899.00      710.00
1014340SX Digital Camera 65K P      990      249.00      199.00
1015340SX digital Camera 65K P      990      249.00      199.00
1016330DX Digital Camera 1024K P  12707      279.00      199.00
1018250 8MM Camcorder 40 X          60073      399.00      320.00
1019250 8mm Camcorder 40 X          60073      399.00      320.00
1020150 8MM Camcorder 20 X          5961      319.00      240.00
1022120 VHS-C Camcorder 40 X        2300      399.00      259.00
1024110 VHS-C Camcorder 20 X        4000      349.00      249.00
1026AR2 35mm Camera 8 X             12444      129.00       95.00
1029AR2 35MM Camera 8 X             11499      109.00       79.00
1028AR3 35MM Camera 10 X            11499      109.00       79.00
1030QX Portable CD Player           22000      169.00       99.00
1032R5 Micro Digital Tape Recorder   1990        89.00       69.00
1034ZT Digital PDA - Commercial     21000      499.00      349.00
1036ZC Digital PDA - Standard       33000      299.00      249.00
END
```

The following request prints the values of PRODNAME in the order in which they are encountered in the input stream:

```
TABLE FILE COLLATE
PRINT PROD_NUM PRODNAME
END
```

On the output, the rows with product numbers 1004 and 1005 differ only in the case of the letter *d* in *HD*. The record with the lowercase *d* is before the record with the uppercase *D*. The rows with record numbers 1006 and 1007 also differ only in the case of the letter *d* in *HD*. In this case, the record with the uppercase *D* is before the record with the lowercase *d*:

Product Number:	Product Name:
-----	-----
1004	2 Hd VCR LCD Menu
1005	2 HD VCR LCD Menu
1006	Combo Player - 4 HD VCR + DVD
1007	Combo Player - 4 Hd VCR + DVD
1008	DVD Upgrade Unit for Cent. VCR
1010	750SL Digital Camcorder 300 X
1012	650DL Digital Camcorder 150 X
1014	340SX Digital Camera 65K P
1015	340SX digital Camera 65K P
1016	330DX Digital Camera 1024K P
1018	250 8MM Camcorder 40 X
1019	250 8mm Camcorder 40 X
1020	150 8MM Camcorder 20 X
1022	120 VHS-C Camcorder 40 X
1024	110 VHS-C Camcorder 20 X
1026	AR2 35mm Camera 8 X
1029	AR2 35MM Camera 8 X
1028	AR3 35MM Camera 10 X
1030	QX Portable CD Player
1032	R5 Micro Digital Tape Recorder
1034	ZT Digital PDA - Commercial
1036	ZC Digital PDA - Standard

The next request sorts the output in BINARY order. The setting COLLATION = BINARY is in effect:

```
TABLE FILE COLLATE
PRINT PROD_NUM
BY PRODNAME
END
```

In an EBCDIC environment, the records with the lowercase letters sort in front of the records with the uppercase letters, so the row with product number 1007 sorts in front of the row with product number 1006:

Product Name: -----	Product Number: -----
AR2 35mm Camera 8 X	1026
AR2 35MM Camera 8 X	1029
AR3 35MM Camera 10 X	1028
Combo Player - 4 Hd VCR + DVD	1007
Combo Player - 4 HD VCR + DVD	1006
DVD Upgrade Unit for Cent. VCR	1008
QX Portable CD Player	1030
R5 Micro Digital Tape Recorder	1032
ZC Digital PDA - Standard	1036
ZT Digital PDA - Commercial	1034
110 VHS-C Camcorder 20 X	1024
120 VHS-C Camcorder 40 X	1022
150 8MM Camcorder 20 X	1020
2 Hd VCR LCD Menu	1004
2 HD VCR LCD Menu	1005
250 8mm Camcorder 40 X	1019
250 8MM Camcorder 40 X	1018
330DX Digital Camera 1024K P	1016
340SX digital Camera 65K P	1015
340SX Digital Camera 65K P	1014
650DL Digital Camcorder 150 X	1012
750SL Digital Camcorder 300 X	1010



In an ASCII environment, the records with the uppercase letters sort in front of the records with the lowercase letters, so the row with product number 1005 sorts in front of the row with product number 1004:

Product Name: -----	Product Number: -----
110 VHS-C Camcorder 20 X	1024
120 VHS-C Camcorder 40 X	1022
150 8MM Camcorder 20 X	1020
2 HD VCR LCD Menu	1005
2 Hd VCR LCD Menu	1004
250 8MM Camcorder 40 X	1018
250 8mm Camcorder 40 X	1019
330DX Digital Camera 1024K P	1016
340SX Digital Camera 65K P	1014
340SX digital Camera 65K P	1015
650DL Digital Camcorder 150 X	1012
750SL Digital Camcorder 300 X	1010
AR2 35MM Camera 8 X	1029
AR2 35mm Camera 8 X	1026
AR3 35MM Camera 10 X	1028
Combo Player - 4 HD VCR + DVD	1006
Combo Player - 4 Hd VCR + DVD	1007
DVD Upgrade Unit for Cent. VCR	1008
QX Portable CD Player	1030
R5 Micro Digital Tape Recorder	1032
ZC Digital PDA - Standard	1036
ZT Digital PDA - Commercial	1034

With COLLATION set to SRV\_CI and a sort on the PRODNAME field, the uppercase and lowercase letters have the same value, so the row displays only once for multiple record numbers. For example, the rows with product numbers 1004 and 1005 display with the same PRODNAME value and the sort field value for the display is the first one in the input stream.

The following shows the output in an EBCDIC environment:

Product Name: -----	Product Number: -----
AR2 35mm Camera 8 X	1026
	1029
AR3 35MM Camera 10 X	1028
Combo Player - 4 HD VCR + DVD	1006
	1007
DVD Upgrade Unit for Cent. VCR	1008
QX Portable CD Player	1030
R5 Micro Digital Tape Recorder	1032
ZC Digital PDA - Standard	1036
ZT Digital PDA - Commercial	1034
110 VHS-C Camcorder 20 X	1024
120 VHS-C Camcorder 40 X	1022
150 8MM Camcorder 20 X	1020
2 Hd VCR LCD Menu	1004
	1005
250 8MM Camcorder 40 X	1018
250 8MM Camcorder 40 X	1019
330DX Digital Camera 1024K P	1016
340SX Digital Camera 65K P	1014
	1015
650DL Digital Camcorder 150 X	1012
750SL Digital Camcorder 300 X	1010

The following shows the output in an ASCII environment:

Product Name: -----	Product Number: -----
110 VHS-C Camcorder 20 X	1024
120 VHS-C Camcorder 40 X	1022
150 8MM Camcorder 20 X	1020
2 Hd VCR LCD Menu	1004
	1005
250 8MM Camcorder 40 X	1018
	1019
330DX Digital Camera 1024K P	1016
340SX Digital Camera 65K P	1014
	1015
650DL Digital Camcorder 150 X	1012
750SL Digital Camcorder 300 X	1010
AR2 35mm Camera 8 X	1026
	1029
AR3 35MM Camera 10 X	1028
Combo Player - 4 HD VCR + DVD	1006
	1007
DVD Upgrade Unit for Cent. VCR	1008
QX Portable CD Player	1030
R5 Micro Digital Tape Recorder	1032
ZC Digital PDA - Standard	1036
ZT Digital PDA - Commercial	1034

**Example: Using Binary and Case-Insensitive Collation Sequence for Selection**

The following request against the COLLATE data source selects records in which the PRODNAME contains the characters 'HD':

```
TABLE FILE COLLATE
PRINT PROD_NUM PRODNAME
WHERE PRODNAME CONTAINS 'HD'
END
```

With COLLATION set to BINARY, only the records with an exact match (uppercase HD) are selected. The output is:

Product Number: -----	Product Name: -----
1005	2 HD VCR LCD Menu
1006	Combo Player - 4 HD VCR + DVD

Running the same request but changing the COLLATION parameter to SRV\_CI selects all records with any combination of uppercase and lowercase values for H and D. The rows are displayed in the order in which they appeared in the data source:

Product Number:	Product Name:
-----	-----
1004	2 Hd VCR LCD Menu
1005	2 HD VCR LCD Menu
1006	Combo Player - 4 HD VCR + DVD
1007	Combo Player - 4 Hd VCR + DVD

Specifying the Sort Order

Sort field values are automatically displayed in ascending order, beginning with the lowest value and continuing to the highest. The default sorting sequence varies for operating systems. On z/OS it is a-z, A-Z, 0-9 for alphanumeric fields; 0-9 for numeric fields. On UNIX and Windows it is 0-9, A-Z, a-z for alphanumeric fields; 0-9 for numeric fields.

You have the option of overriding this default and displaying values in descending order, ranging from the highest value to the lowest value, by including HIGHEST in the sort phrase.

Syntax: How to Specify the Sort Order

{BY|ACROSS} {LOWEST|HIGHEST} *sortfield*

where:

LOWEST

Sorts in ascending order, beginning with the lowest value and continuing to the highest value (a-z, A-Z, 0-9 for alphanumeric fields; 0-9 for numeric fields). This option is the default.

HIGHEST

Sorts in descending order, beginning with the highest value and continuing to the lowest value. You can also use TOP as a synonym for HIGHEST.

*sortfield*

Is the name of the sort field.

Example: Sorting in Ascending Order

The following report request does not specify a particular sorting order, and so, by default, it lists salaries ranging from the lowest to the highest.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME
BY CURR_SAL
END
```

You can specify this same ascending order explicitly by including LOWEST in the sort phrase.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME
BY LOWEST CURR_SAL
END
```

The output is:

<u>CURR SAL</u>	<u>LAST NAME</u>
\$9,000.00	GREENSPAN
\$9,500.00	SMITH
\$11,000.00	STEVENS
\$13,200.00	SMITH
\$16,100.00	MCKNIGHT
\$18,480.00	JONES
	MCCOY
\$21,120.00	ROMANS
\$21,780.00	BLACKWOOD
\$26,862.00	IRVING
\$27,062.00	CROSS
\$29,700.00	BANNING

**Example:**    **Sorting in Descending Order**

The following request lists salaries ranging from the highest to lowest.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME
BY HIGHEST CURR_SAL
END
```

The output is:

<u>CURR_SAL</u>	<u>LAST_NAME</u>
\$29,700.00	BANNING
\$27,062.00	CROSS
\$26,862.00	IRVING
\$21,780.00	BLACKWOOD
\$21,120.00	ROMANS
\$18,480.00	JONES
	MCCOY
\$16,100.00	MCKNIGHT
\$13,200.00	SMITH
\$11,000.00	STEVENS
\$9,500.00	SMITH
\$9,000.00	GREENSPAN

### Specifying Your Own Sort Order

Sort field values are automatically displayed in ascending order, beginning with the lowest value and continuing to the highest.

You can override the default order and display values in your own user-defined sorting sequence. To do this, you need to decide the following:

1. Which sort field values you want to allow. You can specify every sort field value, or a subset of values. When you issue your report request, only records containing those values are included in the report.
2. The order in which you want the values to appear. You can specify any order. For example, you could specify that an A1 sort field containing a single-letter code be sorted in the order A, Z, B, C, Y, and so on.

There are two ways to specify your own sorting order, depending on whether you are sorting rows with BY, or sorting columns with ACROSS:

- ☐ The BY ROWS OVER phrase, for defining your own row sort sequence.
- ☐ The ACROSS COLUMNS AND phrase, for defining your own column sort sequence.

### **Syntax:** How to Define Your Own Sort Order

```
BY sortfield AS 'coltitle' ROWS value1 [AS 'text1']  
OVER value2 [AS 'text2']  
[... OVER valuen [ AS 'textn']]  
END
```

where:

*sortfield*

Is the last BY field in the report.

*coltitle*

Is the column title for the BY field on the report output.

*value1*

Is the sort field value that is first in the sorting sequence.

*AS 'text1'*

Enables you to assign alternate text for the first row, which replaces the field value in the output. Enclose the text in single quotation marks.

*value2*

Is the sort field value that is second in the sorting sequence.

*AS 'text2'*

Enables you to assign alternate text for the second row, which replaces the field value in the output. Enclose the text in single quotation marks.

*valuen*

Is the sort field value that is last in the sorting sequence.

*AS 'textn'*

Enables you to assign alternate text for the last row, which replaces the field value in the output. Enclose the text in single quotation marks.

An alternative syntax is

```
FOR sortfield
value1 OVER value2 [... OVER valuen]
```

which uses the row-based reporting phrase FOR, described in [Creating Financial Reports With Financial Modeling Language \(FML\)](#) on page 1729.

### **Reference: Usage Notes for Defining Your Sort Order**

- ☐ Any sort field value that you do not specify in the BY ROWS OVER phrase is not included in the sorting sequence, and does not appear in the report.
- ☐ Sort field values that contain embedded blank spaces should be enclosed in single quotation marks (').

- ❑ Any sort field value that you do specify in the BY ROWS OVER phrase is included in the report, whether or not there is data.
- ❑ If missing data is included in the report, it must be inserted at the lowest sort level.
- ❑ The name of the sort field is not included in the report.
- ❑ Each report request can contain only one BY ROWS OVER phrase. BY ROWS OVER is not supported with the FOR phrase. For information about the FOR phrase, see [Creating Financial Reports With Financial Modeling Language \(FML\)](#) on page 1729.

### **Example:** Defining Your Row Sort Order

The following illustrates how to sort employees by the banks at which their paychecks are automatically deposited, and how to define your own label in the sorting sequence for the bank field.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME
BY BANK_NAME ROWS 'BEST BANK' OVER STATE
  OVER ASSOCIATED OVER 'BANK ASSOCIATION'
END
```

The output is:

	<u>LAST_NAME</u>
BEST BANK	BANNING
STATE	JONES
ASSOCIATED	IRVING
ASSOCIATED	BLACKWOOD
ASSOCIATED	MCKNIGHT
BANK ASSOCIATION	CROSS

### **Syntax:** How to Define Column Sort Sequence

```
ACROSS sortfield COLUMNS value1 AND value2 [... AND valuen]
```

where:

*sortfield*

Is the name of the sort field.

*value1*

Is the sort field value that is first in the sorting sequence.



*value2*

Is the sort field value that is second in the sorting sequence.

*valuen*

Is the sort field value that is last in the sorting sequence.

### **Reference:** Usage Notes for Defining Column Sort Sequence

- ☐ Any sort field value that you do not specify in the ACROSS COLUMNS AND phrase is not included in the label within the sorting sequence, and does not appear in the report.
- ☐ Sort field values that contain embedded blank spaces should be enclosed in single quotation marks.
- ☐ Any sort field value that you do specify in the ACROSS COLUMNS AND phrase is included in the report, whether or not there is data.
- ☐ When using a COMPUTE with an ACROSS COLUMNS phrase, the COLUMNS should be specified last:

*ACROSS acrossfield [AND] COMPUTE compute\_expression; COLUMNS values*

- ☐ Each report request may contain only one BY ROWS OVER phrase.

### **Example:** Defining Column Sort Sequence

The following illustrates how to sum employee salaries by the bank at which they are automatically deposited, and to define your own label within the sorting sequence for the bank field.

```
TABLE FILE EMPLOYEE
SUM CURR_SAL
ACROSS BANK_NAME COLUMNS 'BEST BANK' AND STATE
AND ASSOCIATED AND 'BANK ASSOCIATION'
END
```

The output is:

BANK_NAME				
BEST BANK	STATE	ASSOCIATED	BANK ASSOCIATION	
\$29,700.00	\$18,480.00	\$64,742.00	\$27,062.00	

## Selecting and Assigning Column Titles to ACROSS Values

When you use the ACROSS COLUMNS phrase to select and order the columns that display on the report output for an ACROSS sort field, you can assign each selected column a new column title using an AS phrase.

### **Syntax:** How to Assign Column Titles To ACROSS Values

```
ACROSS sortfield [AS title]  
  COLUMNS aval1 [AS val1title] [{AND|OR} aval2 [AS val2title] [... {AND|  
OR} avaln [AS valntitle]]]
```

where:

*sortfield*

Is the ACROSS field name.

*title*

Is the title for the ACROSS field name.

AND|OR

Is required to separate the selected ACROSS values. AND and OR are synonyms for this purpose.

*aval1, aval2,... avaln*

Are the selected ACROSS values to display on the report output.

*val1title, val2title ...valntitle*

Are the column titles for the selected ACROSS values.

### **Reference:** Usage Notes for Assigning Column Titles to ACROSS Values

- ❑ Any value you specify as an ACROSS value in the sort phrase will appear on the report output, even if the value is screened out by an IF or WHERE test, or if the value does not exist at all in the data source.

**Note:** For styled output formats, SET HIDENULLACRS=ON removes empty columns in ACROSS groups from the report output.

- ❑ Column titles for ACROSS fields appear on a single line of the report output.
- ❑ Support for AS names for ACROSS values is limited to the TABLE FILE command.

- ❑ When you create a HOLD file with SET ASNAMES = ON, the original field name is propagated to the output Master File, *not* the AS name.

### **Example:** Selecting and Assigning Column Titles to ACROSS Values

The following request against the GGSALES data source selects the columns *Coffee Grinder*, *Latte*, and *Coffee Pot* for the ACROSS field PRODUCT, and assigns each of them a new column title:

```
TABLE FILE GGSALES
SUM
DOLLARS/I8M AS ''
BY REGION
ACROSS PRODUCT AS 'Products'
  COLUMNS 'Coffee Grinder' AS 'Grinder'
    OR Latte AS 'caffellatte'
    AND 'Coffee Pot' AS 'Carafe'
ON TABLE SET PAGE NOPAGE
END
```

The output is:

Region	Products Grinder	caffellatte	Carafe
Midwest	\$666,622	\$2,883,566	\$599,878
Northeast	\$509,200	\$2,808,855	\$590,780
Southeast	\$656,957	\$2,637,562	\$645,303
West	\$603,436	\$2,670,405	\$613,624

### Ranking Sort Field Values

When you sort report rows using the BY phrase, you can indicate the numeric rank of each row. Ranking sort field values is frequently combined with restricting sort field values by rank.

Note that it is possible for several report rows to have the same rank if they have identical sort field values.

The default column title for RANKED BY is RANK. You can change the title using an AS phrase. The RANK field has format I7. Therefore, the RANK column in a report can be up to seven digits. For more information, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.

You can rank aggregated values using the syntax RANKED BY TOTAL. For details, see [Sorting and Aggregating Report Columns](#) on page 176.

### **Syntax:**      **How to Rank Sort Field Values**

```
RANKED [AS 'name'] BY {HIGHEST|LOWEST} [n] sortfield [AS 'text']
```

where:

*name*

Is the new name for the RANK column title.

*sortfield*

Is the name of the sort field. The field can be numeric or alphanumeric.

*n*

Is the number of rank categories to display on the report output.

*text*

Is the column heading to use for the sort field column on the report output.

### **Example:**      **Ranking Sort Field Values**

Issue the following request to display a list of employee names in salary order, indicating the rank of each employee by salary. Note that employees Jones and McCoy have the same rank since their current salary is the same.

```
TABLE FILE EMPLOYEE  
PRINT LAST_NAME  
RANKED AS 'Sequence' BY CURR_SAL  
END
```

The output is:

<u>Sequence</u>	<u>CURR_SAL</u>	<u>LAST_NAME</u>
1	\$9,000.00	GREENSPAN
2	\$9,500.00	SMITH
3	\$11,000.00	STEVENS
4	\$13,200.00	SMITH
5	\$16,100.00	MCKNIGHT
6	\$18,480.00	JONES
		MCCOY
7	\$21,120.00	ROMANS
8	\$21,780.00	BLACKWOOD
9	\$26,862.00	IRVING
10	\$27,062.00	CROSS
11	\$29,700.00	BANNING

### **Example:** Ranking and Restricting Sort Field Values

Ranking sort field values is frequently combined with restricting sort field values by rank, as in the following example.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME
RANKED BY HIGHEST 5 CURR_SAL
END
```

The output is:

<u>RANK</u>	<u>CURR_SAL</u>	<u>LAST_NAME</u>
----	-----	-----
1	\$29,700.00	BANNING
2	\$27,062.00	CROSS
3	\$26,862.00	IRVING
4	\$21,780.00	BLACKWOOD
5	\$21,120.00	ROMANS

### **DENSE and SPARSE Ranking**

The WebFOCUS sort phrases RANK BY and BY {HIGHEST|LOWEST} *n* sort the report output and assign rank numbers to the sequence of data values. When assigning a rank to a data value, by default WebFOCUS does not skip rank numbers. This means that even when multiple data values are assigned the same rank, the rank number for the next group of values is the next sequential integer. This method of assigning rank numbers is called *dense*.

Some of the relational engines assign rank numbers using a method called *sparse*. With sparse ranking, if multiple data values are assigned the same rank number, the next rank number will be the previous rank number plus the number of multiples.

You can use the WebFOCUS RANK parameter to control the type of ranking done by WebFOCUS. In addition, if you are accessing a relational data source, you can set the ranking method to the type of ranking done by your relational engine so that the rank calculation can be optimized. Some relational engines have functions for both dense and sparse ranking. In this case, either setting can be optimized.

### **Reference:** Optimizing Ranking

In order to pass rank processing to a relational engine your request must:

- ☐ Use the SUM (or WRITE or ADD) command to aggregate values.
- ☐ Specify the number of rank categories to be displayed. That is, you must specify a value for *n*:

```
[RANKED] BY [HIGHEST] n
```

### **Syntax:** How to Control the Ranking Method

```
SET RANK={DENSE|SPARSE}
```

where:

DENSE

Specifies dense ranking. With this method, each rank number is the next sequential integer, even when the same rank is assigned to multiple data values. DENSE is the default value.

SPARSE

Specifies sparse ranking. With this method, if the same rank number is assigned to multiple data values, the next rank number will be the previous rank number plus the number of multiples.

Then, in your request, use one of the following forms of the BY phrase:

```
RANKED BY {HIGHEST|LOWEST} [n] sortfield [AS 'text']
```

or

```
BY {HIGHEST|LOWEST} n sortfield [AS 'text']
```

where:

*n*

Is the highest rank number to display on the report output when the RANKED BY phrase is used. When RANKED is not used, it is the number of distinct sort field values to display on the report output when SET RANK=DENSE, and the total number of lines of output for the sort field when SET RANK=SPARSE.

*sortfield*

Is the name of the sort field.

*text*

Is the column heading to be used for the sort field column on the report output.

**Reference: Usage Notes for SET RANK**

- ☐ The RNK. prefix operator is not affected by the RANK parameter.
- ☐ The rank numbers propagated to a HOLD file depend on the RANK parameter setting.

**Example: Ranking Values in a FOCUS Data Source**

The following request against the EMPDATA data source ranks salaries in descending order by division. The RANK parameter is set to DENSE (the default).

```
SET RANK = DENSE
TABLE FILE EMPDATA
PRINT LASTNAME FIRSTNAME
RANKED BY HIGHEST 12 SALARY
BY DIV
ON TABLE SET PAGE NOPAGE
END
```

On the output, six employees are included in rank number 6. With dense ranking, the next rank number is the next highest integer, 7.

RANK	SALARY	DIV	LASTNAME	FIRSTNAME
----	-----	---	-----	-----
1	\$115,000.00	CE	LASTRA	KAREN
2	\$83,000.00	CORP	SANCHEZ	EVELYN
3	\$80,500.00	SE	NOZAWA	JIM
4	\$79,000.00	CORP	SOPENA	BEN
5	\$70,000.00	WE	CASSANOVA	LOIS
6	\$62,500.00	CE	ADAMS	RUTH
		CORP	CVEK	MARCUS
			WANG	JOHN
		NE	WHITE	VERONICA
		SE	BELLA	MICHAEL
			HIRSCHMAN	ROSE
7	\$58,800.00	WE	GOTLIEB	CHRIS
8	\$55,500.00	CORP	VALINO	DANIEL
		NE	PATEL	DORINA
9	\$54,100.00	CE	ADDAMS	PETER
		WE	FERNSTEIN	ERWIN
10	\$52,000.00	NE	LIEBER	JEFF
11	\$50,500.00	SE	LEWIS	CASSANDRA
12	\$49,500.00	CE	ROSENTHAL	KATRINA
		SE	WANG	KATE

Running the same request with SET RANK=SPARSE produces the following output. Since rank category 6 includes six employees, the next rank number is 6 + 6.

RANK	SALARY	DIV	LASTNAME	FIRSTNAME
----	-----	---	-----	-----
1	\$115,000.00	CE	LASTRA	KAREN
2	\$83,000.00	CORP	SANCHEZ	EVELYN
3	\$80,500.00	SE	NOZAWA	JIM
4	\$79,000.00	CORP	SOPENA	BEN
5	\$70,000.00	WE	CASSANOVA	LOIS
6	\$62,500.00	CE	ADAMS	RUTH
		CORP	CVEK	MARCUS
			WANG	JOHN
		NE	WHITE	VERONICA
		SE	BELLA	MICHAEL
			HIRSCHMAN	ROSE
12	\$58,800.00	WE	GOTLIEB	CHRIS



**Example: Limiting the Number of Sort Field Values**

The following request against the EMPDATA data source sorts salaries in descending order by division and prints the 12 highest salaries. The RANK parameter is set to DENSE (the default).

```
SET RANK = DENSE
TABLE FILE EMPDATA
PRINT LASTNAME FIRSTNAME
BY HIGHEST 12 SALARY
BY DIV
ON TABLE SET PAGE NOPAGE
END
```

On the output, 12 distinct salary values are displayed, even though some of the employees have the same salaries.

SALARY	DIV	LASTNAME	FIRSTNAME
-----	---	-----	-----
\$115,000.00	CE	LASTRA	KAREN
\$83,000.00	CORP	SANCHEZ	EVELYN
\$80,500.00	SE	NOZAWA	JIM
\$79,000.00	CORP	SOPENA	BEN
\$70,000.00	WE	CASSANOVA	LOIS
\$62,500.00	CE	ADAMS	RUTH
	CORP	CVEK	MARCUS
		WANG	JOHN
	NE	WHITE	VERONICA
	SE	BELLA	MICHAEL
		HIRSCHMAN	ROSE
\$58,800.00	WE	GOTLIEB	CHRIS
\$55,500.00	CORP	VALINO	DANIEL
	NE	PATEL	DORINA
\$54,100.00	CE	ADDAMS	PETER
	WE	FERNSTEIN	ERWIN
\$52,000.00	NE	LIEBER	JEFF
\$50,500.00	SE	LEWIS	CASSANDRA
\$49,500.00	CE	ROSENTHAL	KATRINA
	SE	WANG	KATE

Running the same request with SET RANK=SPARSE produces the following output. Since six employees have salary \$62,500, that value is counted 6 times so that only 12 lines (seven distinct salary values) display on the output.

SALARY	DIV	LASTNAME	FIRSTNAME
----	---	-----	-----
\$115,000.00	CE	LASTRA	KAREN
\$83,000.00	CORP	SANCHEZ	EVELYN
\$80,500.00	SE	NOZAWA	JIM
\$79,000.00	CORP	SOPENA	BEN
\$70,000.00	WE	CASSANOVA	LOIS
\$62,500.00	CE	ADAMS	RUTH
	CORP	CVEK	MARCUS
		WANG	JOHN
	NE	WHITE	VERONICA
	SE	BELLA	MICHAEL
		HIRSCHMAN	ROSE
\$58,800.00	WE	GOTLIEB	CHRIS

## Grouping Numeric Data Into Ranges

When you sort a report using a numeric sort field, you can group the sort field values together and define the range of each group.

There are several ways of defining groups. You can define groups of:

- ☐ Equal range using the IN-GROUPS-OF phrase.

Each report request can contain a total of five IN-GROUPS-OF phrases plus IN-RANGES-OF phrases. The IN-GROUPS-OF phrase can only be used once per BY field. The first sort field range starts from the lowest value of a multiple of the IN-GROUPS-OF value, and the value displayed is the start point of each range.

- ☐ Equal range using the IN-RANGES-OF phrase.

Each report request can contain a total of five IN-GROUPS-OF phrases plus IN-RANGES-OF phrases. The IN-RANGES-OF phrase can only be used once per BY field, and it generates an additional internal sort phrase that must be counted in the total number of sort phrases. The first sort field range starts from the lowest value of a multiple of the IN-GROUPS-OF value. No message is generated if you specify a range of zero, but the values displayed on the report are unpredictable.

- ☐ Unequal range using the FOR phrase.

- ☐ Tiles. These include percentiles, quartiles, or deciles. For details, see [Grouping Numeric Data Into Tiles](#) on page 170.

The FOR phrase is usually used to produce matrix reports and is part of the Financial Modeling Language (FML). However, you can also use it to create columnar reports that group sort field values in unequal ranges.

The FOR phrase displays the sort value for each individual row. The ranges do not have to be contiguous, that is, you can define your ranges with gaps between them. The FOR phrase is described in more detail in [Creating Financial Reports With Financial Modeling Language \(FML\)](#) on page 1729.

**Note:** If there is not any data for a group, a row for the group still appears in the report.

### **Syntax:** How to Define Groups of Equal Range

```
{BY|ACROSS} sortfield IN-GROUPS-OF value [TOP limit]
```

where:

*sortfield*

Is the name of the sort field. The sort field must be numeric: its format must be I (integer), F (floating-point number), D (decimal number), or P (packed number).

*value*

Is a positive integer that specifies the range by which sort field values are grouped.

*limit*

Is an optional number that defines the highest group label to be included in the report.

### **Example:** Defining Groups of Equal Ranges

The following illustrates how to show which employees fall into which salary ranges, and to define the ranges by \$5,000 increments.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME
BY CURR_SAL IN-GROUPS-OF 5000
END
```

The output is:

CURR_SAL	LAST_NAME
-----	-----
\$5,000.00	SMITH
	GREENSPAN
\$10,000.00	STEVENS
	SMITH
\$15,000.00	JONES
	MCCOY
	MCKNIGHT
\$20,000.00	ROMANS
	BLACKWOOD
\$25,000.00	BANNING
	IRVING
	CROSS

### **Syntax:** How to Define Equal Ranges

```
{BY|ACROSS} sortfield IN-RANGES-OF value [TOP limit]
```

where:

*sortfield*

Is the name of the sort field. The sort field must be numeric: its format must be I (Integer), F (floating-point), D (double-precision), or P (packed).

*value*

Is an integer greater than zero indicating the range by which sort field values are grouped.

*limit*

Is an optional number that defines the highest range label to be included in the report. The range is extended to include all data values higher than this value.

**Note:** IN-RANGES-OF generates an internal sort phrase that must be counted in the total number of sort phrases.

### **Example:** Defining Equal Ranges

```
TABLE FILE EMPLOYEE  
PRINT LAST_NAME  
BY CURR_SAL IN-RANGES-OF 5000  
END
```

The output is:

<u>CURR SAL</u>	<u>LAST NAME</u>
\$5,000.00 - \$9,999.99	SMITH
	GREENSPAN
\$10,000.00 - \$14,999.99	STEVENS
	SMITH
\$15,000.00 - \$19,999.99	JONES
	MCCOY
	MCKNIGHT
\$20,000.00 - \$24,999.99	ROMANS
	BLACKWOOD
\$25,000.00 - \$29,999.99	BANNING
	IRVING
	CROSS

**Syntax:**      **How to Define Custom Groups of Data Values**

```
FOR sortfield
  begin1 TO end1 [OVER begin2 TO end2 ... ]
```

where:

*sortfield*

Is the name of the sort field.

*begin*

Is a value that identifies the beginning of a range.

*end*

Is a value that identifies the end of a range.

### **Example:** Defining Custom Groups of Data Values

The following request displays employee salaries, but it groups them in an arbitrary way. Notice that the starting value of each range prints in the report.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME
FOR CURR_SAL
9000 TO 13500 OVER
14000 TO 19700 OVER
19800 TO 30000
END
```

The output is:

	<u>LAST_NAME</u>
9000	STEVENS
9000	SMITH
9000	SMITH
9000	GREENSPAN
14000	JONES
14000	MCCOY
14000	MCKNIGHT
19800	BANNING
19800	IRVING
19800	ROMANS
19800	BLACKWOOD
19800	CROSS

### Grouping Numeric Data Into Tiles

You can group numeric data into any number of tiles (percentiles, deciles, quartiles, etc.) in tabular reports. For example, you can group student test scores into deciles to determine which students are in the top ten percent of the class, or determine which sales representatives are in the top half of all sales representatives based on total sales.

Grouping is based on the values in the selected vertical (BY) field, and data is apportioned as equally as possible into the number of tile groups you specify.

The following occurs when you group data into tiles:

- ❑ A new column, labeled TILE by default, is added to the report output and displays the tile number assigned to each instance of the tile field. You can change the column heading with an AS phrase. For details on the AS phrase, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.

- ❑ Tiling is calculated within all of the higher-level sort fields in the request, and restarts whenever a sort field at a higher level than the tile field value changes.
- ❑ Instances are counted using the tile field. If the request prints fields from lower level segments, there may be multiple report lines that correspond to one instance of the tile field.
- ❑ Instances with the same tile field value are placed in the same tile. For example, consider the following data, which is to be apportioned into three tiles:

1 5 5 5 8 9

In this case, dividing the instances into groups containing an equal number of records produces the following:

Group	Data Values
1	1,5
2	5,5
3	8,9

However, because all of the same data values must be in the same tile, the fives (5) that are in group 2 are moved to group 1. Group 2 remains empty. The final tiles are:

Tile Number	Data Values
1	1,5,5,5
2	
3	8,9

### **Syntax:** How to Group Numeric Data Into Tiles

```
BY [ {HIGHEST|LOWEST} [k] ] tilefield [AS 'head1']
    IN-GROUPS-OF n TILES [TOP m] [AS 'head2']
```

where:

**HIGHEST**

Sorts the data in descending order so that the highest data values are placed in tile 1.

### LOWEST

Sorts the data in ascending order so that the lowest data values are placed in tile 1. This is the default sort order.

### *k*

Is a positive integer representing the number of tile groups to display in the report. For example, BY HIGHEST 2 displays the two non-empty tiles with the highest data values.

### *tilefield*

Is the field whose values are used to assign the tile numbers.

### *head1*

Is a heading for the column that displays the values of the tile sort field.

### *n*

Is a positive integer not greater than 32,767, specifying the number of tiles to be used in grouping the data. For example, 100 tiles produces percentiles, while 10 tiles produces deciles.

### *m*

Is a positive integer indicating the highest tile value to display in the report. For example, TOP 3 does not display any data row that is assigned a tile number greater than 3.

### *head2*

Is a new heading for the column that displays the tile numbers.

### **Note:**

- ☐ The syntax accepts numbers that are not integers for *k*, *n*, and *m*. On z/OS, values with decimals are rounded to integers; on UNIX and Windows they are truncated. If the numbers supplied are negative or zero, an error message is generated.
- ☐ Both *k* and *m* limit the number of rows displayed within each sort break in the report. If you specify both, the more restrictive value controls the display. If *k* and *m* are both greater than *n* (the number of tiles), *n* is used.



**Example: Grouping Data Into Five Tiles**

The following illustrates how to group data into five tiles.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME
BY DEPARTMENT
BY CURR_SAL IN-GROUPS-OF 5 TILES
END
```

The output is:

<u>DEPARTMENT</u>	<u>CURR SAL</u>	<u>TILE</u>	<u>LAST NAME</u>	<u>FIRST NAME</u>
MIS	\$9,000.00	1	GREENSPAN	MARY
	\$13,200.00	1	SMITH	MARY
	\$18,480.00	2	JONES	DIANE
			MCCOY	JOHN
	\$21,780.00	4	BLACKWOOD	ROSEMARIE
PRODUCTION	\$27,062.00	5	CROSS	BARBARA
	\$9,500.00	1	SMITH	RICHARD
	\$11,000.00	1	STEVENS	ALFRED
	\$16,100.00	2	MCKNIGHT	ROGER
	\$21,120.00	3	ROMANS	ANTHONY
	\$26,862.00	4	IRVING	JOAN
	\$29,700.00	5	BANNING	JOHN

Note that the tiles are assigned within the higher-level sort field DEPARTMENT. The MIS category does not have any data assigned to tile 3. The PRODUCTION category has all five tiles.

**Example: Displaying the First Three Tile Groups**

In this example, the employees with the three lowest salaries are grouped into five tiles.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME
BY DEPARTMENT
BY LOWEST 3 CURR_SAL IN-GROUPS-OF 5 TILES
END
```

The output is:

<u>DEPARTMENT</u>	<u>CURR SAL</u>	<u>TILE</u>	<u>LAST NAME</u>	<u>FIRST NAME</u>
MIS	\$9,000.00	1	GREENSPAN	MARY
	\$13,200.00	1	SMITH	MARY
	\$18,480.00	2	JONES	DIANE
			MCCOY	JOHN
PRODUCTION	\$21,780.00	4	BLACKWOOD	ROSEMARIE
	\$9,500.00	1	SMITH	RICHARD
	\$11,000.00	1	STEVENS	ALFRED
	\$16,100.00	2	MCKNIGHT	ROGER
	\$21,120.00	3	ROMANS	ANTHONY

Note that the request displays three tile groups in each category. Because no data was assigned to tile 3 in the MIS category, tiles 1, 2, and 4 display for that category.

**Example: Displaying Tiles With a Value of Three or Less**

In this example, the employees with the three lowest salaries are listed and grouped into five tiles, but only the tiles that are in the top 3 (tiles 1, 2, or 3) are displayed in the report. Also, the heading for the TILES field has been renamed (using the AS phrase) to DECILES.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME
BY DEPARTMENT
BY LOWEST 3 CURR_SAL IN-GROUPS-OF 5 TILES TOP 3 AS DECILES
END
```

The output is:

<u>DEPARTMENT</u>	<u>CURR SAL</u>	<u>DECILE</u>	<u>LAST NAME</u>	<u>FIRST NAME</u>
MIS	\$9,000.00	1	GREENSPAN	MARY
	\$13,200.00	1	SMITH	MARY
	\$18,480.00	2	JONES	DIANE
			MCCOY	JOHN
PRODUCTION	\$9,500.00	1	SMITH	RICHARD
	\$11,000.00	1	STEVENS	ALFRED
	\$16,100.00	2	MCKNIGHT	ROGER
	\$21,120.00	3	ROMANS	ANTHONY

Because no data was assigned to tile 3 in the MIS category, only tiles 1 and 2 display for that category.

**Reference: Usage Notes for Tiles**

- ❑ If a request retrieves data from segments that are descendants of the segment containing the tile field, multiple report rows may correspond to one instance of the tile field. These additional report rows do not affect the number of instances used to assign the tile values. However, if you retrieve fields from multiple segments and create a single-segment output file, this flat file will have multiple instances of the tile field, and this increased number of instances may affect the tile values assigned. Therefore, when you run the same request against the multi-level file and the single-segment file, different tile assignments may result.
- ❑ Tiles are always calculated on a BY sort field in the request.
- ❑ Only one tiles calculation is supported per request. However, the request can contain up to five (the maximum allowed) non-tile IN-GROUP-OF phrases in addition to the TILES phrase.
- ❑ Comparisons for the purpose of assigning tile numbers use exact data values regardless of their display format. Therefore, if you display a floating-point value as D7, you may not be showing enough significant digits to indicate why values are placed in separate tiles.
- ❑ The tile field can be a real field or a virtual field created with a DEFINE command or a DEFINE in the Master File. The COMPUTE command cannot be used to create a tile field.
- ❑ Empty tiles do not display in the report output.
- ❑ In requests with multiple sort fields, tiles are supported only at the lowest level and only with the BY LOWEST phrase.
- ❑ Tiles are supported with output files. However, the field used to calculate the tiles propagates three fields to a HOLD file (the actual field value, the tile, and a ranking field) unless you set HOLDLIST to PRINTONLY.
- ❑ Tiles are not supported with BY TOTAL, TABLEF, FML, and GRAPH.

**Restricting Sort Field Values by Highest/Lowest Rank**

When you sort report rows using the BY phrase, you can restrict the sort field values to a group of high or low values. You choose the number of fields to include in the report. For example, you can choose to display only the 10 highest (or lowest) sort field values in your report by using BY HIGHEST (or LOWEST).

You can have up to five sort fields with BY HIGHEST or BY LOWEST.

**Syntax:**      **How to Restrict Sort Field Values by Highest/Lowest Rank**

`BY {HIGHEST n|LOWEST n} sortfield`

where:

`HIGHEST n`

Specifies that only the highest *n* sort field values are included in the report. TOP is a synonym for HIGHEST.

`LOWEST n`

Specifies that only the lowest *n* sort field values are included in the report.

`sortfield`

Is the name of the sort field. The sort field can be numeric or alphanumeric.

**Note:** HIGHEST/LOWEST *n* refers to the number of sort field values, not the number of report rows. If several records have the same sort field value that satisfies the HIGHEST/LOWEST *n* criteria, all of them are included in the report.

**Example:**      **Restricting Sort Field Values to a Group**

The following request displays the names of the employees earning the five highest salaries.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME
BY HIGHEST 5 CURR_SAL
END
```

The output is:

<u>CURR SAL</u>	<u>LAST NAME</u>
\$29,700.00	BANNING
\$27,062.00	CROSS
\$26,862.00	IRVING
\$21,780.00	BLACKWOOD
\$21,120.00	ROMANS

## Sorting and Aggregating Report Columns

Using the BY TOTAL phrase, you can apply aggregation and sorting simultaneously to numeric columns in your report in one pass of the data. For BY TOTAL to work correctly, you must have an aggregating display command such as SUM. A non-aggregating display command, such as PRINT, simply retrieves the data without aggregating it. Records are sorted in either ascending or descending sequence, based on your query. Ascending order is the default.

You can also use the BY TOTAL phrase to sort based on temporary values calculated by the COMPUTE command.

**Note:** On z/OS, the sort on the aggregated value is calculated using an external sort package, even if EXTSORT = OFF.

### **Syntax:** How to Sort and Aggregate a Report Column

```
[RANKED] BY [HIGHEST|LOWEST [n] ]
          TOTAL {display_field/COMPUTE name/format=expression;}
```

or

```
[RANKED] BY TOTAL {[HIGHEST|LOWEST [n] ]
                  display_field/COMPUTE name/format=expression;}
```

where:

**RANKED**

Adds a column to the report in which a rank number is assigned to each aggregated sort value in the report output. If multiple rows have the same ranking, the rank number only appears in the first row.

**n**

Is the number of sort field values you wish to display in the report. If *n* is omitted, all values of the calculated sort field are displayed. The default order is from lowest to highest.

**display\_field**

Can be a field name, a field name preceded by an operator (that is, prefixoperator.fieldname), or a calculated value.

A BY TOTAL field is treated as a display field when the internal matrix is created. After the matrix is created, the output lines are aggregated and re-sorted based on all of the sort fields.

### **Example:** Sorting and Aggregating Report Columns

In this example, the salary average is calculated and used as a sort field. The two highest salaries are displayed in the report.

```
TABLE FILE EMPLOYEE
SUM SALARY CNT.SALARY
BY DEPARTMENT
BY HIGHEST 2 TOTAL AVE.SALARY AS 'HIGHEST,AVERAGE,SALARIES'
BY CURR_JOBCODE
END
```

The output is:

	HIGHEST AVERAGE			SALARY
<u>DEPARTMENT</u>	<u>SALARIES</u>	<u>CURR_JOBCODE</u>	<u>SALARY</u>	<u>COUNT</u>
MIS	\$26,418.50	A17	\$52,837.00	2
	\$21,780.00	B04	\$21,780.00	1
PRODUCTION	\$29,700.00	A17	\$29,700.00	1
	\$25,641.00	A15	\$51,282.00	2

**Example: Sorting, Aggregating, and Ranking Report Columns**

In this example, the salary average is calculated and used as a sort field. The two highest salaries are displayed and ranked.

```
TABLE FILE EMPLOYEE
SUM SALARY CNT.SALARY
BY DEPARTMENT
RANKED BY HIGHEST 2 TOTAL AVE.SALARY AS 'HIGHEST,AVERAGE,SALARIES'
BY CURR_JOBCODE
END
```

The output is:

		HIGHEST AVERAGE			SALARY
<u>DEPARTMENT</u>	<u>RANK</u>	<u>SALARIES</u>	<u>CURR_JOBCODE</u>	<u>SALARY</u>	<u>COUNT</u>
MIS	1	\$26,418.50	A17	\$52,837.00	2
	2	\$21,780.00	B04	\$21,780.00	1
PRODUCTION	1	\$29,700.00	A17	\$29,700.00	1
	2	\$25,641.00	A15	\$51,282.00	2

**Example: Sorting and Aggregating Report Columns With COMPUTE**

In this example, the monthly salary is calculated using a COMPUTE within a sort field. The two highest monthly salaries are displayed.

```
TABLE FILE EMPLOYEE
SUM SALARY CNT.SALARY
BY DEPARTMENT
BY HIGHEST 2 TOTAL COMPUTE MONTHLY_SALARY/D12.2M=SALARY/12;
AS 'HIGHEST,MONTHLY,SALARIES'
BY CURR_JOBCODE
END
```

The output is:

	HIGHEST MONTHLY			SALARY
<u>DEPARTMENT</u>	<u>SALARIES</u>	<u>CURR</u>	<u>JOB</u> <u>CODE</u>	<u>SALARY</u>
MIS	\$4,403.08	A17		\$52,837.00
	\$3,019.17	B03		\$36,230.00
PRODUCTION	\$4,273.50	A15		\$51,282.00
	\$2,591.67	B02		\$31,100.00
				COUNT
				2
				2
				2
				2

**Reference:** Usage Notes for BY TOTAL

- ❑ When you use BY HIGHEST/LOWEST *n* with BY TOTAL HIGHEST/LOWEST *n*, the BY TOTAL phrase works on the result of the BY phrase (that is, on the *n* rows that result from the BY phrase).

## Hiding Sort Values

When you sort a report, you can omit the sort field value itself from the report by using the phrase NOPRINT. This can be helpful in several situations; for instance, when you use the same field as a sort field and a display field, or when you want to sort by a field but not display its values in the report output.

**Syntax:** How to Hide Sort Values

```
{BY|ACROSS} sortfield {NOPRINT|SUP-PRINT}
```

where:

*sortfield*

Is the name of the sort field.

You can use SUP-PRINT as a synonym for NOPRINT.

**Example:** Hiding Sort Values

If you want to display a list of employees sorted by the date on which they were hired, but you want the report to contain last name, first name, and then the hire date in the third column, the following request is insufficient.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME HIRE_DATE
END
```

The output is:

<u>LAST_NAME</u>	<u>FIRST_NAME</u>	<u>HIRE_DATE</u>
STEVENS	ALFRED	80/06/02
SMITH	MARY	81/07/01
JONES	DIANE	82/05/01
SMITH	RICHARD	82/01/04
BANNING	JOHN	82/08/01
IRVING	JOAN	82/01/04
ROMANS	ANTHONY	82/07/01
MCCOY	JOHN	81/07/01
BLACKWOOD	ROSEMARIE	82/04/01
MCKNIGHT	ROGER	82/02/02
GREENSPAN	MARY	82/04/01
CROSS	BARBARA	81/11/02

To list the employees in the order in which they were hired, you would sort the report by the HIRE\_DATE field and hide the sort field occurrence using the NOPRINT phrase.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME HIRE_DATE
BY HIRE_DATE NOPRINT
END
```

The output is:

<u>LAST_NAME</u>	<u>FIRST_NAME</u>	<u>HIRE_DATE</u>
STEVENS	ALFRED	80/06/02
SMITH	MARY	81/07/01
MCCOY	JOHN	81/07/01
CROSS	BARBARA	81/11/02
SMITH	RICHARD	82/01/04
IRVING	JOAN	82/01/04
MCKNIGHT	ROGER	82/02/02
BLACKWOOD	ROSEMARIE	82/04/01
GREENSPAN	MARY	82/04/01
JONES	DIANE	82/05/01
ROMANS	ANTHONY	82/07/01
BANNING	JOHN	82/08/01



## Sort Performance Considerations

The sorting procedure analyzes the request being processed and the amount of sort memory available in order to reduce the amount of disk I/O. The sort strategy is controlled by the specifics of the request and the values of the SORTMATRIX and SORTMEMORY parameters.

### **SORTMATRIX**

The SORTMATRIX parameter controls whether to employ in-memory sorting with decreased use of external memory. The syntax is

```
SET SORTMATRIX = {SMALL | LARGE}
```

where:

#### SMALL

Creates a single sort matrix of up to 2048 rows, and uses a binary search based insertion sort with aggregation during retrieval. The maximum number of rows in this matrix has been determined to provide the best performance for this type of sort. If the sort matrix becomes full, it is written to a file called FOCSORT on disk, the in-memory matrix is emptied, and retrieval continues, writing to FOCSORT as many times as necessary. When the end of data is detected, the remaining rows are written to FOCSORT and the merge routine merges all of the sort strings in FOCSORT (which, in extreme cases, may require multiple merge phases), while also completing the aggregation.

#### LARGE

Creates a large matrix or multiple small matrices in memory, when adequate memory is available as determined by the SORTMEMORY parameter. LARGE is the default value. The goal of this strategy is to do as much sorting as possible in internal memory before writing any records to disk. Whether disk I/O is necessary at all in the sorting process depends on the amount of memory allocated for sorting and the size of the request output. If the amount of SORTMEMORY is not large enough to meaningfully make use of the LARGE strategy, the sort will default to the SMALL strategy. The LARGE strategy greatly reduces the need for disk I/O and, if disk I/O is required after all (for very large output), it virtually eliminates the need for multiple merge phases.

### **SORTMEMORY**

The SORTMEMORY parameter controls the amount of internal memory available for sorting. The syntax is

```
SET SORTMEMORY = {n | 512}
```

where:

*n*

Is the positive number of megabytes of memory available for sorting. The default value is 512.

### Sorting With Multiple Display Commands

A request can consist of up to 64 sets of separate display commands (also known as verb phrases), each with its own sort conditions. In order to display all of the information, a meaningful relationship has to exist among the separate sort condition sets. The following rules apply:

- ❑ Up to 64 display commands and their associated sort conditions can be used. The first display command does not have to have any sort condition. Only the last display command may be a detail command, such as PRINT or LIST. Other preceding display commands must be aggregating commands.
- ❑ WHERE and IF criteria apply to the records selected for the report as a whole. WHERE and IF criteria are explained in [Selecting Records for Your Report](#) on page 219.
- ❑ When a sort phrase is used with a display command, the display commands following it must use the same sorting condition in the same order. For example:

```
TABLE FILE EMPLOYEE
SUM ED_HRS
SUM CURR_SAL CNT.CURR_SAL
BY DEPARTMENT
PRINT FIRST_NAME
BY DEPARTMENT
BY LAST_NAME
END
```

The first SUM does not have a sort condition. The second SUM has a sort condition: BY DEPARTMENT. Because of this sort condition, the PRINT command must have BY DEPARTMENT as the first sort condition, and other sort conditions may be added as needed.

#### **Example:** Using Multiple Display and Sort Fields

The following request summarizes several levels of detail in the data source.

```
TABLE FILE EMPLOYEE
SUM CURR_SAL
SUM CURR_SAL BY DEPARTMENT
SUM CURR_SAL BY DEPARTMENT BY LAST_NAME
END
```

The command SUM CURR\_SAL calculates the total amount of current salaries; SUM CURR\_SAL BY DEPARTMENT calculates the total amounts of current salaries in each department; SUM CURR\_SAL BY DEPARTMENT BY LAST\_NAME calculates the total amounts of current salaries for each employee name.

The output is:

<u>CURR_SAL</u>	<u>DEPARTMENT</u>	<u>CURR_SAL</u>	<u>LAST_NAME</u>	<u>CURR_SAL</u>
\$222,284.00	MIS	\$108,002.00	BLACKWOOD	\$21,780.00
			CROSS	\$27,062.00
			GREENSPAN	\$9,000.00
			JONES	\$18,480.00
			MCCOY	\$18,480.00
			SMITH	\$13,200.00
	PRODUCTION	\$114,282.00	BANNING	\$29,700.00
			IRVING	\$26,862.00
			MCKNIGHT	\$16,100.00
			ROMANS	\$21,120.00
			SMITH	\$9,500.00
			STEVENS	\$11,000.00

## Controlling Formatting of Reports With Multiple Display Commands

You can use the SET DUPLICATECOL command to reformat report requests that use multiple display commands, placing aggregated fields in the same column above the displayed field.

By default, each new display command in a request generates additional sort field and display field columns. With DUPLICATECOL set to OFF, each field occupies only one column in the request, with the values from each display command stacked under the values for the previous display command.

### *Syntax:* How to Control the Format of Reports With Multiple Display Commands

```
SET DUPLICATECOL={ON|OFF}
```

where:

ON

Displays the report with each field as a column. This is the default value.

OFF

Displays the report with common fields as a row.

**Example:**    **Displaying Reports With Multiple Display Commands**

The following request sums current salaries and education hours for the entire EMPLOYEE data source and for each department:

```
TABLE FILE EMPLOYEE
SUM CURR_SAL ED_HRS
SUM CURR_SAL ED_HRS BY DEPARTMENT
END
```

With DUPLICATECOL=ON, the output has separate columns for the grand totals and for the departmental totals:

CURR_SAL	ED_HRS	DEPARTMENT	CURR_SAL	ED_HRS
-----	-----	-----	-----	-----
\$222,284.00	351.00	MIS	\$108,002.00	231.00
		PRODUCTION	\$114,282.00	120.00

With DUPLICATECOL=OFF, the output has one column for each field. The grand totals are on the top row of the report, and the departmental totals are on additional rows below the grand totals:

DEPARTMENT	CURR_SAL	ED_HRS
-----	-----	-----
	\$222,284.00	351.00
MIS	\$108,002.00	231.00
PRODUCTION	\$114,282.00	120.00

The following request adds a PRINT command sorted by department and by last name to the previous request:

```
SET SPACES = 1
TABLE FILE EMPLOYEE
SUM CURR_SAL ED_HRS
SUM CURR_SAL ED_HRS BY DEPARTMENT AS 'DEPT'
PRINT FIRST_NAME CURR_SAL ED_HRS BY DEPARTMENT BY LAST_NAME
END
```

With DUPLICATECOL=ON, the output has separate columns for the grand totals, for the departmental totals, and for each last name:

CURR_SAL	ED_HRS	DEPT	CURR_SAL	ED_HRS	LAST_NAME	FIRST_NAME	CURR_SAL	ED_HRS
-----	-----	-----	-----	-----	-----	-----	-----	-----
\$222,284.00	351.00	MIS	\$108,002.00	231.00	BLACKWOOD	ROSEMARIE	\$21,780.00	75.00
					CROSS	BARBARA	\$27,062.00	45.00
					GREENSPAN	MARY	\$9,000.00	25.00
					JONES	DIANE	\$18,480.00	50.00
					MCCOY	JOHN	\$18,480.00	.00
					SMITH	MARY	\$13,200.00	36.00
		PRODUCTION	\$114,282.00	120.00	BANNING	JOHN	\$29,700.00	.00
					IRVING	JOAN	\$26,862.00	30.00
					MCKNIGHT	ROGER	\$16,100.00	50.00
					ROMANS	ANTHONY	\$21,120.00	5.00
					SMITH	RICHARD	\$9,500.00	10.00
					STEVENS	ALFRED	\$11,000.00	25.00

With DUPLICATECOL=OFF, the output has one column for each field. The grand totals are on the top row of the report, the departmental totals are on additional rows below the grand totals, and the values for each last name are on additional rows below their departmental totals:

DEPT	LAST_NAME	FIRST_NAME	CURR_SAL	ED_HRS
			\$222,284.00	351.00
MIS			\$108,002.00	231.00
	BLACKWOOD	ROSEMARIE	\$21,780.00	75.00
	CROSS	BARBARA	\$27,062.00	45.00
	GREENSPAN	MARY	\$9,000.00	25.00
	JONES	DIANE	\$18,480.00	50.00
	MCCOY	JOHN	\$18,480.00	.00
	SMITH	MARY	\$13,200.00	36.00
PRODUCTION			\$114,282.00	120.00
	BANNING	JOHN	\$29,700.00	.00
	IRVING	JOAN	\$26,862.00	30.00
	MCKNIGHT	ROGER	\$16,100.00	50.00
	ROMANS	ANTHONY	\$21,120.00	5.00
	SMITH	RICHARD	\$9,500.00	10.00
	STEVENS	ALFRED	\$11,000.00	25.00

### **Syntax:** How to Style a Report With SET DUPLICATECOL=ON

In a StyleSheet, you can identify the rows you want to style by specifying which display command created those rows:

`VERBSET = n`

where:

*n*

Is the ordinal number of the display command in the report request.

### **Example:** Styling Rows Associated With a Specific Display Command

The following request has two display commands:

1. SUM CURR\_SAL ED\_HRS BY DEPARTMENT (totals by department).
2. PRINT FIRST\_NAME CURR\_SAL ED\_HRS BY DEPARTMENT BY LAST\_NAME (values by employee by department).

```
SET DUPLICATECOL = OFF
TABLE FILE EMPLOYEE
SUM CURR_SAL ED_HRS BY DEPARTMENT
PRINT FIRST_NAME CURR_SAL ED_HRS BY DEPARTMENT BY LAST_NAME

ON TABLE SET STYLE *
TYPE = REPORT, COLUMN= P4, VERBSET = 1, STYLE = ITALIC, COLOR=BLUE,$
TYPE = REPORT, COLUMN= B2, VERBSET = 2, STYLE = UNDERLINE, COLOR = RED,$
ENDSTYLE
END
```

On the output:

- ☐ The fourth displayed column (P4, department total of CURR\_SAL) for the SUM command is italic and blue.
- ☐ The second BY field (LAST\_NAME) for the PRINT command is underlined and red.

When you style specific columns, using P notation means that you count every column that displays on the report output, including BY columns. Therefore, P1 is the DEPARTMENT column, P2 is the LAST\_NAME column (this is also B2, the second BY field column), P3 is the FIRST\_NAME column, P4 is the displayed version of the CURR\_SAL column (the internal matrix has multiple CURR\_SAL columns), and P5 is the displayed ED\_HRS column (the internal matrix has multiple ED\_HRS columns).

The output is:

PAGE 1				
DEPARTMENT	LAST_NAME	FIRST_NAME	CURR_SAL	ED_HRS
MIS			<u>\$108,002.00</u>	231.00
	<u>BLACKWOOD</u>	ROSEMARIE	\$21,780.00	75.00
	<u>CROSS</u>	BARBARA	\$27,062.00	45.00
	<u>GREENSPAN</u>	MARY	\$9,000.00	25.00
	<u>JONES</u>	DIANE	\$18,480.00	50.00
	<u>MCCOY</u>	JOHN	\$18,480.00	.00
	<u>SMITH</u>	MARY	\$13,200.00	36.00
PRODUCTION			<u>\$114,282.00</u>	120.00
	<u>BANNING</u>	JOHN	\$29,700.00	.00
	<u>IRVING</u>	JOAN	\$26,862.00	30.00
	<u>MCKNIGHT</u>	ROGER	\$16,100.00	50.00
	<u>ROMANS</u>	ANTHONY	\$21,120.00	5.00
	<u>SMITH</u>	RICHARD	\$9,500.00	10.00
	<u>STEVENS</u>	ALFRED	\$11,000.00	25.00

### **Reference:** Stacking Duplicate Columns in Multi-Verb Requests Based on AS Names

You can use the SET DUPLICATECOL command to reformat report requests that use multiple display commands, placing aggregated fields in the same column above the displayed field.

By default, each new display command in a request generates additional sort field and display field columns. With DUPLICATECOL set to OFF, each field occupies only one column in the request, with the values from each display command stacked under the values for the previous display command.

In prior releases, the duplicate columns were matched based on field names. Now, fields can also be matched based on AS names. An AS name will not be matched to a field name. When a field has an AS name, it will only be matched to other fields that have the same AS name.

**Example: Stacking Duplicate Columns in Multi-Verb Requests Based on AS Names**

The following request has three display commands. The first sums the CURR\_SAL field. The second sums the SALARY field by department. The third prints the GROSS field by department and last name. Each field is assigned the same AS name, even the CURR\_SAL field.

```
TABLE FILE EMPLOYEE
SUM CURR_SAL AS CURR_SAL ED_HRS
SUM SALARY AS CURR_SAL ED_HRS BY DEPARTMENT AS 'DEPT'
PRINT FIRST_NAME GROSS AS CURR_SAL ED_HRS BY DEPARTMENT BY LAST_NAME
ON TABLE SET DUPLICATECOL OFF
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF, SIZE=10, $
VERBSET=1, COLOR=RED,$
VERBSET=2, COLOR=BLUE,$
VERBSET=3,COLOR=BLACK,$
ENDSTYLE
END
```



The partial output is shown in the following image.

<u>DEPT</u>	<u>LAST_NAME</u>	<u>FIRST_NAME</u>	<u>CURR_SAL</u>	<u>ED_HRS</u>
			\$222,284.00	351.00
MIS			\$160,177.00	231.00
	BLACKWOOD	ROSEMARIE	\$1,815.00	75.00
		ROSEMARIE	\$1,815.00	75.00
		ROSEMARIE	\$1,815.00	75.00
		ROSEMARIE	\$1,815.00	75.00
		ROSEMARIE	\$1,815.00	75.00
	CROSS	BARBARA	\$2,255.00	45.00
		BARBARA	\$2,255.00	45.00
		BARBARA	\$2,255.00	45.00
		BARBARA	\$2,255.00	45.00
		BARBARA	\$2,255.00	45.00
		BARBARA	\$2,147.75	45.00
		BARBARA	\$2,147.75	45.00
		BARBARA	\$2,147.75	45.00
		BARBARA	\$2,147.75	45.00
		BARBARA	\$2,147.75	45.00
	GREENSPAN	MARY	\$750.00	25.00
		MARY	\$750.00	25.00
		MARY	\$750.00	25.00
		MARY	\$720.84	25.00
	JONES	DIANE	\$1,540.00	50.00
		DIANE	\$1,540.00	50.00
		DIANE	\$1,540.00	50.00
		DIANE	\$1,479.50	50.00
	MCCOY	JOHN	\$1,540.00	.00
	SMITH	MARY	\$1,100.00	36.00
		MARY	\$1,100.00	36.00
		MARY	\$1,100.00	36.00
		MARY	\$1,100.00	36.00
		MARY	\$1,100.00	36.00
		MARY	\$1,100.00	36.00
		MARY	\$1,100.00	36.00
		MARY	\$1,100.00	36.00

## Improving Efficiency With External Sorts

When a report is generated, by default it is sorted using an internal sorting procedure. This sorting procedure is optimized for reports of up to approximately 180 to 200K, although many factors affect the size of the data that can be handled by the internal sort.

The FOCSORT file used for the internal sort can grow to any size allowed by the operating system running and the available disk space. The user does not have to break a request up to accommodate massive files. In previous releases, the FOCSORT file was limited to 2 GB and the user received a FOC298 message when the WebFOCUS limit was exceeded. With no limit enforced by WebFOCUS, the operating system provides whatever warning and error handling it has for the management of a FOCSORT file that exceeds its limits.

You can generate larger reports somewhat faster by using dedicated sorting products, such as SyncSort, DFSORT, or, in non-Mainframe environments, the WebFOCUS external sort routines.

To use an external sort, the EXTSORT parameter must be ON. Use of a StyleSheet turns off external sorting.

Note that in Mainframe environments, external sorting is supported with the French, Spanish, German, and Scandinavian National Languages (Swedish, Danish, Finnish, and Norwegian). To specify the National Language Support Environment, use the LANG parameter as described in the *Developing Reporting Applications* manual.

### **Reference:** Requirements for External Sorting

You can use the DFSORT and SyncSort external sort products with any TABLE, FML, GRAPH, or MATCH request in all WebFOCUS Mainframe environments. In other operating environments, WebFOCUS has its own external sort routines.

### **Reference:** Usage Notes for External Sorting in Non-Mainframe Environments

It is probably best not to use external sort if:

- ☐ Your request requires a matrix (cannot be converted to a TABLEF request). If your request needs a matrix and uses external sort, it will go through two sorts, both external and internal, and it will be hard to realize any performance gains.

To tell if your report is convertible to TABLEF, use ? STAT (as described in [How to Query the Sort Type](#) on page 191) or run an abbreviated version of the request with a low record limit and external sort on. If the report statistics are printed after the TABLE output, it was performed as TABLEF; if the statistics are printed before the first screen of TABLE output, it went through TABLE processing because it was not convertible to TABLEF.

- ❑ Your input is sorted or almost sorted.
- ❑ Your system cannot support a large number of work files (for information, see [Sort Work Files and Return Codes](#) on page 192). In this case the internal sort may do a better job since internally it implements about 60 logical work files, all sharing space in FOCSORT.

### ***Procedure:* How to Determine the Type of Sort Used**

To determine which sort is used, the following criteria are evaluated, in this sequence:

1. **BINS.** If an entire report can be sorted within the work area (BINS), the external sort is not invoked, even if EXTSORT is set ON.
2. **EXTERNAL.** If BINS is not large enough to sort the entire report and EXTSORT is set ON, the external sort utility will be invoked.

### ***Syntax:* How to Control External Sorting**

You can turn the external sorting feature on and off using the SET EXTSORT command.

```
SET EXTSORT = {ON|OFF}
```

where:

**ON**

Enables the selective use of a dedicated external sorting product to sort reports. This value is the default in all Mainframe environments.

**OFF**

Uses the internal sorting procedure to sort all reports. This value is the default in all non-Mainframe environments.

### ***Syntax:* How to Query the Sort Type**

To determine which sort is being used for a given report, issue the following command after the report request:

```
? STAT
```

The command displays the following values for the SORT USED parameter:

**FOCUS**

The internal sorting procedure was used to sort the entire report.

#### SQL

You are using a relational data source and the RDBMS supplied data already in order.

#### EXTERNAL

An external sorting product sorted the report.

#### NONE

The report did not require sorting.

### Providing an Estimate of Input Records or Report Size for Sorting

There are two advantages to providing an estimate for the input size (ESTRECORDS) or the report size (ESTLINES):

- ☐ If the request cannot be converted to a TABLEF request and the file size estimate shows that the external sort will be needed, FOCUS initiates the external sort immediately, which makes a FOCUS merge unnecessary. Without the estimate, such a request always performs this merge.
- ☐ In Mainframe environments, FOCUS passes the file size to the external sort, which enables it to allocate work files of the appropriate size.

#### **Syntax:** How to Provide an Estimate of Input Records or Report Size for Sorting

```
ON TABLE SET ESTRECORDS nON TABLE SET ESTLINES n
```

where:

*n*

Is the estimated number of records or lines to be sorted.

### Sort Work Files and Return Codes

In non-Mainframe environments, external sorts use temporary work files to hold intermediate sorting results. For each type of external sort, you must be aware of how sort work files are created and used.

#### **Reference:** Sort Work Files on UNIX, Windows, and OpenVMS

While internal sorting uses only one work file, FOCUSORT (allocated in the EDATEMP directory), external sort allows up to 31 work files, allocated on one or more disk drives (spindles) or directories.

**Warning:** Any one or more of these work files may become very large. Count on using many times the total disk space required by FOCSSORT.

By default, five work files are allocated in the /tmp directory on UNIX, or in the directory pointed to by the TMP environment variable in Windows. This may not be enough sort work space and, even if the files fit in the directory, five files are probably not enough for optimal performance. Also, having all of the sort work files on the same disk may further degrade performance.

You have two other options:

- ❑ Define the TMPDIR shell variable (UNIX) or TMP environment variable (Windows) to point to some suitable writable directory. For best results, this directory should be on a disk with a lot of available space, and not the same disk as the data source or the EDATEMP directory. Again, you will get five temporary work files allocated on the same spindle, with consequent performance degradation.
- ❑ Define 1 to 31 shell variables of the form IBITMPDIR01 ... IBITMPDIR31 to point to one or more writable directories.

If the UNIX TMPDIR or Windows TMP variable is set, it must be "unset" in order to make use of the IBITMPDIR $nn$  variables. The UNIX command for unsetting the TMPDIR variable is:

```
unset TMPDIR
```

The Windows command for unsetting the TMP variable is:

```
SET TMP=
```

Different variables may point to the same directory, if desired. If you wish to allocate  $n$  work files, you must define variables 01 through  $n$ . The first variable missing from the environment determines the number of work files that will be used. (If you define fewer than five, additional files will be allocated using the system default location to make up the difference.) The more work files you allocate, and the more separated they are across different spindles, the better performance you should achieve. The major constraint is the total disk space available.

The work file names are generated by the ANSI tempnam function, however, the names all begin with the characters *srtwk*. If the sorting process ends normally or terminates because of a detectable error (typically, disk space overflow), all of the allocated work files are deleted. There is no explicit way to save them. If there is another type of abnormal termination, *srtwk* files may be left on the disk. You can and should erase them.

**Reference: Sort Work Files on IBM i**

On IBM i (formerly i5/OS), the number of work files is fixed at 9. They are virtual files.

**Reference: WebFOCUS External Sort Return Codes**

The WebFOCUS error message FOC909 is issued for all errors from external sort. An additional three-digit code is supplied, of which the last two digits are of interest. If you get an error number ending in:

- ☐ 16, external sort did not have enough memory allocated. You can try reducing the number of work files.
- ☐ 20, an I/O error occurred; in most cases, this means that one of the disks is not writable or has overflowed. Allocate the work files differently or reduce their number.
- ☐ 28, one of the work files could not be opened. Check to make sure the pathname was specified correctly and that protections allow writing and reading.
- ☐ 32, an internal logical error was detected in the sort processing. Report this problem to Information Builders.

**Mainframe External Sort Utilities and Message Options**

By default, error messages created by a Mainframe external sort product are not displayed. However, you may wish to display these messages on your screen for diagnostic purposes.

**Procedure: How to Select a Sort Utility and Message Options**

You use the SET SORTLIB command to both specify the sort utility used at your site and, for DFSORT and SYNCSORT on z/OS, to display sort messages.

1. Issue the SET SORTLIB command to specify the sort utility being used:

```
SET SORTLIB = {sortutility|DEFAULT}
```

where:

*sortutility*

Can be one of the following:

- ☐ **DFSORT** for DFSORT without messages.
- ☐ **MVSMSGDF** for DFSORT with messages.
- ☐ **SYNCSORT** for SyncSort without messages.

- ❑ **MVMSGSS** for SyncSort with standard messages.
  - ❑ **MVMSGSD** for SyncSort with debug (verbose) messages.
  - ❑ **DEFAULT** for DFSORT. However, It is more efficient and highly recommended that you explicitly specify the sort utility using one of the other values.
2. If you specified a sort option that produces sort messages on z/OS, you must direct the sort messages to the batch output stream or a file.

Allocate DDNAME SYSOUT to the batch output stream or a file on z/OS by inserting the appropriate following DD card into your server batch JCL, if it is not already there. For example, the following DD card allocates DDNAME SYSOUT to the batch output stream:

```
//SYSOUT DD SYSOUT=*
```

## Diagnosing External Sort Errors

When an external sort generates an error, you can generate a trace of sort processing and examine the FOCUS return codes and messages to diagnose the problem.

### *Procedure:* How to Trace Sort Processing

When an external sort problem occurs, one of the following messages is generated:

```
(FOC909) CRITICAL ERROR IN EXTERNAL SORT. RETURN CODE IS: xxxx
(FOC1810) External sort not found
(FOC1899) Load of %1 (external-sort module) under %2 failed
```

In response to these messages, as well as for any other problem with sorting, it is useful to trace sort processing. For information on diagnosing external sort problems, see [Diagnosing External Sort Errors](#) on page 195.

1. Allocate DDNAME FSTRACE to the terminal or a file. The following example sends trace output to the terminal:

```
//FSTRACE DD SYSOUT=*,DCB=(RECFM=FA,LRECL=133,BLKSIZE=133)
```

2. Activate the trace by adding the following commands in any supported profile or a FOCEXEC:

```
SET TRACEUSER = ON
SET TRACEON = SORT/1/FSTRACE
```

### *Reference:* External Sort Messages and Return Codes

When you receive a FOC909 message, it includes a return code:

```
(FOC909) CRITICAL ERROR IN EXTERNAL SORT. RETURN CODE IS: xxxx
```

You may also receive one of the following messages:

```
(FOC1810) External sort not found
(FOC1899) Load of %1 (external-sort module) under %2 failed
```

The following notes apply when this message or a FOC1800 or FOC1899 message is generated by a TABLE request:

- ❑ The most common value for `xxxx` is 16. However, return code 16 is issued for a number of problems, including but not limited to the following:
  - ❑ Syntax errors.
  - ❑ Memory shortage.
  - ❑ I/O errors (depending on installation options).
  - ❑ Space problems with output.
  - ❑ Space problems with work files.

In order to diagnose the error, you must generate external sort messages (using the instructions in [How to Select a Sort Utility and Message Options](#) on page 194 and [How to Trace Sort Processing](#) on page 195) and then reproduce the failure.

For return codes not described below, follow the same procedure described for return code 16.

- ❑ Return code 20 is issued by DFSORT under z/OS if messages were requested (using the MVSMSGDJ option of the SET SORTLIB command), but the SYSOUT DD card is missing. DFSORT terminates after issuing the return code. Under the same conditions, SyncSort attempts to open SYSOUT, producing the following message, and then continues with messages written to the operator or terminal:

```
IEC130I SYSOUT DD STATEMENT MISSING.
```

- ❑ Return code 36 or a FOC1899 message under z/OS means that the external sort module could not be found; check the STEPLIBs allocated.

When REBUILD INDEX invokes an external sort that fails, it generates a message similar to the following:

```
ERROR OCCURRED IN THE SORT yyyyyyyyzzzzzzzz
```

In this case, the return code is yyyyyyyy and it is expressed in hex. The final eight digits (zzzzzzzz) should be ignored.



Translate the return code into decimal and follow the instructions for return codes in a TABLE request.

Note also that when a TABLE request generates a non-zero return code from an external sort, FOCUS is terminated. By contrast, when REBUILD INDEX gets a non-zero return code from an external sort, the REBUILD command is terminated but FOCUS continues.

**Reference: Responding to an Indication of Inadequate Sort Work Space**

Before following these instructions, make sure that external sort messages were generated (for information, see [How to Select a Sort Utility and Message Options](#) on page 194) and that they clearly show that the reason for failure was inadequate sort work space.

1. Make an estimate of the number of lines of output the request will produce.
2. Set the ESTLINES parameter in the request or FOCEXEC. For information, see [Providing an Estimate of Input Records or Report Size for Sorting](#) on page 192.

WebFOCUS will pass this estimate to the external sort utility through the parameter list.

Do not override the DD cards for SORTWKnn, S001WKnn, DFSPARM, or \$SORTPARM without direct instructions from technical support. The instructions in [How to Select a Sort Utility and Message Options](#) on page 194, [How to Trace Sort Processing](#) on page 195, and [Providing an Estimate of Input Records or Report Size for Sorting](#) on page 192 should provide equivalent capabilities.

## Aggregation by External Sort (Mainframe Environments Only)

External sorts can be used to perform aggregation with a significant decrease in processing time in comparison to using the internal sort facility. The gains are most notable with relatively simple requests against large data sources.

When aggregation is performed by an external sort, the statistical variables &RECORDS and &LINES are equal because the external sort products do not return a line count for the answer set. This is a behavior change, and affects any code that checks the value of &LINES. (If you must test &LINES, do not use this feature.)

**Syntax: How to Use Aggregation in Your External Sort**

```
SET EXTAGGR = aggropt
```

where:

*aggropt*

Can be one of the following:

`OFF` disallows aggregation by an external sort.

`NOFLOAT` allows aggregation if there are no floating point data fields present.

`ON` allows aggregation by an external sort. This value is the default.

### **Reference:** Usage Notes for Aggregating With an External Sort

- ☐ You must use SyncSort or DFSORT.
- ☐ Your query should be simple (that is, it should be able to take advantage of the TABLEF facility). For related information, see [Data Retrieval Using TABLEF](#) on page 1845.
- ☐ The PRINT display command may not be used in the query.
- ☐ SET ALL must be equal to OFF.
- ☐ Only the following column prefixes are allowed: SUM, AVG, CNT, FST.
- ☐ Columns can be calculated values or have a row total.
- ☐ When SET EXTAGGR = NOFLOAT and your query aggregates numeric data, the external sort is not called, and aggregation is performed through the internal sorting procedure.

### **Example:** Changing Output by Using an External Sort for Aggregation

If you use SUM on an alphanumeric field in your report request without using an external sort, the last instance of the sorted fields is displayed in the output, by default. Turning on aggregation in the external sort displays the first record instead. However, you can control the order of display using the SUMPREFIX parameter. For information about the SUMPREFIX parameter, see [Changing Retrieval Order With Aggregation](#) on page 199.

The following command turns aggregation ON and leaves SUMPREFIX set to LST (the default) and, therefore, displays the last record:

```
SET EXTAGGR = ON
SET SUMPREFIX = LST
TABLE FILE CAR
SUM CAR BY COUNTRY
END
```

The output is:

COUNTRY	CAR
-----	---
ENGLAND	TRIUMPH
FRANCE	PEUGEOT
ITALY	MASERATI
JAPAN	TOYOTA
W GERMANY	BMW

**Note:** SUMPREFIX is described in [Changing Retrieval Order With Aggregation](#) on page 199.

With SUMPREFIX = FST, the output is:

COUNTRY	CAR
-----	---
ENGLAND	JAGUAR
FRANCE	PEUGEOT
ITALY	ALFA ROMEO
JAPAN	DATSUN
W GERMANY	AUDI

## Changing Retrieval Order With Aggregation

The SUMPREFIX parameter allows you to specify which value will be displayed when aggregating an alphanumeric or smart date field in the absence of any prefix operator. The default value is LST, which will return the physical last value within the sort group. FST will return the first physical value in the sort group. MIN and MAX return either the minimum value or maximum value within the sort group.

The SUMPREFIX command allows users to choose the answer set display order.

### **Syntax:** How to Set Retrieval Order

```
SET SUMPREFIX = {FST|LST|MIN|MAX}
```

where:

#### FST

Displays the first value when alphanumeric or smart date data types are aggregated.

#### LST

Displays the last value when alphanumeric or smart date data types are aggregated. LST is the default value.

#### MIN

Displays the minimum value in the sort order set by your server code page and configuration when alphanumeric or smart date data types are aggregated.

#### MAX

Displays the maximum value in the sort order set by your server code page and configuration when alphanumeric or smart date data types are aggregated.

**Example: Displaying the Minimum Value for an Aggregated Alphanumeric Field**

The following request sets SUMPREFIX to MIN and displays the aggregated PRODUCT\_CATEGORY and DAYSDELAYED values as well as the minimum, maximum, first, and last PRODUCT\_CATEGORY values. In each row, the aggregated PRODUCT\_CATEGORY value matches the MIN.PRODUCT\_CATEGORY value. The DAYSDELAYED numeric field is not affected by the SUMPREFIX value and is aggregated.

```
SET SUMPREFIX = MIN
TABLE FILE wf_retail_lite
SUM PRODUCT_CATEGORY DAYSDELAYED MIN.PRODUCT_CATEGORY MAX.PRODUCT_CATEGORY
  FST.PRODUCT_CATEGORY LST.PRODUCT_CATEGORY
BY BRAND
WHERE BRAND GT 'K' AND BRAND LT 'U'
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

The output is shown in the following image.

Brand	Product Category	Days Delayed	MIN Product Category	MAX Product Category	FST Product Category	LST Product Category
LG	Media Player	339	Media Player	Televisions	Media Player	Televisions
Logitech	Accessories	114	Accessories	Accessories	Accessories	Accessories
Niles Audio	Accessories	150	Accessories	Accessories	Accessories	Accessories
Onkyo	Stereo Systems	140	Stereo Systems	Stereo Systems	Stereo Systems	Stereo Systems
Panasonic	Camcorder	422	Camcorder	Televisions	Camcorder	Televisions
Philips	Stereo Systems	229	Stereo Systems	Stereo Systems	Stereo Systems	Stereo Systems
Pioneer	Accessories	339	Accessories	Stereo Systems	Accessories	Stereo Systems
Polk Audio	Stereo Systems	93	Stereo Systems	Stereo Systems	Stereo Systems	Stereo Systems
Roku	Media Player	85	Media Player	Media Player	Media Player	Media Player
Samsung	Accessories	525	Accessories	Stereo Systems	Accessories	Stereo Systems
Sanyo	Camcorder	298	Camcorder	Stereo Systems	Camcorder	Stereo Systems
Sennheiser	Accessories	128	Accessories	Accessories	Accessories	Accessories
Sharp	Media Player	252	Media Player	Stereo Systems	Media Player	Stereo Systems
Sony	Accessories	1,100	Accessories	Televisions	Accessories	Televisions
Thomson Grass Valley	Video	166	Video	Video	Video	Video
Toshiba	Production		Production	Production	Production	Production
	Media Player	7	Media Player	Media Player	Media Player	Media Player

**Creating a HOLD File With an External Sort (Mainframe Environments Only)**

You can use Mainframe external sort packages to create HOLD files, producing substantial savings in processing time. The gains are most notable with relatively simple requests against large data sources.

**Syntax: How to Create HOLD Files With an External Sort**

```
SET EXTHOLD = {OFF|ON}
```

where:

**OFF**

Disables HOLD files by an external sort.

**ON**

Enables HOLD files by an external sort. This value is the default.

**Reference: Usage Notes for Creating a HOLD File With an External Sort**

- ☐ The default setting of EXTSORT=ON must be in effect.
- ☐ EXTHOLD must be ON.
- ☐ The request must contain a BY field.
- ☐ The type of HOLD file created must be a FOCUS, XFOCUS, ALPHA, or BINARY file.
- ☐ Your query should be simple. AUTOTABLEF analyzes a query and determines whether the combination of display commands and formatting options requires the internal matrix. In cases where it is determined that a matrix is not necessary to satisfy the query, you may avoid the extra internal costs associated with creating the matrix. The internal matrix is stored in a file or data set named FOCSORT. The AUTOTABLEF default is ON, in order to realize performance gains.
- ☐ SET ALL must be OFF.
- ☐ There cannot be an IF/WHERE TOTAL or BY TOTAL in the request.
- ☐ If a request contains a SUM command, EXTAGGR must be set ON, and the only column prefixes allowed are SUM. and FST.

**Hierarchical Reporting: BY HIERARCHY**

Cube data sources such as Essbase or SAP BW are organized into dimensions and facts. Dimensions are often organized into hierarchies. The synonyms for cube data sources have attributes that describe the dimension hierarchies, and WebFOCUS has hierarchical reporting syntax that can automatically report against these hierarchies and display the results indented to show the hierarchical relationships.

WebFOCUS also supports defining dimension hierarchies in synonyms for non-cube data sources that have hierarchical data. Once hierarchical dimensions are defined in a synonym, you can issue hierarchical reporting requests against them. Non-cube synonyms with hierarchical attributes are called *virtual cubes*.

Dimensions are categories of data, such as Region or Time, that you use to analyze and compare business performance. Dimensions consist of data elements that are called members. For example, a Region dimension could have members England and France.

Dimension members are usually organized into hierarchies. Hierarchies can be viewed as tree-like structures where members are the nodes. For example, the Region dimension may have the element World at its top level (the root node). The World element may have children nodes (members) representing continents. Continents, in turn, can have children nodes that represent countries, and countries can have children nodes representing states or cities. Nodes with no children are called leaf nodes.

Measures are numeric values, such as Sales Volume or Net Income, that are used to quantify how your business is performing.

A cube consists of data derived from facts, which are records about individual business transactions. For example, an individual fact record reflects a sales transaction of a certain number of items of a certain product at a certain price, which occurred in a certain store at a certain moment in time. The cube contains summarized fact values for all combinations of measures and members of different dimensions.

A synonym describes a hierarchy using a set of fields that define the hierarchical structure and the relationships between the hierarchy members. WebFOCUS has special hierarchical reporting syntax for reporting on hierarchies.

Hierarchical reporting requests have several phases:

**❑ Phase 1, selecting hierarchy members to display.**

The hierarchical reporting phrase BY or ON HIERARCHY automatically sorts and formats a hierarchy with appropriate indentations that show the parent/child relationships. It also automatically rolls up the measure values for child members to generate the measure values for the parent members.

If you do not want to see the entire hierarchy, you can use the WHEN phrase to select hierarchy members for display. The expression in this WHEN phrase must reference only hierarchy fields, not dimension properties or measures.

**❑ Phase 2, screening the retrieved dimension data.**

WHERE criteria are applied to the leaf nodes of the members selected during phase 1. Therefore, dimension properties can be used in WHERE tests. These tests can also reference hierarchy fields. However, since the selection criteria are always applied to the values at the leaf nodes, they cannot select data based on values that occur at higher levels. For example, in a dimension with Continents, Countries, and Cities, your request will not display any rows if you use WHERE to select at the Country level, but it may if you use it to select at the City level. WHERE tests can also reference measures.

#### ❑ **Phase 3, screening based on aggregated values.**

Measures, being summarized values, can be referenced in WHERE TOTAL tests and COMPUTE commands because those commands are processed after the hierarchy selection and aggregation phases of the request.

### **Syntax:** How to Specify a Hierarchy in a Master File

The data source must have at least one dimension that is organized hierarchically. The declaration for a dimension is:

```
DIMENSION=dimname,CAPTION=dimcaption, $
```

where:

*dimname*

Is a name for the dimension.

*dimcaption*

Is a label for the dimension.

The declaration for a hierarchy within the dimension is:

```
HIERARCHY=hname,CAPTION=' hcaption ',HRY_DIMENSION=dimname,  
HRY_STRUCTURE=RECURSIVE, $
```

where:

*hname*

Is a name for the hierarchy.

*hcaption*

Is a label for the hierarchy.

*dimname*

Is the name of the dimension for which this hierarchy is defined.

Several fields are used to define a parent/child hierarchy. Each has a PROPERTY attribute that describes which hierarchy property it represents. Each hierarchy must have a unique identifier field. This field is called the hierarchy field. If the synonym represents a FOCUS data source, this field must be indexed (FIELDTYPE=I). The declaration for the hierarchy field is:

```
FIELD=hfield,ALIAS=halias,USAGE= An, [ACTUAL=Am,]  
WITHIN='*hierarchy',PROPERTY=UID, [TITLE='title1',] [FIELDTYPE=I,] $
```

where:

*hfield*

Is the field name for the hierarchy field.

*halias*

Is the alias for the hierarchy field. If the data source is relational, this must be the name of the column in the Relational DBMS.

*hierarchy*

Is the name of the hierarchy to which this field belongs.

```
USAGE= An, [ACTUAL=Am,]
```

Are the USAGE format and, if the data source is not a FOCUS data source, the ACTUAL format of the field.

*title1*

Is an optional title for the field.

Other fields defined for the hierarchy include the parent field and the caption field. Each of these fields has the same name as the hierarchy field with a suffix added. Each has a PROPERTY attribute that specifies its role in the hierarchy and a REFERENCE attribute that points to the corresponding hierarchy field.

The following is the declaration for the parent field. The parent field is needed to define the parent/child relationships in the hierarchy:

```
FIELD=hfield_PARENT,ALIAS=parentalias,USAGE=An,[ACTUAL=Am,] [TITLE=ptitle,]  
PROPERTY=PARENT_OF, REFERENCE=hfield, $
```

where:

*hfield*

Is the hierarchy field.



*parentalias*

Is the alias for the parent field. If the data source is relational, this must be the name of the column in the relational DBMS.

USAGE= *An*, [ACTUAL=*Am*,]

Are the USAGE format and, if the data source is not a FOCUS data source, the ACTUAL format of the field.

*ptitle*

Is a column title for the parent field.

The following is the declaration for the caption field. A caption is a descriptive title for each value of the hierarchy field. It is part of the data and, therefore, is different from a TITLE attribute in the Master File, which is a literal title for the column on the report output.

```
FIELD=hfield_CAPTION,ALIAS=capalias,USAGE=Ann, [ACTUAL=Amm, ]
[TITLE=capttitle,]
      PROPERTY=CAPTION, REFERENCE=hfield, $
```

where:

*hfield*

Is the hierarchy field.

*capalias*

Is the alias for the caption field. If the data source is relational, this must be the name of the column in the relational DBMS.

USAGE= *Ann*, [ACTUAL=*Amm*,]

Are the USAGE format and, if the data source is not a FOCUS data source, the ACTUAL format of the field.

*capttitle*

Is a column title for the caption field.

**Example: Sample Master File With a Dimension Hierarchy**

The following Master File is based on the CENTGL Master File, which has an FML hierarchy defined. This version is named NEWGL and it has a dimension hierarchy of accounts in which GL\_ACCOUNT is the hierarchy field, GL\_ACCOUNT\_PARENT is the parent field, and GL\_ACCOUNT\_CAPTION is the caption field. There are other fields based on the hierarchy (GL\_ACCOUNT\_LEVEL, GL\_ROLLUP\_OP, and GL\_ACCOUNT\_TYPE). In addition, there is a measure field (GL\_ACCOUNT\_AMOUNT):

```
FILE=NEWGL          , SUFFIX=FOC, $
SEGNAME=ACCOUNTS    , SEGTYPE=S01
DIMENSION=Accnt, CAPTION=Accnt, $
HIERARCHY=Accnt, CAPTION='Accnt', HRY_DIMENSION=Accnt,
HRV_STRUCTURE=RECURSIVE, $
FIELD=GL_ACCOUNT, GLACCT, A7, WITHIN='*Accnt', PROPERTY=UID,
        TITLE='Ledger, Account', FIELDTYPE=I, $
FIELD=GL_ACCOUNT_PARENT, GLPAR, A7, TITLE=Parent,
        PROPERTY=PARENT_OF, REFERENCE=GL_ACCOUNT, $
FIELD=GL_ACCOUNT_TYPE, GLTYPE, A1, TITLE=Type, $
FIELD=GL_ROLLUP_OP, ROLL, A1, TITLE=Op, $
FIELD=GL_ACCOUNT_LEVEL, GLLEVEL, I3, TITLE=Lev, $
FIELDNAME=GL_ACCOUNT_AMOUNT, GLAMT, D12.2, TITLE=Amount, $
FIELD=GL_ACCOUNT_CAPTION, GLCAP, A30, TITLE=Caption,
        PROPERTY=CAPTION, REFERENCE=GL_ACCOUNT, $
FIELD=SYS_ACCOUNT, ALINE, A6, TITLE='System, Account, Line', MISSING=ON, $
```

The following procedure loads data into this data source, as long as the Master File is available to WebFOCUS (on the path or allocated):

```
CREATE FILE NEWGL NOMSG
-RUN
MODIFY FILE NEWGL
COMPUTE TGL_ACCOUNT_LEVEL/A3=;
COMPUTE TGL_ACCOUNT_AMOUNT/A12=;
FIXFORM GL_ACCOUNT/A4B X3 GL_ACCOUNT_PARENT/A4B X3 GL_ACCOUNT_TYPE/A1B
FIXFORM SYS_ACCOUNT/A4B GL_ROLLUP_OP/A1B
FIXFORM TGL_ACCOUNT_LEVEL/A3B GL_ACCOUNT_CAPTION/A30B
FIXFORM TGL_ACCOUNT_AMOUNT/A12B
COMPUTE GL_ACCOUNT_LEVEL = EDIT(TGL_ACCOUNT_LEVEL);
COMPUTE GL_ACCOUNT_AMOUNT = ATODBL(TGL_ACCOUNT_AMOUNT , '12',
GL_ACCOUNT_AMOUNT);

MATCH GL_ACCOUNT
ON MATCH REJECT
ON NOMATCH INCLUDE
```

DATA					
1000		R.	+	1Profit Before Tax	
2000	1000	R.	+	2Gross Margin	
2100	2000	R.	+	3Sales Revenue	
2200	2100	R.	+	4Retail Sales	
2210	2200	R7001+		5Retail - Television	505.00
2220	2200	R7002+		5Retail - Stereo	505.00
2230	2200	R7003+		5Retail - Video Player	505.00
2240	2200	R7004+		5Retail - Computer	505.00
2250	2200	R7005+		5Retail - Video Camera	505.00
2300	2100	R.	+	4Mail Order Sales	
2310	2300	R7011+		5Mail Order - Television	505.00
2320	2300	R7012+		5Mail Order - Stereo	505.00
2330	2300	R7013+		5Mail Order - Video Player	505.00
2340	2300	R7014+		5Mail Order - Computer	505.00
2350	2300	R7015+		5Mail Order - Video Camera	505.00
2400	2100	R.	+	4Internet Sales	
2410	2400	R7021+		5Internet - Television	505.00
2420	2400	R7022+		5Internet - Stereo	505.00
2430	2400	R7023+		5Internet - Video Player	505.00
2440	2400	R7024+		5Internet - Computer	505.00
2450	2400	R7025+		5Internet - Video Camera	505.00
2500	2000	E.	-	3Cost Of Goods Sold	
2600	2500	E.	+	4Variable Material Costs	
2610	2600	E7101+		5Television COGS	505.00
2620	2600	E7102+		5Stereo COGS	505.00
2630	2600	E7103+		5Video COGS	505.00
2640	2600	E7104+		5Computer COGS	505.00
2650	2600	E7105+		5Video Camera COGS	505.00
2700	2500	E7111+		4Direct Labor	404.00
2800	2500	E7112+		4Fixed Costs	404.00
3000	1000	E.	-	2Total Operating Expenses	
3100	3000	E.	+	3Selling Expenses	
3110	3100	E.	+	4Advertising	
3112	3110	E7202+		5TV/Radio	505.00
3114	3110	E7203+		5Print Media	505.00
3116	3110	E7206+		5Internet Advertising	505.00
3120	3100	E7212+		4Promotional Expenses	404.00
3130	3100	E7213+		4Joint Marketing	404.00
3140	3100	E7214+		4Bonuses/Commissions	404.00
3200	3000	E.	+	3General + Admin Expenses	
3300	3200	E.	+	4Salaries-Corporate	

3310	3300	E7301+	5Salaries-Corp Mgmt	505.00
3320	3300	E7302+	5Salaries-Administration	505.00
3330	3300	E7303+	5IT Contractors	505.00
3400	3200	E. +	4Company Benefits	
3410	3400	E7311+	5Social Security	505.00
3420	3400	E7312+	5Unemployment	505.00
3430	3400	E7313+	5Vacation Pay	505.00
3440	3400	E7314+	5Sick Pay	505.00
3450	3400	E. +	5Insurances	
3451	3450	E7321+	6Medical Insurance	606.00
3452	3450	E7322+	6Dental Insurance	606.00
3453	3450	E7323+	6Pharmacy Insurance	606.00
3454	3450	E7324+	6Disability Insurance	606.00
3455	3450	E7325+	6Life Insurance	606.00
3500	3200	E. +	4Depreciation Expenses	
3510	3500	E7411+	5Equipment	505.00
3520	3500	E7412+	5Building	505.00
3530	3500	E7413+	5Vehicles	505.00
3600	3200	R7414-	4Gain/(Loss) Sale of Equipment	404.00
3700	3200	E. +	4Leasehold Expenses	
3710	3700	E7421+	5Equipment	505.00
3720	3700	E7422+	5Buildings	505.00
3730	3700	R7429-	5Sub-Lease Income	505.00
3800	3200	E7440+	4Interest Expenses	404.00
3900	3200	E. +	4Utilities	
3910	3900	E7451+	5Electric	505.00
3920	3900	E7452+	5Gas	505.00
3930	3900	E7453+	5Telephone	505.00
3940	3900	E7454+	5Water	505.00
3950	3900	E7455+	5Internet Access	505.00
5000	1000	E. -	2Total R+D Costs	
5100	5000	E7511+	3Salaries	303.00
5200	5000	E7521+	3Misc. Equipment	303.00
END				

## Syntax: How to Report on a Hierarchy

In hierarchical reporting, measure values for child dimension members will be rolled up to generate the parent values. In the data source, the parent members should not have values for the measures.

```
SUM measure_field ...
BY hierarchy_field [HIERARCHY [WHEN expression_using_hierarchy_fields;]
[SHOW [TOP|UP n] [TO {BOTTOM|DOWN m}] [byoption [WHEN condition] ...] ]
[WHERE expression_using_dimension_data]
[ON hierarchy_field HIERARCHY [WHEN expression_using_hierarchy_fields;]
[SHOW [TOP|UP n] [TO BOTTOM|DOWN m] [byoption [WHEN condition] ...]]
```

where:

*measure\_field*

Is the field name of a measure.

**BY *hierarchy\_field* HIERARCHY**

Identifies the hierarchy used for sorting. The field must be a hierarchy field.

**ON *hierarchy\_field* HIERARCHY**

Identifies the hierarchy used for sorting. The field must be a hierarchy field. The request must include either a BY phrase or a BY HIERARCHY phrase for this field name.

**WHEN *expression\_using\_hierarchy\_fields*;**

Selects hierarchy members. The WHEN phrase must immediately follow the word HIERARCHY to distinguish it from a WHEN phrase associated with a BY option (such as SUBFOOT). Any expression using only hierarchy fields is supported. The WHEN phrase can be on the BY HIERARCHY command or the ON HIERARCHY command, but not both.

**SHOW**

Specifies which levels to show on the report output relative to the levels selected by the WHEN phrase. If there is no WHEN phrase, the SHOW option is applied to the root node of the hierarchy. The SHOW option can be specified on the BY HIERARCHY phrase or the ON HIERARCHY phrase, but not both.

***n***

Is the number of ascendants above the set of selected members that will have measure values. All ascendants appear on the report to show the hierarchical context of the selected members. However, ascendants that are not included in the SHOW phrase appear on the report with missing data symbols in the report columns that display measures. The default for *n* is 0.

**TOP**

Specifies that ascendant levels to the root node of the hierarchy will be populated with measure values.

**TO**

Is required when specifying a SHOW option for descendant levels.

**BOTTOM**

Specifies all descendants to the leaf nodes of the hierarchy will be populated with measure values. This is the default value.

***m***

Is the number of descendants of each selected level that will display. The default for *m* is BOTTOM, which displays all descendants.

### *byoption*

Is one of the following sort-based options: PAGE-BREAK, REPAGE, RECAP, RECOMPUTE, SKIP-LINE, SUBFOOT, SUBHEAD, SUBTOTAL, SUB-TOTAL, SUMMARIZE, UNDER-LINE. If you specify SUBHEAD or SUBFOOT, you must place the WHEN phrase on the line following the heading or footing text.

### *condition*

Is a logical expression.

### *expression\_using\_dimension\_data*

Screens the rows selected in the BY/ON HIERARCHY and WHEN phrases based on dimension data. The expression can use dimension properties and hierarchy fields. However, the selection criteria are always applied to the values at the leaf nodes. Therefore, you cannot use WHERE to select rows based on hierarchy field values that occur at higher levels. For example, in a dimension with Continents, Countries, and Cities, your request will not display any rows if you use WHERE to select a Country name, but it may if you use it to select a City name.

### **Example:** Reporting on a Dimension Hierarchy

The following request reports on the entire GL\_ACCOUNT hierarchy for the CENTGL2 data source created in the *Describing Data With WebFOCUS Language* manual.

```
TABLE FILE NEWGL
SUM GL_ACCOUNT_AMOUNT
BY GL_ACCOUNT HIERARCHY
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
TYPE=REPORT,GRID=OFF,$
ENDSTYLE
END
```

Partial output is shown in the following image. The accounts are indented to show the hierarchical relationships:

Ledger

<u>Account</u>	<u>Amount</u>
1000	27,169.00
2000	10,908.00
2100	7,575.00
2200	2,525.00
2210	505.00
2220	505.00
2230	505.00
2240	505.00
2250	505.00
2300	2,525.00
2310	505.00
2320	505.00
2330	505.00
2340	505.00
2350	505.00
2400	2,525.00
2410	505.00
2420	505.00
2430	505.00
2440	505.00
2450	505.00
2500	3,333.00
2600	2,525.00
2610	505.00
2620	505.00
2630	505.00
2640	505.00
2650	505.00

The following is the same request using the GL\_ACCOUNT\_CAPTION field:

```
TABLE FILE NEWGL
SUM GL_ACCOUNT_AMOUNT
BY GL_ACCOUNT_CAPTION HIERARCHY
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
TYPE=REPORT,GRID=OFF,$
ENDSTYLE
END
```



Partial output is shown in the following image:

<u>Caption</u>	<u>Amount</u>
Profit Before Tax	27,169.00
Gross Margin	10,908.00
Sales Revenue	7,575.00
Retail Sales	2,525.00
Retail - Television	505.00
Retail - Stereo	505.00
Retail - Video Player	505.00
Retail - Computer	505.00
Retail - Video Camera	505.00
Mail Order Sales	2,525.00
Mail Order - Television	505.00
Mail Order - Stereo	505.00
Mail Order - Video Player	505.00
Mail Order - Computer	505.00
Mail Order - Video Camera	505.00
Internet Sales	2,525.00
Internet - Television	505.00
Internet - Stereo	505.00
Internet - Video Player	505.00
Internet - Computer	505.00
Internet - Video Camera	505.00
Cost Of Goods Sold	3,333.00
Variable Material Costs	2,525.00
Television COGS	505.00
Stereo COGS	505.00
Video COGS	505.00
Computer COGS	505.00
Video Camera COGS	505.00

**Example:**    **Using WHEN to Select Hierarchy Members**

The following request selects certain accounts using the WHEN phrase and populates one level up and one level down from the selected nodes with values. Note that all levels to the root node display on the output for context, but if they are not in the members selected, they are not populated with measure values:

```
TABLE FILE NEWGL
SUM GL_ACCOUNT_AMOUNT
BY GL_ACCOUNT_CAPTION HIERARCHY
WHEN GL_ACCOUNT GT '2000' AND GL_ACCOUNT LT '3000';
  SHOW UP 1 TO DOWN 1
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
TYPE=REPORT,GRID=OFF,$
ENDSTYLE
END
```

The output is shown in the following image:

<u>Caption</u>	<u>Amount</u>
Profit Before Tax	.
Gross Margin	10,908.00
Sales Revenue	7,575.00
Retail Sales	2,525.00
Retail - Television	505.00
Retail - Stereo	505.00
Retail - Video Player	505.00
Retail - Computer	505.00
Retail - Video Camera	505.00
Mail Order Sales	2,525.00
Mail Order - Television	505.00
Mail Order - Stereo	505.00
Mail Order - Video Player	505.00
Mail Order - Computer	505.00
Mail Order - Video Camera	505.00
Internet Sales	2,525.00
Internet - Television	505.00
Internet - Stereo	505.00
Internet - Video Player	505.00
Internet - Computer	505.00
Internet - Video Camera	505.00
Cost Of Goods Sold	3,333.00
Variable Material Costs	2,525.00
Television COGS	505.00
Stereo COGS	505.00
Video COGS	505.00
Computer COGS	505.00
Video Camera COGS	505.00
Direct Labor	404.00
Fixed Costs	404.00

***Example:***     **Using WHERE to Screen Selected Hierarchy Members**

The following request selects members using the WHEN phrase and then screens the output by applying a WHERE phrase to the selected members:

```
TABLE FILE NEWGL
SUM GL_ACCOUNT_AMOUNT GL_ACCOUNT_TYPE
BY GL_ACCOUNT HIERARCHY
WHEN GL_ACCOUNT NE '3000';
  SHOW UP 0 TO DOWN 0
WHERE GL_ACCOUNT_TYPE NE 'E'   ;
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF,$
ENDSTYLE
END
```

The output is shown in the following image:

Ledger

<u>Account</u>	<u>Amount</u>	<u>Type</u>
1000	8,484.00	R
2000	7,575.00	R
2100	7,575.00	R
2200	2,525.00	R
2210	505.00	R
2220	505.00	R
2230	505.00	R
2240	505.00	R
2250	505.00	R
2300	2,525.00	R
2310	505.00	R
2320	505.00	R
2330	505.00	R
2340	505.00	R
2350	505.00	R
2400	2,525.00	R
2410	505.00	R
2420	505.00	R
2430	505.00	R
2440	505.00	R
2450	505.00	R
3000	.	.
3200	909.00	R
3600	404.00	R
3700	505.00	R
3730	505.00	R



## Selecting Records for Your Report

---

When generating a report and selecting fields, you may not want to include every instance of a field. By including selection criteria, you can display only those field values that meet your needs. In effect, you can select a subset of data that you can easily redefine each time you issue the report request.

### In this chapter:

- ☐ [Selecting Records Overview](#)
  - ☐ [Choosing a Filtering Method](#)
  - ☐ [Selections Based on Individual Values](#)
  - ☐ [Selection Based on Aggregate Values](#)
  - ☐ [Applying Selection Criteria to the Internal Matrix Prior to COMPUTE Processing](#)
  - ☐ [Using Compound Expressions for Record Selection](#)
  - ☐ [Using Operators in Record Selection Tests](#)
  - ☐ [Types of Record Selection Tests](#)
  - ☐ [Selections Based on Group Key Values](#)
  - ☐ [Setting Limits on the Number of Records Read](#)
  - ☐ [Selecting Records Using IF Phrases](#)
  - ☐ [Reading Selection Values From a File](#)
  - ☐ [Assigning Screening Conditions to a File](#)
  - ☐ [VSAM Record Selection Efficiencies](#)
- 

### Selecting Records Overview

When developing a report request, you can define criteria that select records based on a variety of factors:

- ☐ The values of an individual field. See [Selections Based on Individual Values](#) on page 220.
- ☐ The aggregate value of a field (for example, the sum or average of field values). See [Selection Based on Aggregate Values](#) on page 228.
- ☐ The existence of missing values for a field, whether field values fall within a range, or whether a field does not contain a certain value. See [Types of Record Selection Tests](#) on page 243.
- ☐ The number of records that exist for a field (for example, the first 50 records), rather than on the field values. See [Setting Limits on the Number of Records Read](#) on page 258.

- ❑ For non-FOCUS data sources that have group keys, you can select records based on group key values. See [Selections Based on Group Key Values](#) on page 257.

In addition, you can take advantage of a variety of record selection efficiencies, including assigning filtering criteria to a data source and reading selection values from a file.

## Choosing a Filtering Method

There are two phrases for selecting records: WHERE and IF. It is recommended that you use WHERE to select records. IF offers a subset of the functionality of WHERE. Everything that you can accomplish with IF, you can also accomplish with WHERE. WHERE can accomplish things that IF cannot.

If you used IF to select records in the past, remember that WHERE and IF are two different phrases, and may require different syntax to achieve the same result.

WHERE syntax is described and illustrated throughout this topic. For details on IF syntax, see [Selecting Records Using IF Phrases](#) on page 259.

## Selections Based on Individual Values

The WHERE phrase selects records from the data source to be included in a report. The data is evaluated according to the selection criteria before it is retrieved from the data source.

You can use as many WHERE phrases as necessary to define your selection criteria. For an illustration, see [Using Multiple WHERE Phrases](#) on page 222. For additional information, see [Using Compound Expressions for Record Selection](#) on page 237.

**Note:** Multiple selection tests on fields that reside on separate paths of a multi-path data source are processed as though connected by either AND or OR operators, based on the setting of a parameter called MULTIPATH. For details, see [Controlling Record Selection in Multi-path Data Sources](#) on page 223.

### **Syntax:** How to Select Records With WHERE

```
WHERE criteria [;]
```

where:

*criteria*

Are the criteria for selecting records to include in the report. The criteria must be defined in a valid expression that evaluates as true or false (that is, a Boolean expression). Expressions are described in detail in [Using Expressions](#) on page 431. Operators that can be used in WHERE expressions (such as, CONTAINS, IS, and GT), are described in [Operators Supported for WHERE and IF Tests](#) on page 238.



;

Is an optional semicolon that can be used to enhance the readability of the request. It does not affect the report.

### **Reference: Usage Notes for WHERE Phrases**

The WHERE phrase can include:

- ☐ Most expressions that would be valid on the right-hand side of a DEFINE expression. However, the logical expression IF ... THEN ... ELSE cannot be used.
- ☐ Real fields, temporary fields, and fields in joined files. If a field name is enclosed in single or double quotation marks, it is treated as a literal string, not a field reference.
- ☐ The operators EQ, NE, GE, GT, LT, LE, CONTAINS, OMITS, FROM ... TO, NOT-FROM ... TO, INCLUDES, EXCLUDES, LIKE, and NOT LIKE.
- ☐ All arithmetic operators (+, -, \*, /, \*\*), as well as, functions (MIN, MAX, ABS, and SQRT).
- ☐ An alphanumeric expression, which can be a literal, or a function yielding an alphanumeric or numeric result using EDIT or DECODE.

Note that files used with DECODE expressions can contain two columns, one for field values and one for numeric decode values.

- ☐ Alphanumeric and date literals enclosed in single quotation marks and date-time literals in the form DT (*date-time literal*).
- ☐ A date literal used in a selection test against a date field cannot contain the day of the week value.
- ☐ Text fields. However, the only operators supported for use with text fields are CONTAINS and OMITS.
- ☐ All functions.

You can build complex selection criteria by joining simple expressions with AND and OR logical operators and, optionally, adding parentheses to specify explicitly the order of evaluation. This is easier than trying to achieve the same effect with the IF phrase, which may require the use of a separate DEFINE command. For details, see [Using Compound Expressions for Record Selection](#) on page 237.

**Example: Using a Simple WHERE Test**

To show only the names and salaries of employees earning more than \$20,000 a year, issue the following request:

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME AND FIRST_NAME AND CURR_SAL
BY LAST_NAME NOPRINT
WHERE CURR_SAL GT 20000
END
```

In this example, CURR\_SAL is a selected field, and CURR\_SAL GT 20000 is the selection criterion. Only those records with a current salary greater than \$20,000 are retrieved. All other records are ignored.

The output is:

<u>LAST_NAME</u>	<u>FIRST_NAME</u>	<u>CURR_SAL</u>
BANNING	JOHN	\$29,700.00
BLACKWOOD	ROSEMARIE	\$21,780.00
CROSS	BARBARA	\$27,062.00
IRVING	JOAN	\$26,862.00
ROMANS	ANTHONY	\$21,120.00

**Example: Using Multiple WHERE Phrases**

You can use as many WHERE phrases as necessary to define your selection criteria. This request uses multiple WHERE phrases so that only those employees in the MIS or Production departments with the last name of Cross or Banning are included in the report.

```
TABLE FILE EMPLOYEE
PRINT EMP_ID LAST_NAME
WHERE SALARY GT 20000
WHERE DEPARTMENT IS 'MIS' OR 'PRODUCTION'
WHERE LAST_NAME IS 'CROSS' OR 'BANNING'
END
```

The output is:

<u>EMP_ID</u>	<u>LAST_NAME</u>
119329144	BANNING
818692173	CROSS

For related information, see [Using Compound Expressions for Record Selection](#) on page 237.

## Controlling Record Selection in Multi-path Data Sources

When you report from a multi-path data source, a parent segment may have children down some paths, but not others. The MULTIPATH parameter allows you to control whether such a parent segment is omitted from the report output.

The MULTIPATH setting also affects the processing of selection tests on independent paths. If MULTIPATH is set to:

- ❑ **COMPOUND**, WHERE or IF tests on separate paths are treated as if they are connected by an AND operator. That is, *all* paths must pass the screening tests in order for the parent to be included in the report output.
- ❑ **SIMPLE**, WHERE or IF tests on separate paths are considered independently, as if an OR operator connected them. Therefore, a parent instance is included in the report if at least one of the paths passes its screening test. A warning message is produced, indicating that if the request contains a test on one path, data is also retrieved from another, independent path. Records on the independent path are retrieved regardless of whether the condition is satisfied on the tested path.

The MULTIPATH settings apply in all types of data sources and in all reporting environments (TABLE, TABLEF, MATCH, GRAPH, and requests with multiple display commands). MULTIPATH also works with alternate views, indexed views, filters, DBA, and joined structures.

### **Syntax:** How to Control Record Selection in Multi-path Data Sources

To set MULTIPATH from the command level or in a stored procedure, use

```
SET MULTIPATH = {SIMPLE|COMPOUND}
```

To set MULTIPATH in a report request, use

```
ON TABLE SET MULTIPATH {SIMPLE|COMPOUND}
```

where:

#### **SIMPLE**

Includes a parent segment in the report output if:

- ❑ It has at least one child that passes its screening conditions.

**Note:** A unique segment is considered a part of its parent segment, and therefore does not invoke independent path processing.

- ❑ It lacks any referenced child on a path, but the child is optional.

The (FOC144) warning message is generated when a request screens data in a multi-path report:

(FOC144) WARNING. TESTING IN INDEPENDENT SETS OF DATA

### COMPOUND

Includes a parent in the report output if it has *all* of its required children. WHERE or IF tests on separate paths are treated as if they are connected by an AND operator. That is, *all* paths must pass the screening tests in order for the parent to be included in the report output. COMPOUND is the default value.

For related information, see [MULTIPATH and SET ALL Combinations](#) on page 226 and [Rules for Determining If a Segment Is Required](#) on page 228.

### **Reference: Requirements and Usage Notes for MULTIPATH = COMPOUND**

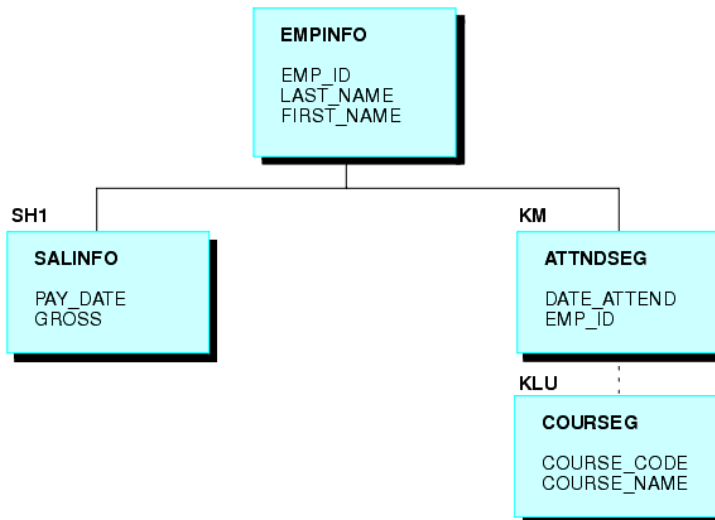
- ❑ The minimum memory requirement for the MULTIPATH = COMPOUND setting is 4K per active segment. If there is insufficient memory, the SIMPLE setting is implemented and a message is returned.

There is no limit to the number of segment instances (rows). However, no single segment instance can have more than 4K of active fields (referenced fields or fields needed for retrieving referenced fields). If this limit is exceeded, the SIMPLE setting is implemented and a message is returned.

- ❑ WHERE criteria that screen on more than one path with the OR operator are not supported.

**Example: Retrieving Data From Multiple Paths**

This example uses the following segments from the EMPLOYEE data source:



The request that follows retrieves data from both paths with MULTIPATH = SIMPLE, and displays data if either criterion is met:

```

SET ALL = OFF
SET MULTIPATH = SIMPLE
TABLE FILE EMPLOYEE
PRINT GROSS DATE_ATTEND COURSE_NAME
BY LAST_NAME BY FIRST_NAME
WHERE PAY_DATE EQ 820730
WHERE COURSE_CODE EQ '103'
END
  
```

The following warning message is generated:

```
(FOC144) WARNING. TESTING IN INDEPENDENT SETS OF DATA
```

Although several employees have not taken any courses, they are included in the report output since they have instances on one of the two paths.

The output is:

LAST_NAME	FIRST_NAME	GROSS	DATE_ATTEND	COURSE_NAME
BANNING	JOHN	\$2,475.00	.	.
BLACKWOOD	ROSEMARIE	\$1,815.00	.	.
CROSS	BARBARA	\$2,255.00	.	.
GREENSPAN	MARY	\$750.00	.	.
IRVING	JOAN	\$2,238.50	.	.
JONES	DIANE	\$1,540.00	82/05/26	BASIC REPORT PREP FOR PROG
MCCOY	JOHN	\$1,540.00	.	.
MCKNIGHT	ROGER	\$1,342.00	.	.
ROMANS	ANTHONY	\$1,760.00	.	.
SMITH	MARY	\$1,100.00	81/11/16	BASIC REPORT PREP FOR PROG
	RICHARD	\$791.67	.	.
STEVENS	ALFRED	\$916.67	.	.

If you run the same request with MULTIPATH = COMPOUND, the employees without instances for COURSE\_NAME are omitted from the report output, and the warning message is not generated.

The output is:

LAST_NAME	FIRST_NAME	GROSS	DATE_ATTEND	COURSE_NAME
-----	-----	-----	-----	-----
JONES	DIANE	\$1,540.00	82/05/26	BASIC REPORT PREP FOR PROG
SMITH	MARY	\$1,100.00	81/11/16	BASIC REPORT PREP FOR PROG

### Reference: MULTIPATH and SET ALL Combinations

The ALL parameter affects independent path processing. The following table uses examples from the EMPLOYEE data source to explain the interaction of ALL and MULTIPATH.

Request	MULTIPATH=SIMPLE	MULTIPATH=COMPOUND
SET ALL = OFF PRINT EMP_ID PAY_DATE DATE_ATTEND	Shows employees who have <i>either</i> SALINFO data <i>or</i> ATTNDSEG data.	Shows employees who have <i>both</i> SALINFO <i>and</i> ATTNDSEG data.
SET ALL = ON PRINT EMP_ID PAY_DATE DATE_ATTEND	Shows employees who have SALINFO data <i>or</i> ATTNDSEG data <i>or</i> no child data at all.	Same as SIMPLE.

Request	MULTIPATH=SIMPLE	MULTIPATH=COMPOUND
<pre>SET ALL = OFF PRINT EMP_ID PAY_DATE DATE_ATTEND WHERE PAY_DATE EQ 980115</pre>	<p>Shows employees who have <i>either</i> SALINFO data for 980115 <i>or</i> any ATTNDSEG data.</p> <p>Produces (FOC144) message.</p>	<p>Shows employees who have <i>both</i> SALINFO data for 980115 <i>and</i> ATTNDSEG data.</p>
<pre>SET ALL = ON PRINT EMP_ID PAY_DATE DATE_ATTEND WHERE PAY_DATE EQ 980115</pre>	<p>Shows employees who have <i>either</i> SALINFO data for 980115 <i>or</i> any ATTNDSEG data.</p> <p>Produces (FOC144) message.</p>	<p>Shows employees who have SALINFO data for 980115. Any DATE_ATTEND data is also shown.</p>
<pre>SET ALL = OFF PRINT ALL.EMP_ID DATE_ATTEND WHERE PAY_DATE EQ 980115</pre>	<p>Shows employees who have <i>either</i> SALINFO data for 980115 <i>or</i> any ATTNDSEG data.</p> <p>Produces (FOC144) message.</p>	<p>Shows employees who have SALINFO data for 980115. Any DATE_ATTEND data is also shown.</p>
<pre>SET ALL = ON or OFF PRINT EMP_ID PAY_DATE DATE_ATTEND WHERE PAY_DATE EQ 980115 <b>AND</b> COURSE_CODE EQ '103'</pre>	<p>Shows employees who have <i>either</i> SALINFO data for 980115 <i>or</i> COURSE 103.</p> <p><b>Note:</b> SIMPLE treats AND in the WHERE clause as OR.</p> <p>Produces (FOC144) message.</p>	<p>Shows employees who have <i>both</i> SALINFO data for 980115 <i>and</i> COURSE 103.</p>

**Note:** SET ALL = PASS is not supported with MULTIPATH = COMPOUND.

For related information about the ALL parameter, see [Handling Records With Missing Field Values](#) on page 971.

### **Reference:** Rules for Determining If a Segment Is Required

The segment rule is applied level by level, descending through the data source/view hierarchy. That is, a parent segment existence depends on the child segment existence, and the child segment depends on the grandchild existence, and so on, for the full data source tree.

The following rules are used to determine if a segment is required or optional:

- ☐ When SET ALL is ON or OFF, a segment with WHERE or IF criteria is required for its parent, and all segments up to the root segment are required for their parents.  
When SET ALL = PASS, a segment with WHERE or IF criteria is optional.
- ☐ IF SET ALL = ON or PASS, all referenced segments with no WHERE or IF criteria are optional for their parents (outer join).
- ☐ IF SET ALL = OFF, all referenced segments are required (inner join).
- ☐ A referenced segment can become optional if its parent segment uses the ALL. field prefix operator.

**Note:** ALL = PASS is not supported for all data adapters and, if it is supported, it may behave slightly differently. Check your specific data adapter documentation for detailed information.

For related information about the ALL parameter, see [Handling Records With Missing Field Values](#) on page 971, and the *Describing Data With WebFOCUS Language* manual.

## Selection Based on Aggregate Values

You can select records based on the aggregate value of a field. For example, on the sum of field values, or on the average of field values, by using the WHERE TOTAL phrase. WHERE TOTAL is very helpful when you employ the aggregate display commands SUM and COUNT, and is required for fields with a prefix operator, such as AVE. and PCT.

In WHERE tests, data is evaluated before it is retrieved. In WHERE TOTAL tests, however, data is selected after all the data has been retrieved and processed. For an example, see [Using WHERE TOTAL for Record Selection](#) on page 229.

### **Syntax:** How to Select Records With WHERE TOTAL

```
WHERE TOTAL criteria[:]
```

where:

*criteria*

Are the criteria for selecting records to include in the report. The criteria must be defined in a valid expression that evaluates as true or false (that is, a Boolean



expression). Expressions are described in detail in [Using Expressions](#) on page 431. Operators that can be used in WHERE expressions (such as, IS and GT) are described in [Operators Supported for WHERE and IF Tests](#) on page 238.

;

Is an optional semicolon that can be used to enhance the readability of the request. It does not affect the report.

### **Reference: Usage Notes for WHERE TOTAL**

- ❑ Any reference to a calculated value, or use of a feature that aggregates values, such as TOT.field, AVE.field, requires the use of WHERE TOTAL.
- ❑ Fields with prefix operators require the use of WHERE TOTAL.
- ❑ WHERE TOTAL tests are performed at the lowest sort level.
- ❑ Alphanumeric and date literals must be enclosed in single quotation marks. Date-time literals must be in the form DT (*date-time literal*).
- ❑ When you use ACROSS with WHERE TOTAL, data that does not satisfy the selection criteria is represented in the report with the NODATA character.
- ❑ If you save the output from your report request in a HOLD file, the WHERE TOTAL test creates a field called WH\$\$\$T1, which contains its internal computations. If there is more than one WHERE TOTAL test, each TOTAL test creates a corresponding WH\$\$\$T field and the fields are numbered consecutively.

### **Example: Using WHERE TOTAL for Record Selection**

The following example sums current salaries by department.

```
TABLE FILE EMPLOYEE
SUM CURR_SAL
BY DEPARTMENT
END
```

The output is:

DEPARTMENT	CURR_SAL
-----	-----
MIS	\$108,002.00
PRODUCTION	\$114,282.00

Now, add a WHERE TOTAL phrase to the request in order to generate a report that lists only the departments where the total of the salaries is more than \$110,000.

```
TABLE FILE EMPLOYEE
SUM CURR_SAL
BY DEPARTMENT
WHERE TOTAL CURR_SAL EXCEEDS 110000
END
```

The values for each department are calculated and then each final value is compared to \$110,000. The output is:

DEPARTMENT	CURR_SAL
-----	-----
PRODUCTION	\$114,282.00

### **Example:** Combining WHERE TOTAL and WHERE for Record Selection

The following request extracts records for the MIS department. Then, CURR\_SAL is summed for each employee. If the total salary for an employee is greater than \$20,000, the values of CURR\_SAL are processed for the report. In other words, WHERE TOTAL screens data after records are selected.

```
TABLE FILE EMPLOYEE
SUM CURR_SAL
BY LAST_NAME AND BY FIRST_NAME
WHERE TOTAL CURR_SAL EXCEEDS 20000
WHERE DEPARTMENT IS 'MIS'
END
```

The output is:

LAST_NAME	FIRST_NAME	CURR_SAL
-----	-----	-----
BLACKWOOD	ROSEMARIE	\$21,780.00
CROSS	BARBARA	\$27,062.00

## Applying Selection Criteria to the Internal Matrix Prior to COMPUTE Processing

WHERE TOTAL tests are applied to the rows of the internal matrix after COMPUTE calculations are processed in the output phase of the report. WHERE\_GROUPED tests are applied to the internal matrix values prior to COMPUTE calculations. The processing then continues with COMPUTE calculations, and then WHERE TOTAL tests. This allows the developer to control the evaluation, and is particularly useful in recursive calculations.

### **Syntax:** How to Apply WHERE\_GROUPED Selection Criteria

```
WHERE_GROUPED expression
```

where:

*expression*

Is an expression that does not refer to more than one row in the internal matrix. For example, it cannot use the LAST operator to refer to or retrieve a value from a prior record.

**Example: Using a WHERE\_GROUPED Test**

The following request has two COMPUTE commands. The first COMPUTE checks to see if the business region value has changed, incrementing a counter if it has. This allows us to sequence the records in the matrix. The second COMPUTE creates a rolling total of the days delayed within the business region.

```
TABLE FILE WF_RETAIL_LITE
SUM  DAYSDELAYED AS DAYS
COMPUTE CTR/I3 = IF BUSINESS_REGION EQ LAST BUSINESS_REGION THEN CTR+1 ELSE
1;
COMPUTE NEWDAYS = IF BUSINESS_REGION EQ LAST BUSINESS_REGION THEN NEWDAYS
+DAYSDELAYED ELSE DAYSDELAYED;
BY BUSINESS_REGION AS Region
BY TIME_MTH
WHERE BUSINESS_REGION NE 'Oceania'
ON TABLE SET PAGE NOPAGE
END
```

The output is shown in the following image.

Region	Sale Month	DAYS	CTR	NEWDAYS
EMEA	1	191	1.00	191.00
	2	205	2.00	396.00
	3	224	3.00	620.00
	4	213	4.00	833.00
	5	185	5.00	1,018.00
	6	234	6.00	1,252.00
	7	223	7.00	1,475.00
	8	234	8.00	1,709.00
	9	249	9.00	1,958.00
	10	290	10.00	2,248.00
	11	1	11.00	2,249.00
North America	1	253	1.00	253.00
	2	205	2.00	458.00
	3	293	3.00	751.00
	4	319	4.00	1,070.00
	5	273	5.00	1,343.00
	6	268	6.00	1,611.00
	7	219	7.00	1,830.00
	8	248	8.00	2,078.00
	9	220	9.00	2,298.00
	10	326	10.00	2,624.00
	11	8	11.00	2,632.00
South America	1	64	1.00	64.00
	2	87	2.00	151.00
	3	29	3.00	180.00
	4	39	4.00	219.00
	5	29	5.00	248.00
	6	49	6.00	297.00
	7	16	7.00	313.00
	8	26	8.00	339.00
	9	54	9.00	393.00
	10	54	10.00	447.00

The following version of the request adds a WHERE TOTAL test to select only those months where DAYSDELAYED exceeded 200 days..

```
TABLE FILE WF_RETAIL_LITE
SUM  DAYSDELAYED AS DAYS
COMPUTE CTR/I3 = IF BUSINESS_REGION EQ LAST BUSINESS_REGION THEN CTR+1 ELSE
1;
COMPUTE NEWDAYS= IF BUSINESS_REGION EQ LAST BUSINESS_REGION THEN NEWDAYS
+DAYSDELAYED ELSE DAYSDELAYED;
BY BUSINESS_REGION AS Region
BY TIME_MTH
WHERE BUSINESS_REGION NE 'Oceania'
WHERE TOTAL DAYSDELAYED GT 200
ON TABLE SET PAGE NOPAGE
END
```

The output is shown in the following image. The COMPUTE calculations for CTR and NEWDAYS was processed prior to eliminating the rows in which TOTAL DAYSDELAYED were 200 or less, so their values are the same as in the original output. This does not correctly reflect the sequence of records and the rolling total of the values that are actually displayed on the output. To do this, we need to select the appropriate months (DAYSDELAYED GT 200) before the COMPUTE expressions are evaluated. This requires WHERE\_GROUPED.

Region	Sale Month	DAYS	CTR	NEWDAYS
EMEA	2	205	2.00	396.00
	3	224	3.00	620.00
	4	213	4.00	833.00
	6	234	6.00	1,252.00
	7	223	7.00	1,475.00
	8	234	8.00	1,709.00
	9	249	9.00	1,958.00
	10	290	10.00	2,248.00
North America	1	253	1.00	253.00
	2	205	2.00	458.00
	3	293	3.00	751.00
	4	319	4.00	1,070.00
	5	273	5.00	1,343.00
	6	268	6.00	1,611.00
	7	219	7.00	1,830.00
	8	248	8.00	2,078.00
	9	220	9.00	2,298.00
	10	326	10.00	2,624.00

The following version of the request replaces the WHERE TOTAL test with a WHERE\_GROUPED test.

```
TABLE FILE WF_RETAIL_LITE
SUM  DAYSDELAYED AS DAYS
COMPUTE CTR/I3 = IF BUSINESS_REGION EQ LAST BUSINESS_REGION THEN CTR+1 ELSE
1;
COMPUTE NEWDAYS= IF BUSINESS_REGION EQ LAST BUSINESS_REGION THEN NEWDAYS
+DAYSDELAYED ELSE DAYSDELAYED;
BY BUSINESS_REGION AS Region
BY TIME_MTH
WHERE BUSINESS_REGION NE 'Oceania'
WHERE_GROUPED DAYSDELAYED GT 200
ON TABLE SET PAGE NOPAGE
END
```

The output is shown in the following image. The COMPUTE calculation for NEWDAYS was processed after eliminating the rows in which TOTAL DAYSDELAYED were 200 or less, so its values are based on fewer rows than the calculations in the original request. This is verified by the CTR values, which are now in a continuous sequence. The rolling total now reflects the values that are actually displayed on the report output.

Region	Sale Month	DAYS	CTR	NEWDAYS
EMEA	2	205	1.00	205.00
	3	224	2.00	429.00
	4	213	3.00	642.00
	6	234	4.00	876.00
	7	223	5.00	1,099.00
	8	234	6.00	1,333.00
	9	249	7.00	1,582.00
	10	290	8.00	1,872.00
North America	1	253	1.00	253.00
	2	205	2.00	458.00
	3	293	3.00	751.00
	4	319	4.00	1,070.00
	5	273	5.00	1,343.00
	6	268	6.00	1,611.00
	7	219	7.00	1,830.00
	8	248	8.00	2,078.00
	9	220	9.00	2,298.00
	10	326	10.00	2,624.00

**Reference: Usage Notes for WHERE\_GROUPED**

- ❑ If the expression refers to multiple rows in the internal matrix, the following message is generated and processing stops.  
(FOC32692) WHERE\_GROUPED CANNOT REFER TO OTHER LINES OF REPORT
- ❑ A COMPUTE that does not reference multiple lines will be evaluated prior to WHERE\_GROUPED tests, and may, therefore, be used in an expression and evaluated as part of a WHERE\_GROUPED test.



- ❑ WHERE\_GROUPED can be optimized for SQL data sources by creating a GROUP BY *fieldname* HAVING *expression* clause, where the expression is the WHERE\_GROUPED selection criteria.

## Using Compound Expressions for Record Selection

You can combine two or more simple WHERE expressions, connected by AND and/or OR operators, to create a compound expression.

By default, when multiple WHERE phrases are evaluated, logical ANDs are processed before logical ORs. In compound expressions, you can use parentheses to change the order of evaluation. All AND and OR operators enclosed in parentheses are evaluated first, followed by AND and OR operators outside of parentheses.

You should always use parentheses in complex expressions to ensure that the expression is evaluated correctly. For example:

```
WHERE (SEATS EQ 2) AND (SEATS NOT-FROM 3 TO 4)
```

This is especially useful when mixing literal OR tests with logical AND and OR tests:

- ❑ In a logical AND or OR test, all field names, test relations, and test values are explicitly referenced and connected by the words OR or AND. For example:

```
WHERE (LAST_NAME EQ 'CROSS') OR (LAST_NAME EQ 'JONES')
```

or

```
WHERE (CURR_SAL GT 20000) AND (DEPARTMENT IS 'MIS')
      AND (CURR_JOBCODE CONTAINS 'A')
```

- ❑ In a literal OR test, the word OR is repeated between test values of a field name, but the field name itself and the connecting relational operator are not repeated. For example:

```
WHERE (LAST_NAME EQ 'CROSS' OR 'JONES')
```

### **Example:** Mixing AND and OR Record Selection Tests

This example illustrates the impact of parentheses on the evaluation of literal ORs and logical ANDs.

In this request, each expression enclosed in parentheses is evaluated first in the order in which it appears. Notice that the first expression contains a literal OR. The result of each expression is then evaluated using the logical AND.

If parentheses are excluded, the logical AND is evaluated before the literal OR.

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL BY LAST_NAME
WHERE (LAST_NAME EQ 'CROSS' OR 'JONES')
AND (CURR_SAL GT 22000)
END
```

The output is:

LAST_NAME	CURR_SAL
-----	-----
CROSS	\$27,062.00

Using Operators in Record Selection Tests

You can include a variety of operators in your WHERE and IF selection tests. Many of the operators are common for WHERE and IF. However, several are supported only for WHERE tests.

*Reference:* Operators Supported for WHERE and IF Tests

You can define WHERE and IF selection criteria using the following operators.

WHERE Operator	IF Operator	Meaning
EQ IS	EQ IS	Tests for and selects values equal to the test expression.
NE IS-NOT	NE IS-NOT	Tests for and selects values not equal to the test expression.
GE	GE FROM IS-FROM	Tests for and selects values greater than or equal to the test value (based on the characters 0 to 9 for numeric values, A to Z and a to z for alphanumeric values).  The test value can be a field value or the result of an expression.
GT EXCEEDS IS-MORE-THAN	GT EXCEEDS IS-MORE-THAN	Tests for and selects values greater than the test value.
LT IS-LESS-THAN	LT IS-LESS-THAN	Tests for and selects values less than the test value.

WHERE Operator	IF Operator	Meaning
LE	LE TO	Tests for and selects values less than or equal to the test value.
GE <i>lower</i> AND ... LE <i>upper</i>		Tests for and selects values within a range of values.
LT <i>lower</i> OR ... GT <i>upper</i>		Tests for and selects values outside of a range of values.
FROM <i>lower</i> TO upper		Tests for and selects values within a range of values.
IS-FROM lower TO upper	IS-FROM lower TO upper	Tests for and selects values within a range of values. For WHERE, this is alternate syntax for FROM lower to UPPER. Both operators produce identical results.
NOT-FROM lower TO upper	NOT-FROM lower TO upper	Tests for and selects values that are outside a range of values.
IS MISSING IS-NOT MISSING NE MISSING	IS MISSING IS-NOT MISSING NE MISSING	Tests whether a field contains missing values. If some instances of the field contain no data, they have missing data. For information on missing data, see <a href="#">Handling Records With Missing Field Values</a> on page 971.
CONTAINS LIKE	CONTAINS LIKE	Tests for and selects values that include a character string matching test value. The string can occur in any position in the value being tested. When used with WHERE, CONTAINS can test alphanumeric fields. When used with IF, it can test both alphanumeric and text fields.

WHERE Operator	IF Operator	Meaning
OMITS NOT LIKE	OMITS UNLIKE	Tests for and selects values that do not include a character string matching test value. The string cannot occur in any position in the value being tested. When used with WHERE, OMITS can test alphanumeric fields. When used with IF, it can test both alphanumeric and text fields.
INCLUDES	INCLUDES	Tests whether a chain of values of a given field in a child segment includes all of a list of literals.
EXCLUDES	EXCLUDES	Tests whether a chain of values of a given field in a child segment excludes all of a list of literals.
IN ( z , x , y )		Selects records based on values found in an unordered list.
NOT ... IN ( z , x , y )		Selects records based on values not found in an unordered list.
IN FILE		Selects records based on values stored in a sequential file.
NOT ... IN FILE		Selects records with field values not found in a sequential file.

**Example:** Using Operators to Compare a Field to One or More Values

The following examples illustrate field selection criteria that use one or more values. You may use the operators: EQ, IS, IS-NOT, EXCEEDS, IS-LESS-THAN, and IN.

**Example 1:** The field LAST\_NAME must equal the value JONES:

```
WHERE LAST_NAME EQ 'JONES'
```

**Example 2:** The field LAST\_NAME begins with 'CR' or 'MC:'

```
WHERE EDIT (LAST_NAME, '99') EQ 'CR' OR 'MC'
```

**Example 3:** The field AREA must not equal the value EAST or WEST:

```
WHERE AREA IS-NOT 'EAST' OR 'WEST'
```

**Example 4:** The value of the field AREA must equal the value of the field REGION:

```
WHERE AREA EQ REGION
```

Note that you cannot compare one field to another in an IF test.

**Example 5:** The ratio between retail cost and dealer cost must be greater than 1.25:

```
WHERE RETAIL_COST/DEALER_COST GT 1.25
```

**Example 6:** The field UNITS must be equal to or less than the value 50, and AREA must not be equal to either NORTH EAST or WEST. Note the use of single quotation marks around NORTH EAST. All alphanumeric strings must be enclosed within single quotation marks.

```
WHERE UNITS LE 50 WHERE AREA IS-NOT 'NORTH EAST' OR 'WEST'
```

**Example 7:** The value of AMOUNT must be greater than 40:

```
WHERE AMOUNT EXCEEDS 40
```

**Example 8:** The value of AMOUNT must be less than 50:

```
WHERE AMOUNT IS-LESS-THAN 50
```

**Example 9:** The value of SALES must be equal to one of the numeric values in the unordered list. Use commas or blanks to separate the list values.

```
WHERE SALES IN (43000,12000,13000)
```

**Example 10:** The value of CAR must be equal to one of the alphanumeric values in the unordered list. Single quotation marks must enclose alphanumeric list values.

```
WHERE CAR IN ('JENSEN','JAGUAR')
```

**Example: Using Variables in Record Selection Tests**

In this example, the field REGION is used in the WHERE test as a variable so that when the report is executed, the user is prompted to select one of the listed values (CE, CORP, NE, SE or WE) of the REGION field. The text that appears after the values is what appears before the drop-down list in the output.

```
TABLE FILE EMPDATA
SUM SALARY
BY DIV
BY DEPT
HEADING
"Current Salary Report"
"for the &REGION Division"
" "
WHERE ( DIV EQ
'&REGION.(CE,CORP,NE,SE,WE).Please select a Region. ');
END
```

The output is:

Select a region from the drop-down list and click *Submit*. The output for the NE region is:

Current Salary Report  
for the NE Division

<u>DIV</u>	<u>DEPT</u>	<u>SALARY</u>
NE	CUSTOMER SUPPORT	\$81,800.00
	MARKETING	\$139,800.00
	SALES	\$82,600.00

## Types of Record Selection Tests

You can select records for your reports using a variety of tests that are implemented using the operators described in [Operators Supported for WHERE and IF Tests](#) on page 238. You can test for:

- ☐ Values that lie within or outside of a range. See [Range Tests With FROM and TO](#) on page 243 and [Range Tests With GE and LE or GT and LT](#) on page 245.
- ☐ Missing or existing data. See [Missing Data Tests](#) on page 246.
- ☐ The existence or absence of a character string. See [Character String Screening With CONTAINS and OMITS](#) on page 247.
- ☐ Partially defined character strings in a data field. See [Screening on Masked Fields](#) on page 248.
- ☐ Literals in a parent segment. See [Qualifying Parent Segments Using INCLUDES and EXCLUDES](#) on page 256.

## Range Tests With FROM and TO

Use the operators FROM ... TO and NOT-FROM ... TO in order to determine whether field values fall within or outside of a given range. You can use either values or expressions to specify the lower and upper boundaries. Range tests can also be applied on the sort control fields. The range test is specified immediately after the sort phrase.

You can also test whether an expression falls within or outside the boundaries.

### **Syntax:** How to Specify a Range Test (FROM and TO)

```
WHERE [TOTAL] {fieldname|expression} {FROM|IS-FROM} lower TO upper
WHERE [TOTAL] fieldname NOT-FROM      lower TO upper
```

where:

*fieldname*

Is any valid field name or alias.

*expression*

Is any valid expression.

*lower*

Are numeric or alphanumeric values or expressions that indicate lower boundaries. You may add parentheses around expressions for readability.

### *upper*

Are numeric or alphanumeric values or expressions that indicate upper boundaries.  
You may add parentheses around expressions for readability.

#### **Example:** Range Test With FROM ... TO

An example of a range test using expressions as boundaries follows:

```
WHERE SALES FROM (DEALER_COST * 1.4) TO (DEALER_COST * 2.0)
```

The following is an example of a range test using expressions as the comparison value and the boundaries:

```
WHERE SALES * 1.5 FROM (DEALER_COST * 1.4) TO (DEALER_COST * 2.0)
```

#### **Example:** Range Test With NOT-FROM ... TO

The following illustrates how you can use the range test NOT-FROM ... TO to display only those records that fall outside of the specified range. In this example, it is all employees whose salaries do not fall in the range between \$12,000 and \$22,000.

```
TABLE FILE EMPLOYEE  
PRINT CURR_SAL  
BY LAST_NAME  
WHERE CURR_SAL NOT-FROM 12000 TO 22000  
END
```

The output is:

LAST_NAME	CURR_SAL
-----	-----
BANNING	\$29,700.00
CROSS	\$27,062.00
GREENSPAN	\$9,000.00
IRVING	\$26,862.00
SMITH	\$9,500.00
STEVENS	\$11,000.00

#### **Example:** Range Tests on Sort Fields With FROM ... TO

The following examples demonstrate how to perform range tests when sorting a field using the BY or ACROSS sort phrases:

```
BY MONTH FROM 4 TO 8
```

or

```
ACROSS MONTH FROM 6 TO 10
```



## Range Tests With GE and LE or GT and LT

The operators GE (greater than or equal to), LE (less than or equal to), GT (greater than), and LT (less than) can be used to specify a range.

GE ... LE enable you to specify values within the range test boundaries.

LT ...GT enable you to specify values outside the range test boundaries.

### **Syntax:** How to Specify Range Tests (GE and LE)

To select values that fall within a range, use

```
WHERE fieldname GE lower AND fieldname LE upper
```

To find records whose values do not fall in a specified range, use

```
WHERE fieldname LT lower OR fieldname GT upper
```

where:

*fieldname*

Is any valid field name or alias.

*lower*

Are numeric or alphanumeric values or expressions that indicate lower boundaries.  
You may add parentheses around expressions for readability.

*upper*

Are numeric or alphanumeric values or expressions that indicate upper boundaries.  
You may add parentheses around expressions for readability.

### **Example:** Selecting Values Inside a Range

This WHERE phrase selects records in which the UNIT value is between 10,000 and 14,000.

```
WHERE UNITS GE 10000 AND UNITS LE 14000
```

This example is equivalent to:

```
WHERE UNITS GE 10000  
WHERE UNITS LE 14000
```

**Example: Selecting Values Outside a Range**

The following illustrates how you can select values that are outside a range of values using the LT and GT operators. In this example, only those employees whose salaries are less than \$12,000 and greater than \$22,000 are included in the output.

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL
BY LAST_NAME
WHERE CURR_SAL LT 12000 OR CURR_SAL GT 22000
END
```

The output is:

LAST_NAME	CURR_SAL
-----	-----
BANNING	\$29,700.00
CROSS	\$27,062.00
GREENSPAN	\$9,000.00
IRVING	\$26,862.00
SMITH	\$9,500.00
STEVENS	\$11,000.00

**Missing Data Tests**

When creating report requests, you may want to test for missing data. This type of test is most useful when fields that have missing data also have the MISSING attribute set to ON in the Master File. For information on missing data, see [Handling Records With Missing Field Values](#) on page 971, and the *Describing Data With WebFOCUS Language* manual.

**Note:** If a test value to screen on an alphanumeric field is a variable and you want to look for missing instances, you must use \_FOC\_MISSING, instead of MISSING, as an alphanumeric literal value in a test must be in single quotation marks, and 'MISSING' is the literal value MISSING, not the MISSING value. The value \_FOC\_MISSING represents the MISSING value whether it is in single quotation marks or not.

**Syntax: How to Test for Missing Data**

```
{WHERE|IF} fieldname {EQ|IS} MISSING
```

where:

*fieldname*

Is any valid field name or alias.

EQ|IS

Are record selection operators. EQ and IS are synonyms.

**Syntax:**      **How to Test for Existing Data**

```
{WHERE|IF} fieldname {NE|IS-NOT} MISSING
```

where:

*fieldname*

Is any valid field name or alias.

NE|IS-NOT

Are record selection operators. NE and IS-NOT are synonyms.

**Character String Screening With CONTAINS and OMITS**

The CONTAINS and OMITS operators test alphanumeric fields when used with WHERE, and both alphanumeric and text fields when used with IF. With CONTAINS, if the characters in the given literal or literals appear anywhere within the characters of the field value, the test is passed.

OMITS is the opposite of CONTAINS; if the characters of the given literal or literals appear anywhere within the characters of the field's value, the test fails.

CONTAINS and OMITS tests are useful when you do not know the exact spelling of a value. As long as you know that a specific string appears within the value, you can retrieve the desired data.

**Example:**      **Selecting Records With CONTAINS and OMITS**

The following examples illustrate several ways to use the CONTAINS and OMITS operators. The field name that is being tested must appear on the left side of the CONTAINS or OMITS operator.

- ❑ In this example, the characters JOHN are contained in JOHNSON, and are selected by the following phrase:

```
WHERE LAST_NAME CONTAINS 'JOHN'
```

The LAST\_NAME field may contain the characters JOHN anywhere in the field.

- ❑ In this example, any last name without the string JOHN is selected:

```
WHERE LAST_NAME OMITS 'JOHN'
```

- ❑ In this example, all names that contain the letters ING are retrieved.

```
TABLE FILE EMPLOYEE
LIST LAST_NAME AND FIRST_NAME
WHERE LAST_NAME CONTAINS 'ING'
END
```

The output is:

LIST	LAST_NAME	FIRST_NAME
----	-----	-----
1	BANNING	JOHN
2	IRVING	JOAN

## Screening on Masked Fields

A mask is an alphanumeric pattern that you supply for comparison to characters in a data field. The data field must have an alphanumeric format (A). You can use the LIKE and NOT LIKE or the IS and IS-NOT operators to perform screening on masked fields.

The wildcard characters for screening on masked fields with:

- ❑ LIKE and NOT LIKE operators are % and \_. The percent allows any following sequence of zero or more characters. The underscore indicates that any character in that position is acceptable. The LIKE operator is supported in expressions that are used to derive temporary fields with either the DEFINE or COMPUTE command.
- ❑ IS (or EQ) and IS-NOT (or NE) operators are \$ and \$\*. The dollar sign indicates that any character in that position is acceptable. The \$\* is shorthand for writing a sequence of dollar signs to fill the end of the mask without specifying a length. This combination can only be used at the end of the mask.

In IF clauses and those WHERE clauses that can be translated into one or more IF clauses, you can treat the \$ and \$\* characters as normal characters rather than wildcards by issuing the SET EQTEST=EXACT command.

**Note:** The IS (or EQ) and IS-NOT (or NE) operators support screening based on a mask for fixed length formats only. If the format is a variable length format, for example, AnV, use the LIKE or NOT LIKE operator to screen based on a mask.

### **Syntax:** How to Screen Fields Based on a Mask (Using LIKE and NOT LIKE)

To search for records with the LIKE operator, use

```
WHERE field LIKE 'mask'
```

To reject records based on the mask value, use either

```
WHERE field NOT LIKE 'mask'
```

or

```
WHERE NOT field LIKE 'mask'
```

where:

*field*

Is any valid field name or alias.

*mask*

Is an alphanumeric or text character string you supply. There are two wildcard characters that you can use in the mask. The underscore (\_) indicates that any character in that position is acceptable, and the percent sign (%) allows any following sequence of zero or more characters.

For related information, see [Restrictions on Masking Characters](#) on page 250.

### **Syntax:** How to Screen Using LIKE and UNLIKE in an IF Phrase

To search for records with the LIKE operator, use

```
IF field LIKE 'mask1' [OR 'mask2' ...]
```

To reject records based on the mask value, use

```
IF field UNLIKE 'mask1' [OR 'mask2' ...]
```

where:

*field*

Is any valid field name or alias.

*mask1, mask2*

Are the alphanumeric patterns you want to use for comparison. The single quotation marks are required if the mask contains blanks. There are two wildcard characters that you can use in a mask. The underscore (\_) indicates that any character in that position is acceptable, and the percent sign (%) allows any following sequence of zero or more characters. Every other character in the mask accepts only itself in that position as a match to the pattern.

### **Syntax:** How to Screen Fields Based on a Mask (Using IS and IS-NOT)

To search for records with the IS operator, use

```
{WHERE|IF} field {IS|EQ} 'mask'
```

To reject records based on the mask value, use

```
{WHERE|IF} field {IS-NOT|NE} 'mask'
```

where:

*field*

Is any valid field name or alias.

IS|IS-NOT

Are record selection operators. EQ is a synonym for IS. NE is a synonym for IS-NOT.

*mask*

Is an alphanumeric or text character string you supply. The wildcard characters that you can use in the mask are the dollar sign (\$) and the combination \$\*. The dollar sign indicates that any character in that position is acceptable. The \$\* combination allows any sequence of zero or more characters. The \$\* is shorthand for writing a sequence of dollar signs to fill the end of the mask without specifying a specific length. This combination can only be used at the end of the mask.

For related information, see [Restrictions on Masking Characters](#) on page 250.

### **Reference:** Restrictions on Masking Characters

- ❑ The wildcard characters dollar sign (\$) and dollar sign with an asterisk (\$\*), which are used with IS operators, are treated as literals with LIKE operators.
- ❑ Masking with the characters \$ and \$\* is not supported for compound WHERE phrases that use the AND or OR logical operators.

### **Example:** Screening on Initial Characters

To list all employees who have taken basic-level courses, where every basic course begins with the word BASIC, issue the following request:

```
TABLE FILE EMPLOYEE  
PRINT COURSE_NAME COURSE_CODE  
BY LAST_NAME BY FIRST_NAME  
WHERE COURSE_NAME LIKE 'BASIC%'  
END
```

The output is:

<u>LAST_NAME</u>	<u>FIRST_NAME</u>	<u>COURSE_NAME</u>	<u>COURSE_CODE</u>
BLACKWOOD	ROSEMARIE	BASIC REPORT PREP NON-PROG	102
CROSS	BARBARA	BASIC REPORT PREP DP MGRS	107
JONES	DIANE	BASIC REPORT PREP FOR PROG	103
SMITH	MARY	BASIC REPORT PREP FOR PROG	103
SMITH	RICHARD	BASIC RPT NON-DP MGRS	108

### **Example:** Screening on Characters Anywhere in a Field

If you want to see which employees have taken a FOCUS course, but you do not know where the word FOCUS appears in the title, bracket the word FOCUS with wildcards (which is equivalent to using the CONTAINS operator):

```
TABLE FILE EMPLOYEE
PRINT COURSE_NAME COURSE_CODE
BY LAST_NAME BY FIRST_NAME
WHERE COURSE_NAME LIKE '%FOCUS%'
END
```

The output is:

LAST_NAME	FIRST_NAME	COURSE_NAME	COURSE_CODE
-----	-----	-----	-----
BLACKWOOD	ROSEMARIE	WHAT'S NEW IN FOCUS	202
JONES	DIANE	FOCUS INTERNALS	203

If you want to list all employees who have taken a 20x-series course, and you know that all of these courses have the same code except for the final character, issue the following request:

```
TABLE FILE EMPLOYEE
PRINT COURSE_NAME COURSE_CODE
BY LAST_NAME BY FIRST_NAME
WHERE COURSE_CODE LIKE '20_'
END
```

The output is:

LAST_NAME	FIRST_NAME	COURSE_NAME	COURSE_CODE
-----	-----	-----	-----
BLACKWOOD	ROSEMARIE	WHAT'S NEW IN FOCUS	202
JONES	DIANE	FOCUS INTERNALS	203
		ADVANCED TECHNIQUES	201

**Example:**     **Screening on Initial Characters and Specific Length**

The following example illustrates how to screen on initial characters and specify the length of the field value you are searching for. In this example, the WHERE phrase states that the last name must begin with BAN and be seven characters in length (the three initial characters BAN and the four placeholders, in this case, the dollar sign). The remaining characters in the field (positions 8 through 15) must be blank.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME
WHERE LAST_NAME IS 'BAN$$$$'
END
```

The output is:

```
LAST_NAME
-----
BANNING
```

**Example:**     **Screening on Records of Unspecified Length**

To retrieve records with unspecified lengths, use the dollar sign followed by an asterisk (\$\*):

```
WHERE LAST_NAME IS 'BAN$*'
```

This phrase searches for last names that start with the letters BAN, regardless of the name length. The characters \$\* reduce typing, and enable you to define a screen mask without knowing the exact length of the field you wish to retrieve.

**Syntax:**     **How to Deactivate Wildcard Characters**

```
SET EQTEST = {WILDCARD|EXACT}
```

where:

WILDCARD

Treats the \$ and \$\* characters as wildcard characters. WILDCARD is the default value.

EXACT

Treats the \$ and \$\* characters as normal characters, not wildcards, in IF tests and in WHERE tests that can be translated to IF tests.

**Example:**     **Selecting Records With SET EQTEST**

The following request against the VIDEOTR2 data source creates two similar email addresses:

❑ handy\$man@usa.com, which has a dollar sign.



- ❑ handyiman@usa.com, which has the letter i in the same position as the \$ character in the other email address.

```

DEFINE FILE VIDEOTR2
SMAIL/A18= IF EMAIL EQ 'handyman@usa.com'
          THEN 'handyman@usa.com'
          ELSE EMAIL;
SMAIL/A18 = STRREP(18,SMAIL,1,'_',1,'$',18,SMAIL);
END
TABLE FILE VIDEOTR2
PRINT SMAIL
BY LASTNAME BY FIRSTNAME
WHERE SMAIL EQ 'handy$man@usa.com'
ON TABLE SET EQTEST WILDCARD
END

```

With SET EQTEST=WILDCARD (the default), the WHERE test *WHERE SMAIL IS 'handy \$man@usa.com'* returns both the record with the \$ in the address and the record with the letter i in the address because the \$ is treated as a wildcard character, and any character in that position causes the record to pass the screening test:

LASTNAME	FIRSTNAME	SMAIL
-----	-----	-----
HANDLER	EVAN	handy\$man@usa.com handyiman@usa.com

Changing the ON TABLE SET command to ON TABLE SET EQTEST EXACT returns just the ONE email address with the \$ character because the dollar sign is now treated as a normal character and only passes the test if there is an exact match:

LASTNAME	FIRSTNAME	SMAIL
-----	-----	-----
HANDLER	EVAN	handy\$man@usa.com

## Using an Escape Character for LIKE

You can use an escape character in the LIKE syntax to treat the masking characters (%) and (\_) as literals within the search pattern, rather than as wildcards. This technique enables you to search for these characters in the data. For related information, see [Screening on Masked Fields](#) on page 248.

**Syntax:**      **How to Use an Escape Character in a WHERE Phrase**

Any single character can be used as an escape character, if prefaced with the word ESCAPE

```
WHERE fieldname LIKE 'mask' ESCAPE 'c'
```

where:

*fieldname*

Is any valid field name or alias to be evaluated in the selection test.

*mask*

Is the search pattern that you supply. The single quotation marks are required.

*c*

Is any single character that you identify as the escape character. If you embed the escape character in the mask, before a % or \_, the % or \_ character is treated as a literal, rather than as a wildcard. The single quotation marks are required.

**Syntax:**      **How to Specify an Escape Character for a Mask in an IF Phrase**

You can assign any single character as an escape character by prefacing it with the word ESCAPE in the LIKE or UNLIKE syntax

```
IF field {LIKE|UNLIKE} 'mask1' ESCAPE 'a' [OR 'mask2' ESCAPE 'b' ...
```

where:

*field*

Is any valid field name or alias to be evaluated in the selection test.

*mask1, mask2*

Are search patterns that you supply. The single quotation marks are required.

*a, b ...*

Are single characters that you identify as escape characters. Each mask can specify its own escape character or use the same character as other masks. If you embed the escape character in the mask, before a % or \_, the % or \_ character is treated as a literal, rather than as a wildcard. The single quotation marks are required if the mask contains blanks.

**Reference:**      **Usage Notes for Escape Characters**

- ☐ The use of an escape character in front of any character other than %, \_, and itself is ignored.
- ☐ The escape character itself can be escaped, thus becoming a normal character in a string (for example, 'abc\%\%').

- ❑ Only one escape character can be used per LIKE phrase in a WHERE phrase.
- ❑ The escape character is only in effect when the ESCAPE syntax is included in the LIKE phrase.
- ❑ Every LIKE phrase can provide its own escape character.
- ❑ If a WHERE criterion is used with literal OR phrases, the ESCAPE must be on the first OR phrase, and applies to all subsequent phrases in that WHERE expression. For example:

```
WHERE field LIKE 'ABCg_' ESCAPE 'g' OR 'ABCg%' OR 'g%ABC'
```

### **Example:** Using the Escape Character in a WHERE Phrase

The VIDEOTR2 data source contains an email address field. To search for the email address with the characters 'handy\_' you can issue the following request:

```
TABLE FILE VIDEOTR2
PRINT CUSTID LASTNAME FIRSTNAME EMAIL
WHERE EMAIL LIKE 'handy_%'
END
```

Because the underscore character functions as a wildcard character, this request returns two instances, only one of which contains the underscore character.

The output is:

CUSTID	LASTNAME	FIRSTNAME	EMAIL
-----	-----	-----	-----
0944	HANDLER	EVAN	handy_man@usa.com
0944	HANDLER	EVAN	handyman@usa.com

To retrieve only the instance that contains the underscore character, you must indicate that the underscore should be treated as a normal character, not a wildcard. The following request retrieves only the instance with the underscore character in the email field:

```
TABLE FILE VIDEOTR2
PRINT CUSTID LASTNAME FIRSTNAME EMAIL
WHERE EMAIL LIKE 'handy\_%' ESCAPE '\'
END
```

The output is:

CUSTID	LASTNAME	FIRSTNAME	EMAIL
-----	-----	-----	-----
0944	HANDLER	EVAN	handy_man@usa.com

**Example:**    **Using an Escape Character in an IF Phrase**

The VIDEOTR2 data source contains an email address field. To search for email addresses with the characters 'handy\_' you can issue the following request:

```
TABLE FILE VIDEOTR2
PRINT CUSTID LASTNAME FIRSTNAME EMAI
IF EMAIL LIKE 'handy_%'
END
```

Because the underscore character functions as a wildcard character, this request returns two instances, only one of which contains the underscore character.

The output is:

CUSTID	LASTNAME	FIRSTNAME	EMAIL
-----	-----	-----	-----
0944	HANDLER	EVAN	handy_man@usa.com
0944	HANDLER	EVAN	handyman@usa.com

To retrieve only the instance that contains the underscore character, you must indicate that the underscore should be treated as a normal character, not a wildcard. The following request retrieves only the instance with the underscore character in the email field:

```
TABLE FILE VIDEOTR2
PRINT CUSTID LASTNAME FIRSTNAME EMAI
IF EMAIL LIKE 'handy\_%' ESCAPE '\'
END
```

The output is:

CUSTID	LASTNAME	FIRSTNAME	EMAIL
-----	-----	-----	-----
0944	HANDLER	EVAN	handy_man@usa.com

**Qualifying Parent Segments Using INCLUDES and EXCLUDES**

You can test whether instances of a given field in a child segment include or exclude all literals in a list using the INCLUDES and EXCLUDES operators. INCLUDES and EXCLUDES retrieve only parent records. You cannot print or list any field in the same segment as the field specified for the INCLUDES or EXCLUDES test.

**Note:** INCLUDES and EXCLUDES work only with multi-segment FOCUS data sources.

**Reference:**    **Usage Notes for INCLUDES and EXCLUDES**

- ❑ Literals containing embedded blanks must be enclosed in single quotation marks.
- ❑ The total number of literals must be 31 or less.
- ❑ To use more than one INCLUDES or EXCLUDES phrase in a request, begin each phrase on a separate line.

- ❑ You can connect the literals you are testing for with ANDs and ORs; however, the ORs are changed to ANDs.

### **Example: Selecting Records With INCLUDES and EXCLUDES**

A request that contains the phrase

```
WHERE JOBCODE INCLUDES A01 OR B01
```

returns employee records with JOBCODE instances for both A01 and B01, as if you had used AND.

In the following example, for a record to be selected, its JOBCODE field must have values of both A01 and B01:

```
WHERE JOBCODE INCLUDES A01 AND B01
```

If either one is missing, the record is not selected for the report.

If the selection criterion is

```
WHERE JOBCODE EXCLUDES A01 AND B01
```

every record that does not have both values is selected for the report.

In the CAR data source, only England produces Jaguars and Jensens, and so the request

```
TABLE FILE CAR
PRINT COUNTRY
WHERE CAR INCLUDES JAGUAR AND JENSEN
END
```

generates this output:

```
COUNTRY
-----
ENGLAND
```

## **Selections Based on Group Key Values**

Some data sources use group keys. A group key is a single key composed of several fields. You can use a group name to refer to group key fields.

To select records based on a group key value, you need to supply the value of each field. The values must be separated by the slash character (/).

Note that a WHERE phrase that refers to a group field cannot be used in conjunction with AND or OR. For related information, see [Using Compound Expressions for Record Selection](#) on page 237.

### **Example:**    **Selecting Records Using Group Keys**

Suppose that a data source has a group key named PRODNO, which contains three separate fields. The first is stored in alphanumeric format, the second as a packed decimal, and the third as an integer. A screening phrase on this group might be:

```
WHERE PRODNO EQ 'RS/62/83'
```

### **Setting Limits on the Number of Records Read**

For some reports, a limited number of records is satisfactory. When the specified number of records is retrieved, record retrieval can stop. This is useful when:

- ☐ You are designing a new report, and you need only a few records from the actual data source to test your design.
- ☐ The database administrator needs to limit the size of reports by placing an upper limit on retrieval from very large data sources. This limit is attached to the user password.
- ☐ You know the number of records that meet the test criteria. You can specify that number so that the search does not continue beyond the last record that meets the criteria. For example, suppose only ten employees use electronic transfer of funds, and you want to retrieve only those records. The record limit would be ten, and retrieval would stop when the tenth record is retrieved. The data source would not be searched any further.

### **Syntax:**    **How to Limit the Number of Records Read**

There are two ways to limit the number of records retrieved. You can use

```
WHERE RECORDLIMIT EQ n
```

where:

*n*

Is a number greater than 0, and indicates the number of records to be retrieved. This syntax can be used with FOCUS and non-FOCUS data sources.

For all non-FOCUS data sources, you can also use

```
WHERE READLIMIT EQ n
```

where:

*n*

Is a number greater than 0, and indicates the number of read operations (not records) to be performed. For details, see the appropriate data adapter manual.

**Tip:** If an attempt is made to apply the READLIMIT test to a FOCUS data source, the request is processed correctly, but the READLIMIT phrase is ignored.

**Note:** Using SET RECORDLIMIT disables AUTOINDEX.

### **Example:** Limiting the Number of Records Read

The following request retrieves four records, generating a four-line report:

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME AND FIRST_NAME AND EMP_ID
WHERE RECORDLIMIT EQ 4
END
```

The output is:

LAST_NAME	FIRST_NAME	EMP_ID
-----	-----	-----
STEVENS	ALFRED	071382660
SMITH	MARY	112847612
JONES	DIANE	117593129
SMITH	RICHARD	119265415

## Selecting Records Using IF Phrases

The IF phrase selects records to be included in a report, and offers a subset of the functionality of WHERE. For a list of supported IF operators, see [Operators Supported for WHERE and IF Tests](#) on page 238.

**Tip:** Unless you specifically require IF syntax (for example, to support legacy applications), we recommend using WHERE.

### **Syntax:** How to Select Records Using the IF Phrase

```
IF fieldname operator literal [OR literal]
```

where:

*fieldname*

Is the field you want to test (the test value).

*operator*

Is the type of selection operator you want. Valid operators are described in [Operators Supported for WHERE and IF Tests](#) on page 238.

*literal*

Can be the MISSING keyword (as described in [Missing Data Tests](#) on page 246) or alphanumeric or numeric values that are in your data source, with the word OR between values.

Note that all literals that contain blanks (for example, New York City) and all date and date-time literals must be enclosed within single quotation marks.

**Note:** The IF phrase alone cannot be used to create compound expressions by connecting simple expressions with AND and OR logical operators. Compound logic requires that the IF phrase be used with the DEFINE command, as described in [Using Expressions](#) on page 431. You can accomplish this more easily with WHERE. See [Using Compound Expressions for Record Selection](#) on page 237.

**Example:**    **Using Multiple IF Phrases**

You can use as many IF phrases as necessary to define all your selection criteria, as illustrated in the following example:

```
TABLE FILE EMPLOYEE
PRINT EMP_ID LAST_NAME
IF SALARY GT 20000
IF DEPARTMENT IS MIS
IF LAST_NAME IS CROSS OR BANNING
END
```

All of these criteria must be satisfied in order for a record to be included in a report. The output is:

EMP_ID	LAST_NAME
-----	-----
818692173	CROSS

**Reading Selection Values From a File**

Instead of typing literal test values in a WHERE or IF phrase, you can store them in a file and refer to the file in the report request. You can then select records based on equality or inequality tests on values stored in the file.

This method has the following advantages:

- ❑ You can save time by coding a large set of selection values once, then using these values as a set in as many report requests as you wish. You also ensure consistency by maintaining the criteria in just one location.



- ❑ If the selection values already exist in a data source, you can quickly create a file of selection values by generating a report and saving the output in a HOLD or SAVE file. You can then read selection values from that file.

Values from a HOLD file (with a data description) can be in either BINARY format (the default) or ALPHA (simple character) format. If you use a SAVE file, it must be in ALPHA format (the default). Using a SAVB file is only valid for alphanumeric values. For information on HOLD and SAVE files, see [Saving and Reusing Your Report Output](#) on page 471.

Note that in z/OS, a HOLD file in BINARY format that is used for selection values must be allocated to ddname HOLD (the default). The other extract files used for this purpose can be allocated to any ddname.

- ❑ You can include entries with mixed-case and special characters.

### **Syntax:** How to Read Selection Values From a File: WHERE field IN file

```
WHERE [NOT] fieldname IN FILE file
```

where:

*fieldname*

Is the name of the selection field. It can be any real or temporary field in the data source.

*file*

Is the name of the file.

Two-part names (app/file) are not supported. The file name is the ddname assigned by a DYNAM or TSO ALLOCATE command for z/OS, or a FILEDEF command for other environments.

For related information, see [Usage Notes for Reading Values From a File](#) on page 262.

### **Syntax:** How to Read Selection Values From a File: WHERE field operator (file)

```
WHERE field1 operator1 (file1) [{OR|AND} field2 operator2 (file2) ... ]
```

where:

*field1*, *field2*

Are any valid field names or aliases.

*operator1*, *operator2*

Can be the EQ, IS, NE, or IS-NOT operator.

*file1, file1*

Are the names of the files.

Two-part names (app/file) are not supported. The file name is the ddname assigned by a DYNAM or TSO ALLOCATE command for z/OS, or a FILEDEF command for other environments.

**Syntax:**     **How to Read Selection Values From a File: IF**

*IF fieldname operator (file) [OR (file) ... ]*

where:

*fieldname*

Is any valid field name or alias.

*operator*

Is the EQ, IS, NE, or IS-NOT operator (see [Operators Supported for WHERE and IF Tests](#) on page 238).

*file*

Is the name of the file.

Two-part names (app/file) are not supported. The file name is the ddname assigned by a DYNAM or TSO ALLOCATE command for z/OS, or a FILEDEF command for other environments.

**Reference:**     **Usage Notes for Reading Values From a File**

In order to read selection criteria from a file, the file must comply with the following rules:

- ☐ Each value in the file must be on a separate line.  
  
For IF, more information can appear on a line, but only the first data value encountered on the line is used.
- ☐ The selection value must start in column one.
- ☐ The values are assumed to be in character format, unless the file name is HOLD, and numeric digits are converted to internal computational numbers where needed (for example, binary integer).
- ☐ The maximum number of values is 32,767.
- ☐ For WHERE, alphanumeric values with embedded blanks or any mathematical operator (-, +, \*, /) must be enclosed in single quotation marks.

- ❑ For WHERE, when a compound WHERE phrase uses IN FILE more than once, the specified files must have the same record formats.

If your list of literals is too large, an error is displayed.

- ❑ For IF, sets of file names may be used, separated by the word OR, and with WHERE, AND. The file names cannot be prefaced with app names. Actual literals may also be mixed with the file names. For example:

```
IF fieldname operator (filename) OR literal...etc...
```

### **Example:** Reading Selection Values From a File (WHERE field IN file)

Create a file named EXPER, which contains the values B141 and B142.

This request uses selection criteria from the file EXPER. You must allocate or FILEDEF the EXPER file prior to running the request. For example, if the file is named exper.ftm and it is in the baseapp application, you can issue the following FILEDEF command:

```
FILEDEF EXPER DISK baseapp/exper.ftm
```

All records for which PRODUCT\_ID has a value of B141 or B142 are selected:

```
TABLE FILE GGPRODS
SUM UNIT_PRICE
BY PRODUCT_DESCRIPTION
WHERE PRODUCT_ID IN FILE EXPER
END
```

If you include the selection criteria directly in the request, the WHERE phrase specifies the values explicitly:

```
WHERE PRODUCT_DESCRIPTION EQ 'B141' or 'B142'
```

The output is:

Product	Unit Price
French Roast	81.00
Hazelnut	58.00

### **Example:** Reading Selection Values From a File With WHERE field operator (file)

The following request against the GGPRODS data source creates a HOLD file named EXPER1 that contains product IDs B141, B142, B143, and B144.

```
TABLE FILE GGPRODS
BY PRODUCT_ID BY PRODUCT_DESCRIPTION
WHERE PRODUCT_ID EQ 'B141' OR 'B142' OR 'B143' OR 'B144'
ON TABLE HOLD AS EXPER1 FORMAT ALPHA
END
```

The following request against the GGPRODS data source creates a HOLD file named EXPER2 that contains product IDs B144, F101, and F102.

```
TABLE FILE GGPRODS
BY PRODUCT_ID BY PRODUCT_DESCRIPTION
WHERE PRODUCT_ID EQ 'B144' OR 'F101' OR 'F102'
ON TABLE HOLD AS EXPER2 FORMAT ALPHA
END
```

The following request selects the values that exist in both EXPER1 AND EXPER2.

```
TABLE FILE GGPRODS
SUM PRODUCT_DESCRIPTION
BY PRODUCT_ID
WHERE PRODUCT_ID EQ (EXPER1) AND PRODUCT_ID IS (EXPER2)
ON TABLE SET PAGE NOPAGE
END
```

The output is:

```
Product
Code      Product
-----
B144      Kona
```

### ***Example:*** Reading Selection Values From a File (IF)

Create a file named EXPER, which contains the values B141 and B142.

This request uses selection criteria from the file EXPER. All records for which PRODUCT\_ID has a value of B141 or B142 are selected:

```
TABLE FILE GGPRODS
SUM UNIT_PRICE
BY PRODUCT_DESCRIPTION
IF PRODUCT_ID IS (EXPER)
END
```

If you include the selection criteria directly in the request, the IF phrase specifies the values explicitly:

```
IF PRODUCT_DESCRIPTION EQ 'B141' or 'B142'
```

The output is:

Product	Unit Price
-----	-----
French Roast	81.00
Hazelnut	58.00

## Assigning Screening Conditions to a File

You can assign screening conditions to a data source, independent of a request, and activate these screening conditions for use in report requests against the data source.

A filter is a packet of definitions that resides at the file level, containing WHERE and/or IF criteria. Whenever a report request is issued against a data source, all filters that have been activated for that data source are in effect. WHERE or IF syntax that is valid in a report request is also valid in a filter.

A filter can be declared at any time before the report request is run. The filters are available to subsequent requests during the session in which the filters have been run. For details, see [How to Declare a Filter](#) on page 266.

Filters allow you to:

- ☐ Declare a common set of screening conditions that apply each time you retrieve data from a data source. You can declare one or more filters for a data source.
- ☐ Declare a set of screening conditions and dynamically turn them on and off.
- ☐ Limit access to data without specifying rules in the Master File.

In an interactive environment, filters also reduce repetitive ad hoc typing.

**Note:** Simply declaring a filter for a data source does *not* make it active. A filter must be activated with a SET command. For details, see [How to Activate or Deactivate Filters](#) on page 268.

**Syntax:**      **How to Declare a Filter**

A filter can be described by the following declaration

```
FILTER FILE filename [CLEAR|ADD]
  [filter-defines;]
  NAME=filtername1 [,DESC=text]
  where-if phrases .
  .
  .
  NAME=filternamen [,DESC=text]
  where-if phrases END
```

where:

*filename*

Is the name of the Master File to which the filters apply.

**CLEAR**

Deletes any existing filter phrases, including any previously defined virtual fields.

**ADD**

Enables you to add new filter phrases to an existing filter declaration without clearing previously defined filters.

*filter-defines*

Are virtual fields declared for use in filters. For more information, see [Usage Notes for Virtual Fields Used in Filters](#) on page 266.

*filtername1...filternamen*

Is the name by which the filter is referenced in subsequent SET FILTER commands. This name may be up to eight characters long and must be unique for a particular file name.

*text*

Describes the filter for documentation purposes. Text must fit on one line.

*where-if phrases*

Are screening conditions that can include all valid syntax. They may refer to data source fields and virtual fields in the Master File. They may not refer to virtual fields declared using a DEFINE command, or to other filter names.

**Reference:**      **Usage Notes for Virtual Fields Used in Filters**

Virtual fields used in filters:

- ☐ Are exclusively local to (or usable by) filters in a specific filter declaration.
- ☐ Cannot be referenced in a DEFINE or TABLE command.

- ☐ Support any syntax valid for virtual fields in a DEFINE command.
- ☐ Cannot reference virtual fields in a DEFINE command, but can reference virtual fields in the Master File.
- ☐ Do not count toward the display field limit, unlike virtual fields in DEFINE commands.
- ☐ Must all be declared before the first named filter.
- ☐ Must each end with a semi-colon.
- ☐ Cannot be enclosed between the DEFINE FILE and END commands.
- ☐ Cannot reuse a virtual field name for a the same file.

### **Example:** Declaring Filters

The first example creates the filter named UK, which consists of one WHERE condition. It also adds a definition for the virtual field MARK\_UP to the set of virtual fields already being used in filters for the CAR data source.

When a report request is issued for CAR, with UK activated, the condition WHERE MARK\_UP is greater than 1000 is automatically added to the request.

**Note:** The virtual field MARK\_UP cannot be explicitly displayed or referenced in the TABLE request.

```

FILTER FILE CAR ADD
MARK_UP/D7=RCOST-DCOST;
NAME=UK
WHERE MARK_UP GT 1000
END

```

The second example declares three named filters for the CAR data source: ASIA, UK, and LUXURY. The filter ASIA contains a textual description, for documentation purposes only. CLEAR, on the first line, erases any previously existing filters for CAR, as well any previously defined virtual fields used in filters for CAR, before it processes the new definitions.

```

FILTER FILE CAR CLEAR
NAME=ASIA,DESC=Asian cars only
IF COUNTRY EQ JAPAN
NAME=UK
IF COUNTRY EQ ENGLAND
NAME=LUXURY
IF RETAIL_COST GT 50000
END

```

### **Syntax:**      **How to Activate or Deactivate Filters**

Filters can be activated and deactivated with the command

```
SET FILTER= {*|xx|yy|zz|} IN {file|*} {ON|OFF}
```

where:

\*

Denotes all declared filters. This is the default value.

*xx, yy, zz*

Are the names of filters as declared in the NAME = syntax of the FILTER FILE command.

*file*

Is the name of the data source to which you are assigning screening conditions. \* denotes all data sources.

ON

Activates all (\*) or specifically named filters for the data source or all data sources (\*). The maximum number of filters you can activate for a data source is limited by the number of WHERE/IF phrases that the filters contain, not to exceed the limit of WHERE/IF criteria in any single report request.

OFF

Deactivates all (\*) or specifically named filters for the data source or all data sources (\*). OFF is the default value.

**Note:** The SET FILTER command is limited to one line. To activate more filters than fit on one line, issue additional SET FILTER commands. As long as you specify ON, the effect is cumulative.

### **Example:**      **Activating and Deactivating Filters**

The following commands activate A, B, C, D, E, F, and deactivate G (assuming that it was set ON, previously):

```
SET FILTER = A B C IN CAR ON  
SET FILTER = D E F IN CAR ON  
SET FILTER = G IN CAR OFF
```



The following commands activate some filters and deactivate others:

```
SET FILTER = UK LUXURY IN CAR ON
...
TABLE FILE CAR
PRINT COUNTRY MODEL RETAIL_COST
END
...
SET FILTER = LUXURY IN CAR OFF
TABLE FILE CAR
PRINT COUNTRY MODEL RETAIL_COST
END
```

The first SET FILTER command activates the filters UK and LUXURY, assigned to the CAR data source, and applies their screening conditions to any subsequent report request against the CAR data source.

The second SET FILTER command deactivates the filter LUXURY for the CAR data source. Unless LUXURY is reactivated, any subsequent report request against CAR will not apply the conditions in LUXURY, but will continue to apply UK.

### **Syntax:** How to Query the Status of Filters

To determine the status of existing filters, use

```
? FILTER [{file|*}] [SET] [ALL]
```

where:

*file*

Is the name of a Master File.

\*

Displays filters for all Master Files for which filters have been declared.

SET

Displays only active filters.

ALL

Displays all information about the filter, including its description and the exact WHERE/IF definition.

**Example:    Querying Filters**

To query filters, issue the following command:

```
FILTER FILE CAR CLEAR
NAME=BOTH, DESC=Asian and British cars only
IF COUNTRY EQ JAPAN AND ENGLAND
END
SET FILTER =BOTH IN CAR ON
TABLE FILE CAR
PRINT CAR RETAIL_COST
BY COUNTRY
END
```

The output is:

COUNTRY	CAR	RETAIL_COST
-----	---	-----
ENGLAND	JAGUAR	8,878
	JAGUAR	13,491
	JENSEN	17,850
	TRIUMPH	5,100
JAPAN	DATSUN	3,139
	TOYOTA	3,339

The following example queries filters for all data sources:

```
? FILTER
```

If no filters are defined, the following message displays:

```
NO FILTERS DEFINED
```

If filters are defined, the following screen displays:

Set	File	Filter name	Description
---	---	-----	-----
	CAR	ROB	Rob's selections
*	CAR	PETER	Peter's selections for CAR
*	EMPLOYEE	DAVE	Dave's tests
	EMPLOYEE	BRAD	Brad's tests

To query filters for the CAR data source, issue:

```
? FILTER CAR
```

If no filters are defined for the CAR data source, the following message displays:

```
NO FILTERS DEFINED FOR FILE NAMED CAR
```

If filters are defined for the CAR data source, the following screen displays:

Set	File	Filter name	Description
	CAR	ROB	Rob's selections
*	CAR	PETER	Peter's selections for CAR

To see all active filters, issue the following command:

```
? FILTER * SET
```

The output is:

Set	File	Filter name	Description
*	CAR	PETER	Peter's selections for CAR
*	EMPLOYEE	DAVE	Dave's tests

The asterisk in the first column indicates that a filter is activated.

## Preserving Filters Across Joins

By default, filters defined on the host data source are cleared by a JOIN command. However, filters can be maintained when a JOIN command is issued, by issuing the SET KEEPFILTERS=ON command.

Setting KEEPFILTERS to ON reinstates filter definitions and their individual declared status after a JOIN command. The set of filters and virtual fields defined prior to each join is called a context (see your documentation on SET KEEPDEFINES and on DEFINE FILE SAVE for information about contexts as they relate to virtual fields). Each new JOIN or DEFINE FILE command creates a new context.

If a new filter is defined after a JOIN command, it cannot have the same name as any previously defined filter unless you issue the FILTER FILE command with the CLEAR option. The CLEAR option clears all filter definitions for that data source in all contexts.

When a JOIN is cleared, each filter definition that was in effect prior to the JOIN command and that was not cleared, is reinstated with its original status. Clearing a join by issuing the JOIN CLEAR join\_name command removes all of the contexts and filter definitions that were created after the JOIN join\_name command was issued.

**Note:** When an error occurs because of a reference to field that does not exist in the original FILTER FILE, the filter is disabled even though KEEPFILTERS is set to ON.

**Syntax: How to Preserve Filter Definitions With KEEPFILTERS**

```
SET KEEPFILTERS = {OFF|ON}
```

where:

OFF

Does not preserve filters issued prior to a join. OFF is the default value.

ON

Preserves filters across joins.

**Example: Preserving Filters With KEEPFILTERS**

The first filter, UNITPR, is defined prior to issuing any joins, but after setting KEEPFILTERS to ON:

```
SET KEEPFILTERS = ON
FILTER FILE VIDEOTRK
PERUNIT/F5 = TRANSTOT/QUANTITY;
NAME=UNITPR
WHERE PERUNIT GT 2
WHERE LASTNAME LE 'CRUZ'
END
```

The ? FILTER command shows that the filter named UNITPR was created but not activated (activation is indicated by an asterisk in the SET column of the display:

```
? FILTER
```

```
SET FILE      FILTER NAME DESCRIPTION
-----
VIDEOTRK UNITPR
```

Next the filter is activated:

```
SET FILTER= UNITPR IN VIDEOTRK ON
```

The ? FILTER query shows that the filter is now activated:

```
? FILTER
```

```
SET FILE      FILTER NAME DESCRIPTION
-----
*  VIDEOTRK UNITPR
```

The following TABLE request is issued against the filtered data source:

```
TABLE FILE VIDEOTRK
SUM QUANTITY TRANSTOT BY LASTNAME
END
```

The output shows that the TABLE request retrieved only the data that satisfies the UNITPR filter:

```
NUMBER OF RECORDS IN TABLE=          6  LINES=          3
ACCESS LIMITED BY FILTERS
```

PAUSE.. PLEASE ISSUE CARRIAGE RETURN WHEN READY

LASTNAME	QUANTITY	TRANSTOT
CHANG	3	31.00
COLE	2	18.98
CRUZ	2	16.00

Now, the VIDEOTRK data source is joined to the MOVIES data source. The ? FILTER query shows that the join did not clear the UNITPR filter:

```
JOIN MOVIECODE IN VIDEOTRK TO ALL MOVIECODE IN MOVIES AS J1
```

The ? FILTER command shows that the UNITPR filter still exists and is still activated:

```
? FILTER
```

```
SET FILE      FILTER NAME DESCRIPTION
-----
*  VIDEOTRK UNITPR
```

Next a new filter, YEARS1, is created and activated for the join between VIDEOTRK and MOVIES:

```
FILTER FILE VIDEOTRK
YEARS/I5 = (EXPDATE - TRANSDATE)/365;
NAME=YEARS1
WHERE YEARS GT 1
END
SET FILTER= YEARS1 IN VIDEOTRK ON
```

The ? FILTER query shows that both the UNITPR and YEARS1 filters exist and are activated:

```
? FILTER
```

```
SET FILE      FILTER NAME DESCRIPTION
-----
*  VIDEOTRK UNITPR
*  VIDEOTRK YEARS1
```

Now, J1 is cleared. The output of the ? FILTER command shows that the YEARS1 filter that was created after the JOIN command was issued no longer exists. The UNITPR filter created prior to the JOIN command still exists with its original status:

```
JOIN CLEAR J1
? FILTER

SET FILE      FILTER NAME DESCRIPTION
-----
*    VIDEOTRK UNITPR
```

VSAM Record Selection Efficiencies

The most efficient way to retrieve selected records from a VSAM KSDS data source is by applying an IF screening test against the primary key. This results in a direct reading of the data using the data source's index. Only those records that you request are retrieved from the file. The alternative method of retrieval, the sequential read, forces the data adapter to retrieve all the records into storage.

Selection criteria that are based on the entire primary key, or on a subset of the primary key, cause direct reads using the index. A partial key is any contiguous part of the primary key beginning with the first byte.

IF selection tests performed against virtual fields can take advantage of these efficiencies as well, if the full or partial key is embedded in the virtual field.

The EQ and IS relations realize the greatest performance improvement over sequential reads. When testing on a partial key, equality logic is used to retrieve only the first segment instance of the screening value. To retrieve subsequent instances, NEXT logic is used.

Screening relations GE, FROM, FROM-TO, GT, EXCEEDS, IS-MORE-THAN, and NOT-FROM-TO all obtain some benefit from direct reads. The following example uses the index to find the record containing primary key value 66:

```
IF keyfield GE 66
```

It then continues to retrieve records by sequential processing, because VSAM stores records in ascending key sequence. The direct read is not attempted when the IF screening conditions NE, IS-NOT, CONTAINS, OMITS, LT, IS-LESS-THAN, LE, and NOT-FROM are used in the report request.

Reporting From Files With Alternate Indexes

Similar performance improvement is available for ESDS and KSDS files that use alternate indexes. An alternate index provides access to records in a key sequenced data set based on a key other than the primary key.

All benefits and limitations inherent with screening on the primary or partial key are applicable to screening on the alternate index or partial alternate index.

**Note:** It is not necessary to take an explicit indexed view to use the index.





## Creating Temporary Fields

---

When you create a report, you are not restricted to the fields that exist in your data source. If you can generate the information you want from the existing data, you can create a temporary field to evaluate and display it. A temporary field takes up no storage space in the data source. It is created only when needed.

### In this chapter:

- ☐ [What Is a Temporary Field?](#)
  - ☐ [Defining a Virtual Field](#)
  - ☐ [Creating a Calculated Value](#)
  - ☐ [Assigning Column Reference Numbers](#)
  - ☐ [Using FORECAST in a COMPUTE Command](#)
  - ☐ [Calculating Trends and Predicting Values With FORECAST](#)
  - ☐ [Calculating Trends and Predicting Values With Multivariate REGRESS](#)
  - ☐ [Using Text Fields in DEFINE and COMPUTE](#)
  - ☐ [Creating Temporary Fields Independent of a Master File](#)
- 

### What Is a Temporary Field?

A temporary field is a field whose value is not stored in the data source, but can be calculated from the data that is there, or assigned an absolute value. A temporary field takes up no storage space in the data source, and is created only when needed.

When you create a temporary field, you determine its value by writing an expression. You can combine fields, constants, and operators in an expression to produce a single value. For example, if your data contains salary and deduction amounts, you can calculate the ratio of deductions to salaries using the following expression:

```
deduction / salary
```

You can specify the expression yourself, or you can use one of the many supplied functions that perform specific calculations or manipulations. In addition, you can use expressions and functions as building blocks for more complex expressions, as well as use one temporary field to evaluate another.

### **Reference:** Types of Temporary Fields

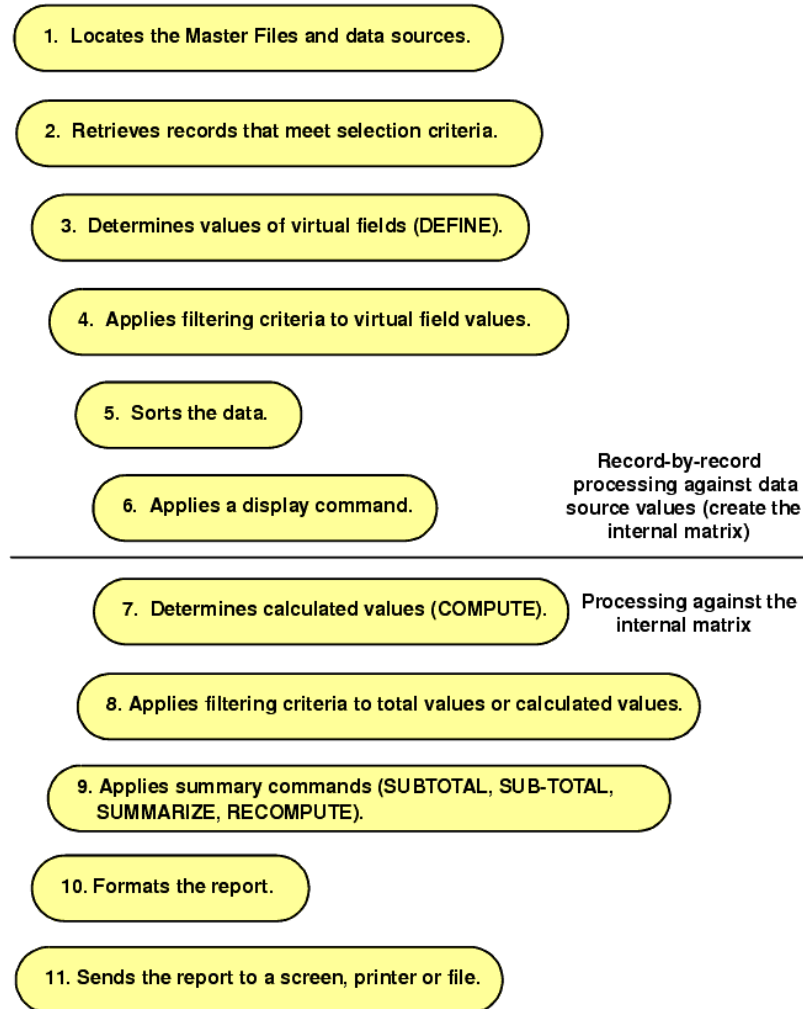
You can use two types of temporary fields (a *virtual field* and a *calculated value*), which differ in how they are evaluated:

A virtual field (DEFINE) is evaluated as each record that meets the selection criteria is retrieved from the data source. The result of the expression is treated as though it were a real field stored in the data source.

A calculated value (COMPUTE) is evaluated after all the data that meets the selection criteria is retrieved, sorted, and summed. Therefore, the calculation is performed using the aggregated values of the fields.

**Reference: Evaluation of Temporary Fields**

The following illustration shows how a request processes, and when each type of temporary field is evaluated:



**Example:**     **Distinguishing Between Virtual Fields and Calculated Values**

In the following example, both the DRATIO field (virtual field) and the CRATIO (calculated value) use the same expression DELIVER\_AMT/OPENING\_AMT, but do not return the same result. The value for CRATIO is calculated after all records have been selected, sorted, and aggregated. The virtual field DRATIO is calculated for each retrieved record.

```
DEFINE FILE SALES
DRATIO = DELIVER_AMT/OPENING_AMT;
END
TABLE FILE SALES
SUM DELIVER_AMT AND OPENING_AMT AND DRATIO
COMPUTE CRATIO = DELIVER_AMT/OPENING_AMT;
END
```

The output is:

<u>DELIVER_AMT</u>	<u>OPENING_AMT</u>	<u>DRATIO</u>	<u>CRATIO</u>
760	724	28.41	1.05

**Reference:**     **Selecting a Temporary Field**

The following is to help you choose the kind of temporary field you need.

**Choose a virtual field** when you want to:

- ☐ Use the temporary field to select data for your report. You cannot use a calculated value, since it is evaluated after data selection takes place.
- ☐ Use the temporary field to sort on data values. A calculated value is evaluated after the data is sorted. With the BY TOTAL phrase, you can sort on this type of field.

**Choose a calculated value** when you want to:

- ☐ Evaluate the temporary field using total values or prefix operators (which operate on total values). You cannot use a virtual field, since it is evaluated before any totaling takes place.
- ☐ Evaluate the temporary field using fields from different paths in the data structure. You cannot use a virtual field, since it is evaluated before the relationship between data in the different paths is established.

**Defining a Virtual Field**

A virtual field can be used in a request as though it is a real data source field. The calculation that determines the value of a virtual field is performed on each retrieved record that passes any screening conditions on real fields. The result of the expression is treated as though it were a real field stored in the data source.

You can define a virtual field in the following ways:

- ❑ **In a Master File.** These virtual fields are available whenever the data source is used for reporting. These fields cannot be cleared by JOIN or DEFINE FILE commands.

For more information, see the *Describing Data With WebFOCUS Language* manual.

- ❑ **In a procedure.** A virtual field created in a procedure lasts only for that procedure.

**Tip:** If your environment supports the KEEPDEFINES parameter, you can set KEEPDEFINES to ON to protect virtual fields from being cleared by a subsequent JOIN command. For details, see [Joining Data Sources](#) on page 1005.

### **Reference: Usage Notes for Creating Virtual Fields**

- ❑ If you do not use the KEEPDEFINES parameter, when a JOIN is issued, all pre-existing virtual fields for that data source are cleared except those defined in the Master File.
- ❑ To join structures using a virtual field with the source, make sure the DEFINE follows the JOIN command. Otherwise, the JOIN command clears the temporary field. For an explanation of reporting on joined data sources, see [Joining Data Sources](#) on page 1005.
- ❑ If no field in the expression is in the Master File or has been defined, use the WITH command to identify the logical home of the defined calculation. See [Establishing a Segment Location for a Virtual Field](#) on page 290.
- ❑ WITH can be used to move the logical home for the virtual field to a segment lower than that to which it would otherwise be assigned (for example, to count instances in a lower segment).
- ❑ You may define fields simultaneously (in addition to fields defined in the Master File) for as many data sources as desired. The total length of all virtual fields and real fields cannot exceed 32,000 characters.
- ❑ When you specify virtual fields in a request, they count toward the display field limit. For details on determining the maximum number of display fields that can be used in a request, see [Displaying Report Data](#) on page 43.
- ❑ Virtual fields are only available when the data source is used for reporting. Virtual fields cannot be used with MODIFY.
- ❑ A DEFINE command may not contain qualified field names on the left-hand side of the expression. If the same field name exists in more than one segment, and that field must be redefined or recomputed, use the REDEFINES command.

- ❑ Using a self-referencing DEFINE such as `x=x+1` disables AUTOPATH (see the *Developing Reporting Applications* manual).
- ❑ Field names used in the expression that defines the virtual field cannot be enclosed in single or double quotation marks. Any character string enclosed in quotation marks is treated as a literal string, not a field reference.
- ❑ A DEFINE FILE command overwrites a DEFINE in the Master File with same name as long as you do not redefine the format (which is not allowed).

### **Syntax:** How to Create a Virtual Field

Before you begin a report request, include

```
DEFINE FILE filename[.view_fieldname] [CLEAR|ADD]
fieldname[/format] [(GEOGRAPHIC_ROLE = georole)] [TITLE
'line1[,line2 ...']
[DESCRIPTION 'description']=expression;
fieldname[/format][WITH realfield]=expression;
fieldname[/format] REDEFINES qualifier.fieldname=expression;
.
.
.
END
```

where:

*filename*

Is the name of the data source for which you are defining the virtual field.

If the report request specifies an alternate view, use *filename* in conjunction with *view\_fieldname*.

All fields used to define the virtual field must lie on a single path in the data source. If they do not, you can use an alternate view, which requires alternate view DEFINE commands.

For an alternate view, virtual fields cannot have qualified field names. For information on alternate views, see [Rotating a Data Structure for Enhanced Retrieval](#) on page 1839.

The DEFINE FILE command line must be on a separate line from its virtual field definitions.

*view\_fieldname*

Is the field on which an alternate view is based in the corresponding request. You may need to use an alternate view if the fields used do not lie on a single path in the normal view.

CLEAR

Clears previously defined virtual fields associated with the specified data source. CLEAR is the default value.

**ADD**

Enables you to specify additional virtual fields for a data source without releasing any existing virtual fields. Omitting ADD produces the same results as the CLEAR option.

*fieldname*

Is a name that complies with WebFOCUS field naming rules. Indexed field names in FOCUS data sources must be less than or equal to 12 characters. It can be the name of a new virtual field that you are defining, or an existing field declared in the Master File, which you want to redefine.

The name can include any combination of letters, digits, and underscores (\_), and should begin with a letter.

Do not use field names of the type  $C_n$ ,  $E_n$ , or  $X_n$  (where  $n$  is any sequence of one or two digits), because they are reserved for other uses.

*format*

Is the format of the field. The default value is D12.2. For information on field formats, see the *Describing Data With WebFOCUS Language* manual.

*georole*

Is a valid geographic role. Geographic roles can be names, postal codes, ISO (International Organization for Standardization) codes, FIPS (Federal Information Processing Standards) codes, or NUTS (Nomenclature of Territorial Units for Statistics) codes. The following is a list of supported geographic roles.

- ☐ **ADDRESS\_FULL.** Full address.
- ☐ **ADDRESS\_LINE.** Number and street name.
- ☐ **CITY.** City name.
- ☐ **CONTINENT.** Continent name.
- ☐ **CONTINENT\_ISO2.** Continent ISO-3166 code.
- ☐ **COUNTRY.** Country name.
- ☐ **COUNTRY\_FIPS.** Country FIPS code.
- ☐ **COUNTRY\_ISO2.** Country ISO-3166-2 code.
- ☐ **COUNTRY\_ISO3.** Country ISO-3166-3 code.
- ☐ **GEOMETRY\_AREA.** Geometry area.
- ☐ **GEOMETRY\_LINE.** Geometry line.
- ☐ **GEOMETRY\_POINT.** Geometry point.

- ☐ **LATITUDE.** Latitude.
- ☐ **LONGITUDE.** Longitude.
- ☐ **NUTS0.** Country name (NUTS level 0).
- ☐ **NUTS0\_CC.** Country code (NUTS level 0).
- ☐ **NUTS1.** Region name (NUTS level 1).
- ☐ **NUTS1\_CC.** Region code (NUTS level 1).
- ☐ **NUTS2.** Province name (NUTS level 2).
- ☐ **NUTS2\_CC.** Province code (NUTS level 2).
- ☐ **NUTS3.** District name (NUTS level 3).
- ☐ **NUTS3\_CC.** District code (NUTS level 3).
- ☐ **POSTAL\_CODE.** Postal code.
- ☐ **STATE.** State name.
- ☐ **STATE\_FIPS.** State FIPS code.
- ☐ **STATE\_ISO\_SUB.** US State ISO subdivision code.
- ☐ **USCITY.** US city name.
- ☐ **USCITY\_FIPS.** US city FIPS code.
- ☐ **USCOUNTY.** US county name.
- ☐ **USCOUNTY\_FIPS.** US county FIPS code.
- ☐ **USSTATE.** US state name.
- ☐ **USSTATE\_ABBR.** US state abbreviation.
- ☐ **USSTATE\_FIPS.** US state FIPS code.
- ☐ **ZIP3.** US 3-digit postal code.
- ☐ **ZIP5.** US 5-digit postal code.

**WITH** *realfield*

Associates a virtual field with a data source segment containing a real field. For more information, see [Usage Notes for Creating Virtual Fields](#) on page 281.



*line1, line2...*

Are the lines of default column title to be displayed for the virtual field unless overridden by an AS phrase.

*description*

Is the description to be associated with the virtual field, enclosed in single quotation marks. The description displays in the tools that browse Master Files.

**REDEFINES** *qualifier.fieldname*

Enables you to redefine or recompute a field whose name exists in more than one segment. If you change the format of the field when redefining it, the length in the new format must be the same as or shorter than the original. In addition, conversion between alphanumeric and numeric data types is not supported.

*expression*

Can be an arithmetic or logical expression or function, evaluated to establish the value of *fieldname* (see [Using Expressions](#) on page 431). You must end each expression with a semicolon except for the last one, where the semicolon is optional.

Fields in the expression can be real data fields, data fields in data sources that are cross-referenced or joined, or previously defined virtual fields. For related information, see [Usage Notes for Creating Virtual Fields](#) on page 281.

**END**

Is required to end the DEFINE FILE command. END must be on its own line in the procedure.

**Note:** For information about missing attributes for virtual fields, see [MISSING Attribute in a DEFINE or COMPUTE Command](#) on page 975.

### **Example:** Defining a Virtual Field

In the following request, the value of **RATIO** is calculated by dividing the value of **DELIVER\_AMT** by **OPENING\_AMT**. The **DEFINE** command creates **RATIO** as a virtual field, which is used in the request as though it were a real field in the data source.

```
DEFINE FILE SALES
RATIO = DELIVER_AMT/OPENING_AMT;
END
TABLE FILE SALES
PRINT DELIVER_AMT AND OPENING_AMT AND RATIO
WHERE DELIVER_AMT GT 50
END
```

The output is:

DELIVER_AMT	OPENING_AMT	RATIO
-----	-----	-----
80	65	1.23
100	100	1.00
80	90	.89

### **Example:** Redefining a Field

The following request redefines the salary field in the EMPDATA data source to print asterisks for job titles that contain the word EXECUTIVE:

```
SET EXTENDNUM=OFF
DEFINE FILE EMPDATA
SALARY REDEFINES EMPDATA.SALARY =
  IF TITLE CONTAINS 'EXECUTIVE' THEN ELSE
    EMPDATA.SALARY;
END
TABLE FILE EMPDATA
SUM SALARY BY TITLE
WHERE TITLE CONTAINS 'MANAGER' OR 'MARKETING' OR 'SALES'
ON TABLE SET PAGE OFF
END
```

The output is:

TITLE	SALARY
-----	-----
EXEC MANAGER	\$54,100.00
EXECUTIVE MANAGER	*****
MANAGER	\$270,500.00
MARKETING DIRECTOR	\$176,800.00
MARKETING EXECUTIVE	*****
MARKETING SUPERVISOR	\$50,500.00
SALES EXECUTIVE	*****
SALES MANAGER	\$70,000.00
SALES SPECIALIST	\$82,000.00
SENIOR SALES EXEC.	\$43,400.00

**Example: Redefining a Field That Has the Same Name in Multiple Segments**

The following request joins the EMPDATA data source to itself. This creates a two-segment structure in which the names are the same in both segments. The request then redefines the salary field in the top segment (tag name ORIG) so that all names starting with the letter L are replaced by asterisks, and redefines the salary field in the child segment (tag name NEW) so that all names starting with the letter M are replaced by asterisks:

```
SET EXTENDNUM=OFF
JOIN PIN IN EMPDATA TAG ORIG TO PIN IN EMPDATA TAG NEW AS AJ
DEFINE FILE EMPDATA
SALARY/D12.2M REDEFINES ORIG.SALARY = IF LASTNAME LIKE 'L%' THEN
                                     999999999999 ELSE ORIG.SALARY;
SALARY/D12.2M REDEFINES NEW.SALARY = IF LASTNAME LIKE 'M%' THEN
                                     999999999999 ELSE NEW.SALARY * 1.2;
END
TABLE FILE EMPDATA
PRINT ORIG.SALARY AS 'ORIGINAL' NEW.SALARY AS 'NEW'
BY LASTNAME
WHERE LASTNAME FROM 'HIRSCHMAN' TO 'OLSON'
ON TABLE SET PAGE NOPAGE
END
```

The output is:

LASTNAME	ORIGINAL	NEW
-----	-----	---
HIRSCHMAN	\$62,500.00	\$75,000.00
KASHMAN	\$33,300.00	\$39,960.00
LASTRA	*****	\$138,000.00
LEWIS	*****	\$60,600.00
LIEBER	*****	\$62,400.00
LOPEZ	*****	\$31,680.00
MARTIN	\$49,000.00	*****
MEDINA	\$39,000.00	*****
MORAN	\$30,800.00	*****
NOZAWA	\$80,500.00	\$96,600.00
OLSON	\$30,500.00	\$36,600.00

**Defining Multiple Virtual Fields**

You may wish to have more than one set of virtual fields for the same data source, and to use some or all of the virtual fields in the request. The ADD option enables you to specify additional virtual fields without clearing existing ones. If you omit the ADD option, previously defined virtual fields in that data source are cleared.

If you want to clear a virtual field for a particular data source, use the CLEAR option.

**Syntax:**      **How to Add a Virtual Field to Existing Virtual Fields**

```
DEFINE FILE filename ADD
```

where:

*filename*

Is the data source.

**Example:**      **Adding Virtual Fields**

The following annotated example illustrates the use of the ADD and CLEAR options for virtual fields:

```
1. DEFINE FILE CAR
   ETYPE/A2=DECODE STANDARD (OHV O OHC O ELSE L);
   END
2. DEFINE FILE CAR ADD
   TAX/D8.2=IF MPG LT 15 THEN .06*RCOST
       ELSE .04*RCOST;
   FCOST = RCOST+TAX;
   END
```

1. The first DEFINE command creates the TYPE virtual field for the CAR data source. For information about the DECODE function, see the *Using Functions* manual.
2. Two or more virtual fields, TAX and FCOST, are created for the CAR data source. The ADD option allows you to reference ETYPE, TAX, and FCOST in future requests.

**Displaying Virtual Fields**

You can display all virtual fields with the ? DEFINE command.

**Syntax:**      **How to Display Virtual Fields**

```
? DEFINE
```

For more information, see the *Developing Reporting Applications* manual.

**Procedure:**      **How to Display Virtual Fields**

Click the *Defined Fields* tab in the Define tool.

## Clearing a Virtual Field

The following can clear a virtual field created in a procedure:

- ❑ A DEFINE FILE *filename* CLEAR command.
- ❑ A subsequent DEFINE command (without the ADD option), against the same data source.
- ❑ A join. When a join is created for a data source, all pre-existing virtual fields for that data source are cleared except those defined in the Master File. This may affect virtual fields used in an expression.
- ❑ A change in the value of the FIELDNAME SET parameter.

Unlike fields created in a procedure, virtual fields in the Master File are not cleared in the above ways.

To clear all virtual fields for all data sources, issue the following command:

```
DEFINE FILE * CLEAR
END
```

### **Example:** Clearing Virtual Fields

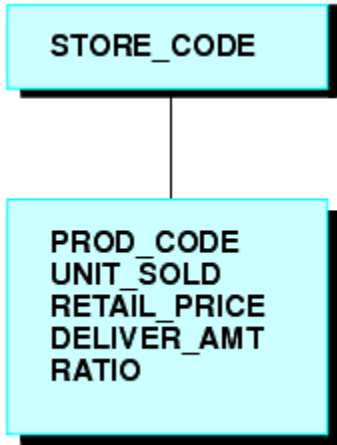
The following annotated example illustrates the use of the CLEAR options for virtual fields:

```
1. DEFINE FILE CAR
   ETYPE/A2=DECODE STANDARD (OHV O OHC O ELSE L);
   END
2. DEFINE FILE CAR CLEAR
   COST = RCOST-DCOST;
   END
```

1. The first DEFINE command creates the TYPE virtual field for the CAR data source. For information about the DECODE function, see the *Using Functions* manual.
2. The CLEAR option clears the previously defined virtual fields, and only the COST virtual field in the last DEFINE is available for further requests.

## Establishing a Segment Location for a Virtual Field

Virtual fields have a logical location in the data source structure, just like permanent data source fields. The logical home of a virtual field is on the lowest segment that has to be accessed in order to evaluate the expression, and determines the time of execution for that field. Consider the following data source structure and DEFINE command:



```
DEFINE RATIO = DELIVER_AMT/RETAIL_PRICE ;
```

The expression for RATIO includes at least one real data source field. As far as report capabilities are concerned, the field RATIO is just like a real field in the Master File, and is located in the lowest segment.

In some applications, you can have a virtual field evaluated by an expression that contains no real data source fields. Such an expression might refer only to temporary fields or literals. For example,

```
NCOUNT/I5 = NCOUNT+1;
```

or

```
DATE/YMD = '19990101';
```

Since neither expression contains a data source field (NCOUNT and the literal do not exist in the Master File), their logical positions in the data source cannot be determined. You have to specify in which segment you want the expression to be placed. To associate a virtual field with a specific segment, use the WITH phrase. The field name following WITH may be any real field in the Master File.

For FOCUS data sources, you may be able to increase the retrieval speed with an external index on the virtual field. In this case, you can associate the index with a target segment outside of the segment containing the virtual field. See the *Developing Reporting Applications* manual for more information on external indexes.

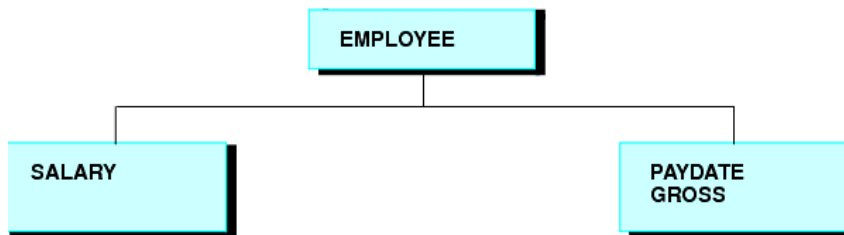
### **Example: Establishing a Segment Location**

The field NCOUNT is placed in the same segment as the UNITS field. NCOUNT is calculated each time a new segment instance is retrieved.

```
DEFINE FILE GGSales
NCOUNT/I5 WITH UNITS = NCOUNT+1;
END
```

### **Defining Virtual Fields Using a Multi-Path Data Source**

Calculations of a virtual field may include fields from all segments of a data source, but they must lie in a unique top-to-bottom path. Different virtual fields may, of course, lie along different paths. For example, consider the following data source structure:



This data source structure does not permit you to write the following expression:

```
NEWAMT = SALARY+GROSS;
```

The expression is invalid because the structure implies that there can be several SALARY segments for a given EMPLOYEE, and it is not clear which SALARY to associate with which GROSS.

To accomplish such an operation, you can use the alternate view option explained in *Improving Report Processing* on page 1839.

### **Increasing the Speed of Calculations in Virtual Fields**

Virtual fields are compiled into machine code in order to increase the speed of calculations.

## Preserving Virtual Fields Using DEFINE FILE SAVE and RETURN

Occasionally, new code needs to be added to an existing application. When adding code, there is always the possibility of over-writing existing virtual fields by reusing their names inadvertently.

The DEFINE FILE SAVE command forms a new context for virtual fields. Each new context creates a new layer or command environment. When you first enter the new environment, all of the virtual fields defined in the previous layer are available in the new layer. Overwriting or clearing a virtual field definition affects only the current layer. You can return to the default context with the DEFINE FILE RETURN command, and the virtual field definitions remain intact.

Therefore, all the virtual fields that are created in the new application can be removed before returning to the calling application, without affecting existing virtual fields in that application.

For an example of DEFINE FILE SAVE and DEFINE FILE RETURN, see [Joining Data Sources](#) on page 1005.

**Note:** A JOIN command can be issued after a DEFINE FILE SAVE command. However, in order to clear the join context, you must issue a JOIN CLEAR command if the join is still in effect. If only virtual fields and DEFINE FILE ADD were issued after a DEFINE FILE SAVE command, you can clear them by issuing a DEFINE FILE RETURN command.

### **Syntax:** How to Protect Virtual Fields From Being Overwritten

```
DEFINE FILE filename SAVE  
fld1/format1=expression1;  
fld2/format2=expression2;  
END  
TABLE FILE filename ...  
MODIFY FILE filename ...  
DEFINE FILE filename RETURN  
END
```

where:

#### **SAVE**

Creates a new context for virtual fields.

#### *filename*

Is the name of the Master File that gets a new context and has the subsequent virtual fields applied before the DEFINE FILE RETURN command is issued.

#### **RETURN**

Clears the current context if it was created by DEFINE FILE SAVE, and restores the previous context.



## Applying Dynamically Formatted Virtual Fields to Report Columns

Dynamic formatting enables you to apply different formats to specific data in a column by using a temporary field that contains dynamic data settings.

Before you can format a report column using the dynamic format, you must create the report, then apply the temporary field to a column in the report. For example, you can create a temporary field that contains different decimal currency formats for countries like Japan (which uses no decimal places) and England (which uses 2 decimal places). These currency formats are considered dynamic formats. You can then apply the temporary field containing the dynamic formatting to a Sales column. In a report, the Sales column reflects the different currency formats for each country.

The field that contains the format specifications can be:

- ☐ A real field in the data source.
- ☐ A temporary field created with a DEFINE command.
- ☐ A DEFINE in the Master File.
- ☐ A COMPUTE command. If the field is created with a COMPUTE command, the command must appear in the request prior to using the calculated field for reformatting.

The field that contains the formats must be alphanumeric, and at least eight characters in length. Only the first eight characters are used for formatting.

The field-based format may specify a length longer than the length of the original field. However, if the new length is more than one-third larger than the original length, the report column width may not be large enough to hold the value (indicated by asterisks in the field).

You can apply a field-based format to any type of field. However, the new format must be compatible with the original format:

- ☐ A numeric field can be reformatted to any other numeric format with any edit format options.
- ☐ An alphanumeric field can be reformatted to a different length.
- ☐ Any date field can be reformatted to any other date format type.
- ☐ Any date-time field can be reformatted to any other date-time format.

If the field-based format is invalid or specifies an impermissible type of conversion, the field displays with plus signs (+++++) on the report output.

**Syntax:**      **How to Define and Apply a Format Field**

- ❑ With a DEFINE command:

```
DEFINE FILE filename
format_field/A8 = expression;
END
```

- ❑ In a Master File:

```
DEFINE format_field/A8 = expression; $
```

- ❑ In a request:

```
COMPUTE format_field/A8 = expression;
```

where:

*format\_field*

Is the name of the field that contains the format for each row.

*expression*

Is the expression that assigns the format values to the format field.

After the format field is defined, you can apply it in a report request:

```
TABLE FILE filename
display fieldname/format_field[/just]
END
```

where:

*display*

Is any valid display command.

*fieldname*

Is a field in the request to be reformatted.

*format\_field*

Is the name of the field that contains the formats. If the name of the format field is the same as an explicit format, the explicit format is used. For example, a field named I8 cannot be used for field-based reformatting, because it is interpreted as the explicit format I8.

*just*

Is a justification option: L, R, or C. The justification option can be placed before or after the format field, separated from the format by a slash.

**Reference: Usage Notes for Field-Based Reformatting**

- ❑ Field-based reformatting is supported for TABLE and TABLEF. It works with StyleSheets, joins, and any type of data source.
- ❑ Field-based reformatting is not supported for MODIFY, Maintain, MATCH, GRAPH, RECAP, FOOTING, HEADING, or text fields.
- ❑ Although you can use a DEFINE or COMPUTE command to create the format field, you *cannot* apply a field-based format to a calculated or virtual field.
- ❑ Field-based reformatting cannot be used on a BY sort field. It does work with an ACROSS field.
- ❑ If a report column is produced using field-based reformatting, the format used for a total or subtotal of the column is taken from the previous detail line.
- ❑ Explicit reformatting creates two display fields internally for each field that is reformatted. Field-based reformatting creates three display fields.
- ❑ Field-based formats are applied at the final output phase of report processing, while specific formats are applied at the final output phase of report processing, while specific formats are applied prior to performing calculations. Therefore, the dynamically reformatted field will perform calculations, including summation, using the original format, while a field reformatted using a specific format will use the new format for calculations. Thus, there may be numeric differences in the final output because of rounding when using packed fields that reduce the precision.
- ❑ Field-based reformatting works for alphanumeric fields in a HOLD file, although three fields are stored in the file for each field that is reformatted. To prevent the extra fields from being propagated to the HOLD file, specify SET HOLDLIST=PRINTONLY.
- ❑ If the number of decimal places varies between rows, the decimal points are not aligned in the report output.

**Example: Creating Dynamically Formatted Fields**

The following request formats the DOLLARS2 field according to the value of the CATEGORY field and shows the numeric differences in sums using dynamic and static reformatting:

```
DEFINE FILE GGSales
MYFORMAT/A8=DECODE CATEGORY ('Coffee' 'P15.3' 'Gifts' 'P15.0' ELSE
'P15.2');
DOLLARS2/P15.2 = DOLLARS + .5;
END
```

```
TABLE FILE GGSales
SUM DOLLARS2/MYFORMAT AS 'Dynamic' DOLLARS2/P10.2 AS 'Specific'
BY CATEGORY
ON TABLE SUBTOTAL
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
END
```

The output is shown in the following image:

<u>Category</u>	<u>Dynamic</u>	<u>Specific</u>
Coffee	17232175.000	17232175.00
Food	17230053.00	17230053.00
Gifts	11696221.	11696220.50
<b>TOTAL</b>	<b>46158449.</b>	<b>46158448.50</b>

### Passing Function Calls Directly to a Relational Engine Using SQL.Function Syntax

The SQL adapters can pass virtual fields that call certain SQL scalar functions to the relational engine for processing. This enables you to use SQL functions in a request even when they have no equivalent in the WebFOCUS language. The function must be row-based and have a parameter list that consists of a comma-delimited list of column names or constants. In order to reference the function in an expression, prefix the function name with *SQL.*

If the virtual field is in the Master File, both TABLE requests and those SQL requests that qualify for Automatic Passthru (APT) can access the field. If the virtual field is created by a DEFINE FILE command, TABLE requests can access the field. The function name and parameters are passed without translation to the relational engine. Therefore, the expression that creates the DEFINE field must be optimized, or the request will fail.

#### **Reference:** Usage Notes for Direct SQL Function Calls

- ❑ The expression containing the *SQL.function* call must be optimized or the request will fail with the following message:

```
(FOC32605) NON OPTIMIZABLE EXPRESSION WITH SQL. SYNTAX
```

- ❑ The function must be a row-based scalar function and have a parameter list that consists of a comma-delimited list of column names or constants. If the function uses anything other than a list of comma separated values, the SQL. syntax cannot be used to pass it.

- ❑ Constant DEFINE fields must be assigned a segment location using the WITH phrase.
- ❑ Expressions should be declared as DEFINE fields, which are supported as parameters to an SQL function.
- ❑ Data types are not supported as parameters to an SQL function. Examples of data type arguments are CHAR and INT for the CONVERT function and ISO, EUR, JIS, and USA for the CHAR function.

### **Example:** Calling the SQL CONCAT Function in a Request

This example uses the WebFOCUS Retail demo sample. You can create this sample data source for a relational adapter by right-clicking the application in which you want to place this sample, and selecting *New* and then *Samples* from the context menu. Then, select *WebFOCUS - Retail Demo* from the *Sample procedures and data for* drop-down list and click *Create*.

The following request against the WebFOCUS Retail demo data source uses the SQL CONCAT function to concatenate the product category with the product subcategory.

```
SET TRACEUSER = ON
SET TRACEOFF = ALL
SET TRACEON = STMTRACE//CLIENT
SET TRACESTAMP=OFF
SET XRETRIEVAL = OFF

DEFINE FILE WF_RETAIL
CAT_SUBCAT/A50 = SQL.CONCAT(PRODUCT_CATEGORY, PRODUCT_SUBCAT);
END

TABLE FILE WF_RETAIL
PRINT CAT_SUBCAT
BY PRODUCT_CATEGORY NOPRINT
END
```

The trace output shows that the SQL function call was passed to the RDBMS.

```
SELECT
CONCAT(T2."PRODUCT_CATEGORY",T2."PRODUCT_SUBCAT"),
T2."PRODUCT_CATEGORY",
T2."PRODUCT_SUBCAT"
FROM
wfr_product T2
ORDER BY
T2."PRODUCT_CATEGORY"
FOR FETCH ONLY;
```

## Creating a Calculated Value

A calculated value is a temporary field that is evaluated after all the data that meets the selection criteria is retrieved, sorted, and summed. Calculated values are available only for the specified report request.

You specify the COMPUTE command in the body of the report request, following the display command and optionally introduced by AND. You can compute more than one field with a single COMPUTE command.

### **Reference:** Usage Notes for Calculated Field Values

The following apply to the use of calculated values:

- ❑ If you specify any optional COMPUTE phrases (such as, AS, IN, or NORPINT), and you compute additional fields following these phrases, you must repeat the commands COMPUTE or AND COMPUTE before specifying the additional fields.
- ❑ You can rename and justify column totals and row totals. For information, see the examples in [Including Totals and Subtotals](#) on page 369.
- ❑ Expressions in a COMPUTE command can include fields with prefix operators (see [Manipulating Display Fields With Prefix Operators](#) on page 60). For more information on valid expressions, see [Using Expressions](#) on page 431.
- ❑ Fields referred to in a COMPUTE command are counted toward the display field limit, and appear in the internal matrix. For details on determining the maximum number of display fields that can be used in a request, see [Displaying Report Data](#) on page 43.
- ❑ Field names used in the expression that defines the calculated field cannot be enclosed in single or double quotation marks. Any character string enclosed in quotation marks is treated as a literal string, not a field reference.
- ❑ When using a COMPUTE with an ACROSS COLUMNS phrase, the COLUMNS should be specified last:

`ACROSS acrossfield [AND] COMPUTE compute_expression; COLUMNS values`

**Syntax:**      **How to Create a Calculated Value**

```
COMPUTE fld [/format] [(GEOGRAPHIC_ROLE = georole)] = expression;
  [AS 'title'] [NOPRINT] [IN [+n]]
```

where:

*fld*

Is the name of the calculated value.

The name can be any name that complies with WebFOCUS field naming rules.

Do not use field names of the type *Cn*, *En*, and *Xn* (where *n* is any sequence of one or two digits), because they are reserved for other uses.

*format*

Is the format of the field. The default is D12.2. For information on formats, see the *Describing Data With WebFOCUS Language* manual.

*georole*

Is a valid geographic role. Geographic roles can be names, postal codes, ISO (International Organization for Standardization) codes, FIPS (Federal Information Processing Standards) codes, or NUTS (Nomenclature of Territorial Units for Statistics) codes. The following is a list of supported geographic roles.

- ☐ **ADDRESS\_FULL.** Full address.
- ☐ **ADDRESS\_LINE.** Number and street name.
- ☐ **CITY.** City name.
- ☐ **CONTINENT.** Continent name.
- ☐ **CONTINENT\_ISO2.** Continent ISO-3166 code.
- ☐ **COUNTRY.** Country name.
- ☐ **COUNTRY\_FIPS.** Country FIPS code.
- ☐ **COUNTRY\_ISO2.** Country ISO-3166-2 code.
- ☐ **COUNTRY\_ISO3.** Country ISO-3166-3 code.
- ☐ **GEOMETRY\_AREA.** Geometry area.
- ☐ **GEOMETRY\_LINE.** Geometry line.
- ☐ **GEOMETRY\_POINT.** Geometry point.

- ☐ **LATITUDE.** Latitude.
- ☐ **LONGITUDE.** Longitude.
- ☐ **NUTS0.** Country name (NUTS level 0).
- ☐ **NUTS0\_CC.** Country code (NUTS level 0).
- ☐ **NUTS1.** Region name (NUTS level 1).
- ☐ **NUTS1\_CC.** Region code (NUTS level 1).
- ☐ **NUTS2.** Province name (NUTS level 2).
- ☐ **NUTS2\_CC.** Province code (NUTS level 2).
- ☐ **NUTS3.** District name (NUTS level 3).
- ☐ **NUTS3\_CC.** District code (NUTS level 3).
- ☐ **POSTAL\_CODE.** Postal code.
- ☐ **STATE.** State name.
- ☐ **STATE\_FIPS.** State FIPS code.
- ☐ **STATE\_ISO\_SUB.** US State ISO subdivision code.
- ☐ **USCITY.** US city name.
- ☐ **USCITY\_FIPS.** US city FIPS code.
- ☐ **USCOUNTY.** US county name.
- ☐ **USCOUNTY\_FIPS.** US county FIPS code.
- ☐ **USSTATE.** US state name.
- ☐ **USSTATE\_ABBR.** US state abbreviation.
- ☐ **USSTATE\_FIPS.** US state FIPS code.
- ☐ **ZIP3.** US 3-digit postal code.
- ☐ **ZIP5.** US 5-digit postal code.

*expression*

Can be an arithmetic and/or logical expression or function (see [Using Expressions](#) on page 431). Each field used in the expression must be part of the request. Each expression must end with a semicolon (;).



**NOPRINT**

Suppresses printing of the field. For more information, see [Laying Out the Report Page](#) on page 1249.

**AS 'title'**

Changes the name of the calculated value. For more information, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.

**IN [+n]**

Specifies the location of the column. For more information, see [Using Headings, Footings, Titles, and Labels](#) on page 1431. IN only works in an HTML report when the STYLEMODE SET parameter is set to FIXED or OFF.

### **Syntax:**      **How to Create a Calculated Value Without a Calculation**

```
COMPUTE fld [/format]= ;
```

where:

*fld*

Is the name of the calculated value.

The name can be any name that complies with WebFOCUS field naming rules.

Do not use field names of the type Cn, En, and Xn (where n is any sequence of one or two digits), because they are reserved for other uses.

*format*

Is the format of the field. The default is D12.2. For information on formats, see the [Describing Data With WebFOCUS Language](#) manual.

### **Example:**      **Calculating a Field Value**

In the following example, the COMPUTE command creates a temporary field REVENUE based on the product of UNIT\_SOLD and RETAIL\_PRICE, and displays this information for New York City. The format D12.2M indicates the field format for REVENUE and the AS command changes the default column headings for UNIT\_SOLD and RETAIL\_PRICE. REVENUE is only available for this report request.

```
TABLE FILE SALES
HEADING CENTER
"NEW YORK PROFIT REPORT"
" "
SUM UNIT_SOLD AS 'UNITS,SOLD' RETAIL_PRICE AS 'RETAIL,PRICE'
COMPUTE REVENUE/D12.2M = UNIT_SOLD * RETAIL_PRICE;
BY PROD_CODE AS 'PROD,CODE'
WHERE CITY EQ 'NEW YORK'
END
```

The output is:

NEW YORK PROFIT REPORT			
PROD CODE	UNITS SOLD	RETAIL PRICE	REVENUE
----	-----	-----	-----
B10	30	\$.85	\$25.50
B17	20	\$1.89	\$37.80
B20	15	\$1.99	\$29.85
C13	15	\$1.99	\$29.85
C14	18	\$2.05	\$36.90
C17	12	\$2.09	\$25.08
D12	20	\$2.09	\$41.80
E1	30	\$.89	\$26.70
E2	33	\$.99	\$32.67
E3	35	\$1.09	\$38.15

## Using Positional Column Referencing With Calculated Values

In a COMPUTE command, it is sometimes convenient to refer to a field by its report column position, rather than its name. This option is especially useful when the same field is specified for several report columns.

Column referencing becomes essential when you are using the same field name in a variety of ways. The following image shows that columns produced by display commands (whether displayed or not) can be referred to as C1 for the first column, C2 for the second column, and so forth. The BY field columns are not counted.

For additional information about column reference numbers, see [Assigning Column Reference Numbers](#) on page 304.

### **Example:** Using Positional Column Referencing

The following example demonstrates positional field references in a COMPUTE command:

```
TABLE FILE CAR
SUM AVE.DEALER_COST
SUM AVE.DEALER_COST AND COMPUTE RATIO=C1/C2;
BY COUNTRY
END
```

The columns produced by display commands can be referred to as C1 for the first column (AVE.DEALER\_COST), C2 for the second column (AVE.DEALER\_COST BY COUNTRY), and so forth. The BY field columns are not counted.

The output is:

AVE DEALER_COST	COUNTRY	AVE DEALER_COST	RATIO
-----	-----	-----	-----
7,989	ENGLAND	9,463	.84
	FRANCE	4,631	1.73
	ITALY	10,309	.77
	JAPAN	2,756	2.90
	W GERMANY	7,795	1.02

### Using ACROSS With Calculated Values

If the COMPUTE command is issued immediately following an ACROSS phrase, only a recap type of the calculation is performed once for all columns. COMPUTE is used as part of a display command, so a new column is calculated for each set of values.

#### *Example:* Using COMPUTE as Part of a Display Command

```
TABLE FILE SALES
SUM UNIT_SOLD
COMPUTE NEWVAL = UNIT_SOLD * RETAIL_PRICE;
ACROSS CITY
END
```

The first page of output is:

CITY	NEWARK		STAMFORD		UNIONDALE		
NEW YORK	NEWVAL	UNIT_SOLD	NEWVAL	UNIT_SOLD	NEWVAL	UNIT_SOLD	NEWVAL
UNIT_SOLD							
-----	-----	-----	-----	-----	-----	-----	-----
162	1,764.18	42	104.16	376	4,805.28	65	297.70

#### *Example:* Using ACROSS With Calculated Values

In the following COMPUTE command, C1, C2, C3, C4, C5, and C6 are positional column references, and the COMPUTE command follows the ACROSS phrase. The COMPUTE is performed once for the report, and the results are displayed to the right of all sort groups.

```
TABLE FILE SALES
SUM UNIT_SOLD AND RETURNS
WHERE DATE GE '010' AND DATE LE '1031'
ACROSS DATE
COMPUTE
TOT_UNITS/D5=C1 + C3 + C5;
TOT_RETURNS = C2 + C4 + C6;
END
```

The output is:

DATE	10/18	10/19	TOT_UNITS	TOT_RETURNS		TOT_UNITS
10/17						
TOT_RETURNS						
UNIT_SOLD	RETURNS	UNIT_SOLD	RETURNS	UNIT_SOLD	RETURNS	
-----						
162	15	78	2	29	1	269
18.00						

Sorting Calculated Values

You can sort a report by a virtual field or a calculated value. To sort by a calculated value, you must use the BY TOTAL phrase in your request. For details, see [Sorting and Aggregating Report Columns](#) on page 176.

Screening on Calculated Values

You can screen on values produced by COMPUTE commands by using the WHERE TOTAL test, as described in [Selecting Records for Your Report](#) on page 219.

Assigning Column Reference Numbers

Column notation assigns a sequential column number to each column in the internal matrix created for a report request. If you want to control the creation of column reference numbers for the columns that are used in your report, use the CNOTATION column notation command.

Because column numbers refer to columns in the internal matrix, they are assigned after retrieval and aggregation of data are completed. Columns created and displayed in a report are stored in the internal matrix, and columns that are not displayed in a report may also be generated and stored in the internal matrix. Columns stored in the internal matrix include calculated values, reformatted field values, BY fields, fields with the NOPRINT option, and certain RECAP calculations such as FORECAST and REGRESS. Every other column in the internal matrix is assigned a column number by default which means you have to account for all internally generated columns if you want to refer to the appropriate column value in your request.

You can change the default assignment of column reference numbers by using the SET CNOTATION command which can assign column numbers only to columns that display in the report output or to all fields referenced in the report request. You can use column notation in COMPUTE and RECAP commands to refer to these columns in your request.

**Syntax:**      **How to Control the Creation of Column Reference Numbers**

```
SET CNOTATION={ALL | PRINTONLY | EXPLICIT}
```

where:

ALL

Assigns column reference numbers to every column in the internal matrix. ALL is the default value.

PRINTONLY

Assigns column reference numbers only to columns that display in the report output.

EXPLICIT

Assigns column reference numbers to all fields referenced in the request, whether displayed or not.

**Using Column Notation in a Report Request**

To create a column reference in a request, you can:

- ☐ Preface the column number with a C in a non-FML request.
- ☐ Use the column number as an index in conjunction with a row label in an FML request. With this type of notation, you can specify a specific column, a relative column number, or a sequence or series of columns.
- ☐ Refer to a particular cell in an FML request using the notation E(r,c), where *r* is a row number and *c* is a column number.

**Example:**      **Using Column Notation in a Non-FML Request With CNOTATION=ALL**

In the following request with CNOTATION=ALL, the product of C1 and C2 *does not* calculate TRANSTOT times QUANTITY because the reformatting generates additional columns.

```
SET CNOTATION = ALL
TABLE FILE VIDEOTRK
SUM TRANSTOT/D12.2 QUANTITY/D12.2
AND COMPUTE
PRODUCT = C1 * C2;
BY TRANSDATE
END
```

The output is:

TRANSDATE	TRANSTOT	QUANTITY	PRODUCT
91/06/17	57.03	12.00	3,252.42
91/06/18	21.25	2.00	451.56
91/06/19	38.17	5.00	1,456.95
91/06/20	14.23	3.00	202.49
91/06/21	44.72	7.00	1,999.88
91/06/24	126.28	12.00	15,946.63
91/06/25	47.74	8.00	2,279.11
91/06/26	40.97	2.00	1,678.54
91/06/27	60.24	9.00	3,628.85
91/06/28	31.00	3.00	961.00

BY fields do not get a column reference, so the first column reference is for TRANSTOT with its original format, then the reformatted version. Next is QUANTITY with its original format and then the reformatted version. Last is the calculated value, PRODUCT.

**Example: Using Column Notation in a Non-FML Request With CNOTATION=PRINTONLY**

Setting CNOTATION=PRINTONLY assigns column references to the output columns only. In this case, the product of C1 and C2 does calculate TRANSTOT times QUANTITY.

```
SET CNOTATION = PRINTONLY

TABLE FILE VIDEOTRK
SUM TRANSTOT/D12.2 QUANTITY/D12.2
AND COMPUTE
PRODUCT = C1 * C2;
BY TRANSDATE
END
```

The output is:

TRANSDATE	TRANSTOT	QUANTITY	PRODUCT
91/06/17	57.03	12.00	684.36
91/06/18	21.25	2.00	42.50
91/06/19	38.17	5.00	190.85
91/06/20	14.23	3.00	42.69
91/06/21	44.72	7.00	313.04
91/06/24	126.28	12.00	1,515.36
91/06/25	47.74	8.00	381.92
91/06/26	40.97	2.00	81.94
91/06/27	60.24	9.00	542.16
91/06/28	31.00	3.00	93.00

**Example: Using CNOTATION=PRINTONLY With Column Numbers in an FML Request**

In the following request, the reformatting of fields generates additional columns in the internal matrix. In the second RECAP expression, note that because of the CNOTATION setting:

❑ TOTCASH(1) refers to total cash in displayed column 1.

- ❑ TOTCASH(2) refers to total cash in displayed column 2.
- ❑ The resulting calculation is displayed in column 2 of the row labeled CASH GROWTH(%).
- ❑ The RECAP value is only calculated for the column specified.

```

SET CNOTATION=PRINTONLY
DEFINE FILE LEDGER
CUR_YR/I5C=AMOUNT;
LAST_YR/I5C=.87*CUR_YR - 142;
END
TABLE FILE LEDGER
SUM CUR_YR/F9.2 AS 'CURRENT,YEAR'
LAST_YR/F9.2 AS 'LAST,YEAR'

FOR ACCOUNT
1010 AS 'CASH ON HAND'                OVER
1020 AS 'DEMAND DEPOSITS'            OVER
1030 AS 'TIME DEPOSITS'              OVER
BAR                                   OVER
RECAP TOTCASH/F9.2C= R1 + R2 + R3; AS 'TOTAL CASH' OVER
" "                                   OVER
RECAP GROCASH(2)/F9.2C=100*TOTCASH(1)/TOTCASH(2) - 100;
AS 'CASH GROWTH(%)'
END

```

The output is:

	CURRENT <u>YEAR</u>	LAST <u>YEAR</u>
CASH ON HAND	8784.00	7216.00
DEMAND DEPOSITS	4494.00	3483.00
TIME DEPOSITS	7961.00	6499.00
	-----	-----
TOTAL CASH	21239.00	17198.00
CASH GROWTH(%)		23.50

**Example:**    **Using CNOTATION=PRINTONLY to RECAP Over Contiguous Columns in an FML Request**

In this example, the RECAP calculation for ATOT occurs only for displayed columns 2 and 3, as specified in the request. No calculation is performed for displayed column 1.

```
SET CNOTATION=PRINTONLY
DEFINE FILE LEDGER
CUR_YR/I5C=AMOUNT;
LAST_YR/I5C=.87*CUR_YR - 142;
NEXT_YR/I5C=1.13*CUR_YR + 222;
END
TABLE FILE LEDGER
SUM NEXT_YR/F9.2 CUR_YR/F9.2 LAST_YR/F9.2
FOR ACCOUNT
10$$ AS 'CASH'                                OVER
1100 AS 'ACCOUNTS RECEIVABLE'                 OVER
1200 AS 'INVENTORY'                           OVER
BAR                                             OVER
RECAP ATOT(2,3)/I5C = R1 + R2 + R3;
AS 'ASSETS ACTUAL'
END
```

The output is:

	<u>NEXT_YR</u>	<u>CUR_YR</u>	<u>LAST_YR</u>
CASH	25992.00	21239.00	17198.00
ACCOUNTS RECEIVABLE	21941.00	18829.00	15954.00
INVENTORY	31522.00	27307.00	23329.00
	-----	-----	-----
ASSETS ACTUAL		67,375	56,478



**Example: Using CNOTATION=PRINTONLY With Relative Column Addressing in an FML Request**

This example computes the change in cash (CHGCASH) for displayed columns 1 and 2.

```
SET CNOTATION=PRINTONLY
DEFINE FILE LEDGER
CUR_YR/I5C=AMOUNT;
LAST_YR/I5C=.87*CUR_YR - 142;
NEXT_YR/I5C=1.13*CUR_YR + 222;
END
TABLE FILE LEDGER
SUM NEXT_YR/F9.2 CUR_YR/F9.2 LAST_YR/F9.2
FOR ACCOUNT
10$$$ AS 'TOTAL CASH' LABEL TOTCASH          OVER
" "                                           OVER
RECAP CHGCASH(1,2)/I5C = TOTCASH(*) - TOTCASH(*+1); AS 'CHANGE IN CASH'
END
```

The output is:

	<u>NEXT_YR</u>	<u>CUR_YR</u>	<u>LAST_YR</u>
TOTAL CASH	25992.00	21239.00	17198.00
CHANGE IN CASH	4,752	4,044	

**Example: Using CNOTATION=PRINTONLY With Cell Notation in an FML Request**

In this request, two RECAP expressions derive VARIANCEs (EVAR and WVAR) by subtracting values in four displayed columns (1, 2, 3, 4) in row three (PROFIT); these values are identified using cell notation (r,c).

```
SET CNOTATION=PRINTONLY
TABLE FILE REGION
SUM E_ACTUAL/F9.2 E_BUDGET/F9.2 W_ACTUAL/F9.2 W_BUDGET/F9.2
FOR ACCOUNT
3000 AS 'SALES'          OVER
3100 AS 'COST'           OVER
BAR                      OVER
RECAP PROFIT/I5C = R1 - R2; OVER
" "                     OVER
RECAP EVAR(1)/I5C = E(3,1) - E(3,2);
AS 'EAST VARIANCE'      OVER
RECAP WVAR(3)/I5C = E(3,3) - E(3,4);
AS 'WEST VARIANCE'
END
```

The output is:

	<u>E ACTUAL</u>	<u>E BUDGET</u>	<u>W ACTUAL</u>	<u>W BUDGET</u>
SALES	6000.00	4934.00	7222.00	7056.00
COST	4650.00	3760.00	5697.00	5410.00
PROFIT	1,350	1,174	1,525	1,646
EAST VARIANCE	176			
WEST VARIANCE			-121	

**Example: Using NOPRINT, Field Reformatting, and COMPUTE With Column Notation**

The following request has a field that is not printed, several reformatted fields and three calculated values. With SET CNOTATION=PRINTONLY, the column references result in correct output.

```

SET CNOTATION = PRINTONLY
DEFINE FILE LEDGER
CUR_YR/I5C=AMOUNT;
LAST_YR/I5C=.87*CUR_YR - 142;
NEXT_YR/I5C=1.13*CUR_YR + 222;
END
TABLE FILE LEDGER
SUM NEXT_YR NOPRINT CUR_YR
COMPUTE AMT2/D6 = AMOUNT *2;
LAST_YR/D5      AMOUNT NEXT_YR
COMPUTE AMT3/D6 = AMOUNT*3;
COMPUTE AMT4/D6 = AMOUNT*4;
FOR ACCOUNT
10$$ AS 'CASH'                                OVER
1100 AS 'ACCTS. REC.'                          OVER
1200 AS 'INVENTORY'                          OVER
BAR
RECAP ATOT/I8C = R1 + R2 + R3; AS 'TOTAL'      OVER
RECAP DIFF(2,10,2)/D8 = ATOT(*) - ATOT(*-1);
END

```

The output is:

	<u>CUR_YR</u>	<u>AMT2</u>	<u>LAST_YR</u>	<u>AMOUNT</u>	<u>NEXT_YR</u>	<u>AMT3</u>	<u>AMT4</u>
CASH	21,239	42,478	17,198	21,239	25,992	63,717	84,956
ACCTS. REC.	18,829	37,658	15,954	18,829	21,941	56,487	75,316
INVENTORY	27,307	54,614	23,329	27,307	31,522	81,921	109,228
TOTAL	67,375	134,750	56,481	67,375	79,455	202,125	269,500
DIFF		67,375		10,894		122,670	

### **Example:** Using Column Notation With NOPRINT in a non-FML Request

The following request, sums TRANSTOT, QUANTITY, and TRANSCODE by TRANSDATE. TRANSTOT has the NOPRINT option, so it is not displayed on the report output. The request also calculates the following fields using COMPUTE commands:

- ❑ TTOT2, which has the same value as TRANSTOT and displays on the report output.
- ❑ UNIT\_COST1, which is calculated by dividing column1 by column2.
- ❑ UNIT\_COST2, which is calculated by dividing column1 by QUANTITY.

```
SET CNOTATION = ALL
TABLE FILE VIDEOTRK
SUM TRANSTOT/D7.2 NOPRINT QUANTITY/D7.2 TRANSCODE
  COMPUTE TTOT2/D7.2 = C1;
  COMPUTE UNIT_COST1/D7.2 = C1/C2;
  COMPUTE UNIT_COST2/D7.2 = C1/QUANTITY;
BY TRANSDATE
END
```

With this request, only CNOTATION=EXPLICIT produces the correct output. The following discussion illustrates why the EXPLICIT setting is needed.

With CNOTATION=ALL, all fields in the internal matrix are assigned column numbers. In particular, the request creates the following column references:

- ❑ C1 is TRANSTOT with its original format.
- ❑ C2 is TRANSTOT with format D7.2.
- ❑ C3 is QUANTITY with its original format.
- ❑ C4 is QUANTITY with format D7.2.
- ❑ C5 is TRANSCODE.

UNIT\_COST1 is C1/C2. These column numbers have both been assigned to TRANSTOT, so UNIT\_COST1 always equals 1. UNIT\_COST2 is C1 (TRANSTOT) divided by QUANTITY. The output is:

TRANSDATE	QUANTITY	TRANSCODE	TTOT2	UNIT_COST1	UNIT_COST2
91/06/17	12.00	10	57.03	1.00	4.75
91/06/18	2.00	2	21.25	1.00	10.63
91/06/19	5.00	4	38.17	1.00	7.63
91/06/20	3.00	3	14.23	1.00	4.74
91/06/21	7.00	6	44.72	1.00	6.39
91/06/24	12.00	9	126.28	1.00	10.52
91/06/25	8.00	7	47.74	1.00	5.97
91/06/26	2.00	2	40.97	1.00	20.48
91/06/27	9.00	7	60.24	1.00	6.69
91/06/28	3.00	3	31.00	1.00	10.33

With CNOTATION = PRINTONLY, the field TRANSTOT, which has the NOPRINT option, is not assigned any column numbers. QUANTITY with its original format is not assigned a column number because it is not displayed on the report output. The reformatted QUANTITY field is displayed and is assigned a column number. Therefore, the request creates the following column references:

❑ C1 is QUANTITY with format D7.2.

❑ C2 is TRANSCODE.

UNIT\_COST1 is C1/C2, QUANTITY/TRANSCODE. UNIT\_COST2 is C1 (QUANTITY) divided by QUANTITY. Therefore, UNIT\_COST2 always equals 1. The output is:

TRANSDATE	QUANTITY	TRANSCODE	TTOT2	UNIT_COST1	UNIT_COST2
91/06/17	12.00	10	12.00	1.20	1.00
91/06/18	2.00	2	2.00	1.00	1.00
91/06/19	5.00	4	5.00	1.25	1.00
91/06/20	3.00	3	3.00	1.00	1.00
91/06/21	7.00	6	7.00	1.17	1.00
91/06/24	12.00	9	12.00	1.33	1.00
91/06/25	8.00	7	8.00	1.14	1.00
91/06/26	2.00	2	2.00	1.00	1.00
91/06/27	9.00	7	9.00	1.29	1.00
91/06/28	3.00	3	3.00	1.00	1.00

With CNOTATION = EXPLICIT, the reformatted TRANSTOT field is explicitly referenced in the request, so it is assigned a column number even though it is not displayed. However, the TRANSTOT field with its original format is not assigned a column number. The QUANTITY field with its original format is not assigned a column number because it is not explicitly referenced in the request. The reformatted QUANTITY field is assigned a column number. Therefore, the request creates the following column references:

❑ C1 is TRANSTOT with format D7.2.

❑ C2 is QUANTITY with format D7.2.

❑ C3 is TRANSCODE.

UNIT\_COST1 is C1/C2, TRANSTOT/QUANTITY. UNIT\_COST2 is C1 (TRANSTOT) divided by QUANTITY. Therefore, UNIT\_COST2 always equals UNIT\_COST1. The output is:

TRANSDATE	QUANTITY	TRANSCODE	TTOT2	UNIT_COST1	UNIT_COST2
91/06/17	12.00	10	57.03	4.75	4.75
91/06/18	2.00	2	21.25	10.63	10.63
91/06/19	5.00	4	38.17	7.63	7.63
91/06/20	3.00	3	14.23	4.74	4.74
91/06/21	7.00	6	44.72	6.39	6.39
91/06/24	12.00	9	126.28	10.52	10.52
91/06/25	8.00	7	47.74	5.97	5.97
91/06/26	2.00	2	40.97	20.48	20.48
91/06/27	9.00	7	60.24	6.69	6.69
91/06/28	3.00	3	31.00	10.33	10.33

### Example: Using Cell Notation in an FML Request

In the following request, CUR\_YR has the NOPRINT option. The CHGCASH RECAP expression is supposed to subtract CUR\_YR from LAST\_YR and NEXT\_YR.

```
SET CNOTATION = ALL
DEFINE FILE LEDGER
CUR_YR/I7C = AMOUNT;
LAST_YR/I5C = .87*CUR_YR - 142;
NEXT_YR/I5C = 1.13*CUR_YR + 222;
END
TABLE FILE LEDGER
SUM CUR_YR/I5C NOPRINT LAST_YR NEXT_YR
FOR ACCOUNT
10$$ AS 'TOTAL CASH ' LABEL TOTCASH OVER
" " OVER
RECAP CHGCASH(1,3)/I5C=(TOTCASH(*) - TOTCASH(1));
AS 'CHANGE FROM CURRENT'
END
```

When CNOTATION = ALL, C1 refers to the CUR\_YR field with its original format, C2 refers to the reformatted value, C3 is LAST\_YR, and C4 is NEXT\_YR. Since there is an extra column and the RECAP only refers to columns 1 and 3, the calculation for NEXT\_YR - CUR\_YR is not performed. The output is:

	LAST_YR	NEXT_YR
	-----	-----
TOTAL CASH	17,195	25,991
CHANGE FROM CURRENT	-4,044	

When CNOTATION = PRINTONLY, the CUR\_YR field is not assigned any column number, so there is no column 3. Therefore, no calculations are performed. The output is:

	LAST_YR	NEXT_YR
	-----	-----
TOTAL CASH	17,195	25,991
CHANGE FROM CURRENT		

When CNOTATION = EXPLICIT, the reformatted version of the CUR\_YR field is C1 because it is referenced in the request even though it is not displayed. Both calculations are performed correctly. The output is:

	LAST_YR	NEXT_YR
	-----	-----
TOTAL CASH	17,195	25,991
CHANGE FROM CURRENT	-4,044	4,752

**Reference: Usage Notes for Column Numbers**

- ☐ BY fields are not assigned column numbers.
- ☐ ACROSS columns are assigned column numbers.
- ☐ Calculated fields are assigned column numbers.
- ☐ Column numbers outside the range of the columns created in the request are allowed under the following circumstances (and are treated as containing the value zero):
  - ☐ When specified in a COMPUTE command issued after an ACROSS phrase.
  - ☐ In a cell reference in an FML RECAP command.

In those cases, it is not possible to know in advance how many columns will be generated by the syntax. Using a column number outside of the range in any other context generates the following message:

(FOC258) FIELDNAME OR COMPUTATIONAL ELEMENT NOT RECOGNIZED: *column*

**Using FORECAST in a COMPUTE Command**

A version of the FORECAST feature was implemented for use in a RECAP command. However, the use of RECAP imposes limitations on placement of the FORECAST field in the output and use of sort fields.

Using FORECAST in a COMPUTE command eliminates these limitations and enables you to place the FORECAST calculation in a Master File. For the COMPUTE version of FORECAST, each type of calculation has its own version of the FORECAST function.

## Calculating Trends and Predicting Values With FORECAST

You can calculate trends in numeric data and predict values beyond the range of those stored in the data source by using the FORECAST feature. FORECAST can be used in a report or graph request.

The calculations you can make to identify trends and forecast values are:

- ❑ **Simple moving average (FORECAST\_MOVAVE).** Calculates a series of arithmetic means using a specified number of values from a field. For details, see [FORECAST\\_MOVAVE: Using a Simple Moving Average](#) on page 317.
- ❑ **Exponential moving average.** Calculates a weighted average between the previously calculated value of the average and the next data point. There are three methods for using an exponential moving average:
  - ❑ **Single exponential smoothing (FORECAST\_EXPAVE).** Calculates an average that allows you to choose weights to apply to newer and older values. For details, see [FORECAST\\_EXPAVE: Using Single Exponential Smoothing](#) on page 322.
  - ❑ **Double exponential smoothing (FORECAST\_DOUBLEXP).** Accounts for the tendency of data to either increase or decrease over time without repeating. For details, see [FORECAST\\_DOUBLEXP: Using Double Exponential Smoothing](#) on page 326.
  - ❑ **Triple exponential smoothing (FORECAST\_SEASONAL).** Accounts for the tendency of data to repeat itself in intervals over time. For details, see [FORECAST\\_SEASONAL: Using Triple Exponential Smoothing](#) on page 328.
- ❑ **Linear regression analysis (FORECAST\_LINEAR).** Derives the coefficients of a straight line that best fits the data points and uses this linear equation to estimate values. For details, see [FORECAST\\_LINEAR: Using a Linear Regression Equation](#) on page 332.

When predicting values in addition to calculating trends, FORECAST continues the same calculations beyond the data points by using the generated trend values as new data points. For the linear regression technique, the calculated regression equation is used to derive trend and predicted values.

FORECAST performs the calculations based on the data provided, but decisions about their use and reliability are the responsibility of the user. Therefore, the user is responsible for determining the reliability of the FORECAST predictions, based on the many factors that determine how accurate a prediction will be.

## FORECAST Processing

You invoke FORECAST processing by including one of the FORECAST functions in a COMPUTE command. FORECAST performs the specified calculation for all the existing data points and then continues them to generate the number of predicted values that you request. The parameters needed for FORECAST include the field to use in the calculations, the number of predictions to generate, and whether to display the input field values or the calculated values on the report output for the rows that represent existing data points.

FORECAST operates on the lowest sort field in the request. This is either the last ACROSS field in the request or, if the request does not contain an ACROSS field, it is the last BY field. The FORECAST calculations start over when the highest-level sort field changes its value. In a request with multiple display commands, FORECAST operates on the last ACROSS field (or if there are no ACROSS fields, the last BY field) of the last display command. When using an ACROSS field with FORECAST, the display command must be SUM or COUNT.

### **Reference:** Usage Notes for FORECAST

- ❑ The sort field used for FORECAST must be in a numeric or date format.
- ❑ When using simple moving average and exponential moving average methods, data values should be spaced evenly in order to get meaningful results.
- ❑ The use of column notation is not supported in the FORECAST expression. Column notation continues to be supported as before outside of this expression. The process of generating the FORECAST values creates extra columns that are not printed in the report output. The number and placement of these additional columns varies depending on the specific request.
- ❑ Missing values may lead to unexpected or unusable results and are not recommended for use with FORECAST\_LINEAR.
- ❑ If you use the ESTRECORDS parameter to enable the external sort to better estimate the amount of sort work space needed, you must take into account that FORECAST with predictions creates additional records in the output.
- ❑ In a styled report, you can assign specific attributes to values predicted by FORECAST with the StyleSheet attribute WHEN=FORECAST. For example, to make the predicted values display with the color red, use the following syntax in the TABLE request:

```
ON TABLE SET STYLE *  
TYPE=DATA,COLUMN=MYFORECASTSORTFIELD,WHEN=FORECAST,COLOR=RED,$  
ENDSTYLE
```



**Reference: FORECAST Limits**

The following are not supported with a COMPUTE command that uses FORECAST:

- ❑ BY TOTAL command.
- ❑ MORE, MATCH, FOR, and OVER phrases.

**FORECAST\_MOVAVE: Using a Simple Moving Average**

A simple moving average is a series of arithmetic means calculated with a specified number of values from a field. Each new mean in the series is calculated by dropping the first value used in the prior calculation, and adding the next data value to the calculation.

Simple moving averages are sometimes used to analyze trends in stock prices over time. In this scenario, the average is calculated using a specified number of periods of stock prices. A disadvantage to this indicator is that because it drops the oldest values from the calculation as it moves on, it loses its memory over time. Also, mean values are distorted by extreme highs and lows, since this method gives equal weight to each point.

Predicted values beyond the range of the data values are calculated using a moving average that treats the calculated trend values as new data points.

The first complete moving average occurs at the  $n^{\text{th}}$  data point because the calculation requires  $n$  values. This is called the lag. The moving average values for the lag rows are calculated as follows: the first value in the moving average column is equal to the first data value, the second value in the moving average column is the average of the first two data values, and so on until the  $n^{\text{th}}$  row, at which point there are enough values to calculate the moving average with the number of values specified.

**Syntax: How to Calculate a Simple Moving Average Column**

```
FORECAST_MOVAVE(display, infield, interval,  
                npredict, npoint1)
```

where:

*display*

Keyword

Specifies which values to display for rows of output that represent existing data. Valid values are:

- ❑ **INPUT\_FIELD.** This displays the original field values for rows that represent existing data.

- ❑ **MODEL\_DATA.** This displays the calculated values for rows that represent existing data.

**Note:** You can show both types of output for any field by creating two independent COMPUTE commands in the same request, each with a different display option.

*infield*

Is any numeric field. It can be the same field as the result field, or a different field. It cannot be a date-time field or a numeric field with date display options.

*interval*

Is the increment to add to each sort field value (after the last data point) to create the next value. This must be a positive integer. To sort in descending order, use the BY HIGHEST phrase. The result of adding this number to the sort field values is converted to the same format as the sort field.

For date fields, the minimal component in the format determines how the number is interpreted. For example, if the format is YMD, MDY, or DMY, an interval value of 2 is interpreted as meaning two days. If the format is YM, the 2 is interpreted as meaning two months.

*npredict*

Is the number of predictions for FORECAST to calculate. It must be an integer greater than or equal to zero. Zero indicates that you do not want predictions, and is only supported with a non-recursive FORECAST. For the SEASONAL method, npredict is the number of *periods* to calculate. The number of *points* generated is:

*nperiod \* npredict*

*npoint1*

Is the number of values to average for the MOVAVE method.

**Example: Calculating a New Simple Moving Average Column**

This request defines an integer value named PERIOD to use as the independent variable for the moving average. It predicts three periods of values beyond the range of the retrieved data. The MOVAVE column on the report output shows the calculated moving average numbers for existing data points.

```

DEFINE FILE GGSALES
SDATE/YYM = DATE;
SYEAR/Y = SDATE;
SMONTH/M = SDATE;
PERIOD/I2 = SMONTH;
END
TABLE FILE GGSALES
SUM UNITS DOLLARS
COMPUTE MOVAVE/D10.1= FORECAST_MOVAVE(MODEL_DATA, DOLLARS,1,3,3);
BY CATEGORY BY PERIOD
WHERE SYEAR EQ 97 AND CATEGORY NE 'Gifts'
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END

```

The output is:

<u>Category</u>	<u>PERIOD</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>	<u>MOVAVE</u>
Coffee	1	61666	801123	801,123.0
	2	54870	682340	741,731.5
	3	61608	765078	749,513.7
	4	57050	691274	712,897.3
	5	59229	720444	725,598.7
	6	58466	742457	718,058.3
	7	60771	747253	736,718.0
	8	54633	655896	715,202.0
	9	57829	730317	711,155.3
	10	57012	724412	703,541.7
	11	51110	620264	691,664.3
	12	58981	762328	702,334.7
	13	0	0	694,975.6
	14	0	0	719,879.4
	15	0	0	705,729.9
Food	1	54394	672727	672,727.0
	2	54894	699073	685,900.0
	3	52713	642802	671,534.0
	4	58026	718514	686,796.3
	5	53289	660740	674,018.7
	6	58742	734705	704,653.0
	7	60127	760586	718,677.0
	8	55622	695235	730,175.3
	9	55787	683140	712,987.0
	10	57340	713768	697,381.0
	11	57459	710138	702,348.7
	12	57290	705315	709,740.3
	13	0	0	708,397.8
	14	0	0	707,817.7
	15	0	0	708,651.9

In the report, the number of values to use in the average is 3 and there are no UNITS or DOLLARS values for the generated PERIOD values.

Each average (MOVAVE value) is computed using DOLLARS values where they exist. The calculation of the moving average begins in the following way:

- ☐ The first MOVAVE value (801,123.0) is equal to the first DOLLARS value.
- ☐ The second MOVAVE value (741,731.5) is the mean of DOLLARS values one and two:  
 $(801,123 + 682,340) / 2$ .

- ❑ The third MOVAVE value (749,513.7) is the mean of DOLLARS values one through three:  
 $(801,123 + 682,340 + 765,078) / 3$ .
- ❑ The fourth MOVAVE value (712,897.3) is the mean of DOLLARS values two through four:  
 $(682,340 + 765,078 + 691,274) / 3$ .

For predicted values beyond the supplied values, the calculated MOVAVE values are used as new data points to continue the moving average. The predicted MOVAVE values (starting with 694,975.6 for PERIOD 13) are calculated using the previous MOVAVE values as new data points. For example, the first predicted value (694,975.6) is the average of the data points from periods 11 and 12 (620,264 and 762,328) and the moving average for period 12 (702,334.7). The calculation is:  $694,975 = (620,264 + 762,328 + 702,334.7) / 3$ .

**Example:**     **Displaying Original Field Values in a Simple Moving Average Column**

This request defines an integer value named PERIOD to use as the independent variable for the moving average. It predicts three periods of values beyond the range of the retrieved data. It uses the keyword INPUT\_FIELD as the first argument in the FORECAST parameter list. The trend values do not display in the report. The actual data values for DOLLARS are followed by the predicted values in the report column.

```
DEFINE FILE GGSales
SDATE/YYM = DATE;
SYEAR/Y = SDATE;
SMONTH/M = SDATE;
PERIOD/I2 = SMONTH;
END
TABLE FILE GGSales
SUM UNITS DOLLARS
COMPUTE MOVAVE/D10.1 = FORECAST_MOVAVE(INPUT_FIELD,DOLLARS,1,3,3);
BY CATEGORY BY PERIOD
WHERE SYEAR EQ 97 AND CATEGORY NE 'Gifts'
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

The output is shown in the following image:

<u>Category</u>	<u>PERIOD</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>	<u>MOVAVE</u>
Coffee	1	61666	801123	801,123.0
	2	54870	682340	682,340.0
	3	61608	765078	765,078.0
	4	57050	691274	691,274.0
	5	59229	720444	720,444.0
	6	58466	742457	742,457.0
	7	60771	747253	747,253.0
	8	54633	655896	655,896.0
	9	57829	730317	730,317.0
	10	57012	724412	724,412.0
	11	51110	620264	620,264.0
	12	58981	762328	762,328.0
	13	0	0	694,975.6
	14	0	0	719,879.4
	15	0	0	705,729.9
Food	1	54394	672727	672,727.0
	2	54894	699073	699,073.0
	3	52713	642802	642,802.0
	4	58026	718514	718,514.0
	5	53289	660740	660,740.0
	6	58742	734705	734,705.0
	7	60127	760586	760,586.0
	8	55622	695235	695,235.0
	9	55787	683140	683,140.0
	10	57340	713768	713,768.0
	11	57459	710138	710,138.0
	12	57290	705315	705,315.0
	13	0	0	708,397.8
	14	0	0	707,817.7
	15	0	0	708,651.9

### FORECAST\_EXPAVE: Using Single Exponential Smoothing

The single exponential smoothing method calculates an average that allows you to choose weights to apply to newer and older values.

The following formula determines the weight given to the newest value.

$$k = 2 / (1 + n)$$

where:

$k$

Is the newest value.

*n*

Is an integer greater than one. Increasing *n* increases the weight assigned to the earlier observations (or data instances), as compared to the later ones.

The next calculation of the exponential moving average (EMA) value is derived by the following formula:

$$\text{EMA} = (\text{EMA} * (1-k)) + (\text{datavalue} * k)$$

This means that the newest value from the data source is multiplied by the factor *k* and the current moving average is multiplied by the factor (1-*k*). These quantities are then summed to generate the new EMA.

**Note:** When the data values are exhausted, the last data value in the sort group is used as the next data value.

### **Syntax:** How to Calculate a Single Exponential Smoothing Column

```
FORECAST_EXPAVE(display, infield, interval,  
               npredict, npoint1)
```

where:

*display*

Keyword

Specifies which values to display for rows of output that represent existing data. Valid values are:

- ☐ **INPUT\_FIELD.** This displays the original field values for rows that represent existing data.
- ☐ **MODEL\_DATA.** This displays the calculated values for rows that represent existing data.

**Note:** You can show both types of output for any field by creating two independent COMPUTE commands in the same request, each with a different display option.

*infield*

Is any numeric field. It can be the same field as the result field, or a different field. It cannot be a date-time field or a numeric field with date display options.

*interval*

Is the increment to add to each sort field value (after the last data point) to create the next value. This must be a positive integer. To sort in descending order, use the BY HIGHEST phrase. The result of adding this number to the sort field values is converted to the same format as the sort field.

For date fields, the minimal component in the format determines how the number is interpreted. For example, if the format is YMD, MDY, or DMY, an interval value of 2 is interpreted as meaning two days. If the format is YM, the 2 is interpreted as meaning two months.

### *npredict*

Is the number of predictions for FORECAST to calculate. It must be an integer greater than or equal to zero. Zero indicates that you do not want predictions, and is only supported with a non-recursive FORECAST.

### *npoint1*

For EXPAVE, this number is used to calculate the weights for each component in the average. This value must be a positive whole number. The weight,  $k$ , is calculated by the following formula:

$$k=2/(1+npoint1)$$

## **Example:** Calculating a Single Exponential Smoothing Column

The following defines an integer value named PERIOD to use as the independent variable for the moving average. It predicts three periods of values beyond the range of retrieved data.

```
DEFINE FILE GGSALES
SDATE/YYM = DATE;
SYEAR/Y = SDATE;
SMONTH/M = SDATE;
PERIOD/I2 = SMONTH;
END
TABLE FILE GGSALES
SUM UNITS DOLLARS
COMPUTE EXPAVE/D10.1= FORECAST_EXPAVE(MODEL_DATA,DOLLARS,1,3,3);
BY CATEGORY BY PERIOD
WHERE SYEAR EQ 97 AND CATEGORY NE 'Gifts'
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
```



The output is shown in the following image:

Category	PERIOD	Unit Sales	Dollar Sales	EXPAVE
-----	-----	-----	-----	-----
Coffee	1	61666	801123	801,123.0
	2	54870	682340	741,731.5
	3	61608	765078	753,404.8
	4	57050	691274	722,339.4
	5	59229	720444	721,391.7
	6	58466	742457	731,924.3
	7	60771	747253	739,588.7
	8	54633	655896	697,742.3
	9	57829	730317	714,029.7
	10	57012	724412	719,220.8
	11	51110	620264	669,742.4
	12	58981	762328	716,035.2
	13	0	0	739,181.6
	14	0	0	750,754.8
	15	0	0	756,541.4
Food	1	54394	672727	672,727.0
	2	54894	699073	685,900.0
	3	52713	642802	664,351.0
	4	58026	718514	691,432.5
	5	53289	660740	676,086.3
	6	58742	734705	705,395.6
	7	60127	760586	732,990.8
	8	55622	695235	714,112.9
	9	55787	683140	698,626.5
	10	57340	713768	706,197.2
	11	57459	710138	708,167.6
	12	57290	705315	706,741.3
	13	0	0	706,028.2
	14	0	0	705,671.6
	15	0	0	705,493.3

In the report, three predicted values of EXPAVE are calculated within each value of CATEGORY. For values outside the range of the data, new PERIOD values are generated by adding the interval value (1) to the prior PERIOD value.

Each average (EXPAVE value) is computed using DOLLARS values where they exist. The calculation of the moving average begins in the following way:

- ❑ The first EXPAVE value (801,123.0) is the same as the first DOLLARS value.
- ❑ The second EXPAVE value (741,731.5) is calculated as follows. Note that because of rounding and the number of decimal places used, the value derived in this sample calculation varies slightly from the one displayed in the report output:

`n=3 (number used to calculate weights)`

`k = 2/(1+n) = 2/4 = 0.5`

`EXPAVE = (EXPAVE*(1-k))+(new-DOLLARS*k) = (801123*0.5) + (682340*0.50) = 400561.5 + 341170 = 741731.5`

- ❑ The third EXPAVE value (753,404.8) is calculated as follows:

$$\text{EXPAVE} = (\text{EXPAVE} * (1 - k)) + (\text{new-DOLLARS} * k) = (741731.5 * 0.5) + (765078 * 0.5) = 370865.75 + 382539 = 753404.75$$

## FORECAST\_DOUBLEEXP: Using Double Exponential Smoothing

Double exponential smoothing produces an exponential moving average that takes into account the tendency of data to either increase or decrease over time without repeating. This is accomplished by using two equations with two constants.

- ❑ The first equation accounts for the current time period and is a weighted average of the current data value and the prior average, with an added component (b) that represents the trend for the previous period. The weight constant is k:

$$\text{DOUBLEEXP}(t) = k * \text{datavalue}(t) + (1 - k) * ((\text{DOUBLEEXP}(t-1)) + b(t-1))$$

- ❑ The second equation is the calculated trend value, and is a weighted average of the difference between the current and previous average and the trend for the previous time period. b(t) represents the average trend. The weight constant is g:

$$b(t) = g * (\text{DOUBLEEXP}(t) - \text{DOUBLEEXP}(t-1)) + (1 - g) * (b(t-1))$$

These two equations are solved to derive the smoothed average. The first smoothed average is set to the first data value. The first trend component is set to zero. For choosing the two constants, the best results are usually obtained by minimizing the mean-squared error (MSE) between the data values and the calculated averages. You may need to use nonlinear optimization techniques to find the optimal constants.

The equation used for forecasting beyond the data points with double exponential smoothing is

$$\text{forecast}(t+m) = \text{DOUBLEEXP}(t) + m * b(t)$$

where:

*m*

Is the number of time periods ahead for the forecast.

### Syntax:

## How to Calculate a Double Exponential Smoothing Column

```
FORECAST_DOUBLEEXP(display, infield,  
interval, npredict, npoint1, npoint2)
```

where:

#### *display*

Keyword

Specifies which values to display for rows of output that represent existing data. Valid values are:

- ☐ **INPUT\_FIELD.** This displays the original field values for rows that represent existing data.
- ☐ **MODEL\_DATA.** This displays the calculated values for rows that represent existing data.

**Note:** You can show both types of output for any field by creating two independent COMPUTE commands in the same request, each with a different display option.

#### *infield*

Is any numeric field. It can be the same field as the result field, or a different field. It cannot be a date-time field or a numeric field with date display options.

#### *interval*

Is the increment to add to each sort field value (after the last data point) to create the next value. This must be a positive integer. To sort in descending order, use the BY HIGHEST phrase. The result of adding this number to the sort field values is converted to the same format as the sort field.

For date fields, the minimal component in the format determines how the number is interpreted. For example, if the format is YMD, MDY, or DMY, an interval value of 2 is interpreted as meaning two days. If the format is YM, the 2 is interpreted as meaning two months.

#### *npredict*

Is the number of predictions for FORECAST to calculate. It must be an integer greater than or equal to zero. Zero indicates that you do not want predictions, and is only supported with a non-recursive FORECAST.

#### *nperiod*

For the SEASONAL method, it is a positive whole number that specifies the number of data points in a period.

#### *npoint1*

For DOUBLEXP, this number is used to calculate the weights for each component in the average. This value must be a positive whole number. The weight,  $k$ , is calculated by the following formula:

$$k=2/(1+npoint1)$$

*npoint2*

For DOUBLEXP, this positive whole number is used to calculate the weights for each term in the trend. The weight, g, is calculated by the following formula:

$$g=2/(1+npoint2)$$

**Example:**     **Calculating a Double Exponential Smoothing Column**

The following sums the ACTUAL\_YTD field of the CENTSTMT data source by period, and calculates a single exponential and double exponential moving average. The report columns show the calculated values for existing data points.

```
TABLE FILE CENTSTMT
SUM ACTUAL_YTD
COMPUTE EXP/D15.1 = FORECAST_EXP(AVE(MODEL_DATA,ACTUAL_YTD,1,0,3));
DOUBLEXP/D15.1 = FORECAST_DOUBLEXP(MODEL_DATA,ACTUAL_YTD,1,0,3,3);
BY PERIOD
WHERE GL_ACCOUNT LIKE '3%%%'
ON TABLE SET STYLE *
GRID=OFF,$
END
```

The output is shown in the following image:

PERIOD	YTD		
	Actual	EXP	DOUBLEXP
2002/01	12,957,681.	12,957,681.0	12,957,681.0
2002/02	25,441,971.	19,199,826.0	22,439,246.3
2002/03	39,164,321.	29,182,073.5	34,791,885.1
2002/04	52,733,326.	40,957,699.8	48,845,816.0
2002/05	66,765,920.	53,861,809.9	63,860,955.9
2002/06	80,952,492.	67,407,150.9	79,188,052.9

**FORECAST\_SEASONAL: Using Triple Exponential Smoothing**

Triple exponential smoothing produces an exponential moving average that takes into account the tendency of data to repeat itself in intervals over time. For example, sales data that is growing and in which 25% of sales always occur during December contains both trend and seasonality. Triple exponential smoothing takes both the trend and seasonality into account by using three equations with three constants.

For triple exponential smoothing you, need to know the number of data points in each time period (designated as L in the following equations). To account for the seasonality, a seasonal index is calculated. The data is divided by the prior season index and then used in calculating the smoothed average.

- ❑ The first equation accounts for the current time period, and is a weighted average of the current data value divided by the seasonal factor and the prior average adjusted for the trend for the previous period. The weight constant is k:

$$SEASONAL(t) = k * (datavalue(t)/I(t-L)) + (1-k) * (SEASONAL(t-1) + b(t-1))$$

- ❑ The second equation is the calculated trend value, and is a weighted average of the difference between the current and previous average and the trend for the previous time period. b(t) represents the average trend. The weight constant is g:

$$b(t) = g * (SEASONAL(t) - SEASONAL(t-1)) + (1-g) * (b(t-1))$$

- ❑ The third equation is the calculated seasonal index, and is a weighted average of the current data value divided by the current average and the seasonal index for the previous season. I(t) represents the average seasonal coefficient. The weight constant is p:

$$I(t) = p * (datavalue(t)/SEASONAL(t)) + (1 - p) * I(t-L)$$

These equations are solved to derive the triple smoothed average. The first smoothed average is set to the first data value. Initial values for the seasonality factors are calculated based on the maximum number of full periods of data in the data source, while the initial trend is calculated based on two periods of data. These values are calculated with the following steps:

1. The initial trend factor is calculated by the following formula:

$$b(0) = (1/L) ((y(L+1)-y(1))/L + (y(L+2)-y(2))/L + \dots + (y(2L) - y(L))/L)$$

2. The calculation of the initial seasonality factor is based on the average of the data values within each period, A(j) (1<=j<=N):

$$A(j) = ( y((j-1)L+1) + y((j-1)L+2) + \dots + y(jL) ) / L$$

3. Then, the initial periodicity factor is given by the following formula, where N is the number of full periods available in the data, L is the number of points per period and n is a point within the period (1<= n <= L):

$$I(n) = ( y(n)/A(1) + y(L+n)/A(2) + \dots + y((N-1)L+n)/A(N) ) / N$$

The three constants must be chosen carefully. The best results are usually obtained by choosing the constants to minimize the mean-squared error (MSE) between the data values and the calculated averages. Varying the values of npoint1 and npoint2 affect the results, and some values may produce a better approximation. To search for a better approximation, you may want to find values that minimize the MSE.

The equation used to forecast beyond the last data point with triple exponential smoothing is:

$$\text{forecast}(t+m) = (\text{SEASONAL}(t) + m * b(t)) / I(t-L+\text{MOD}(m/L))$$

where:

*m*

Is the number of periods ahead for the forecast.

### **Syntax:** How to Calculate a Triple Exponential Smoothing Column

```
FORECAST_SEASONAL(display, infield,  
interval, npredict, nperiod, npoint1, npoint2, npoint3)
```

where:

*display*

Keyword

Specifies which values to display for rows of output that represent existing data. Valid values are:

- ☐ **INPUT\_FIELD.** This displays the original field values for rows that represent existing data.
- ☐ **MODEL\_DATA.** This displays the calculated values for rows that represent existing data.

**Note:** You can show both types of output for any field by creating two independent COMPUTE commands in the same request, each with a different display option.

*infield*

Is any numeric field. It can be the same field as the result field, or a different field. It cannot be a date-time field or a numeric field with date display options.

*interval*

Is the increment to add to each sort field value (after the last data point) to create the next value. This must be a positive integer. To sort in descending order, use the BY HIGHEST phrase. The result of adding this number to the sort field values is converted to the same format as the sort field.

For date fields, the minimal component in the format determines how the number is interpreted. For example, if the format is YMD, MDY, or DMY, an interval value of 2 is interpreted as meaning two days. If the format is YM, the 2 is interpreted as meaning two months.

#### *npredict*

Is the number of predictions for FORECAST to calculate. It must be an integer greater than or equal to zero. Zero indicates that you do not want predictions, and is only supported with a non-recursive FORECAST. For the SEASONAL method, npredict is the number of *periods* to calculate. The number of *points* generated is:

$$nperiod * npredict$$

#### *nperiod*

For the SEASONAL method, is a positive whole number that specifies the number of data points in a period.

#### *npoint1*

For SEASONAL, this number is used to calculate the weights for each component in the average. This value must be a positive whole number. The weight, k, is calculated by the following formula:

$$k=2/(1+npoint1)$$

#### *npoint2*

For SEASONAL, this positive whole number is used to calculate the weights for each term in the trend. The weight, g, is calculated by the following formula:

$$g=2/(1+npoint2)$$

#### *npoint3*

For SEASONAL, this positive whole number is used to calculate the weights for each term in the seasonal adjustment. The weight, p, is calculated by the following formula:

$$p=2/(1+npoint3)$$

**Example: Calculating a Triple Exponential Smoothing Column**

In the following, the data has seasonality but no trend. Therefore, *npoint2* is set high (1000) to make the trend factor negligible in the calculation:

```
TABLE FILE VIDEOTRK
SUM TRANSTOT
COMPUTE SEASONAL/D10.1 = FORECAST_SEASONAL(MODEL_DATA,TRANSTOT,
1,3,3,3,1000,1);
BY TRANSDATE
WHERE TRANSDATE NE '19910617'
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

In the output, *npredict* is 3. Therefore, three periods (nine points, *nperiod* \* *npredict*) are generated.

TRANSDATE	TRANSTOT	SEASONAL
91/06/18	21.25	21.3
91/06/19	38.17	31.0
91/06/20	14.23	34.6
91/06/21	44.72	53.2
91/06/24	126.28	75.3
91/06/25	47.74	82.7
91/06/26	40.97	73.7
91/06/27	60.24	62.9
91/06/28	31.00	66.3
91/06/29		45.7
91/06/30		94.1
91/07/01		53.4
91/07/02		72.3
91/07/03		140.0
91/07/04		75.8
91/07/05		98.9
91/07/06		185.8
91/07/07		98.2

**FORECAST\_LINEAR: Using a Linear Regression Equation**

The linear regression equation estimates values by assuming that the dependent variable (the new calculated values) and the independent variable (the sort field values) are related by a function that represents a straight line:

$$y = mx + b$$

where:

*y*

Is the dependent variable.



$x$ 

Is the independent variable.

 $m$ 

Is the slope of the line.

 $b$ 

Is the y-intercept.

FORECAST\_LINEAR uses a technique called Ordinary Least Squares to calculate values for  $m$  and  $b$  that minimize the sum of the squared differences between the data and the resulting line.

The following formulas show how  $m$  and  $b$  are calculated.

$$m = \frac{(\sum xy - (\sum x \cdot \sum y)/n)}{(\sum x^2 - (\sum x)^2/n)}$$

$$b = (\sum y)/n - (m \cdot (\sum x)/n)$$

where:

 $n$ 

Is the number of data points.

 $y$ 

Is the data values (dependent variables).

 $x$ 

Is the sort field values (independent variables).

Trend values, as well as predicted values, are calculated using the regression line equation.

**Syntax:**      **How to Calculate a Linear Regression Column**

```
FORECAST_LINEAR(display, infield, interval,  
               npredict)
```

where:

*display*

Keyword

Specifies which values to display for rows of output that represent existing data. Valid values are:

- ☐ **INPUT\_FIELD.** This displays the original field values for rows that represent existing data.
- ☐ **MODEL\_DATA.** This displays the calculated values for rows that represent existing data.

**Note:** You can show both types of output for any field by creating two independent COMPUTE commands in the same request, each with a different display option.

*infield*

Is any numeric field. It can be the same field as the result field, or a different field. It cannot be a date-time field or a numeric field with date display options.

*interval*

Is the increment to add to each sort field value (after the last data point) to create the next value. This must be a positive integer. To sort in descending order, use the BY HIGHEST phrase. The result of adding this number to the sort field values is converted to the same format as the sort field.

For date fields, the minimal component in the format determines how the number is interpreted. For example, if the format is YMD, MDY, or DMY, an interval value of 2 is interpreted as meaning two days. If the format is YM, the 2 is interpreted as meaning two months.

*npredict*

Is the number of predictions for FORECAST to calculate. It must be an integer greater than or equal to zero. Zero indicates that you do not want predictions, and is only supported with a non-recursive FORECAST.

**Example: Calculating a New Linear Regression Field**

The following request calculates a regression line using the VIDEOTRK data source of QUANTITY by TRANSDATE. The interval is one day, and three predicted values are calculated.

```
TABLE FILE VIDEOTRK
SUM QUANTITY
COMPUTE FORTOT=FORECAST_LINEAR(MODEL_DATA,QUANTITY,1,3);
BY TRANSDATE
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

The output is shown in the following image:

<u>TRANSDATE</u>	<u>QUANTITY</u>	<u>FORTOT</u>
06/17/91	12	6.63
06/18/91	2	6.57
06/19/91	5	6.51
06/20/91	3	6.45
06/21/91	7	6.39
06/24/91	12	6.21
06/25/91	8	6.15
06/26/91	2	6.09
06/27/91	9	6.03
06/28/91	3	5.97
06/29/91		5.91
06/30/91		5.85
07/01/91		5.79

**Note:**

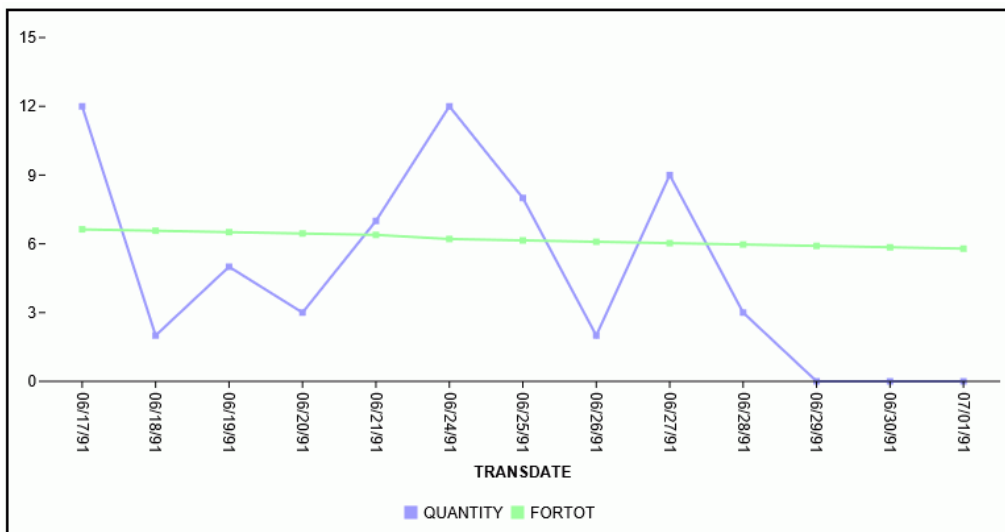
- ☐ Three predicted values of FORTOT are calculated. For values outside the range of the data, new TRANSDATE values are generated by adding the interval value (1) to the prior TRANSDATE value.
- ☐ There are no QUANTITY values for the generated FORTOT values.
- ☐ Each FORTOT value is computed using a regression line, calculated using all of the actual data values for QUANTITY.

TRANSDATE is the independent variable (x) and QUANTITY is the dependent variable (y).  
The equation is used to calculate QUANTITY FORECAST trend and predicted values.

The following version of the request charts the data values and the regression line.

```
GRAPH FILE VIDEOTRK
SUM QUANTITY
COMPUTE FORTOT=FORECAST_LINEAR(MODEL_DATA,QUANTITY,1,3);
BY TRANSDATE
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
END
```

The output is shown in the following image.



### Distinguishing Data Rows From Predicted Rows

To make the report output easier to interpret, you can create a field that indicates whether the FORECAST value in each row is a predicted value. To do this, define a virtual field whose value is a constant other than zero. Rows in the report output that represent actual records in the data source will appear with a value that is not zero. Rows that represent predicted values will display zero. You can also propagate this field to a HOLD file.

**Example: Distinguishing Data Rows From Predicted Rows**

In the following example, the DATA\_ROW virtual field has the value 1 for each row in the data source. It has the value zero for the predicted rows. The PREDICT field is calculated as YES for predicted rows, and NO for rows containing data. In addition, the StyleSheet attribute WHEN=FORECAST is used to display the predicted values for the FORTOT field in red.

```
DEFINE FILE VIDEOTRK
DATA_ROW/I1 = 1;
END
TABLE FILE VIDEOTRK
SUM TRANSTOT DATA_ROW
COMPUTE
PREDICT/A3 = IF DATA_ROW NE 0 THEN 'NO' ELSE 'YES' ;
FORTOT/D12.2=FORECAST_LINEAR(MODEL_DATA,TRANSTOT,1,3);
BY TRANSDATE
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
TYPE=DATA, COLUMN=FORTOT, WHEN=FORECAST, COLOR=RED,$
ENDSTYLE
END
```

The output is shown in the following image:

<u>TRANSDATE</u>	<u>TRANSTOT</u>	<u>DATA ROW</u>	<u>PREDICT</u>	<u>FORTOT</u>
06/17/91	57.03	10	NO	38.58
06/18/91	21.25	2	NO	40.32
06/19/91	38.17	4	NO	42.07
06/20/91	14.23	3	NO	43.81
06/21/91	44.72	6	NO	45.55
06/24/91	126.28	9	NO	50.78
06/25/91	47.74	7	NO	52.52
06/26/91	40.97	2	NO	54.26
06/27/91	60.24	7	NO	56.00
06/28/91	31.00	3	NO	57.74
06/29/91		0	YES	59.48
06/30/91		0	YES	61.23
07/01/91		0	YES	62.97

## Calculating Trends and Predicting Values With FORECAST

You can calculate trends in numeric data and predict values beyond the range of those stored in the data source by using the FORECAST feature. FORECAST can be used in a report or graph request.

The calculations you can make to identify trends and forecast values are:

- ❑ **Simple moving average (MOVAVE).** Calculates a series of arithmetic means using a specified number of values from a field. For details, see [Using a Simple Moving Average](#) on page 343.
- ❑ **Exponential moving average.** Calculates a weighted average between the previously calculated value of the average and the next data point. There are three methods for using an exponential moving average:
  - ❑ **Single exponential smoothing (EXPAVE).** Calculates an average that allows you to choose weights to apply to newer and older values. For details, see [Using Single Exponential Smoothing](#) on page 346.
  - ❑ **Double exponential smoothing (DOUBLEXP).** Accounts for the tendency of data to either increase or decrease over time without repeating. For details, see [Using Double Exponential Smoothing](#) on page 349.
  - ❑ **Triple exponential smoothing (SEASONAL).** Accounts for the tendency of data to repeat itself in intervals over time. For details, see [Using Triple Exponential Smoothing](#) on page 350.
- ❑ **Linear regression analysis (REGRESS).** Derives the coefficients of a straight line that best fits the data points and uses this linear equation to estimate values. For details, see [Usage Notes for Creating Virtual Fields](#) on page 281.

When predicting values in addition to calculating trends, FORECAST continues the same calculations beyond the data points by using the generated trend values as new data points. For the linear regression technique, the calculated regression equation is used to derive trend and predicted values.

FORECAST performs the calculations based on the data provided, but decisions about their use and reliability are the responsibility of the user. Therefore, FORECAST predictions are not always reliable, and many factors determine how accurate a prediction will be.

## FORECAST Processing

You invoke FORECAST processing by including FORECAST in a RECAP command. In this command, you specify the parameters needed for generating estimated values, including the field to use in the calculations, the type of calculation to use, and the number of predictions to generate. The RECAP field that contains the result of FORECAST can be a new field (non-recursive) or the same field used in the FORECAST calculations (recursive):

- ❑ In a recursive FORECAST, the RECAP field that contains the results is also the field used to generate the FORECAST calculations. In this case, the original field is not printed even if it was referenced in the display command, and the RECAP column contains the original field values followed by the number of predicted values specified in the FORECAST syntax. No trend values display in the report. However, the original column is stored in an output file unless you set HOLDLIST to PRINTONLY.
- ❑ In a non-recursive FORECAST, a new field contains the results of FORECAST calculations. The new field is displayed in the report along with the original field when it is referenced in the display command. The new field contains trend values and forecast values when specified.

FORECAST operates on the last ACROSS field in the request. If the request does not contain an ACROSS field, it operates on the last BY field. The FORECAST calculations start over when the highest-level sort field changes its value. In a request with multiple display commands, FORECAST operates on the last ACROSS field (or if there are no ACROSS fields, the last BY field) of the last display command. When using an ACROSS field with FORECAST, the display command must be SUM or COUNT.

**Note:** Although you pass parameters to FORECAST using an argument list in parentheses, FORECAST is not a function. It can coexist with a function of the same name, as long as the function is not specified in a RECAP command.

### **Syntax:** How to Calculate Trends and Predict Values

MOVAVE calculation

```
ON sortfield RECAP result_field[/fmt] = FORECAST(infield, interval,
  npredict, 'MOVAVE', npoint1) sendstyle
```

EXPAVE calculation

```
ON sortfield RECAP result_field[/fmt] = FORECAST(infield, interval,
  npredict, 'EXPAVE', npoint1);
```

### DOUBLEXP calculation

```
ON sortfield RECAP fld1[/fmt] = FORECAST(infield,  
interval, npredict, 'DOUBLEXP', npoint1, npoint2);
```

### SEASONAL calculation

```
ON sortfield RECAP fld1[/fmt] = FORECAST(infield,  
interval, npredict, 'SEASONAL', nperiod, npoint1, npoint2, npoint3);
```

### REGRESS calculation

```
ON sortfield RECAP result_field[/fmt] = FORECAST(infield, interval,  
npredict, 'REGRESS');
```

where:

#### *sortfield*

Is the last ACROSS field in the request. This field must be in numeric or date format. If the request does not contain an ACROSS field, FORECAST works on the last BY field.

#### *result\_field*

Is the field containing the result of FORECAST. It can be a new field, or the same as *infield*. This must be a numeric field; either a real field, a virtual field, or a calculated field.

**Note:** The word FORECAST and the opening parenthesis must be on the same line as the syntax *sortfield*=.

#### *fmt*

Is the display format for *result\_field*. The default format is D12.2. If *result\_field* was previously reformatted using a DEFINE or COMPUTE command, the format specified in the RECAP command is respected.

#### *infield*

Is any numeric field. It can be the same field as *result\_field*, or a different field. It cannot be a date-time field or a numeric field with date display options.

#### *interval*

Is the increment to add to each *sortfield* value (after the last data point) to create the next value. This must be a positive integer. To sort in descending order, use the BY HIGHEST phrase. The result of adding this number to the *sortfield* values is converted to the same format as *sortfield*.

For date fields, the minimal component in the format determines how the number is interpreted. For example, if the format is YMD, MDY, or DMY, an interval value of 2 is interpreted as meaning two days; if the format is YM, the 2 is interpreted as meaning two months.



*npredict*

Is the number of predictions for FORECAST to calculate. It must be an integer greater than or equal to zero. Zero indicates that you do not want predictions, and is only supported with a non-recursive FORECAST. For the SEASONAL method, npredict is the number of *periods* to calculate. The number of *points* generated is:

$$nperiod * npredict$$

*nperiod*

For the SEASONAL method, is a positive whole number that specifies the number of data points in a period.

*npoint1*

Is the number of values to average for the MOVAVE method. For EXPAVE, DOUBLEXP, and SEASONAL, this number is used to calculate the weights for each component in the average. This value must be a positive whole number. The weight, k, is calculated by the following formula:

$$k=2/(1+npoint1)$$

*npoint2*

For DOUBLEXP and SEASONAL, this positive whole number is used to calculate the weights for each term in the trend. The weight, g, is calculated by the following formula:

$$g=2/(1+npoint2)$$

*npoint3*

For SEASONAL, this positive whole number is used to calculate the weights for each term in the seasonal adjustment. The weight, p, is calculated by the following formula:

$$p=2/(1+npoint3)$$

**Reference: Usage Notes for FORECAST**

- ☐ The sort field used for FORECAST must be in a numeric or date format.
- ☐ When using simple moving average and exponential moving average methods, data values should be spaced evenly in order to get meaningful results.
- ☐ When using a RECAP command with FORECAST, the command can contain only the FORECAST syntax. FORECAST does not recognize any syntax after the closing semicolon (;). To specify options such as AS or IN:
  - ☐ In a non-recursive FORECAST request, use an empty COMPUTE command prior to the RECAP.

- ❑ In a recursive FORECAST request, specify the options when the field is first referenced in the report request.
- ❑ The use of column notation is not supported in a request that includes FORECAST. The process of generating the FORECAST values creates extra columns that are not printed in the report output. The number and placement of these additional columns varies depending on the specific request.
- ❑ A request can contain up to seven non-FORECAST RECAP commands and up to seven additional FORECAST RECAP commands.
- ❑ The left side of a RECAP command used for FORECAST supports the CURR attribute for creating a currency-denominated field.
- ❑ Recursive and non-recursive REGRESS are not supported in the same request when the display command is SUM, ADD, or WRITE.
- ❑ Missing values are not supported with REGRESS.
- ❑ If you use the ESTRECORDS parameter to enable the external sort to estimate better the amount of sort work space needed, you must take into account that FORECAST with predictions creates additional records in the output.
- ❑ In a styled report, you can assign specific attributes to values predicted by FORECAST with the StyleSheet attribute WHEN=FORECAST. For example, to make the predicted values display with the color red, use the following syntax in the TABLE request:

```
ON TABLE SET STYLE
*TYPE=DATA,COLUMN=MYFORECASTSORTFIELD,WHEN=FORECAST,COLOR=RED,
$ENDSTYLE
```

### **Reference:** FORECAST Limits

The following are not supported with a RECAP command that uses FORECAST:

- ❑ BY TOTAL command.
- ❑ MORE, MATCH, FOR, and OVER phrases.
- ❑ SUMMARIZE and RECOMPUTE are not supported for the same sort field used for FORECAST.
- ❑ MISSING attribute.

## Using a Simple Moving Average

A simple moving average is a series of arithmetic means calculated with a specified number of values from a field. Each new mean in the series is calculated by dropping the first value used in the prior calculation, and adding the next data value to the calculation.

Simple moving averages are sometimes used to analyze trends in stock prices over time. In this scenario, the average is calculated using a specified number of periods of stock prices. A disadvantage to this indicator is that because it drops the oldest values from the calculation as it moves on, it loses its memory over time. Also, mean values are distorted by extreme highs and lows, since this method gives equal weight to each point.

Predicted values beyond the range of the data values are calculated using a moving average that treats the calculated trend values as new data points.

The first complete moving average occurs at the  $n^{\text{th}}$  data point because the calculation requires  $n$  values. This is called the lag. The moving average values for the lag rows are calculated as follows: the first value in the moving average column is equal to the first data value, the second value in the moving average column is the average of the first two data values, and so on until the  $n^{\text{th}}$  row, at which point there are enough values to calculate the moving average with the number of values specified.

### *Example:* Calculating a New Simple Moving Average Column

This request defines an integer value named PERIOD to use as the independent variable for the moving average. It predicts three periods of values beyond the range of the retrieved data.

```
DEFINE FILE GGSales
  SDATE/YYM = DATE;
  SYEAR/Y = SDATE;
  SMONTH/M = SDATE;
  PERIOD/I2 = SMONTH;
END
TABLE FILE GGSales
  SUM UNITS DOLLARS
  BY CATEGORY BY PERIOD
  WHERE SYEAR EQ 97 AND CATEGORY NE 'Gifts'
  ON PERIOD RECAP MOVAVE/D10.1= FORECAST(DOLLARS,1,3,'MOVAVE',3);
END
```

The output is:

<u>Category</u>	<u>PERIOD</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>	<u>MOVAVE</u>
Coffee	1	61666	801123	801,123.0
	2	54870	682340	741,731.5
	3	61608	765078	749,513.7
	4	57050	691274	712,897.3
	5	59229	720444	725,598.7
	6	58466	742457	718,058.3
	7	60771	747253	736,718.0
	8	54633	655896	715,202.0
	9	57829	730317	711,155.3
	10	57012	724412	703,541.7
	11	51110	620264	691,664.3
	12	58981	762328	702,334.7
	13	0	0	694,975.6
	14	0	0	719,879.4
	15	0	0	705,729.9
Food	1	54394	672727	672,727.0
	2	54894	699073	685,900.0
	3	52713	642802	671,534.0
	4	58026	718514	686,796.3
	5	53289	660740	674,018.7
	6	58742	734705	704,653.0
	7	60127	760586	718,677.0
	8	55622	695235	730,175.3
	9	55787	683140	712,987.0
	10	57340	713768	697,381.0
	11	57459	710138	702,348.7
	12	57290	705315	709,740.3
	13	0	0	708,397.8
	14	0	0	707,817.7
	15	0	0	708,651.9

In the report, the number of values to use in the average is 3 and there are no UNITS or DOLLARS values for the generated PERIOD values.

Each average (MOVAVE value) is computed using DOLLARS values where they exist. The calculation of the moving average begins in the following way:

- ❑ The first MOVAVE value (801,123.0) is equal to the first DOLLARS value.
- ❑ The second MOVAVE value (741,731.5) is the mean of DOLLARS values one and two:  $(801,123 + 682,340) / 2$ .

- ❑ The third MOVAVE value (749,513.7) is the mean of DOLLARS values one through three:  
 $(801,123 + 682,340 + 765,078) / 3$ .
- ❑ The fourth MOVAVE value (712,897.3) is the mean of DOLLARS values two through four:  
 $(682,340 + 765,078 + 691,274) / 3$ .

For predicted values beyond the supplied values, the calculated MOVAVE values are used as new data points to continue the moving average. The predicted MOVAVE values (starting with 694,975.6 for PERIOD 13) are calculated using the previous MOVAVE values as new data points. For example, the first predicted value (694,975.6) is the average of the data points from periods 11 and 12 (620,264 and 762,328) and the moving average for period 12 (702,334.7). The calculation is:  $694,975 = (620,264 + 762,328 + 702,334.7) / 3$ .

**Example:**    **Using an Existing Field as a Simple Moving Average Column**

This request defines an integer value named PERIOD to use as the independent variable for the moving average. It predicts three periods of values beyond the range of the retrieved data. It uses the same name for the RECAP field as the first argument in the FORECAST parameter list. The trend values do not display in the report. The actual data values for DOLLARS are followed by the predicted values in the report column.

```
DEFINE FILE GGSales
  SDATE/YYM = DATE;
  SYEAR/Y = SDATE;
  SMONTH/M = SDATE;
  PERIOD/I2 = SMONTH;
END
TABLE FILE GGSales
  SUM UNITS DOLLARS
  BY CATEGORY BY PERIOD
  WHERE SYEAR EQ 97 AND CATEGORY NE 'Gifts'
  ON PERIOD RECAP DOLLARS/D10.1 = FORECAST(DOLLARS,1,3,'MOVAVE',3);
END
```

The output is:

<u>Category</u>	<u>PERIOD</u>	<u>Unit Sales</u>	<u>DOLLARS</u>
Coffee	1	61666	801,123.0
	2	54870	682,340.0
	3	61608	765,078.0
	4	57050	691,274.0
	5	59229	720,444.0
	6	58466	742,457.0
	7	60771	747,253.0
	8	54633	655,896.0
	9	57829	730,317.0
	10	57012	724,412.0
	11	51110	620,264.0
	12	58981	762,328.0
	13	0	694,975.0
	14	0	719,879.0
	15	0	705,729.0
Food	1	54394	672,727.0
	2	54894	699,073.0
	3	52713	642,802.0
	4	58026	718,514.0
	5	53289	660,740.0
	6	58742	734,705.0
	7	60127	760,586.0
	8	55622	695,235.0
	9	55787	683,140.0
	10	57340	713,768.0
	11	57459	710,138.0
	12	57290	705,315.0
	13	0	708,397.0
	14	0	707,817.0
	15	0	708,651.0

## Using Single Exponential Smoothing

The single exponential smoothing method calculates an average that allows you to choose weights to apply to newer and older values.

The following formula determines the weight given to the newest value.

$$k = 2 / (1 + n)$$

where:

$k$

Is the newest value.

*n*

Is an integer greater than one. Increasing *n* increases the weight assigned to the earlier observations (or data instances), as compared to the later ones.

The next calculation of the exponential moving average (EMA) value is derived by the following formula:

$$\text{EMA} = (\text{EMA} * (1-k)) + (\text{datavalue} * k)$$

This means that the newest value from the data source is multiplied by the factor *k* and the current moving average is multiplied by the factor (1-*k*). These quantities are then summed to generate the new EMA.

**Note:** When the data values are exhausted, the last data value in the sort group is used as the next data value.

### **Example:** Calculating a Single Exponential Smoothing Column

The following defines an integer value named PERIOD to use as the independent variable for the moving average. It predicts three periods of values beyond the range of retrieved data.

```
DEFINE FILE GGSales
  SDATE/YYM = DATE;
  SYEAR/Y = SDATE;
  SMONTH/M = SDATE;
  PERIOD/I2 = SMONTH;
END
TABLE FILE GGSales
  SUM UNITS DOLLARS
  BY CATEGORY BY PERIOD
  WHERE SYEAR EQ 97 AND CATEGORY NE 'Gifts'
  ON PERIOD RECAP EXPAVE/D10.1= FORECAST(DOLLARS,1,3,'EXPAVE',3);
END
```

The output is:

<u>Category</u>	<u>PERIOD</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>	<u>EXPAVE</u>
Coffee	1	61666	801123	801,123.0
	2	54870	682340	741,731.5
	3	61608	765078	753,404.8
	4	57050	691274	722,339.4
	5	59229	720444	721,391.7
	6	58466	742457	731,924.3
	7	60771	747253	739,588.7
	8	54633	655896	697,742.3
	9	57829	730317	714,029.7
	10	57012	724412	719,220.8
	11	51110	620264	669,742.4
	12	58981	762328	716,035.2
	13	0	0	716,035.2
	14	0	0	716,035.2
	15	0	0	716,035.2
Food	1	54394	672727	672,727.0
	2	54894	699073	685,900.0
	3	52713	642802	664,351.0
	4	58026	718514	691,432.5
	5	53289	660740	676,086.3
	6	58742	734705	705,395.6
	7	60127	760586	732,990.8
	8	55622	695235	714,112.9
	9	55787	683140	698,626.5
	10	57340	713768	706,197.2
	11	57459	710138	708,167.6
	12	57290	705315	706,741.3
	13	0	0	706,028.3
	14	0	0	705,671.6
	15	0	0	705,493.3

In the report, three predicted values of EXPAVE are calculated within each value of CATEGORY. For values outside the range of the data, new PERIOD values are generated by adding the interval value (1) to the prior PERIOD value.

Each average (EXPAVE value) is computed using DOLLARS values where they exist. The calculation of the moving average begins in the following way:

- ❑ The first EXPAVE value (801,123.0) is the same as the first DOLLARS value.
- ❑ The second EXPAVE value (741,731.5) is calculated as follows. Note that because of rounding and the number of decimal places used, the value derived in this sample calculation varies slightly from the one displayed in the report output:



`n=3 (number used to calculate weights)`

`k = 2/(1+n) = 2/4 = 0.5`

`EXPAVE = (EXPAVE*(1-k))+(new-DOLLARS*k) = (801123*0.5) + (682340*0.50) = 400561.5 + 341170 = 741731.5`

- ❑ The third EXPAVE value (753,404.8) is calculated as follows:

`EXPAVE = (EXPAVE*(1-k))+(new-DOLLARS*k) = (741731.5*0.5)+(765078*0.50) = 370865.75 + 382539 = 753404.75`

For predicted values beyond those supplied, the last EXPAVE value is used as the new data point in the exponential smoothing calculation. The predicted EXPAVE values (starting with 706,741.6) are calculated using the previous average and the new data point. Because the previous average is also used as the new data point, the predicted values are always equal to the last trend value. For example, the previous average for period 13 is 706,741.6, and this is also used as the next data point. Therefore, the average is calculated as follows:  $(706,741.6 * 0.5) + (706,741.6 * 0.5) = 706,741.6$

`EXPAVE = (EXPAVE * (1-k)) + (new-DOLLARS * k) = (706741.6*0.5) + (706741.6*0.50) = 353370.8 + 353370.8 = 706741.6`

## Using Double Exponential Smoothing

Double exponential smoothing produces an exponential moving average that takes into account the tendency of data to either increase or decrease over time without repeating. This is accomplished by using two equations with two constants.

- ❑ The first equation accounts for the current time period and is a weighted average of the current data value and the prior average, with an added component (b) that represents the trend for the previous period. The weight constant is k:

`DOUBLEEXP(t) = k * datavalue(t) + (1-k) * ((DOUBLEEXP(t-1) + b(t-1))`

- ❑ The second equation is the calculated trend value, and is a weighted average of the difference between the current and previous average and the trend for the previous time period. b(t) represents the average trend. The weight constant is g:

`b(t) = g * (DOUBLEEXP(t)-DOUBLEEXP(t-1)) + (1 - g) * (b(t-1))`

These two equations are solved to derive the smoothed average. The first smoothed average is set to the first data value. The first trend component is set to zero. For choosing the two constants, the best results are usually obtained by minimizing the mean-squared error (MSE) between the data values and the calculated averages. You may need to use nonlinear optimization techniques to find the optimal constants.

The equation used for forecasting beyond the data points with double exponential smoothing is

$$\text{forecast}(t+m) = \text{DOUBLEXP}(t) + m * b(t)$$

where:

$m$

Is the number of time periods ahead for the forecast.

**Example:**     **Calculating a Double Exponential Smoothing Column**

The following defines an integer value named PERIOD to use as the independent variable for the moving average. The double exponential smoothing method estimates the trend in the data points better than the single smoothing method:

```
SET HISTOGRAM = OFF
TABLE FILE CENTSMT
SUM ACTUAL_YTD
  BY PERIOD
  ON PERIOD RECAP EXP/D15.1 = FORECAST(ACTUAL_YTD,1,0,'EXPAVE',3);
  ON PERIOD RECAP DOUBLEXP/D15.1 = FORECAST(ACTUAL_YTD,1,0,
    'DOUBLEXP',3,3);
WHERE GL_ACCOUNT LIKE '3%%%'
END
```

The output is:

PERIOD	YTD		
	Actual	EXP	DOUBLEXP
2002/01	12,957,681.	12,957,681.0	12,957,681.0
2002/02	25,441,971.	19,199,826.0	22,439,246.3
2002/03	39,164,321.	29,182,073.5	34,791,885.1
2002/04	52,733,326.	40,957,699.8	48,845,816.0
2002/05	66,765,920.	53,861,809.9	63,860,955.9
2002/06	80,952,492.	67,407,150.9	79,188,052.9

**Using Triple Exponential Smoothing**

Triple exponential smoothing produces an exponential moving average that takes into account the tendency of data to repeat itself in intervals over time. For example, sales data that is growing and in which 25% of sales always occur during December contains both trend and seasonality. Triple exponential smoothing takes both the trend and seasonality into account by using three equations with three constants.

For triple exponential smoothing you, need to know the number of data points in each time period (designated as L in the following equations). To account for the seasonality, a seasonal index is calculated. The data is divided by the prior season index and then used in calculating the smoothed average.

- ❑ The first equation accounts for the current time period, and is a weighted average of the current data value divided by the seasonal factor and the prior average adjusted for the trend for the previous period. The weight constant is k:

$$SEASONAL(t) = k * (datavalue(t)/I(t-L)) + (1-k) * (SEASONAL(t-1) + b(t-1))$$

- ❑ The second equation is the calculated trend value, and is a weighted average of the difference between the current and previous average and the trend for the previous time period. b(t) represents the average trend. The weight constant is g:

$$b(t) = g * (SEASONAL(t) - SEASONAL(t-1)) + (1-g) * (b(t-1))$$

- ❑ The third equation is the calculated seasonal index, and is a weighted average of the current data value divided by the current average and the seasonal index for the previous season. I(t) represents the average seasonal coefficient. The weight constant is p:

$$I(t) = p * (datavalue(t)/SEASONAL(t)) + (1 - p) * I(t-L)$$

These equations are solved to derive the triple smoothed average. The first smoothed average is set to the first data value. Initial values for the seasonality factors are calculated based on the maximum number of full periods of data in the data source, while the initial trend is calculated based on two periods of data. These values are calculated with the following steps:

1. The initial trend factor is calculated by the following formula:

$$b(0) = (1/L) ((y(L+1)-y(1))/L + (y(L+2)-y(2))/L + \dots + (y(2L) - y(L))/L)$$

2. The calculation of the initial seasonality factor is based on the average of the data values within each period, A(j) (1<=j<=N):

$$A(j) = ( y((j-1)L+1) + y((j-1)L+2) + \dots + y(jL) ) / L$$

3. Then, the initial periodicity factor is given by the following formula, where N is the number of full periods available in the data, L is the number of points per period and n is a point within the period (1<= n <= L):

$$I(n) = ( y(n)/A(1) + y(L+n)/A(2) + \dots + y((N-1)L+n)/A(N) ) / N$$

The three constants must be chosen carefully. The best results are usually obtained by choosing the constants to minimize the mean-squared error (MSE) between the data values and the calculated averages. Varying the values of *npoint1* and *npoint2* affect the results, and some values may produce a better approximation. To search for a better approximation, you may want to find values that minimize the MSE.

The equation used to forecast beyond the last data point with triple exponential smoothing is:

$$\text{forecast}(t+m) = (\text{SEASONAL}(t) + m * b(t)) / I(t-L+\text{MOD}(m/L))$$

where:

*m*

Is the number of periods ahead for the forecast.

### ***Example:*** Calculating a Triple Exponential Smoothing Column

In the following, the data has seasonality but no trend. Therefore, *npoint2* is set high (1000) to make the trend factor negligible in the calculation:

```
SET HISTOGRAM = OFF
TABLE FILE VIDEOTRK
SUM TRANSTOT
BY  TRANSDATE
ON  TRANSDATE RECAP SEASONAL/D10.1 = FORECAST(TRANSTOT,1,3,'SEASONAL',
      3,3,1000,1);
WHERE TRANSDATE NE '19910617'
END
```

In the output, *npredict* is 3. Therefore, three periods (nine points, *nperiod* \* *npredict*) are generated.

TRANSDATE	TRANSTOT	SEASONAL
91/06/18	21.25	21.3
91/06/19	38.17	31.0
91/06/20	14.23	34.6
91/06/21	44.72	53.2
91/06/24	126.28	75.3
91/06/25	47.74	82.7
91/06/26	40.97	73.7
91/06/27	60.24	62.9
91/06/28	31.00	66.3
91/06/29		45.7
91/06/30		94.1
91/07/01		53.4
91/07/02		72.3
91/07/03		140.0
91/07/04		75.8
91/07/05		98.9
91/07/06		185.8
91/07/07		98.2

## Using a Linear Regression Equation

The Linear Regression Equation estimates values by assuming that the dependent variable (the new calculated values) and the independent variable (the sort field values) are related by a function that represents a straight line:

$$y = mx + b$$

where:

*y*  
Is the dependent variable.

*x*  
Is the independent variable.

*m*  
Is the slope of the line.

*b*  
Is the y-intercept.

REGRESS uses a technique called Ordinary Least Squares to calculate values for *m* and *b* that minimize the sum of the squared differences between the data and the resulting line.

The following formulas show how  $m$  and  $b$  are calculated.

$$m = \frac{(\sum xy - (\sum x \cdot \sum y)/n)}{(\sum x^2 - (\sum x)^2/n)}$$

$$b = (\sum y)/n - (m \cdot (\sum x)/n)$$

where:

$n$

Is the number of data points.

$y$

Is the data values (dependent variables).

$x$

Is the sort field values (independent variables).

Trend values, as well as predicted values, are calculated using the regression line equation.

### **Example:** Calculating a New Linear Regression Field

```
TABLE FILE CAR
PRINT MPG
BY DEALER_COST
WHERE MPG NE 0.0
  ON DEALER_COST RECAP FORMPG=FORECAST(MPG,1000,3,'REGRESS');
END
```

The output is:

<u>DEALER_COST</u>	<u>MPG</u>	<u>FORMPG</u>
2,886	27	25.51
4,292	25	23.65
4,631	21	23.20
4,915	21	22.82
5,063	23	22.63
5,660	21	21.83
	21	21.83
5,800	24	21.65
6,000	24	21.38
7,427	16	19.49
8,300	18	18.33
8,400	18	18.20
10,000	18	16.08
11,000	18	14.75
11,194	9	14.50
14,940	11	9.53
15,940	0	8.21
16,940	0	6.88
17,940	0	5.55

**Note:**

- ☐ Three predicted values of FORMPG are calculated. For values outside the range of the data, new DEALER\_COST values are generated by adding the interval value (1,000) to the prior DEALER\_COST value.
- ☐ There are no MPG values for the generated DEALER\_COST values.
- ☐ Each FORMPG value is computed using a regression line, calculated using all of the actual data values for MPG.

DEALER\_COST is the independent variable (x) and MPG is the dependent variable (y). The equation is used to calculate MPGFORECAST trend and predicted values.

In this case, the equation is approximately as follows:

$$\text{FORMPG} = (-0.001323 * \text{DEALER\_COST}) + 29.32$$

The predicted values are (the values are not exactly as calculated by FORECAST because of rounding, but they show the calculation process).

<b>DEALER_COST</b>	<b>Calculation</b>	<b>FORMPG</b>
15,940	$(-0.001323 * 15,940) + 29.32$	8.23
16,940	$(-0.001323 * 16,940) + 29.32$	6.91

DEALER_COST	Calculation	FORMPG
17,940	$(-0.001323 * 17,940) + 29.32$	5.59

FORECAST Reporting Techniques

You can use FORECAST multiple times in one request. However, all FORECAST requests must specify the same sort field, interval, and number of predictions. The only things that can change are the RECAP field, method, field used to calculate the FORECAST values, and number of points to average. If you change any of the other parameters, the new parameters are ignored.

If you want to move a FORECAST column in the report output, use an empty COMPUTE command for the FORECAST field as a placeholder. The data type (I, F, P, D) must be the same in the COMPUTE command and the RECAP command.

To make the report output easier to interpret, you can create a field that indicates whether the FORECAST value in each row is a predicted value. To do this, define a virtual field whose value is a constant other than zero. Rows in the report output that represent actual records in the data source will appear with this constant. Rows that represent predicted values will display zero. You can also propagate this field to a HOLD file.

*Example:*    **Generating Multiple FORECAST Columns in a Request**

This example calculates moving averages and exponential averages for both the DOLLARS and BUDDOLLARS fields in the GGSales data source. The sort field, interval, and number of predictions are the same for all of the calculations.

```
DEFINE FILE GGSales
  SDATE/YYM = DATE;
  SYEAR/Y = SDATE;
  SMONTH/M = SDATE;
  PERIOD/I2 = SMONTH;
END
TABLE FILE GGSales
  SUM DOLLARS AS 'DOLLARS' BUDDOLLARS AS 'BUDGET'
  BY CATEGORY NOPRINT BY PERIOD AS 'PER'
  WHERE SYEAR EQ 97 AND CATEGORY EQ 'Coffee'
  ON PERIOD RECAP DOLMOVAVE/D10.1= FORECAST(DOLLARS,1,0,'MOVAVE',3);
  ON PERIOD RECAP DOEXPAVE/D10.1= FORECAST(DOLLARS,1,0,'EXPAVE',4);
  ON PERIOD RECAP BUDMOVAVE/D10.1 = FORECAST(BUDDOLLARS,1,0,'MOVAVE',3);
  ON PERIOD RECAP BUDEXPAVE/D10.1 = FORECAST(BUDDOLLARS,1,0,'EXPAVE',4);
END
```



The output is shown in the following image.

<u>PER</u>	<u>DOLLARS</u>	<u>BUDGET</u>	<u>DOLMOVAVE</u>	<u>DOLEXPAVE</u>	<u>BUDMOVAVE</u>	<u>BUDEXPAVE</u>
1	801123	801375	801,123.0	801,123.0	801,375.0	801,375.0
2	682340	725117	741,731.5	753,609.8	763,246.0	770,871.8
3	765078	810367	749,513.7	758,197.1	778,953.0	786,669.9
4	691274	717688	712,897.3	731,427.8	751,057.3	759,077.1
5	720444	739999	725,598.7	727,034.3	756,018.0	751,445.9
6	742457	742586	718,058.3	733,203.4	733,424.3	747,901.9
7	747253	773136	736,718.0	738,823.2	751,907.0	757,995.6
8	655896	685170	715,202.0	705,652.3	733,630.7	728,865.3
9	730317	753760	711,155.3	715,518.2	737,355.3	738,823.2
10	724412	709397	703,541.7	719,075.7	716,109.0	727,052.7
11	620264	630452	691,664.3	679,551.0	697,869.7	688,412.4
12	762328	718837	702,334.7	712,661.8	686,228.7	700,582.3

### **Example:** Moving the FORECAST Column

The following example places the DOLLARS field after the MOVAVE field by using an empty COMPUTE command as a placeholder for the MOVAVE field. Both the COMPUTE command and the RECAP command specify formats for MOVAVE (of the same data type), but the format of the RECAP command takes precedence.

```

DEFINE FILE GGSales
  SDATE/YYM = DATE;
  SYEAR/Y = SDATE;
  SMONTH/M = SDATE;
  PERIOD/I2 = SMONTH;
END
TABLE FILE GGSales
SUM  UNITS
COMPUTE MOVAVE/D10.2 = ;
DOLLARS
  BY CATEGORY BY PERIOD
  WHERE SYEAR EQ 97 AND CATEGORY EQ 'Coffee'
  ON PERIOD RECAP MOVAVE/D10.1= FORECAST(DOLLARS,1,3,'MOVAVE',3);
END

```

The output is shown in the following image.

<u>Category</u>	<u>PERIOD</u>	<u>Unit Sales</u>	<u>MOVAVE</u>	<u>Dollar Sales</u>
Coffee	1	61666	801,123.0	801123
	2	54870	741,731.5	682340
	3	61608	749,513.7	765078
	4	57050	712,897.3	691274
	5	59229	725,598.7	720444
	6	58466	718,058.3	742457
	7	60771	736,718.0	747253
	8	54633	715,202.0	655896
	9	57829	711,155.3	730317
	10	57012	703,541.7	724412
	11	51110	691,664.3	620264
	12	58981	702,334.7	762328
	13	0	694,975.6	0
	14	0	719,879.4	0
	15	0	705,729.9	0

## **Example:** Distinguishing Data Rows From Predicted Rows

In the following example, the DATA\_ROW virtual field has the value 1 for each row in the data source. It has the value zero for the predicted rows. The PREDICT field is calculated as YES for predicted rows, and NO for rows containing data.

```

DEFINE FILE CAR
DATA_ROW/I1 = 1;
END
TABLE FILE CAR
PRINT DATA_ROW
COMPUTE PREDICT/A3 = IF DATA_ROW EQ 1 THEN 'NO' ELSE 'YES' ;
MPG
BY DEALER_COST
WHERE MPG GE 20
ON DEALER_COST RECAP FORMPG/D12.2=FORECAST(MPG,1000,3,'REGRESS');
ON DEALER_COST RECAP MPG =FORECAST(MPG,1000,3,'REGRESS');
END

```

The output is:

<u>DEALER_COST</u>	<u>DATA_ROW</u>	<u>PREDICT</u>	<u>MPG</u>	<u>FORMPG</u>
2,886	1	NO	27.00	25.65
4,292	1	NO	25.00	23.91
4,631	1	NO	21.00	23.49
4,915	1	NO	21.00	23.14
5,063	1	NO	23.00	22.95
5,660	1	NO	21.00	22.21
	1	NO	21.00	22.21
5,800	1	NO	24.20	22.04
6,000	1	NO	24.20	21.79
7,000	0	YES	20.56	20.56
8,000	0	YES	19.32	19.32
9,000	0	YES	18.08	18.08

## Calculating Trends and Predicting Values With Multivariate REGRESS

The REGRESS method derives a linear equation that best fits a set of numeric data points, and uses this equation to create a new column in the report output. The equation can be based on one to three independent variables.

This method estimates values by assuming that the dependent variable ( $y$ , the new calculated values) and the independent variables ( $x1$ ,  $x2$ ,  $x3$ ) are related by the following linear equation:

$$y = a1*x1 [+ a2*x2 [+ a3*x3]] + b$$

When there is one independent variable, the equation represents a straight line. This produces the same values as FORECAST using the REGRESS method. When there are two independent variables, the equation represents a plane, and with three independent variables, it represents a hyperplane. You should use this technique when you have reason to believe that the dependent variable can be approximated by a linear combination of the independent variables.

REGRESS uses a technique called Ordinary Least Squares to calculate values for the coefficients ( $a1$ ,  $a2$ ,  $a3$ , and  $b$ ) that minimize the sum of the squared differences between the data and the resulting line, plane, or hyperplane.

### **Syntax:** How to Create a Multivariate Linear Regression Column

```
ON {sortfield} RECAP y[/fmt] = REGRESS(n, x1, [x2, [x3,]] z);
```

where:

*sortfield*

Is a field in the data source. It cannot be the same field as any of the parameters to REGRESS. A new linear regression equation is derived each time the sort field value changes.

*y*

Is the new numeric column calculated by applying the regression equation. You cannot DEFINE or COMPUTE a field with this name.

*fmt*

Is the display format for  $y$ . If it is omitted, the default format is D12.2.

*n*

Is a whole number from 1 to 3 indicating the number of independent variables.

*x1, x2, x3*

Are the field names to be used as the independent variables. All of these variables must be numeric and be independent of each other.

*z*

Is an existing numeric field that is assumed to be approximately linearly dependent on the independent variables and is used to derive the regression equation.

### **Reference:** Usage Notes for REGRESS

- ☐ The (By) sort field used with REGRESS must be in a numeric or date format.
- ☐ REGRESS cannot operate on an ACROSS field.
- ☐ If any of the independent variables are also sort fields, they cannot be referenced in the request prior to the REGRESS sort field.
- ☐ FORECAST and REGRESS cannot be used in the same request, and only one REGRESS is supported in a request. Non-REGRESS RECAP commands are supported.
- ☐ The RECAP command used with REGRESS can contain only the REGRESS syntax. REGRESS does not recognize any syntax after the closing semicolon (;).
- ☐ Although you pass parameters to REGRESS using an argument list in parentheses, REGRESS is not a function. It can coexist with a user-written subroutine of the same name, as long as the user-written subroutine is not specified in a RECAP command.
- ☐ BY TOTAL is not supported.
- ☐ MORE, MATCH, FOR, and OVER are not supported.
- ☐ The process of generating the REGRESS values creates extra columns that are not printed in the report output. The number and placement of these additional columns varies depending on the specific request. Therefore, use of column notation is not supported in a request that includes REGRESS.
- ☐ SUMMARIZE and RECOMPUTE are not supported for the same sort field used for REGRESS.
- ☐ REGRESS is not supported for the FOCUS GRAPH facility.
- ☐ The left side of a RECAP command used for REGRESS supports the CURR attribute for creating a currency-denominated field.
- ☐ Fields with missing values cannot be used in the regression.

- ❑ Larger amounts of data produce more useful results.

**Example: Creating a Multivariate Linear Regression Column**

The following request uses the GGSales data source to calculate an estimated DOLLARS column. The BUDUNITS, UNITS, and BUDDOLLARS fields are the independent variables. The DOLLARS field provides the actual values to be estimated:

```
DEFINE FILE GGSales
  YEAR/Y = DATE;
  MONTH/M = DATE;
  PERIOD/I2 = MONTH;
END

TABLE FILE GGSales
PRINT BUDUNITS UNITS BUDDOLLARS DOLLARS
BY PERIOD
ON PERIOD
RECAP EST_DOLLARS/F8 = REGRESS(3, BUDUNITS, UNITS, BUDDOLLARS, DOLLARS);
WHERE CATEGORY EQ 'Coffee'
WHERE REGION EQ 'West'
WHERE UNITS GT 1600 AND UNITS LT 1700
END
```

The output is:

<u>PERIOD</u>	<u>Budget Units</u>	<u>Unit Sales</u>	<u>Budget Dollars</u>	<u>Dollar Sales</u>	<u>EST DOLLARS</u>
1	1665	1678	21645	23492	0
	1725	1669	22425	21697	0
2	1613	1685	22582	18535	19065
	1568	1682	23520	25230	19089
	1847	1668	18470	25020	19725
3	1646	1656	23044	19872	19871
	1759	1615	17590	17765	17765
	1498	1637	16478	21281	21280
	1653	1694	21489	16940	16940
4	1457	1671	21855	20052	0
5	1662	1674	24930	18414	0
6	1825	1695	23725	25425	32228
	1870	1620	24310	24300	30866
	1712	1640	22256	16400	31859
7	1727	1623	24178	17853	-26821
	1733	1647	17330	24705	-13637
8	1830	1652	20130	23128	51170
	1451	1660	17412	19920	15755
	1556	1643	18672	18073	20517
9	1464	1663	14640	23282	18249
	1463	1663	21945	19956	23005
10	1464	1667	17568	25005	25005
	1711	1623	20532	22722	22722
	1701	1626	18711	21138	21138
	1473	1616	14730	16160	16160
11	1403	1601	21045	17611	0
12	1796	1696	17960	25440	0

## Using Text Fields in DEFINE and COMPUTE

Text fields can be assigned to alphanumeric fields and receive assignment from alphanumeric fields. If an alphanumeric field is assigned the value of a text field that is too long for the alphanumeric field, the value is truncated before being assigned to the alphanumeric field.

**Note:** COMPUTE commands in Maintain do not support text fields.

**Example: Assigning the Result of an Alphanumeric Expression to a Text Field**

This example uses the COURSES data source, which contains a text field, to create an alphanumeric field named ADESC, which truncates the text field at 36 characters, and a new text field named NEWDESC, which is a text version of ADESC:

```
DEFINE FILE COURSES
ADESC/A36   = DESCRIPTION;
NEWDESC/TX36 = ADESC;
END
```

```
TABLE FILE COURSES
PRINT ADESC NEWDESC
END
```

The output is:

ADESC	NEWDESC
-----	-----
This course provides the DP professi	This course provides the DP professi
Anyone responsible for designing FOC	Anyone responsible for designing FOC
This is a course in FOCUS efficienci	This is a course in FOCUS efficienci

**Creating Temporary Fields Independent of a Master File**

The temporary fields you create with the DEFINE and COMPUTE commands are tied to a specific Master File, and in the case of values calculated with the COMPUTE command, to a specific request. However, you can create temporary fields that are independent of either a Master File or a request using the DEFINE FUNCTION command.

A DEFINE function is a named group of calculations that use any number of input values and produce a return value. When calling a DEFINE function, you must first define the function.

A DEFINE function can be called in most of the same situations that are valid for Information Builders-supplied functions. Data types are defined with each argument. When substituting values for these arguments, the format must match the defined format. Alphanumeric arguments shorter than the specified format are padded with blanks, while longer alphanumeric arguments are truncated.

All calculations within the function are done in double precision. Format conversions occur only across equal signs (=) in the assignments that define temporary fields.

## **Syntax:**      **How to Define a Function**

```
DEFINE FUNCTION name (argument1/format1,..., argumentn/formatn)
[tempvariablea/formatata [TITLE 'line1[,line2 ...]'
  [DESCRIPTION 'description'] = expressiona;]
.
.
.
[tempvariablex/formatx = expressionx;]
name/format = [result_expression];
END
```

where:

*name*

Is the name of the function, up to 64 characters. This must be the last field calculated in the function, and is used to return the value of the function to the calling procedure.

*argument1...argumentn*

Are the argument names. They can be any names that comply with WebFOCUS field naming rules.

*format1...formatn*

Are the formats of the function arguments.

If the format of an argument is alphanumeric, the argument value must also be alphanumeric. Shorter arguments are padded on the right with blanks, and longer arguments are truncated.

If the format of an argument is numeric, the argument value must also be numeric. To prevent unexpected results, you must be consistent in your use of data types.

*tempvariablea...tempvariablex*

Are temporary fields. Temporary fields hold intermediate values used in the function. You can define as many temporary fields as you need.

*tempformatata...tempformatx*

Are the formats of the temporary fields.

*line1,line2 ...*

Are the lines of default column title to be displayed for the virtual field unless overridden by an AS phrase.



*description*

Is the description to be associated with the virtual field, enclosed in single quotation marks. The description displays in the tools that browse Master Files.

*expressiona...expressionx*

Are the expressions that calculate the temporary field values. The expressions can use parameters, constants, and other temporary fields defined in the same function.

*format*

Is the format of the value the function returns.

*result\_expression*

Is the expression that calculates the value returned by the function. The expression can use parameters, constants, and temporary fields defined in the same function.

All names defined in the body of the function are local to the function. The last field defined before the END command in the function definition must have the same name as the function, and represents the return value for the function.

**Reference: DEFINE Function Limits and Restrictions**

- ❑ The number of functions you can define and use in a session is virtually unlimited.
- ❑ A DEFINE function is cleared with the DEFINE FUNCTION CLEAR command. It is not cleared by issuing a join, or by any WebFOCUS command.
- ❑ When an expression tries to use a cleared function, an error appears.
- ❑ DEFINE functions can call other DEFINE functions, but cannot call themselves.
- ❑ If you overwrite or clear a DEFINE function, a subsequent attempt to use a temporary field that refers to the function generates the following warning:

```
(FOC03665) Error loading external function '%1'
```

**Example: Defining a Function**

The following example creates and calls the SUBTRACT function. SUBTRACT performs a calculation with the arguments VAL1 and VAL2.

```
DEFINE FUNCTION SUBTRACT (VAL1/D8, VAL2/D8)
  SUBTRACT/D8.2 = VAL1 - VAL2;
END
```

```
TABLE FILE MOVIES
PRINT TITLE LISTPR IN 35 WHOLESALPR AND
COMPUTE PROFIT/D8.2 = SUBTRACT(LISTPR,WHOLESALPR);
BY CATEGORY
WHERE CATEGORY EQ 'MYSTERY' OR 'ACTION'
END
```

The output is:

<u>CATEGORY</u>	<u>TITLE</u>	<u>LISTP R</u>	<u>WHOLESALE PR</u>	<u>PROFIT</u>
ACTION	JAWS	19.95	10.99	8.96
	ROBOCOP	19.98	11.50	8.48
	TOTAL RECALL	19.99	11.99	8.00
	TOP GUN	14.95	9.99	4.96
	RAMBO III	19.95	10.99	8.96
MYSTERY	REAR WINDOW	19.98	9.00	10.98
	VERTIGO	19.98	9.00	10.98
	FATAL ATTRACTION	29.98	15.99	13.99
	NORTH BY NORTHWEST	19.98	9.00	10.98
	DEAD RINGERS	25.99	15.99	10.00
	MORNING AFTER, THE	19.95	9.99	9.96

PSYCHO	19.98	9.00	10.98
BIRDS, THE	19.98	9.00	10.98
SEA OF LOVE	59.99	30.00	29.99

**Procedure:** How to Display DEFINE Functions

Issue the following command from the Command Console:

? FUNCTION

**Example:** Displaying DEFINE Functions

Issuing the command

? FUNCTION

displays information similar to the following:

FUNCTIONS	CURRENTLY	ACTIVE	
NAME	FORMAT	PARAMETER	FORMAT
-----	-----	-----	-----
SUBTRACT	D8.2	VAL1	D8
		VAL2	D8

If you issue the ? FUNCTION command when no functions are defined, the following appears:

NO FUNCTIONS CURRENTLY IN EFFECT

***Syntax:***      **How to Clear DEFINE Functions**

```
DEFINE FUNCTION {name|*} CLEAR
```

where:

*name*

Is the name of the function name to clear.

\*

Clears all active DEFINE functions.

## Including Totals and Subtotals

---

To help interpret detailed information in a report, you can summarize the information using row and column totals, grand totals, and subtotals. You can use these summary lines in a report to clarify or highlight information.

**In this chapter:**

- ☐ [Calculating Row and Column Totals](#)
  - ☐ [Including Section Totals and a Grand Total](#)
  - ☐ [Including Subtotals](#)
  - ☐ [Recalculating Values for Subtotal Rows](#)
  - ☐ [Summarizing Alphanumeric Columns](#)
  - ☐ [Manipulating Summary Values With Prefix Operators](#)
  - ☐ [Combinations of Summary Commands](#)
  - ☐ [Producing Summary Columns for Horizontal Sort Fields](#)
  - ☐ [Performing Calculations at Sort Field Breaks](#)
  - ☐ [Suppressing Grand Totals](#)
  - ☐ [Conditionally Displaying Summary Lines and Text](#)
- 

### Calculating Row and Column Totals

You can produce totals for rows or columns of numbers in a report. Use:

- ☐ **ROW-TOTAL** to display a new column containing the sum of all numbers in each row.
- ☐ **COLUMN-TOTAL** to display a final row on the report, which contains the totals for each column of numbers.

You can use row totals and column totals in matrix reports (created by using a BY and an ACROSS in your report request), rename row and column total titles, and include calculated values in your row or column totals. You can also create row totals using ACROSS-TOTAL.

Note that when producing totals in a report, if one field is summed, the format of the row total is the same as the format of the field. For example, if the format of the CURR\_SAL field is D12.2M, the format of the row total for CURR\_SAL is also D12.2M. When you are summing fields with different formats, the default format of D12.2 is used for the total.

You can rename the default column titles using the AS phrase and align the labels for row and column totals. For details, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.

### **Syntax:**      **How to Calculate Row and Column Totals**

```
display_command fieldname AND ROW-TOTAL [alignment][/format] [AS 'name']  
display_command fieldname AND COLUMN-TOTAL [alignment][AS 'name']
```

where:

*display\_command*

Is one of the following commands: PRINT, LIST, SUM, or COUNT.

*fieldname*

Is the name of the field for which to calculate row and/or column totals.

*alignment*

Specifies the alignment of the ROW-TOTAL or COLUMN-TOTAL label. Possible values are:

*/R* right justifies the label.

*/L* left justifies the label.

*/C* centers the label.

Note that these alignment settings are ignored in HTML output. If you are working in WebFOCUS, to take advantage of column alignment features, you can include the command SET STYLE=OFF in the report request or generate your output in PDF, or in another format that supports these features.

*format*

Reformats the ROW-TOTAL.

*name*

Is the label for the ROW-TOTAL or COLUMN-TOTAL.

You may also specify row or column totals with the ON TABLE command. Field names are optional with COLUMN-TOTAL, and cannot be listed with ROW-TOTAL. Use the following syntax:

```
ON TABLE COLUMN-TOTAL [alignment][AS 'name'] [field field field]  
ON TABLE ROW-TOTAL [alignment][/format] [AS 'name']
```

**Example: Calculating Row and Column Totals**

The following request illustrates the use of ROW-TOTAL and COLUMN-TOTAL. The column and row total labels are "TOTAL" by default. You can change them using an AS phrase.

```
TABLE FILE SALES
SUM RETURNS DAMAGED AND ROW-TOTAL AND COLUMN-TOTAL
BY PROD_CODE
END
```

The output is:

PROD_CODE	RETURNS	DAMAGED	TOTAL
B10	13	10	23
B12	4	3	7
B17	4	2	6
B20	1	2	3
C13	3	0	3
C17	0	0	0
C7	5	4	9
D12	3	2	5
E1	4	7	11
E2	9	4	13
E3	12	11	23
TOTAL	58	45	103

**Example: Specifying Column Totals With ON TABLE**

The following request illustrates the use of COLUMN-TOTAL with the ON TABLE command.

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL
BY LAST_NAME
ON TABLE COLUMN-TOTAL
END
```

The output is:

LAST_NAME	CURR_SAL
-----	-----
BANNING	\$29,700.00
BLACKWOOD	\$21,780.00
CROSS	\$27,062.00
GREENSPAN	\$9,000.00
IRVING	\$26,862.00
JONES	\$18,480.00
MCCOY	\$18,480.00
MCKNIGHT	\$16,100.00
ROMANS	\$21,120.00
SMITH	\$13,200.00
	\$9,500.00
STEVENS	\$11,000.00
TOTAL	\$222,284.00

### **Example:** Using Row and Column Totals in a Matrix Report

The following request illustrates the use of ROW-TOTAL and COLUMN-TOTAL in a matrix report (created by using the BY and ACROSS phrases together).

```
TABLE FILE EMPLOYEE
SUM CURR_SAL AND ROW-TOTAL AND COLUMN-TOTAL
BY BANK_NAME
ACROSS DEPARTMENT
END
```

The output is:

BANK_NAME	DEPARTMENT		
TOTAL	MIS	PRODUCTION	
-----	-----	-----	-----
	\$40,680.00	\$41,620.00	\$82,300.00
ASSOCIATED	\$21,780.00	\$42,962.00	\$64,742.00
BANK ASSOCIATION	\$27,062.00	.	\$27,062.00
BEST BANK	.	\$29,700.00	\$29,700.00
STATE	\$18,480.00	.	\$18,480.00
TOTAL	\$108,002.00	\$114,282.00	\$222,284.00



**Example: Including Calculated Values in Row and Column Totals**

The following request illustrates the inclusion of the calculated value, PROFIT, in row and column totals.

```
TABLE FILE CAR
SUM DCOST RCOST
COMPUTE PROFIT/D12=RCOST-DCOST;
ROW-TOTAL/L/D12 AS 'TOTAL_COST'
BY COUNTRY
ON TABLE COLUMN-TOTAL/L AS 'FINAL_TOTAL'
END
```

The output is:

COUNTRY	DEALER_COST	RETAIL_COST	PROFIT	
TOTAL_COST				
-----	-----	-----	-----	-----
ENGLAND	37,853	45,319	7,466	90,638
FRANCE	4,631	5,610	979	11,220
ITALY	41,235	51,065	9,830	102,130
JAPAN	5,512	6,478	966	12,956
W GERMANY	54,563	64,732	10,169	129,464
FINAL_TOTAL	143,794	173,204	29,410	346,408

**Reference: Using ROW-TOTAL With ACROSS and Multiple Display Commands**

When a request has an ACROSS sort field, each ACROSS value displays a column for each field displayed on the report output. For example, the following request, each state has a column for units and a column for dollars:

```
TABLE FILE GGSALES
SUM UNITS AS 'U' DOLLARS AS 'D' BY CITY
ACROSS ST
IF ST EQ 'CA'
IF BU DUNITS NE MISSING
END
```

The output is:

	State	
	CA	
City	U	D
-----	-----	-----
Los Angeles	298070	3772014
San Francisco	312500	3870258

When you specify a row total with ACROSS, the row total is calculated separately for each column in each ACROSS group. For example, in the following request the row total has a column for units and a column for dollars:

```
TABLE FILE GGSales
SUM UNITS AS 'U' DOLLARS AS 'D' BY CITY
ACROSS ST
IF ST EQ 'CA'
IF BUDUNITS NE MISSING
   ON TABLE ROW-TOTAL
END
```

The output is:

City	State CA			TOTAL	
		U	D	U	D
Los Angeles		298070	3772014	298070	3772014
San Francisco		312500	3870258	312500	3870258

When the request also has multiple display commands, each additional command adds additional columns to each ACROSS group on the report output.

The first column of the row total group is calculated by adding the first column from each display command under each ACROSS value, the second column adds the second column from each display command, and so on.

For example, the following request has a SUM command for units and dollars and another SUM command for budgeted units and budgeted dollars. The row total has a column for the sum of units and budgeted units and another column for the sum of dollars and budgeted dollars:

```
TABLE FILE GGSales
SUM UNITS AS 'U' DOLLARS AS 'D' BY CITY
SUM BUDUNITS AS 'BU' BUDDOLLARS AS 'BD' BY CITY
ACROSS ST
IF ST EQ 'CA'
IF BUDUNITS NE MISSING
   ON TABLE ROW-TOTAL
END
```

The output is:

City	State CA						
		U	D	BU	BD	BU	BD
Los Angeles		298070	3772014	295637	3669484	593707	7441498
San Francisco		312500	3870258	314725	3916863	627225	7787121

If the different display commands do not all specify the same number of fields, some columns will not be represented in the row total. For example, in the following request, the second SUM command has a column for budgeted units but not for budgeted dollars. Therefore, the row total group has no column for dollars:

```
TABLE FILE GGSALES
SUM UNITS AS 'U' DOLLARS AS 'D' BY CITY
SUM BUDUNITS AS 'BU'          BY CITY
ACROSS ST
IF ST EQ 'CA'
IF BUDUNITS NE MISSING
ON TABLE ROW-TOTAL
END
```

The output is:

City	State CA	U	D	BU	TOTAL BU
Los Angeles		298070	3772014	295637	593707
San Francisco		312500	3870258	314725	627225

In this case, you can use column notation to calculate the row total properly. For example, the following request calculates the row total column by adding the units, dollars, and budgeted units columns together:

```
TABLE FILE GGSALES
SUM UNITS AS 'U' DOLLARS AS 'D' BY CITY
SUM BUDUNITS AS 'BU'          BY CITY
ACROSS ST
COMPUTE TOTAL/I10 = C1 + C2 +C3; AS 'ROW-TOTAL'
IF ST EQ 'CA'
IF BUDUNITS NE MISSING
END
```

The output is:

City	State CA	U	D	BU	ROW-TOTAL
Los Angeles		298070	3772014	295637	4365721
San Francisco		312500	3870258	314725	4497483

Producing Row Totals for Horizontal (ACROSS) Sort Field Values

You can produce row totals for horizontal (ACROSS) sort field values. Row totals for horizontal sort fields, referenced by ACROSS-TOTAL, are different from standard row totals because only horizontal sort field values, referenced by ACROSS, are included in the total. Integer, single precision floating point, double precision floating point, packed, and long packed fields can all be totaled.

**Syntax:**      How to Produce Row Totals for Horizontal (ACROSS) Sort Field Values

```
ACROSS sortfield ACROSS-TOTAL [AS 'name'] [COLUMNS col1 AND col2 ...]
```

where:

*sortfield*

Is the name of the field being sorted across.

*name*

Is the new name for the ACROSS-TOTAL column title.

*col1, col2*

Are the titles of the ACROSS columns you want to include in the total.

**Example:**      Producing Row Totals for Horizontal (ACROSS) Sort Field Values

The following illustrates how to generate a row total for horizontal (ACROSS) sort field values. Notice that the summed values in the TOTAL TITLE COUNT column only reflect the values in the (RATING) PG and R columns. The values in the COPIES column are not included since they are not horizontal (ACROSS) sort field values.

```
TABLE FILE MOVIES
SUM COPIES BY CATEGORY
COUNT TITLE BY CATEGORY
ACROSS RATING ACROSS-TOTAL
COLUMNS PG AND R
END
```

The output is:

CATEGORY	COPIES	RATING		TOTAL
		PG	R	
ACTION	14	2	3	5
COMEDY	16	4	1	5
DRAMA	2	0	1	1
FOREIGN	5	2	3	5
MUSICALS	2	1	1	2
MYSTERY	17	2	5	7
SCI/FI	3	0	3	3

**Reference: Usage Notes for ACROSS-TOTAL**

- ❑ Stacking headings in ACROSS-TOTAL is not supported.
- ❑ Attempting to use ACROSS-TOTAL with other types of fields (alphanumeric, text, and dates) produces blank columns.
- ❑ In cases of multiple ACROSS columns with ACROSS-TOTAL, there are additional columns with subtotaled values.
- ❑ The results of ROW-TOTAL and ACROSS-TOTAL are the same if there is only a single display field or single display command in the procedure.
- ❑ The maximum number of ACROSS-TOTAL components is five.
- ❑ ACROSS-TOTAL populates the ACROSSVALUE component in a StyleSheet. For an example of styling an ACROSS-TOTAL component, see [Identifying Row Totals \(ACROSS-TOTAL\) for Horizontal Sort Data](#) on page 1184.

**Including Section Totals and a Grand Total**

Frequently, reports contain detailed information that is broken down into subsections, for which simple column and row totals may not provide adequate summaries. In these instances, it is more useful to look at subtotals for particular sections, and a grand total.

You can add the following commands to your requests to create section subtotals and grand totals:

- ❑ SUB-TOTAL and SUBTOTAL
- ❑ SUMMARIZE and RECOMPUTE (used with calculated values)
- ❑ RECAP and COMPUTE

Each command produces grand totals and/or subtotals by using different information. Subtotals produce totals every time a specified sort field value changes, and are independent of record selection criteria. You can further control when subtotals are produced by specifying WHEN criteria (see [Conditionally Displaying Summary Lines and Text](#) on page 429). You can control whether subtotals display above or below the data. For information, see [How to Control Placement of Summary Lines](#) on page 380. You can also suppress grand totals using the NOTOTAL command. For details, see [Suppressing Grand Totals](#) on page 427.

By default, a blank line is generated before a subtotal on the report output. You can eliminate these automatic blank lines by issuing the SET DROPBLNKLINE=ON command.

**Note:** When the request has a PAGE-BREAK command, the GRANDTOTAL is on a page by itself.

You can use prefix operators with SUBTOTAL, SUB-TOTAL, SUMMARIZE, and RECOMPUTE. For details, see [Manipulating Summary Values With Prefix Operators](#) on page 390. In addition, you can combine different summary operations in a single request. For information, see [Combinations of Summary Commands](#) on page 409.

**Example: Using Section Totals and Grand Totals**

The following request illustrates how to create a subtotal every time the department value changes. The grand total is automatically produced when you use the SUBTOTAL command.

```
TABLE FILE EMPLOYEE
SUM DED_AMT BY DED_CODE BY DEPARTMENT
BY BANK_ACCT
WHERE BANK_ACCT NE 0
WHERE DED_CODE EQ 'CITY' OR 'FED'
ON DEPARTMENT SUBTOTAL
END
```

The first and last portions of the output are:

DED_CODE	DEPARTMENT	BANK_ACCT	DED_AMT
-----	-----	-----	-----
CITY	MIS	40950036	\$14.00
		122850108	\$31.75
		163800144	\$82.70
*TOTAL DEPARTMENT MIS			\$128.45
	PRODUCTION	160633	\$7.42
		136500120	\$18.25
		819000702	\$60.20
*TOTAL DEPARTMENT PRODUCTION			\$85.87
FED	MIS	40950036	\$1,190.77
		122850108	\$2,699.80
		163800144	\$7,028.30
*TOTAL DEPARTMENT MIS			\$10,918.87
	PRODUCTION	160633	\$631.12
		136500120	\$1,552.10
		819000702	\$5,120.04
*TOTAL DEPARTMENT PRODUCTION			\$7,303.26
TOTAL			\$18,436.45

## Including Subtotals

You can use the SUBTOTAL and SUB-TOTAL commands to sum individual values, such as columns of numbers, each time a named sort field changes value.

- ❑ SUB-TOTAL displays a subtotal when the sort field changes value, and for any higher-level sort fields when their values change.
- ❑ SUBTOTAL displays a subtotal only when the specified sort field changes value. It does not give subtotals for higher-level fields.

Both SUB-TOTAL and SUBTOTAL produce grand totals. You can suppress grand totals using the NOTOTAL command. See [Suppressing Grand Totals](#) on page 427.

The subtotal is calculated every time the sort field value changes or, if WHEN criteria are applied to the sort field, every time the WHEN conditions are met.

A BY, ACROSS, or ON phrase is required to initialize the syntax.

### **Syntax:**      **How to Create Subtotals**

```
{BY|ON} fieldname {SUB-TOTAL|SUBTOTAL} [MULTILINES]
      [field1 [AND] field2...] [AS 'text'] [WHEN expression;
```

where:

*fieldname*

Must be the name of a field in a sort phrase. A BY phrase can include a summary command. The number of fields to subtotal multiplied by the number of levels of subtotals counts in the number of display fields permitted for the request. For details on determining the maximum number of display fields that can be used in a request, see [Displaying Report Data](#) on page 43.

SUB-TOTAL|SUBTOTAL

SUB-TOTAL displays subtotals for numeric values when the BY|ON field changes value, and for any higher-level sort fields when their values change.

SUBTOTAL displays a subtotal only when the specified sort field changes value.

MULTILINES

Suppresses the printing of a subtotal line for every sort break that has only one detail line, since the subtotal value is equal to this one value. Note that MULTI-LINES is a synonym for MULTILINES. MULTILINES is not supported with horizontal (ACROSS) sort fields.

*field1, field2, ...*

Denotes a list of specific fields to subtotal. This list overrides the default, which includes all numeric display fields. The list can include numeric and alphanumeric fields.

You can use the asterisk (\*) wildcard character instead of a field list to indicate that all fields, numeric and alphanumeric, should be included on the summary lines.

**AS** *'text'*

Enables you to specify a different label. For related information, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.

**WHEN** *expression*

Specifies the conditional display of subtotals as determined by a Boolean expression. You must end the expression with a semicolon.

### **Syntax:** How to Control Placement of Summary Lines

**SET SUBTOTALS** = {[ABOVE](#) | [BELOW](#)}

where:

[ABOVE](#)

Places summary lines above the detail lines and displays the sort field values on every detail line of the report output.

[BELOW](#)

Places summary lines below the detail lines. BELOW is the default value.

**Note:** SET SUBTOTALS = ABOVE is not supported with format XLSX, EXL07, or EXL2K FORMULA.

### **Example:** Placing Subtotals Above the Data

The following request against the EMPLOYEE data source sums deduction amounts and gross salaries by department, deduction code, and last name. It then subtotals the deduction amounts and gross salaries for each department. The following request places the subtotals below the detail lines (the default):

```
TABLE FILE EMPLOYEE
SUM DED_AMT GROSS
BY DEPARTMENT
BY DED_CODE
  BY LAST_NAME
WHERE BANK_ACCT NE 0
WHERE DED_CODE EQ 'FICA' OR 'CITY'
  ON DEPARTMENT SUBTOTAL
  ON TABLE SET SUBTOTALS BELOW
  ON TABLE SET PAGE NOPAGE
END
```



The output is:

DEPARTMENT	DED_CODE	LAST_NAME	DED_AMT	GROSS
-----	-----	-----	-----	-----
MIS	CITY	BLACKWOOD	\$31.76	\$9,075.00
		CROSS	\$82.69	\$22,013.77
		JONES	\$14.01	\$6,099.50
	FICA	BLACKWOOD	\$2,223.37	\$9,075.00
		CROSS	\$5,788.01	\$22,013.77
		JONES	\$980.64	\$6,099.50
*TOTAL DEPARTMENT MIS			\$9,120.47	\$74,376.54
PRODUCTION	CITY	BANNING	\$7.42	\$2,475.00
		IRVING	\$60.24	\$17,094.00
		MCKNIGHT	\$18.26	\$9,129.99
	FICA	BANNING	\$519.75	\$2,475.00
		IRVING	\$4,216.53	\$17,094.00
		MCKNIGHT	\$1,278.21	\$9,129.99
*TOTAL DEPARTMENT PRODUCTION			\$6,100.40	\$57,397.98
TOTAL			\$15,220.88	\$131,774.52

The following is the same request, but with the subtotals placed above the detail lines:

```
TABLE FILE EMPLOYEE
SUM DED_AMT GROSS
BY DEPARTMENT
BY DED_CODE
  BY LAST_NAME
WHERE BANK_ACCT NE 0
WHERE DED_CODE EQ 'FICA' OR 'CITY'
  ON DEPARTMENT SUBTOTAL
  ON TABLE SET SUBTOTALS ABOVE
  ON TABLE SET PAGE NOPAGE
END
```

On the output, the grand total line comes first, then the subtotal for the MIS department followed by the detail lines for the MIS department, followed by the subtotal for the PRODUCTION department and its detail lines. Note that all sort field values display on each line of the report output:

DEPARTMENT	DED_CODE	LAST_NAME	DED_AMT	GROSS
-----	-----	-----	-----	-----
TOTAL			\$15,220.88	\$131,774.52
*TOTAL	DEPARTMENT	MIS	\$9,120.47	\$74,376.54
MIS	CITY	BLACKWOOD	\$31.76	\$9,075.00
MIS	CITY	CROSS	\$82.69	\$22,013.77
MIS	CITY	JONES	\$14.01	\$6,099.50
MIS	FICA	BLACKWOOD	\$2,223.37	\$9,075.00
MIS	FICA	CROSS	\$5,788.01	\$22,013.77
MIS	FICA	JONES	\$980.64	\$6,099.50
*TOTAL	DEPARTMENT	PRODUCTION	\$6,100.40	\$57,397.98
PRODUCTION	CITY	BANNING	\$7.42	\$2,475.00
PRODUCTION	CITY	IRVING	\$60.24	\$17,094.00
PRODUCTION	CITY	MCKNIGHT	\$18.26	\$9,129.99
PRODUCTION	FICA	BANNING	\$519.75	\$2,475.00
PRODUCTION	FICA	IRVING	\$4,216.53	\$17,094.00
PRODUCTION	FICA	MCKNIGHT	\$1,278.21	\$9,129.99

**Reference: Usage Notes for Subtotals**

- ❑ When using a SUM or COUNT command with only one sort phrase in the request, SUB-TOTAL and SUBTOTAL produce the same result as the value of the SUM or COUNT command. However, when using a PRINT command with one sort phrase, SUBTOTAL is useful because there can be many values within a sort break.
- ❑ All SUB-TOTALs display up to and including the point where the sort break occurs, so only the innermost point of subtotalling should be requested. For instance, if the BY fields are

```
BY AREA
BY PROD_CODE
BY DATE SUB-TOTAL
```

then, when AREA changes, subtotals are displayed for DATE, PROD\_CODE, and AREA on three lines (one under the other).

- ❑ If you use a WHERE TOTAL or IF TOTAL test, the display of the sort field value for the subtotal line is suppressed unless PRINTPLUS is ON. For details about using PRINTPLUS in WebFOCUS, see *Using PRINTPLUS* on page 1591.
- ❑ Subtotals display on the next line if the subtotal text does not fit on the line prior to the displayed field columns.

- ❑ If a report request has multiple BY phrases, with SUBTOTAL/SUMMARIZE/RECOMPUTE/SUB-TOTAL at several levels, and MULTILINES or MULTI-LINES is specified at any one of those levels, it applies to all levels.

**Note:** ON BYfield SUBFOOT applies only to the level specified.

### *Example:* Generating Subtotals

The following request illustrates how to create a subtotal for SALES every time the country value changes.

```
TABLE FILE CAR
SUM AVE.MPG AND SALES AND AVE.RETAIL_COST
BY COUNTRY SUB-TOTAL SALES
BY BODYTYPE
END
```

The output is:

COUNTRY	BODYTYPE	AVE MPG	SALES	AVE RETAIL_COST
-----	-----	----	-----	-----
ENGLAND	CONVERTIBLE	16	0	8,878
	HARDTOP	25	0	5,100
	SEDAN	10	12000	15,671
*TOTAL ENGLAND			12000	
FRANCE	SEDAN	21	0	5,610
*TOTAL FRANCE			0	
ITALY	COUPE	11	12400	19,160
	ROADSTER	21	13000	6,820
	SEDAN	21	4800	5,925
*TOTAL ITALY				
JAPAN	SEDAN	14	78030	3,239
*TOTAL JAPAN			78030	
W GERMANY	SEDAN	20	88190	9,247
*TOTAL W GERMANY			88190	
TOTAL			208420	

**Example: Comparing SUB-TOTAL and SUBTOTAL**

The following request illustrates how to create a subtotal for the numeric fields DED\_AMT and GROSS when the department value changes, and for the higher-level sort field (DED\_CODE) when its value changes.

```
TABLE FILE EMPLOYEE
SUM DED_AMT GROSS BY DED_CODE BY DEPARTMENT
BY BANK_ACCT
WHERE BANK_ACCT NE 0
ON DEPARTMENT SUB-TOTAL
END
```

If you use SUBTOTAL instead of SUB-TOTAL, the totals for DED\_AMT and GROSS display only when the DEPARTMENT value changes.

The first portion of the output is:

DED_CODE	DEPARTMENT	BANK_ACCT	DED_AMT	GROSS
-----	-----	-----	-----	-----
CITY	MIS	40950036	\$14.00	\$6,099.50
		122850108	\$31.75	\$9,075.00
		163800144	\$82.70	\$22,013.75
*TOTAL DEPARTMENT MIS			\$128.45	\$37,188.25
	PRODUCTION	160633	\$7.42	\$2,475.00
		136500120	\$18.25	\$9,130.00
		819000702	\$60.20	\$17,094.00
*TOTAL DEPARTMENT PRODUCTION			\$85.87	\$28,699.00
*TOTAL DED_CODE CITY			\$214.32	\$65,887.25

The last portion of the output is:

DED_CODE	DEPARTMENT	BANK_ACCT	DED_AMT	GROSS
-----	-----	-----	-----	-----
STAT	MIS	40950036	\$196.13	\$6,099.50
		122850108	\$444.65	\$9,075.00
		163800144	\$1,157.60	\$22,013.75
*TOTAL DEPARTMENT MIS			\$1,798.38	\$37,188.25
	PRODUCTION	160633	\$103.95	\$2,475.00
		136500120	\$255.65	\$9,130.00
		819000702	\$843.32	\$17,094.00
*TOTAL DEPARTMENT PRODUCTION			\$1,202.92	\$28,699.00
*TOTAL DED_CODE STAT			\$3,001.30	\$65,887.25
TOTAL			\$41,521.18	\$461,210.75

## Recalculating Values for Subtotal Rows

You can use the SUMMARIZE and RECOMPUTE commands instead of SUB-TOTAL and SUBTOTAL to recalculate the result of a COMPUTE command. SUMMARIZE is similar to SUB-TOTAL in that it recomputes values at every sort break. RECOMPUTE is similar to SUBTOTAL in that it recalculates only at the specified sort break.

SUMMARIZE recomputes grand totals for the entire report. If you wish to suppress grand totals, you can include the NOTOTAL command in your request. See [Suppressing Grand Totals](#) on page 427.

### **Syntax:** How to Subtotal Calculated Values

```
{BY|ON} fieldname {SUMMARIZE|RECOMPUTE} [MULTILINES]
      [field1 [AND] field2...] [AS 'text'] [WHEN expression;
```

where:

*fieldname*

Must be the name of a field in a sort phrase. A BY phrase can include a summary command. The number of fields to summarize multiplied by the number of levels of summary commands counts in the number of display fields for the request. For details on determining the maximum number of display fields that can be used in a request, see [Displaying Report Data](#) on page 43.

SUMMARIZE

Recomputes values at every sort break.

RECOMPUTE

Recalculates values only at the specified sort break.

MULTILINES

Suppresses the printing of a subtotal line for every sort break that has only one detail line, since the subtotal value is equal to this one value. Note that MULTI-LINES is a synonym for MULTILINES. MULTILINES is not supported with horizontal (ACROSS) sort fields.

You can also suppress grand totals using the NOTOTAL command, as described in [Suppressing Grand Totals](#) on page 427.

AS '*text*'

Enables you to specify a different label. For related information, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.

*field1, field2, ...*

Denotes a list of specific fields to be subtotaled after the RECOMPUTE or SUMMARIZE. This list overrides the default, which includes all numeric display fields.

You can use the asterisk (\*) wildcard character instead of a field list to indicate that all fields, numeric and alphanumeric, should be included on the summary lines. You can either use the asterisk to display all columns or reference the specific columns, numeric and alphanumeric, you want to display.

WHEN *expression*

Specifies the conditional display of subtotals based on a Boolean expression. You must end the expression with a semicolon.

You may also generate subtotals for the recalculated values with the ON TABLE command. Use the following syntax:

ON TABLE SUMMARIZE

**Example:** Using SUMMARIZE

The following request illustrates the use of SUMMARIZE to recalculate DG\_RATIO at the specified sort break, DEPARTMENT, and for the higher-level sort break, PAY\_DATE:

```
TABLE FILE EMPLOYEE
SUM GROSS DED_AMT AND COMPUTE
DG_RATIO/F4.2=DED_AMT/GROSS;
BY HIGHEST PAY_DATE BY DEPARTMENT
BY BANK_ACCT
WHERE BANK_ACCT NE 0
ON DEPARTMENT SUMMARIZE
END
```

The first portion of the output is:

PAY_DATE	DEPARTMENT	BANK_ACCT	GROSS	DED_AMT	DG_RATIO
82/08/31	MIS	40950036	\$1,540.00	\$725.34	.47
		122850108	\$1,815.00	\$1,261.40	.69
		163800144	\$2,255.00	\$1,668.69	.74
*TOTAL DEPARTMENT MIS			\$5,610.00	\$3,655.43	.65
	PRODUCTION	160633	\$2,475.00	\$1,427.24	.58
		136500120	\$1,342.00	\$522.28	.39
		819000702	\$2,238.50	\$1,746.03	.78
*TOTAL DEPARTMENT PRODUCTION			\$6,055.50	\$3,695.55	.61
*TOTAL PAY_DATE 82/08/31			\$11,665.50	\$7,350.98	.63

The last portion of the output is:

PAY_DATE	DEPARTMENT	BANK_ACCT	GROSS	DED_AMT	DG_RATIO
-----	-----	-----	-----	-----	-----
82/01/29	PRODUCTION	819000702	\$2,035.00	\$1,241.33	.61
*TOTAL DEPARTMENT PRODUCTION			\$2,035.00	\$1,241.33	.61
*TOTAL PAY_DATE 82/01/29			\$4,182.75	\$2,648.12	.63
81/12/31	MIS	163800144	\$2,147.75	\$1,406.79	.66
*TOTAL DEPARTMENT MIS			\$2,147.75	\$1,406.79	.66
*TOTAL PAY_DATE 81/12/31			\$2,147.75	\$1,406.79	.66
81/11/30	MIS	163800144	\$2,147.75	\$1,406.79	.66
*TOTAL DEPARTMENT MIS			\$2,147.75	\$1,406.79	.66
*TOTAL PAY_DATE 81/11/30			\$2,147.75	\$1,406.79	.66
TOTAL			\$65,887.25	\$41,521.18	.63

**Tip:** If you use SUB-TOTAL or SUBTOTAL rather than SUMMARIZE, the values of DG\_RATIO are added.

### **Example:** Using RECOMPUTE

The following request illustrates the use of RECOMPUTE to recalculate DG\_RATIO only at the specified sort break, DEPARTMENT.

```
TABLE FILE EMPLOYEE
SUM GROSS DED_AMT AND COMPUTE
DG_RATIO/F4.2=DED_AMT/GROSS;
BY HIGHEST PAY_DATE BY DEPARTMENT
BY BANK_ACCT
WHERE BANK_ACCT NE 0
ON DEPARTMENT RECOMPUTE
END
```

## Summarizing Alphanumeric Columns

The first portion of the output is:

PAY_DATE	DEPARTMENT	BANK_ACCT	GROSS	DED_AMT	DG_RATIO
-----	-----	-----	-----	-----	-----
82/08/31	MIS	40950036	\$1,540.00	\$725.34	.47
		122850108	\$1,815.00	\$1,261.40	.69
		163800144	\$2,255.00	\$1,668.69	.74
*TOTAL DEPARTMENT MIS			\$5,610.00	\$3,655.43	.65
	PRODUCTION	160633	\$2,475.00	\$1,427.24	.58
		136500120	\$1,342.00	\$522.28	.39
		819000702	\$2,238.50	\$1,746.03	.78
*TOTAL DEPARTMENT PRODUCTION			\$6,055.50	\$3,695.55	.61
82/07/30	MIS	40950036	\$1,540.00	\$725.34	.47
		122850108	\$1,815.00	\$1,261.40	.69

The last portion of the output is:

PAY_DATE	DEPARTMENT	BANK_ACCT	GROSS	DED_AMT	DG_RATIO
-----	-----	-----	-----	-----	-----
82/01/29	MIS	163800144	\$2,147.75	\$1,406.79	.66
*TOTAL DEPARTMENT MIS			\$2,147.75	\$1,406.79	.66
	PRODUCTION	819000702	\$2,035.00	\$1,241.33	.61
*TOTAL DEPARTMENT PRODUCTION			\$2,035.00	\$1,241.33	.61
81/12/31	MIS	163800144	\$2,147.75	\$1,406.79	.66
*TOTAL DEPARTMENT MIS			\$2,147.75	\$1,406.79	.66
81/11/30	MIS	163800144	\$2,147.75	\$1,406.79	.66
*TOTAL DEPARTMENT MIS			\$2,147.75	\$1,406.79	.66
TOTAL			\$65,887.25	\$41,521.18	.63

## Summarizing Alphanumeric Columns

By default, subtotals (using the SUBTOTAL and SUB-TOTAL commands) and recalculations (using the RECOMPUTE and SUMMARIZE commands) only display values for numeric report columns. However, you can include alphanumeric columns on these summary lines by either specifying the columns you want to display on the summary lines or by using the asterisk wildcard character to display all fields on the summary lines.



The alphanumeric value displayed on a SUBTOTAL or SUB-TOTAL line is either the first, minimum, maximum, or last alphanumeric value within the sort group, depending on the value of the SUMPREFIX parameter. On a RECOMPUTE or SUMMARIZE line, alphanumeric values are recalculated using the summary values for that line.

**Syntax:**      **How to Include All Columns on Summary Lines**

*ON sortfield summarycommand \**

where:

*sortfield*

Is the sort field for which a change in value triggers the summary line.

*summarycommand*

Is SUBTOTAL, SUB-TOTAL, RECOMPUTE, or SUMMARIZE.

\*

Indicates that all fields, numeric and alphanumeric, should be included on the summary lines. You can either use the asterisk to display all columns or reference the specific columns you want to display.

**Example:**      **Including Alphanumeric Fields on Summary Lines**

The following request against the GGSALES data source computes the alphanumeric equivalents of the DOLLARS and UNITS fields, creates an alphanumeric version of the formula for the ratio between DOLLARS and UNITS, and computes the numeric ratio between DOLLARS and UNITS. The RECOMPUTE \* command recomputes all values on a change of value for the state sort field:

```
SET SUMPREFIX=FST
TABLE FILE GGSALES
SUM PRODUCT DOLLARS/I8M AS 'Dollars' IN 22 UNITS AS 'Units'
COMPUTE Formula/A19 = EDIT(DOLLARS) | '/' | EDIT(UNITS) | '=';
COMPUTE Ratio/F8      = DOLLARS/UNITS;
BY ST
BY CATEGORY NOPRINT
WHERE ST EQ 'CA' OR 'IL'
ON ST RECOMPUTE *
ON TABLE SET PAGE NOPAGE
END
```

On the output, the alphanumeric formula is recomputed using the summed numeric fields. However, the product value is taken from the first product within each sort value, as that field is not recomputed and SUMPREFIX=FST:

State	Product	Dollars	Units	Formula	Ratio
-----	-----	-----	-----	-----	-----
CA	Capuccino	\$2,957,852	237246	02957852/00237246=	12
	Biscotti	\$2,770,508	222844	02770508/00222844=	12
	Coffee Grinder	\$1,935,863	152276	01935863/00152276=	13
*TOTAL	CA				
	Capuccino	\$7,664,223	612366	07664223/00612366=	13
IL	Espresso	\$1,398,779	109581	01398779/00109581=	13
	Biscotti	\$1,561,904	120976	01561904/00120976=	13
	Coffee Grinder	\$1,050,243	83541	01050243/00083541=	13
*TOTAL	IL				
	Espresso	\$4,010,926	314098	04010926/00314098=	13
TOTAL	Capuccino	\$11,675,149	926464	11675149/00926464=	13

Note that if the SUBTOTAL summary command had been used, the formula would not have been recomputed and would have displayed the values from the first line within each sort group.

**Reference: Usage Notes for Summarizing Alphanumeric Columns**

- ☐ Date fields and numeric and alphanumeric fields with date formatting options are not included on summary lines.
- ☐ Column total lines follow the same rules as summary lines.
- ☐ Summary values for ACROSS sort fields are supported.

**Manipulating Summary Values With Prefix Operators**

You can use the SUBTOTAL, SUB-TOTAL, RECOMPUTE, and SUMMARIZE commands at the ON TABLE level to specify the type of summary operation to use to produce the grand total line on the report.

In addition, prefix operators can be used with the summary options SUBTOTAL, SUB-TOTAL, RECOMPUTE, and SUMMARIZE at both the sort break and grand total levels. If the same field was aggregated using multiple prefix operators in the SUM command, you can use the prefix operator along with the field name to differentiate between the fields with multiple operators in the summary command.

Prefix operations on summary lines are performed on the retrieved, selected, and summed values that become the detail lines in the report. Unlike field-based prefix operations, they are not performed on each incoming record.

Each type of summary has its own purpose, and handles the prefix operators appropriately for the type of summary information to be displayed. For example, using AVE. at a sort field break produces the average within the sort group.

Alphanumeric fields can also be displayed on summary lines. In order to do this, you must either explicitly list the alphanumeric field name on the summary command, or use the asterisk (\*) wildcard to display all fields.

Different operations from two ON phrases for the same sort break display on the same summary line, and allow a mixture of operations on summary lines. The grand total line populates all fields populated by any summary command, even fields that are not specified in the grand total command.

If the same field is referenced in more than one ON phrase for the same sort break, the last function specified is applied.

The following prefix operators are supported for numeric fields:

- ☐ ASQ.
- ☐ AVE.
- ☐ CNT.
- ☐ FST.
- ☐ LST.
- ☐ MAX.
- ☐ MIN.
- ☐ SUM.

The following prefix operators are supported for alphanumeric fields:

- ☐ FST.
- ☐ LST.
- ☐ MAX.
- ☐ MIN.

- ❑ SUM. (means LST. if SUMPREFIX=LST or FST. if SUMPREFIX=FST)

## **Syntax:** How to Use Prefix Operators With Summary Values

```
{BY|ON} breakfield [AS 'text1'] sumoption [MULTILINES]
      [pref. ] [*| [field1 [[pref2. ] field2 ...]]]
      [AS 'text2'] [WHEN expression;
```

To replace the default grand total, use the following syntax

```
ON TABLE sumoption [pref. ] [field1 [[pref2. ] field2 ...]] [AS 'text2']
```

where:

*breakfield*

Is the sort field whose change in value triggers the summary operation. A BY phrase can include a summary command. When the value of the sort field changes, it triggers the summary operation.

*sumoption*

Can be one of the following: SUBTOTAL, SUB-TOTAL, RECOMPUTE, or SUMMARIZE.

'*text1*'

Is the column heading to use for the break field on the report output.

MULTILINES

Suppresses the printing of a summary line for every sort break that has only one detail line. Note that MULTILINES suppresses the summary line even if a prefix operator is used to specify a different operation for the summary line. MULTI-LINES is a synonym for MULTILINES. MULTILINES is not supported with horizontal (ACROSS) sort fields.

*pref.*

Is a prefix operator. When specified without a field list, the prefix operator is applied to every numeric column in the report output and every numeric column is populated with values on the summary row.

\*

Includes all display fields on the summary line. If a prefix operator is specified, it is applied to all fields. If the prefix operator is not supported with alphanumeric fields, alphanumeric fields are not included on the summary line.

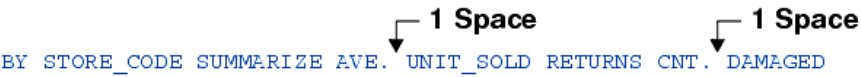
[*field1* [*field2* ... *fieldn*]]

Produces the type of summary specified by *sumoption* for the listed fields. If no field names are listed, the summary is produced for every numeric column in the report output.

*pref.* *field1* [*field2* ... *fieldn*] [*pref2.* *fieldm* ...]

The first prefix operator is applied to *field1* through *fieldn*. The second prefix operator is applied to *fieldm*. Only the fields specified are populated with values on the

summary row. Each prefix operator must be separated by a blank space from the following field name. For example:



*'text2'*

Is the text that prints on the left of the summary row.

*expression*

Is an expression that determines whether the summary operation is performed at each break.

### **Reference:** Usage Notes for Summary Prefix Operators

- ☐ COLUMN-TOTAL does not support prefix operators.
- ☐ Prefix operators PCT., RPCT., AND TOT. are not supported.
- ☐ Double prefix operators (such as PCT.CNT.) are not supported.
- ☐ When an ACROSS field is used in the request, the same field name displays over multiple columns (ACROSS groups) in the report output. A prefix operator applied to such a field on a summary line is applied to all of those columns.
- ☐ The SUM. prefix operator produces the same summary values as a summary phrase with no prefix operator.
- ☐ SUMMARIZE and RECOMPUTE apply the calculations defined in the associated COMPUTE command to the summary values. Therefore, in order to perform the necessary calculations, the SUMMARIZE or RECOMPUTE command must calculate all of the fields referenced in the COMPUTE command.
- ☐ If the same field is referenced by more than one summary operation with different prefix operators at each level, the default grand total (one produced without an ON TABLE *summaryoption* command) applies the operation specified by the first operator used in the report request (the left-most sort field in the output).

### **Example:** Using Prefix Operators With SUBTOTAL

The following example uses prefix operators to calculate the:

- ☐ Average list price by rating.
- ☐ Sum copies by category within the rating field.

Notice that the subtotal row for each rating contains a value only in the LISTPR column, and the subtotal row for each category contains a value only in the COPIES column. The grand total line contains values only for the columns that were subtotaled. Note the blank space between each prefix operator and the field name that follows it:

```
TABLE FILE MOVIES
PRINT COPIES LISTPR WHOLESALPR TITLE/A23
  BY RATING
  BY CATEGORY
  WHERE CATEGORY EQ 'CHILDREN' OR 'CLASSIC'
  WHERE RATING EQ 'G' OR 'NR'
  ON RATING SUBTOTAL AVE. LISTPR AS '*Ave: '
  ON CATEGORY SUBTOTAL SUM. COPIES AS '*Sum: '
END
```

The output is:

RATING	CATEGORY	COPIES	LISTPR	WHOLESALPR	TITLE
G	CHILDREN	2	44.95	29.99	SHAGGY DOG, THE
		2	29.95	12.50	ALICE IN WONDERLAND
		3	26.99	12.00	BAMBI
*Sum:	CHILDREN	7			
	CLASSIC	3	89.95	40.99	GONE WITH THE WIND
*Sum:	CLASSIC	3			
*Ave:	G		47.96		
NR	CHILDREN	1	19.95	10.00	SMURFS, THE
		1	19.95	9.75	SCOOBY-DOO-A DOG IN THE
		1	14.95	7.65	SESAME STREET-BEDTIME S
		1	14.98	7.99	ROMPER ROOM-ASK MISS MO
		1	29.95	15.99	SLEEPING BEAUTY
*Sum:	CHILDREN	5			
	CLASSIC	1	24.98	14.99	EAST OF EDEN
		3	39.99	20.00	CITIZEN KANE
		1	29.95	15.99	CYRANO DE BERGERAC
		1	19.99	10.95	MARTY
		2	19.99	10.95	MALTESE FALCON, THE
		2	19.95	9.99	ON THE WATERFRONT
		2	89.99	40.99	MUTINY ON THE BOUNTY
		2	19.99	10.95	PHILADELPHIA STORY, THE
		2	19.98	10.99	CAT ON A HOT TIN ROOF
		2	29.95	15.00	CASABLANCA
*Sum:	CLASSIC	18			
*Ave:	NR		27.64		
TOTAL		33	31.91		

**Example: Using SUBTOTAL at the Sort Break and Grand Total Levels**

The following example adds the ON TABLE SUBTOTAL command to the request in the previous example (*Using Prefix Operators With SUBTOTAL* on page 393) at the sort break level to calculate the minimum number of copies and maximum list price on the grand total line for the entire report:

```
TABLE FILE MOVIES
PRINT COPIES LISTPR WHOLESALPR TITLE/A23
  BY RATING
  BY CATEGORY
  WHERE CATEGORY EQ 'CHILDREN' OR 'CLASSIC'
  WHERE RATING   EQ 'G' OR 'NR'
  ON RATING      SUBTOTAL AVE. LISTPR AS '*Ave:  '
  ON CATEGORY    SUBTOTAL SUM. COPIES AS '*Sum:  '
  ON TABLE SUBTOTAL MIN. COPIES MAX. LISTPR
END
```

The output is exactly the same as in the previous request, except for the grand total line:

RATING	CATEGORY	COPIES	LISTPR	WHOLESALEPR	TITLE
-----	-----	-----	-----	-----	-----
G	CHILDREN	2	44.95	29.99	SHAGGY DOG, THE
		2	29.95	12.50	ALICE IN WONDERLAND
		3	26.99	12.00	BAMBI
*Sum:	CHILDREN	7			
	CLASSIC	3	89.95	40.99	GONE WITH THE WIND
*Sum:	CLASSIC	3			
*Ave:	G		47.96		
NR	CHILDREN	1	19.95	10.00	SMURFS, THE
		1	19.95	9.75	SCOOBY-DOO-A DOG IN THE
		1	14.95	7.65	SESAME STREET-BEDTIME S
		1	14.98	7.99	ROMPER ROOM-ASK MISS MO
		1	29.95	15.99	SLEEPING BEAUTY
*Sum:	CHILDREN	5			
	CLASSIC	1	24.98	14.99	EAST OF EDEN
		3	39.99	20.00	CITIZEN KANE
		1	29.95	15.99	CYRANO DE BERGERAC
		1	19.99	10.95	MARTY
		2	19.99	10.95	MALTESE FALCON, THE
		2	19.95	9.99	ON THE WATERFRONT
		2	89.99	40.99	MUTINY ON THE BOUNTY
		2	19.99	10.95	PHILADELPHIA STORY, THE
		2	19.98	10.99	CAT ON A HOT TIN ROOF
		2	29.95	15.00	CASABLANCA
*Sum:	CLASSIC	18			
*Ave:	NR		27.64		
TOTAL		1	89.99		

## Example: Differentiating Between Fields With Multiple Prefix Operators

The following request uses both the MAX. and MIN. prefix operators with the UNITS field. On the summary commands, these are differentiated by referencing them as MAX.UNITS and MIN.UNITS.

```
TABLE FILE GGSales
SUM MAX.UNITS MIN.UNITS
  BY REGION
  BY ST
ON REGION RECOMPUTE MAX. MAX.UNITS MIN. MIN.UNITS
WHERE DATE GE 19971001
WHERE REGION EQ 'West' OR 'Northeast'
ON TABLE RECOMPUTE MIN. MAX.UNITS MAX. MIN.UNITS
ON TABLE SET PAGE NOPAGE
END
```



On the report output, the summary for each region displays the maximum of the state maximum values and the minimum of the state minimum values. The summary for the entire report displays the minimum of the state maximum values and the maximum of the state minimum values. The report output is shown in the following image:

<u>Region</u>	<u>State</u>	<u>MAX Unit Sales</u>	<u>MIN Unit Sales</u>
Northeast	CT	1770	101
	MA	1780	146
	NY	1797	73
*TOTAL Northeast		1797	73
West	CA	1794	72
	WA	1787	257
*TOTAL West		1794	72
TOTAL		1770	257

**Example: Displaying an Alphanumeric Field on a Summary Line**

The following request displays the sum of the list price field and the minimum value of the director field by rating:

```
TABLE FILE MOVIES
PRINT COPIES LISTPR WHOLESALPR DIRECTOR
BY RATING
BY CATEGORY
WHERE CATEGORY EQ 'CHILDREN' OR 'CLASSIC'
WHERE RATING EQ 'G' OR 'NR'
WHERE DIRECTOR NE ' '
ON RATING SUBTOTAL SUM. LISTPR MIN. DIRECTOR AS '*A/N:'
END
```

The output is:

RATING	CATEGORY	COPIES	LISTPR	WHOLESALEPR	DIRECTOR
-----	-----	-----	-----	-----	-----
G	CHILDREN	2	44.95	29.99	BARTON C.
		2	29.95	12.50	GEROMINI
		3	26.99	12.00	DISNEY W.
	CLASSIC	3	89.95	40.99	FLEMING V
*A/N: G			191.84		BARTON C.
NR	CHILDREN	1	29.95	15.99	DISNEY W.
	CLASSIC	1	24.98	14.99	KAZAN E.
		3	39.99	20.00	WELLES O.
		1	29.95	15.99	GORDON M.
		1	19.99	10.95	MANN D.
		2	19.99	10.95	HUSTON J.
		2	19.95	9.99	KAZAN E.
		2	89.99	40.99	MILESTONE L.
		2	19.99	10.95	CUKOR G.
		2	19.98	10.99	BROOKS R.
		2	29.95	15.00	CURTIZ M.
*A/N: NR			344.71		BROOKS R.
TOTAL			536.55		BARTON C.

**Example:**    **Displaying All Fields on a Summary Line**

The following request displays the sum of every display field on the subtotal line. The director field is alphanumeric, so the last value displays:

```
TABLE FILE MOVIES
PRINT COPIES LISTPR WHOLESALEPR DIRECTOR
BY RATING
BY CATEGORY
WHERE CATEGORY EQ 'CHILDREN' OR 'CLASSIC'
WHERE RATING   EQ 'G' OR 'NR'
WHERE DIRECTOR NE ' '
ON RATING SUBTOTAL SUM. * AS '*All: '
END
```

The output is:

RATING	CATEGORY	COPIES	LISTPR	WHOLESALEPR	DIRECTOR
-----	-----	-----	-----	-----	-----
G	CHILDREN	2	44.95	29.99	BARTON C.
		2	29.95	12.50	GEROMINI
		3	26.99	12.00	DISNEY W.
	CLASSIC	3	89.95	40.99	FLEMING V
*All:	G	10	191.84	95.48	FLEMING V
NR	CHILDREN	1	29.95	15.99	DISNEY W.
		1	24.98	14.99	KAZAN E.
	CLASSIC	3	39.99	20.00	WELLES O.
		1	29.95	15.99	GORDON M.
		1	19.99	10.95	MANN D.
		2	19.99	10.95	HUSTON J.
		2	19.95	9.99	KAZAN E.
		2	89.99	40.99	MILESTONE L.
		2	19.99	10.95	CUKOR G.
		2	19.98	10.99	BROOKS R.
		2	29.95	15.00	CURTIZ M.
*All:	NR	19	344.71	176.79	CURTIZ M.
TOTAL		29	536.55	272.27	CURTIZ M.

## Controlling Summary Line Processing

When processing summary lines, you can control whether SUBTOTAL and RECOMPUTE commands are propagated to the grand total row of a report.

If the summary line contains fields with and without prefix operators, those fields without prefix operators are processed as though they were specified with the operator SUM.

The function of the SET SUMMARYLINES command is to make the processing of SUBTOTAL, SUB-TOTAL, SUMMARIZE, and RECOMPUTE on the grand total line consistent with how they work for sort field breaks. The setting that invokes this type of processing is SET SUMMARYLINES=EXPLICIT.

When SUBTOTAL and RECOMPUTE are used at a sort break level, they do not propagate to other sort breaks. SUB-TOTAL and SUMMARIZE propagate to all higher level sort breaks.

The grand total can be considered the highest level sort field in a request. However, by default, all of the summary options, not just SUB-TOTAL and SUMMARIZE, propagate to the grand total level.

The SET SUMMARYLINES=EXPLICIT command prevents the propagation of SUBTOTAL and RECOMPUTE to the grand total. In addition, if all summary commands in the request specify field lists, only the specified fields are aggregated and displayed on the grand total line.

When SUBTOTAL and RECOMPUTE are the only summary commands used in the request, a grand total line is produced only if it is explicitly specified in the request using the ON TABLE SUBTOTAL/SUB-TOTAL/RECOMPUTE/SUMMARIZE phrase. If the ON TABLE phrase specifies a field list, only those fields are aggregated and displayed.

Note that you can always suppress the grand total line using the ON TABLE NOTOTAL command in the request.

### **Syntax:** How to Control Summary Line Processing

```
SET SUMMARYLINES = {NEW|OLD|EXPLICIT}
```

where:

#### [NEW](#)

Propagates all summary operations to the grand total line. Fields listed in a summary command are populated only on summary lines created by that summary command and on summary lines created by propagation of that summary command. NEW is the default value.

The alphanumeric value displayed on a SUBTOTAL or SUB-TOTAL line is either the first or last alphanumeric value within the sort group, depending on the value of the SUMPREFIX parameter. On a RECOMPUTE or SUMMARIZE line, alphanumeric values are recalculated using the summary values for that line.

#### [OLD](#)

This value is no longer supported and is processed as NEW.

#### [EXPLICIT](#)

Does not propagate SUBTOTAL and RECOMPUTE to the grand total line. Fields listed in a summary command are populated only on summary lines created by that summary command and on summary lines created by propagation of that summary command.

**Note:** This command is not supported in a request using the ON TABLE SET syntax.

### **Reference:** Usage Notes for SET SUMMARYLINES

- ☐ SET SUMMARYLINES is not supported within a TABLE request (ON TABLE).
- ☐ If COLUMN-TOTAL is specified in the request, all numeric fields are totaled on the grand total line unless the COLUMN-TOTAL phrase lists specific fields. If the COLUMN-TOTAL phrase lists specific fields, those fields and any fields propagated by SUB-TOTAL or SUMMARIZE commands are totaled.

- ❑ A summary command with a list of field names populates only those columns on the associated summary line.

For example:

```
TABLE FILE MOVIES
PRINT COPIES LISTPR WHOLESALPR
  BY RATING
  BY CATEGORY
  WHERE CATEGORY EQ 'CHILDREN'
  WHERE RATING   EQ 'G'
  ON RATING      SUBTOTAL LISTPR AS '*LIST'
  ON CATEGORY    SUBTOTAL COPIES AS '*COPY'
END
```

The output has subtotals for COPIES on the CATEGORY sort break and for LISTPR on the RATING sort break. Both columns are populated on the grand total line. WHOLESALPR is not referenced in either SUBTOTAL command and, therefore, is not on any summary line:

RATING	CATEGORY	COPIES	LISTPR	WHOLESALPR
-----				
G	CHILDREN	2	44.95	29.99
		2	29.95	12.50
		3	26.99	12.00
*COPY	CHILDREN	7		
*LIST	G		101.89	
TOTAL		7	101.89	

**Example: Using SET SUMMARYLINES With SUBTOTAL**

The following request using the MOVIES data source has a sort break for CATEGORY that subtotals the COPIES field and a sort break for RATING that subtotals the LISTPR field:

```
TABLE FILE MOVIES
SUM COPIES LISTPR WHOLESALPR
  BY RATING
  BY CATEGORY
  WHERE CATEGORY EQ 'CHILDREN'
  WHERE RATING   EQ 'G'
  ON RATING SUBTOTAL COPIES
  ON CATEGORY SUBTOTAL LISTPR
END
```

Running the request with SUMMARYLINES=NEW subtotals COPIES only for the RATING sort break and subtotals LISTPR only for the CATEGORY sort break but propagates both to the grand total line:

RATING	CATEGORY	COPIES	LISTPR	WHOLESALEPR
-----	-----	-----	-----	-----
G	CHILDREN	7	101.89	54.49
*TOTAL CHILDREN			101.89	
*TOTAL G		7		
TOTAL		7	101.89	

Running the request with SUMMARYLINES=EXPLICIT subtotals COPIES only for the RATING sort break and subtotals LISTPR only for the CATEGORY sort break. It does not produce a grand total line:

RATING	CATEGORY	COPIES	LISTPR	WHOLESALEPR
-----	-----	-----	-----	-----
G	CHILDREN	7	101.89	54.49
*TOTAL CHILDREN			101.89	
*TOTAL G		7		

Adding the phrase ON TABLE SUBTOTAL WHOLESALEPR with SUMMARYLINES=EXPLICIT produces a grand total line with the WHOLESALEPR field subtotaled:

RATING	CATEGORY	COPIES	LISTPR	WHOLESALEPR
-----	-----	-----	-----	-----
G	CHILDREN	7	101.89	54.49
*TOTAL CHILDREN			101.89	
*TOTAL G		7		
TOTAL				54.49

**Example: Using COLUMN-TOTAL With SET SUMMARYLINES=EXPLICIT**

The following request using the MOVIES data source has a sort break for CATEGORY for which subtotals the COPIES field and a sort break for RATING that subtotals the LISTPR field. It also has an ON TABLE COLUMN-TOTAL phrase:

```
SET SUMMARYLINES=EXPLICIT
TABLE FILE MOVIES
SUM COPIES LISTPR WHOLESALPR
BY RATING
BY CATEGORY
WHERE CATEGORY EQ 'CHILDREN'
WHERE RATING EQ 'G'
ON RATING SUBTOTAL COPIES
ON CATEGORY SUBTOTAL LISTPR
ON TABLE COLUMN-TOTAL
END
```

The grand total line displays a column total for all numeric columns because of the ON TABLE COLUMN-TOTAL phrase:

RATING	CATEGORY	COPIES	LISTPR	WHOLESALPR
-----	-----	-----	-----	-----
G	CHILDREN	7	101.89	54.49
*TOTAL CHILDREN			101.89	
*TOTAL G		7		
TOTAL		7	101.89	54.49

The following request has an ON TABLE SUBTOTAL WHOLESALPR command. It also has an ON TABLE COLUMN-TOTAL phrase:

```
SET SUMMARYLINES=EXPLICIT
TABLE FILE MOVIES
SUM COPIES LISTPR WHOLESALPR
BY RATING
BY CATEGORY
WHERE CATEGORY EQ 'CHILDREN'
WHERE RATING EQ 'G'
ON RATING SUBTOTAL COPIES
ON CATEGORY SUBTOTAL LISTPR
ON TABLE SUBTOTAL WHOLESALPR
ON TABLE COLUMN-TOTAL
END
```

The grand total line displays a column total only for the WHOLESALEPR column because of the ON TABLE SUBTOTAL command:

RATING	CATEGORY	COPIES	LISTPR	WHOLESALEPR
-----	-----	-----	-----	-----
G	CHILDREN	7	101.89	54.49
*TOTAL CHILDREN			101.89	
*TOTAL G		7		
TOTAL				54.49

Using SUB-TOTAL instead of SUBTOTAL causes COPIES and LISTPR to be aggregated on the grand total line. WHOLESALEPR is totaled because it is listed in the COLUMN-TOTAL phrase. The subtotal for LISTPR propagates to the RATING sort break as well as to the grand total:

```
SET SUMMARYLINES=EXPLICIT
TABLE FILE MOVIES
SUM COPIES LISTPR WHOLESALEPR
BY RATING
BY CATEGORY
WHERE CATEGORY EQ 'CHILDREN'
WHERE RATING EQ 'G'
ON RATING SUB-TOTAL COPIES
ON CATEGORY SUB-TOTAL LISTPR
ON TABLE COLUMN-TOTAL WHOLESALEPR
END
```

The output is:

RATING	CATEGORY	COPIES	LISTPR	WHOLESALEPR
-----	-----	-----	-----	-----
G	CHILDREN	7	101.89	54.49
*TOTAL CHILDREN			101.89	
*TOTAL G		7	101.89	
TOTAL		7	101.89	54.49

Using Prefix Operators With Calculated Values

If a request includes the RECOMPUTE or SUMMARIZE command, the expression specified in the associated COMPUTE command is applied using values from the summary line. The columns used to recompute the expression can have prefix operators. The recomputed column, regardless of the prefix operator specified for it, applies these input values to the expression specified in the COMPUTE command. Therefore, any supported prefix operator can be specified for the recomputed report column without affecting the calculated value.



All fields used in the COMPUTE command must be displayed by the RECOMPUTE or SUMMARIZE command in order to be populated. If any field used in the expression is not populated, the calculated value returned for the expression is unpredictable.

**Example: Using Prefix Operators With RECOMPUTE**

The first request creates a calculated field named DIFF, which is the difference between DOLLARS and BUDDOLLARS. This value is then recomputed for each region, without using prefix operators.

```
TABLE FILE GGSALES
SUM UNITS DOLLARS BUDDOLLARS
AND COMPUTE DIFF/I10 = DOLLARS-BUDDOLLARS;
  BY REGION
  BY CATEGORY
  WHERE CATEGORY EQ 'Food' OR 'Coffee'
  WHERE REGION EQ 'West' OR 'Midwest'
  ON REGION RECOMPUTE
END
```

The recomputed value is the difference between the totals for DOLLARS and BUDDOLLARS.

Region	Category	Unit Sales	Dollar Sales	Budget Dollars	DIFF
-----	-----	-----	-----	-----	----
Midwest	Coffee	332777	4178513	4086032	92481
	Food	341414	4338271	4220721	117550
*TOTAL Midwest		674191	8516784	8306753	210031
West	Coffee	356763	4473517	4523963	-50446
	Food	340234	4202337	4183244	19093
*TOTAL West		696997	8675854	8707207	-31353
TOTAL		1371188	17192638	17013960	178678

The following request uses prefix operators in the RECOMPUTE command to calculate the maximum DOLLARS and the minimum BUDDOLLARS and then recompute DIFF. No matter which prefix operator we specify for DIFF, it is calculated as the difference between the values in the DOLLARS and BUDDOLLARS columns. If any of the fields used in the calculation (DOLLARS, BUDDOLLARS, and DIFF) do not display on the summary row, the calculation cannot be performed.

```
TABLE FILE GGSales
SUM UNITS DOLLARS BUDDOLLARS
AND COMPUTE DIFF/I10 = DOLLARS-BUDDOLLARS;
  BY REGION
  BY CATEGORY
  WHERE CATEGORY EQ 'Food' OR 'Coffee'
  WHERE REGION EQ 'West' OR 'Midwest'
  ON REGION RECOMPUTE MAX. DOLLARS MIN. BUDDOLLARS AVE. DIFF
END
```

The output is:

Region	Category	Unit Sales	Dollar Sales	Budget Dollars	DIFF
-----	-----	-----	-----	-----	-----
Midwest	Coffee	332777	4178513	4086032	92481
	Food	341414	4338271	4220721	117550
*TOTAL Midwest			4338271	4086032	252239
West	Coffee	356763	4473517	4523963	-50446
	Food	340234	4202337	4183244	19093
*TOTAL West			4473517	4183244	290273

**Example:** Using RECOMPUTE at the Sort Break and Grand Total Levels

The following example adds the ON TABLE RECOMPUTE command to the previous request (*Using Prefix Operators With RECOMPUTE* on page 405) to calculate the average values for each column. Notice that the value of DIFF is calculated as the difference between the values in the Dollar Sales and the Budget Dollars columns on the grand total line:

```
TABLE FILE GGSales
SUM UNITS DOLLARS BUDDOLLARS
AND COMPUTE DIFF/I10 = DOLLARS-BUDDOLLARS;
  BY REGION
  BY CATEGORY
  WHERE CATEGORY EQ 'Food' OR 'Coffee'
  WHERE REGION EQ 'West' OR 'Midwest'
  ON REGION RECOMPUTE MAX. DOLLARS MIN. BUDDOLLARS DIFF
  ON TABLE RECOMPUTE AVE.
END
```

The output is:

Region	Category	Unit Sales	Dollar Sales	Budget Dollars	DIFF
-----	-----	-----	-----	-----	-----
Midwest	Coffee	332777	4178513	4086032	92481
	Food	341414	4338271	4220721	117550
*TOTAL Midwest			4338271	4086032	252239
West	Coffee	356763	4473527	4523963	-50436
	Food	340234	4202338	4183244	19094
*TOTAL West			4473527	4183244	290283
TOTAL		342797	4298162	4253490	44672

### Using Multiple SUB-TOTAL or SUMMARIZE Commands With Prefix Operators

SUB-TOTAL and SUMMARIZE propagate their operations to all higher-level sort fields. If a request uses SUB-TOTAL or SUMMARIZE at multiple sort levels, more than one prefix operator may apply to the same field.

When a SUB-TOTAL or SUMMARIZE command on a lower-level sort field propagates up to the higher levels, it applies its prefix operators only to those fields that did not already have different prefix operators specified at the higher level. For any field that had a prefix operator specified at a higher level, the original prefix operator is applied at the level at which it was first specified and to the grand total line, unless a different operator is specified for the grand total line.

#### *Example:* Using Multiple SUB-TOTAL Commands With Prefix Operators

The following illustrates prefix operators work in a request that has multiple SUB-TOTAL commands, each with a different prefix operator.

```

DEFINE FILE GGSales
YEAR/YY = DATE;
END

TABLE FILE GGSales
SUM  UNITS DOLLARS/D10.2 BUDDOLLARS
    BY YEAR
    BY ST
    BY REGION
    BY CATEGORY
WHERE REGION EQ 'West' OR 'Midwest'
WHERE ST      EQ 'CA' OR 'IL'
WHERE YEAR EQ '1996' OR '1997'
    ON YEAR SUB-TOTAL CNT. UNITS AS '*CNT. UNITS:'
    ON ST SUB-TOTAL AVE. DOLLARS AS '*AVE. $:'
    ON REGION SUB-TOTAL MIN. AS '*MIN.:'
END

```

In the following report output, some of the values have been manually italicized or bolded for clarity:

- ❑ Outlined rows are the rows generated by the SUB-TOTAL commands.
- ❑ Subtotal values in the normal typeface are the count of unit sales generated by the command ON YEAR SUB-TOTAL CNT. UNITS. This is the topmost summary command, and therefore does not propagate to any other summary lines.
- ❑ Subtotal values in italic are average dollar sales generated by the command ON ST SUB-TOTAL AVE. DOLLARS. This is the second summary command, and therefore propagates to the DOLLARS column of summary lines for the YEAR sort field.

- ❑ Subtotal values in boldface are minimums within their sort groups generated by the command ON REGION SUB-TOTAL MIN. This is the last summary command, and therefore propagates to all other summary lines, but only calculates minimum values for those columns not already populated with a count or an average.

<u>YEAR</u>	<u>State</u>	<u>Region</u>	<u>Category</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>	<u>Budget Dollars</u>
1996	CA	West	Coffee	117529	1,484,873.00	1452548
			Food	116389	1,443,083.00	1414902
			Gifts	74948	947,783.00	946382
*MIN.: West				<b>74948</b>	<b>947,783.00</b>	<b>946382</b>
*AVE. \$: CA				<b>74948</b>	1,291,913.00	<b>946382</b>
	IL	Midwest	Coffee	52348	683,559.00	628333
			Food	58777	768,715.00	742943
			Gifts	38405	481,515.00	487090
*MIN.: Midwest				<b>38405</b>	<b>481,515.00</b>	487090
*AVE. \$: IL				<b>38405</b>	644,596.33	487090
*CNT. UNITS: 1996				6	968,254.67	<b>487090</b>
1997	CA	West	Coffee	118044	1,453,013.00	1507092
			Food	106322	1,325,429.00	1302582
			Gifts	77328	988,080.00	961841
*MIN.: West				<b>77328</b>	<b>988,080.00</b>	<b>961841</b>
*AVE. \$: CA				<b>77328</b>	1,255,507.33	<b>961841</b>
	IL	Midwest	Coffee	57233	715,220.00	737931
			Food	59293	754,132.00	737912
			Gifts	41527	521,260.00	532647
*MIN.: Midwest				<b>41527</b>	<b>521,260.00</b>	<b>532647</b>
*AVE. \$: IL				<b>41527</b>	663,537.33	<b>532647</b>
*CNT. UNITS: 1997				6	959,522.33	<b>532647</b>
TOTAL				12	963,888.50	<b>487090</b>

## Combinations of Summary Commands

You can specify a different summary operation for each sort break (BY or ACROSS field).

If you have multiple summary commands for the same sort field, the following message displays and the last summary command specified in the request is used:

(FOC36359) MORE THAN 1 SUBTOTAL/SUB-TOTAL/RECOMPUTE/SUMMARIZE

There is more than one SUBTOTAL/SUB-TOTAL/RECOMPUTE/SUMMARIZE on the same key field which is not allowed. The last one specified will override the rest.

SUMMARIZE and SUB-TOTAL, which propagate their summary operations to higher level sort breaks, skip those fields at higher level sort breaks that have their own summary commands. The propagation of summary operations depends on whether prefix operator processing is used for summary lines. If prefix operators are:

- ☐ Not used on summary lines, if any summary command specifies a field list, only the fields specified on the summary line field lists are populated on the report.
- ☐ Used on summary lines, SUB-TOTAL and SUMMARIZE propagate to:
  - ☐ All fields at higher level sort breaks that do not have their own summary command.
  - ☐ Fields not specified in the field list at higher level sort breaks that do have their own summary commands (columns that would have been empty). Note that this is the only technique that allows different fields at the same sort break to have different summary options.

Prefix operators on summary lines result in the same values whether the command is RECOMPUTE/SUMMARIZE or SUBTOTAL/SUB-TOTAL. For a computed field, the prefix operator is not applied, and the value is recalculated using the expression in the COMPUTE command and the values from the summary line.

When you use different summary commands for different sort fields, the default grand total row inherits the summary command associated with the first sort field in the request. You can change the operation performed at the grand total level by using the ON TABLE phrase to specify a specific summary command.

**Note:** The grand total is considered the highest sort level. Therefore, although you can use the SUMMARIZE or SUB-TOTAL command at the grand total level, these commands apply only to the grand total and are not propagated to any other line on the report. On the grand total level SUMMARIZE operates as a RECOMPUTE command, and SUB-TOTAL operates as a SUBTOTAL command.

**Example: Using SUBTOTAL and RECOMPUTE in a Request**

In the following request, the first sort field specified is COPIES, which is associated with the RECOMPUTE command. Therefore, on the grand total line, the value of RATIO is correctly recomputed and the values of LISTPR and WHOLESalePR are summed (because this is the default operation when the field is not calculated by a COMPUTE command).

```
TABLE FILE MOVIES
PRINT DIRECTOR LISTPR WHOLESalePR
COMPUTE RATIO = LISTPR/WHOLESalePR;
BY COPIES
BY RATING
WHERE COPIES LT 3
WHERE DIRECTOR EQ 'DISNEY W.' OR 'HITCHCOCK A.'
ON COPIES RECOMPUTE AS '*REC: '
ON RATING SUBTOTAL AS '*SUB: '
END
```

The output is:

COPIES	RATING	DIRECTOR	LISTPR	WHOLESalePR	RATIO
-----	-----	-----	-----	-----	-----
1	NR	DISNEY W.	29.95	15.99	1.87
*SUB:	NR		29.95	15.99	1.87
*REC:	1		29.95	15.99	1.87
2	NR	HITCHCOCK A.	19.98	9.00	2.22
*SUB:	NR		19.98	9.00	2.22
	PG	HITCHCOCK A.	19.98	9.00	2.22
		HITCHCOCK A.	19.98	9.00	2.22
*SUB:	PG		39.96	18.00	4.44
2	PG13	HITCHCOCK A.	19.98	9.00	2.22
*SUB:	PG13		19.98	9.00	2.22
	R	HITCHCOCK A.	19.98	9.00	2.22
*SUB:	R		19.98	9.00	2.22
*REC:	2		99.90	45.00	2.22
TOTAL			129.85	60.99	2.13

If you reverse the BY fields, the grand total line sums the RATIO values as well as the LISTPR and WHOLESalePR values because the SUBTOTAL command controls the grand total line:

TOTAL			129.85	60.99	12.97
-------	--	--	--------	-------	-------

You can change the operation performed at the grand total level by adding the following command to the request:

```
ON TABLE RECOMPUTE
```

The grand total line then displays the recomputed values:

TOTAL	129.85	60.99	2.13
-------	--------	-------	------

### **Example:** Using SUB-TOTAL With Multiple Summary Commands

In the following request, the SUB-TOTAL command propagates its operation to the DIRECTOR sort field (see the total line for HITCHCOCK, on which the RATIO values are subtotaled, not recomputed).

SUB-TOTAL is not propagated to the RATING sort field which has its own RECOMPUTE command, and for this sort field the RATIO value is recomputed. The grand total line is recomputed because RECOMPUTE is performed on a higher level sort field than SUB-TOTAL.

```
TABLE FILE MOVIES
PRINT LISTPR WHOLESalePR
COMPUTE RATIO = LISTPR/WHOLESalePR;
BY DIRECTOR
BY RATING
BY COPIES
WHERE COPIES LT 3
WHERE DIRECTOR EQ 'HITCHCOCK A.'
ON COPIES SUB-TOTAL AS '*SUB: '
ON RATING RECOMPUTE AS '*REC: '
END
```



The output is:

DIRECTOR	RATING	COPIES	LISTPR	WHOLESALEPR	RATIO
-----	-----	-----	-----	-----	-----
HITCHCOCK A.	NR	2	19.98	9.00	2.22
*SUB: 2			19.98	9.00	2.22
*REC: NR			19.98	9.00	2.22
	PG	2	19.98	9.00	2.22
			19.98	9.00	2.22
*SUB: 2			39.96	18.00	4.44
*REC: PG			39.96	18.00	2.22
	PG13	2	19.98	9.00	2.22
*SUB: 2			19.98	9.00	2.2
*REC: PG13			19.98	9.00	2.2
HITCHCOCK A.	R	2	19.98	9.00	2.2
*SUB: 2			19.98	9.00	2.2
*REC: R			19.98	9.00	2.2
*TOTAL DIRECTOR HITCHCOCK A.			99.90	45.00	11.1
TOTAL			99.90	45.00	2.2

### **Example:** Using Multiple Summary Commands With Prefix Operators

The following request prints the average value of LISTPR and the recomputed value of RATIO on the lines associated with sort field RATING. The SUB-TOTAL command associated with sort field COPIES is propagated to all fields on the DIRECTOR sort field lines and to the WHOLESALEPR and RATIO1 columns associated with the RATING sort field. The grand total line is suppressed for this request.

```
TABLE FILE MOVIES
PRINT LISTPR WHOLESALEPR
COMPUTE RATIO/D6.2 = LISTPR/WHOLESALEPR;
COMPUTE RATIO1/D6.2 = LISTPR/WHOLESALEPR;
BY DIRECTOR
BY RATING
BY COPIES
WHERE COPIES LT 3
  WHERE DIRECTOR EQ 'KAZAN E.'
  ON RATING RECOMPUTE AVE. LISTPR RATIO AS '*REC: '
  ON COPIES SUB-TOTAL AS '*SUB: '
  ON TABLE NOTOTAL
END
```

On the output:

- ❑ The values of WHOLESalePR and RATIO1 on the row labeled \*REC are subtotals because of propagation of the SUB-TOTAL command to the fields not specified in the RECOMPUTE command.
- ❑ The LISTPR value is an average and the value of RATIO (which has the same definition as RATIO1) is recomputed because these two fields are specified in the RECOMPUTE command.
- ❑ The SUB-TOTAL command is propagated to the DIRECTOR row.

The output is:

DIRECTOR	RATING	COPIES	LISTPR	WHOLESalePR	RATIO	RATIO1
-----	-----	-----	-----	-----	-----	-----
KAZAN E.	NR	1	24.98	14.99	1.67	1.67
*SUB: 1			24.98	14.99	1.67	1.67
		2	19.95	9.99	2.00	2.00
*SUB: 2			19.95	9.99	2.00	2.00
*REC: NR			22.46	24.98	.90	3.66
*TOTAL DIRECTOR KAZAN E.			44.93	24.98	3.66	3.66

### Example: Propagation of Summary Commands With Field Lists

In the following request, the RECOMPUTE command has a field list.

```
TABLE FILE MOVIES
PRINT LISTPR WHOLESalePR
COMPUTE RATIO/D6.2 = LISTPR/WHOLESalePR;
COMPUTE RATIO1/D6.2 = LISTPR/WHOLESalePR;
BY DIRECTOR
BY RATING
BY COPIES
WHERE COPIES LT 3
  WHERE DIRECTOR EQ 'KAZAN E.'
  ON RATING RECOMPUTE LISTPR RATIO AS '*REC: '
  ON COPIES SUB-TOTAL AS '*SUB: '
END
```

SUB-TOTAL propagates to all of the columns that would otherwise be unpopulated. The grand total line inherits the RECOMPUTE command for the fields listed in its field list, and the SUB-TOTAL command propagates to the other columns:

DIRECTOR	RATING	COPIES	LISTPR	WHOLESALEPR	RATIO	RATIO1
-----	-----	-----	-----	-----	-----	-----
KAZAN E.	NR	1	24.98	14.99	1.67	1.67
*SUB: 1			24.98	14.99	1.67	1.67
		2	19.95	9.99	2.00	2.00
*SUB: 2			19.95	9.99	2.00	2.00
*REC: NR			44.93	24.98	1.80	3.66
*TOTAL DIRECTOR KAZAN E.			44.93	24.98	3.66	3.66
TOTAL			44.93	24.98	1.80	3.66

**Reference:** Usage Notes for Combinations of Summary Commands

- ❑ SET SUMMARYLINES=EXPLICIT affects propagation of summary commands to the grand total line by making it consistent with the behavior for any sort break. Therefore, with this setting in effect, SUB-TOTAL and SUMMARIZE propagate to the grand total line but SUBTOTAL and RECOMPUTE do not.

**Producing Summary Columns for Horizontal Sort Fields**

The summary commands SUBTOTAL, SUB-TOTAL, SUMMARIZE, and RECOMPUTE can be used with horizontal sort breaks.

When a request has multiple display fields and an ACROSS sort field, the report output has multiple columns under each ACROSS value. If you want to apply a summary field to some of the columns for each ACROSS value, but not others, you can specify the field names you want summarized. This technique is most useful for report requests that use the OVER phrase to place the fields on separate rows

**Syntax:** How to Produce a Summary Operation on a Horizontal Sort Field

```
{ACROSS|ON} acrossfield [AS 'text1'] sumoption [AS 'text2']
      [COLUMNS c1 [AND c2 ...]]
```

or

```
ACROSS acrossfieldsumoption [field1field2 ... fieldn]
```

or

*ACROSS acrossfield*

*ON acrossfieldsumoption [field1field2 ... fieldn]*

where:

*acrossfield*

Is the ACROSS field whose for which you want to generate the summary option. The end of the values for the ACROSS field triggers the summary operation.

*sumoption*

Can be one of the following: SUBTOTAL, SUB-TOTAL, RECOMPUTE, or SUMMARIZE.

*'text1'*

Is the column heading to use for the break field on the report output.

*'text2'*

Is the text that prints on the top of the summary column.

*COLUMNSc1, c2 ...*

Lists the specific ACROSS values that you want to display on the report output in the order in which you want them. This list of values cannot be specified in an ON phrase. If it is specified in an ACROSS phrase, it must be the last option specified in the ACROSS phrase.

*field1field2 ... fieldn*

Are the fields that will have the summary command applied. If no fields are listed, all fields will be summarized.

### **Reference: Usage Notes for Summaries on ACROSS Fields**

- ☐ SUMMARIZE and SUB-TOTAL operate on the ACROSS field for which they are specified and for all higher level ACROSS fields. They do not operate on BY fields. SUBTOTAL and RECOMPUTE operate only on the ACROSS field for which they are specified. However, the summary is not produced until the higher level ACROSS field changes value.
- ☐ SUMMARIZE and SUB-TOTAL commands specified for a BY field operate on that BY and all higher level BY fields. They do not operate on ACROSS fields.
- ☐ ROW-TOTAL, ACROSS-TOTAL, SUBTOTAL, and SUB-TOTAL sum the values in the columns. Unlike SUMMARIZE and RECOMPUTE, they do not reapply calculations other than sums.
- ☐ Summary commands specified in an ON TABLE phrase operate on columns, not rows.
- ☐ With ACROSS, summary columns only display at the end of the ACROSS group (when the higher-level ACROSS field changes value).

- ☐ Different operations from two ON phrases for the same sort break display in the same summary column, and allow a mixture of operations on summary columns.
- ☐ If the same field is referenced in more than one ON phrase for the same sort break, the last summary command specified is applied.
- ☐ You can specify a different summary operation for each sort break.
- ☐ The SUMMARYLINES parameter does not affect processing for ACROSS fields.
- ☐ When used with OVERs, the rows containing fields not to be summarized will be blank.
- ☐ Prefix operators are supported on summary lines:
  - ☐ The following prefix operators are supported for numeric fields: ASQ., AVE., CNT., FST., LST., MAX., MIN., SUM.
  - ☐ Prefix operators PCT., RPCT., AND TOT. are not supported.
  - ☐ Double prefix operators (such as PCT.CNT.) are not supported.
  - ☐ The SUM. prefix operator produces the same summary values as a summary phrase with no prefix operator.
  - ☐ SUMMARIZE and RECOMPUTE apply the calculations defined in the associated COMPUTE command to the summary values. Therefore, in order to perform the necessary calculations, the SUMMARIZE or RECOMPUTE command must specify all of the fields referenced in the COMPUTE command.
  - ☐ If the same field has summary operations with different prefix operators at each level, the appropriate calculation is done at each level for the prefix operator specified.
  - ☐ SUB-TOTAL and SUMMARIZE propagate their operations to all higher-level sort fields. If a request uses SUB-TOTAL or SUMMARIZE at multiple sort levels, more than one prefix operator may apply to the same field. When a SUB-TOTAL or SUMMARIZE command on a lower-level sort field propagates up to the higher levels, it applies its prefix operators only to those fields that did not already have a prefix operator specified at the higher level. For any field that had a prefix operator specified at a higher level, the original prefix operator is applied at the level at which it was first specified.
  - ☐ Prefix operators on summary lines result in the same values whether the command is RECOMPUTE/SUMMARIZE or SUBTOTAL/SUB-TOTAL. For a computed field, the prefix operator is not applied, and the value is recalculated using the expression in the COMPUTE command and the values from the summary line.

- ❑ If an ACROSS field has an ACROSS-TOTAL phrase and a summary command with a prefix operator, the prefix operator is applied, not the ACROSS-TOTAL.

**Example:**    **Using Summary Commands With ACROSS**

The following request sums units and dollars and calculates the unit cost by product and across region and month. The ACROSS MNTH RECOMPUTE command creates totals of units and dollars, and recomputes the calculated value for the selected months within regions. The ACROSS REGION RECOMPUTE command does the same for the selected regions. The ON TABLE SUMMARIZE command creates summary rows. It has no effect on columns:

```
DEFINE FILE GGSALES
MNTH/MTr   = DATE;
END
TABLE FILE GGSALES
SUM
  UNITS/I5 AS 'UNITS'                                OVER
  DOLLARS/I6 AS 'DOLLARS'                            OVER
  COMPUTE DOLLPER/I6 = DOLLARS/UNITS; AS 'UNIT COST'
BY PRODUCT
ACROSS REGION RECOMPUTE AS 'Region Sum' COLUMNS 'Northeast' AND 'West'
ACROSS MNTH   RECOMPUTE AS 'Month Sum'  COLUMNS 'November' AND 'December'
WHERE DATE FROM '19971101' TO '19971231';
WHERE PRODUCT EQ 'Capuccino' OR 'Espresso';
ON TABLE SUMMARIZE AS 'Grand Total'

END
```

The output is:

PAGE 1								
		Region						
		Northeast			West			Region Sum
		MNTH						
		NOVEMBER	DECEMBER	Month Sum	NOVEMBER	DECEMBER	Month Sum	
Product								
Capuccino	UNITS	2282	1188	3470	2535	4051	6586	10056
	DOLLARS	25994	13668	39662	31153	57421	88574	128236
	UNIT COST	11	11	11	12	14	13	12
Espresso	UNITS	1947	2403	4350	3088	3732	6820	11170
	DOLLARS	23629	30605	54234	36123	51400	87523	141757
	UNIT COST	12	12	12	11	13	12	12
Grand Total	UNITS	4229	3591	7820	5623	7783	13406	21226
	DOLLARS	49623	44273	93896	67276	108821	176097	269993
	UNIT COST	11	12	12	11	13	13	12

**Example: Subtotaling One Field Within an ACROSS Group**

The following request against the GGSALES data source sums the DOLLARS and UNITS fields by CATEGORY and across REGION, but subtotals only the UNITS field.

```
TABLE FILE GGSALES
SUM DOLLARS AS 'Dollars' OVER
UNITS AS 'Units'
  BY CATEGORY
  ACROSS REGION SUBTOTAL UNITS
WHERE REGION EQ 'Midwest' OR 'West'
ON TABLE SET PAGE NOPAGE
END
```

The output shows that only the rows with the UNITS values are subtotaled.

		Region		
		Midwest	West	TOTAL
Category				
Coffee	Dollars	4178513	4473517	
	Units	332777	356763	689540
Food	Dollars	4338271	4202337	
	Units	341414	340234	681648
Gifts	Dollars	2883881	2977092	
	Units	230854	235042	465896

**Example:** Summarizing a Calculated Value in an ACROSS Group

The following request against the GGSALES data source sums the DOLLARS and UNITS fields and calculates DOLLARS PER UNIT across REGION. The request also has a higher-level ACROSS field, CATEGORY, so the SUMMARIZE command propagates to both ACROSS fields.

```
SET BYPANEL = ON
TABLE FILE GGSALES
SUM DOLLARS AS 'Dollars' OVER
UNITS AS 'Units' OVER
AND COMPUTE DPERU/D9.2 = DOLLARS/UNITS;
ACROSS CATEGORY
ACROSS REGION
ON REGION SUMMARIZE DPERU
WHERE REGION EQ 'Midwest' OR 'West'
WHERE CATEGORY EQ 'Food' OR 'Gifts'
ON TABLE PCHOLD FORMAT PDF
END
```

The first panel of output shows:

- ☐ The values of DOLLARS, UNITS, and DPERU for the Midwest and West regions under the Food category.
- ☐ The summary column, which has a value just for the DPERU row. Note that for ACROSS, the summary column for REGION appears only after the higher-level ACROSS field, CATEGORY, changes value.
- ☐ The values of DOLLARS, UNITS, and DPERU for the Midwest and West regions under the Gifts category.



PAGE 1.1

	Category			Gifts	
	Food				
	Region				
	Midwest	West	TOTAL	Midwest	West
Dollars	4338271	4202337		2883881	2977092
Units	341414	340234		230854	235042
DPERU	12.71	12.35	12.53	12.49	12.67

The second panel has the total column for the Gifts category and the grand total column. Each of those only has a value in the DPERU row.

PAGE 1.2

	Category	
	Region	TOTAL
	TOTAL	
Dollars		
Units		
DPERU	12.58	12.55

**Example:** Using Prefix Operators in a Summary Command With ACROSS

The following request against the GGSALES data source sums the DOLLARS and UNITS fields ACROSS CATEGORY and ACROSS REGION, with a SUMMARIZE command on the REGION field. The request also has a higher-level ACROSS field, CATEGORY, so the SUMMARIZE command propagates to both ACROSS fields. The SUMMARIZE command specifies the AVE. prefix operator for the DOLLARS field.

```
SET BYPANEL = ON
TABLE FILE GGSALES
SUM DOLLARS AS 'Dollars' OVER
UNITS AS 'Units'
  ACROSS CATEGORY
  ACROSS REGION
  ON REGION SUMMARIZE AVE. DOLLARS
  WHERE REGION EQ 'Midwest' OR 'West'
  WHERE CATEGORY EQ 'Food' OR 'Gifts'
  ON TABLE PCHOLD FORMAT PDF
END
```

The first panel of output shows:

- ❑ The values of DOLLARS and UNITS for the Midwest and West regions under the Food category.

- ❑ The summary column, which has a value just for the DOLLARS row. Note that for ACROSS, the summary column for REGION appears only after the higher-level ACROSS field, CATEGORY, changes value.
- ❑ The values of DOLLARS and UNITS for the Midwest and West regions under the Gifts category.

PAGE 1.1

	Category			Gifts	
	Food				
	Region				
	Midwest	West	TOTAL	Midwest	West
Dollars	4338271	4202337	4270304	2883881	2977092
Units	341414	340234		230854	235042

The second panel has the total column for the Gifts category and the grand total column. Each of those only has a value in the DOLLARS row.

PAGE 1.2

	Category	
		TOTAL
	Region	
	TOTAL	
Dollars	2930486	3600395
Units		

**Example: Using Combinations of ACROSS Summary Commands**

The following request against the GGSALES data source sums the DOLLARS and UNITS fields ACROSS CATEGORY and ACROSS REGION, with a SUMMARIZE command on the REGION field and a SUBTOTAL command on the CATEGORY field. The SUMMARIZE command specifies average DOLLARS and minimum UNITS. The SUBTOTAL command specifies minimum DOLLARS.

```

SET BYPANEL = ON
TABLE FILE GGSales
SUM DOLLARS AS 'Dollars' OVER
UNITS AS 'Units'
  ACROSS CATEGORY
  ACROSS REGION
    ON CATEGORY SUBTOTAL MIN. DOLLARS
    ON REGION SUMMARIZE AVE. DOLLARS MIN. UNITS
  WHERE REGION EQ 'Midwest' OR 'West'
  WHERE CATEGORY EQ 'Food' OR 'Gifts'
  ON TABLE PCHOLD FORMAT PDF
END

```

On the output, all of the TOTAL columns have the minimum UNITS. The TOTAL columns associated with the REGION sort field have the average DOLLARS, but the TOTAL column associated with the CATEGORY sort field has the minimum DOLLARS because SUMMARIZE does not change the prefix operator associated with a higher-level sort field.

PAGE 1.1

	Category			Gifts	
	Food				
	Region				
	Midwest	West	TOTAL	Midwest	West
Dollars	4338271	4202337	4270304	2883881	2977092
Units	341414	340234	340234	230854	235042

PAGE 1.2

	Category	
	TOTAL	
	Region	
	TOTAL	
Dollars	2930486	2883881
Units	230854	230854

### Performing Calculations at Sort Field Breaks

You can use the RECAP and COMPUTE commands to create subtotal values in a calculation. The subtotal values are not displayed. Only the result of the calculation is shown on the report.

**Syntax:**      **How to Use Subtotals in Calculations**

Both the RECAP and COMPUTE commands have similar syntax to other total and subtotal commands.

```
{BY|ON} fieldname1 {RECAP|COMPUTE} fieldname2[/format] = expression;  
[WHEN expression;
```

where:

*fieldname1*

Is the field in the BY phrase. Each time the BY field changes value, a new recap value is calculated.

*fieldname2*

Is the field name that contains the result of the expression.

*/format*

Can be any valid format. The default is D12.2.

*expression*

Can be any valid expression, as described in [Using Expressions](#) on page 431. You must end the expression with a semicolon.

WHEN *expression*

Is for use with RECAP only. It specifies the conditional display of RECAP lines as determined by a Boolean expression (see [Conditionally Displaying Summary Lines and Text](#) on page 429). You must end the expression with a semicolon.

**Reference:**      **Usage Notes for RECAP and COMPUTE**

- ❑ RECAP uses the current value of the named sort field, the current subtotal values of any computational fields that appear as display fields, or the last value for alphanumeric fields.
- ❑ RECAP reserves space at the bottom of the page to ensure that a RECAP will not be alone at the top of the next page while the data it is recapping is on the previous page. The same technique is used for subtotals and grand totals, but not for subfootings or COMPUTEs.
- ❑ The field names in the expression must be fields that appear on the report. That is, they must be display fields or sort control fields.

- ❑ Each RECAP value displays on a separate line. However, if the request contains a RECAP command and SUBFOOT text, the RECAP value displays only in the SUBFOOT text and must be specified in the text using a spot marker. (For details, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.)
- ❑ The calculations in a RECAP or COMPUTE can appear anywhere under the control break, along with any text. (For details, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.)
- ❑ In an ON phrase, a COMPUTE command is the same as a RECAP command.
- ❑ The limit for ON *sortfield* RECAP phrases is 64 for each sort field.
- ❑ You can specify multiple recap calculations in one RECAP phrase. Use the following syntax:

```
ON sortfield RECAP field1/format= ... ;field2/format= ... ;
.
.
.
```

### **Example:** Using RECAP

The following request illustrates the use of RECAP (DEPT\_NET) to determine net earnings for each department:

```
TABLE FILE EMPLOYEE
SUM DED_AMT AND GROSS
BY DEPARTMENT BY PAY_DATE
ON DEPARTMENT RECAP DEPT_NET/D8.2M = GROSS-DED_AMT;
WHEN PAY_DATE GT 820101
END
```

The output is:

DEPARTMENT	PAY_DATE	DED_AMT	GROSS
-----	-----	-----	-----
MIS	81/11/30	\$1,406.79	\$2,147.75
	81/12/31	\$1,406.79	\$2,147.75
	82/01/29	\$1,740.89	\$3,247.75
	82/02/26	\$1,740.89	\$3,247.75
	82/03/31	\$1,740.89	\$3,247.75
	82/04/30	\$3,386.73	\$5,890.84
	82/05/28	\$3,954.35	\$6,649.50
	82/06/30	\$4,117.03	\$7,460.00
	82/07/30	\$4,117.03	\$7,460.00
	82/08/31	\$4,575.72	\$9,000.00
** DEPT_NET		\$22,311.98	
PRODUCTION	81/11/30	\$141.66	\$833.33
	81/12/31	\$141.66	\$833.33
	82/01/29	\$1,560.09	\$3,705.84
	82/02/26	\$2,061.69	\$4,959.84
	82/03/31	\$2,061.69	\$4,959.84
	82/04/30	\$2,061.69	\$4,959.84
	82/05/28	\$3,483.88	\$7,048.84
	82/06/30	\$3,483.88	\$7,048.84
	82/07/30	\$3,483.88	\$7,048.84
	82/08/31	\$4,911.12	\$9,523.84
** DEPT_NET		\$27,531.14	

**Example:**    **Using Multiple RECAP Commands**

You can include multiple RECAP or COMPUTE commands in a request. This option enables you to perform different calculations at different control breaks.

The following request illustrates the use of multiple RECAP commands.

```
TABLE FILE SALES
SUM UNIT_SOLD AND RETURNS
WHERE AREA EQ 'U'
BY DATE BY AREA BY PROD_CODE
ON DATE RECAP
DATE_RATIO=RETURNS/UNIT_SOLD;
ON AREA UNDER-LINE RECAP
AREA_RATIO=RETURNS/UNIT_SOLD;
END
```

The output is:

DATE	AREA	PROD_CODE	UNIT_SOLD	RETURNS
10/17	U	B10	30	2
		B17	20	2
		B20	15	0
		C17	12	0
		D12	20	3
		E1	30	4
		E3	35	4
** AREA_RATIO				.09
** DATE_RATIO				.09
-----				
10/18	U	B10	13	1
** AREA_RATIO				.08
** DATE_RATIO				.08
-----				
10/19	U	B12	29	1
** AREA_RATIO				.03
** DATE_RATIO				.03
-----				

### Suppressing Grand Totals

You can use the NOTOTAL command to suppress grand totals in a report.

Suppressing the grand total is useful when there is only one value at a sort break, since the grand total value is equal to that one value. Using the NOTOTAL command prevents the report from displaying a grand total line for every sort break that has only one detail line. You can also suppress subtotals using the MULTILINES command. For details, see [How to Create Subtotals](#) on page 379.

**Syntax:** **How to Suppress Grand Totals**

To suppress grand totals, add the following syntax to your request:

ON TABLE NOTOTAL

### ***Example:*** Suppressing Grand Totals

The following request includes the NOTOTAL phrase to suppress grand totals for CURR\_SAL, GROSS, and DED\_AMT.

```
TABLE FILE EMPLOYEE
SUM CURR_SAL AND GROSS AND DED_AMT
BY EMP_ID
BY BANK_ACCT
WHERE BANK_ACCT NE 0
ON BANK_ACCT SUB-TOTAL
ON TABLE NOTOTAL
END
```



The output is:

<u>EMP ID</u>	<u>BANK ACCT</u>	<u>CURR SAL</u>	<u>GROSS</u>	<u>DED</u>	<u>AMT</u>
117593129	40950036	\$18,480.00	\$6,099.50	\$2,866.18	
*TOTAL 40950036		\$18,480.00	\$6,099.50	\$2,866.18	
*TOTAL 117593129		\$18,480.00	\$6,099.50	\$2,866.18	
119329144	160633	\$29,700.00	\$2,475.00	\$1,427.24	
*TOTAL 160633		\$29,700.00	\$2,475.00	\$1,427.24	
*TOTAL 119329144		\$29,700.00	\$2,475.00	\$1,427.24	
123764317	819000702	\$26,862.00	\$17,094.00	\$11,949.44	
*TOTAL 819000702		\$26,862.00	\$17,094.00	\$11,949.44	
*TOTAL 123764317		\$26,862.00	\$17,094.00	\$11,949.44	
326179357	122850108	\$21,780.00	\$9,075.00	\$6,307.00	
*TOTAL 122850108		\$21,780.00	\$9,075.00	\$6,307.00	
*TOTAL 326179357		\$21,780.00	\$9,075.00	\$6,307.00	
451123478	136500120	\$16,100.00	\$9,130.00	\$3,593.92	
*TOTAL 136500120		\$16,100.00	\$9,130.00	\$3,593.92	
*TOTAL 451123478		\$16,100.00	\$9,130.00	\$3,593.92	
818692173	163800144	\$27,062.00	\$22,013.75	\$15,377.40	
*TOTAL 163800144		\$27,062.00	\$22,013.75	\$15,377.40	
*TOTAL 818692173		\$27,062.00	\$22,013.75	\$15,377.40	

## Conditionally Displaying Summary Lines and Text

In addition to using summary lines to control the look and content of your report, you can specify WHEN criteria to control the conditions under which summary lines appear for each vertical (BY) sort field value. WHEN is supported with SUBFOOT, SUBHEAD, SUBTOTAL, SUBTOTAL, SUMMARIZE, RECOMPUTE, and RECAP.

**Example: Conditionally Displaying Summary Lines and Text**

In a sales report that covers four regions (Midwest, Northeast, Southeast, and West), you may only want to display a subtotal when total dollar sales are greater than \$11,500,000. The following request accomplishes this by including criteria that trigger the display of a subtotal when dollar sales exceed \$11,500,000 and subfooting text when dollar sales are less than \$11,500,000.

```
TABLE FILE GGSales
SUM UNITS DOLLARS
BY REGION
BY CATEGORY
ON REGION SUBTOTAL
WHEN DOLLARS GT 11500000
SUBFOOT
"The total for the <REGION region is less than 11500000."
WHEN DOLLARS LT 11500000
END
```

The output is:

Region	Category	Unit Sales	Dollar Sales
-----			
Midwest	Coffee	332777	4178513
	Food	341414	4338271
	Gifts	230854	2883881
The total for the Midwest region is less than 11500000.			
Northeast	Coffee	335778	4164017
	Food	353368	4379994
	Gifts	227529	2848289
The total for the Northeast region is less than 11500000.			
Southeast	Coffee	350948	4415408
	Food	349829	4308731
	Gifts	234455	2986240
*TOTAL Southeast		935232	11710379
West	Coffee	356763	4473517
	Food	340234	4202337
	Gifts	235042	2977092
*TOTAL West		932039	11652946
TOTAL		3688991	46156290

## Using Expressions

---

An expression combines field names, constants, and operators in a calculation that returns a single value. You can use an expression in a variety of commands to assign a value to a temporary field or Dialogue Manager amper variable, or use it in screening. You can combine simpler ones to build increasingly complex expressions.

When you write an expression, you can specify the operation yourself, or you can use one of the many supplied functions that perform specific calculations or data manipulation. These functions operate on one or more arguments, and return a single value as a result. To use a function, you simply call it. For details about functions, see the *Using Functions* manual.

**In this chapter:**

- ☐ [Using Expressions in Commands and Phrases](#)
  - ☐ [Types of Expressions](#)
  - ☐ [Creating a Numeric Expression](#)
  - ☐ [Creating a Date Expression](#)
  - ☐ [Creating a Date-Time Expression](#)
  - ☐ [Creating a Character Expression](#)
  - ☐ [Creating a Variable Length Character Expression](#)
  - ☐ [Creating a Logical Expression](#)
  - ☐ [Creating a Conditional Expression](#)
- 

### Using Expressions in Commands and Phrases

You can use an expression in various commands and phrases. An expression may not exceed 40 lines and must end with a semicolon, except in WHERE and WHEN phrases, in which the semicolon is optional.

The commands that support expressions, and their basic syntax, are summarized here. For complete syntax with an explanation, see the applicable documentation.

You can use an expression when you:

- ❑ Create a temporary field, and assign a value to that field. The field can be created in a Master File using the DEFINE attribute, or using a DEFINE or COMPUTE command:

- ❑ DEFINE command preceding a report request:

```
DEFINE FILE filename
  fieldname [/format] = expression;
  .
  .
  .
END
```

- ❑ DEFINE attribute in a Master File:

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

- ❑ COMPUTE command in a report request:

```
COMPUTE fieldname [/format] = expression;
```

- ❑ Define record selection criteria and criteria that control report formatting.

```
{WHERE|IF} logical_expression[:]  
  WHEN logical_expression[:]
```

- ❑ Determine branching in Dialogue Manager, or assign a value to a Dialogue Manager ampers variable.

```
-IF logical_expression [THEN] GOTO label1 [ELSE GOTO label2];  
  
-SET &name = expression;
```

- ❑ Perform a calculation with the RECAP command in the Financial Modeling Language (FML).

```
RECAP name [(n)] [/format] = expression;
```

## Types of Expressions

An expression can be one of the following:

- ❑ **Numeric.** Use numeric expressions to perform calculations that use numeric constants (integer or decimal) and fields. For example, you can write an expression to compute the bonus for each employee by multiplying the current salary by the desired percentage as follows:

```
COMPUTE BONUS/D12.2 = CURR_SAL * 0.05 ;
```

A numeric expression returns a numeric value. For details, see [Creating a Numeric Expression](#) on page 434.

- ❑ **Date.** Use date expressions to perform numeric calculations on dates. For example, you can write an expression to determine when a customer can expect to receive an order by adding the number of days in transit to the date on which you shipped the order as follows:

```
COMPUTE DELIVERY/MDY = SHIPDATE + 5 ;
```

There are two types of date expressions:

- ❑ Date expressions, which return a date, a component of a date, or an integer that represents the number of days, months, quarters, or years between two dates. For details, see [Creating a Date Expression](#) on page 440.
- ❑ Date-time expressions, which you can create using a variety of specialized date-time functions, each of which returns a different kind of value. For details on these functions, see the *Using Functions* manual.
- ❑ **Character.** Use character expressions to manipulate alphanumeric constants or fields. For example, you can write an expression to extract the first initial from an alphanumeric field as follows:

```
COMPUTE FIRST_INIT/A1 = EDIT (FIRST_NAME, '9$$$$$$$') ;
```

A character expression returns an alphanumeric value. For details, see [Creating a Character Expression](#) on page 458.

**Note:** Text fields can be assigned to alphanumeric fields and receive assignment from alphanumeric fields. Text fields can also participate in expressions using the operators CONTAINS and OMITS.

- ❑ **Logical.** Use logical expressions to evaluate the relationship between two values. A logical expression returns TRUE or FALSE. For details, see [Creating a Logical Expression](#) on page 465.
- ❑ **Conditional.** Use conditional expressions to assign values based on the result of logical expressions. A conditional expression (IF ... THEN ... ELSE) returns a numeric or alphanumeric value. For details, see [Creating a Conditional Expression](#) on page 467.

## Expressions and Field Formats

When you use an expression to assign a value to a field, make sure that you give the field a format that is consistent with the value returned by the expression. For example, if you use a character expression to concatenate a first name and last name and assign it to the field FULL\_NAME, make sure you define the field as character.

### *Example:* Assigning a Field Format of Sufficient Length

The following example contains a character expression that concatenates a first name and last name to derive the full name. It assigns the field FULL\_NAME an alphanumeric format of sufficient length to accommodate the concatenated name:

```
DEFINE FILE EMPLOYEE
FULL_NAME/A25 = FIRST_NAME | LAST_NAME;
END
TABLE FILE EMPLOYEE
PRINT FULL_NAME
WHERE LAST_NAME IS 'BLACKWOOD'
END
```

The output is:

```
FULL_NAME
-----
ROSEMARIE BLACKWOOD
```

## Creating a Numeric Expression

A numeric expression performs a calculation that uses numeric constants, fields, operators, and functions to return a numeric value. When you use a numeric expression to assign a value to a field, that field must have a numeric format. The default format is D12.2.

A numeric expression can consist of the following components, shown below in bold:

- ☐ A numeric constant. For example:

```
COMPUTE COUNT/I2 = 1 ;
```

- ☐ A numeric constant in scientific notation. For example:

```
COMPUTE COST/D12.2 = EXPN(8E+3) ;
```

For syntax usage, see [How to Express a Number in Scientific Notation](#) on page 435.

- ☐ A numeric field. For example:

```
COMPUTE RECOUNT/I2 = COUNT ;
```

- ☐ Two numeric constants or fields joined by an arithmetic operator. For example:

```
COMPUTE BONUS/D12.2 = CURR_SAL * 0.05 ;
```

For a list of arithmetic operators, see [Arithmetic Operators](#) on page 437.

- ❑ A numeric function. For example:

```
COMPUTE LONGEST_SIDE/D12.2 = MAX (WIDTH, HEIGHT) ;
```

- ❑ Two or more numeric expressions joined by an arithmetic operator. For example:

```
COMPUTE PROFIT/D12.2 = (RETAIL_PRICE - UNIT_COST) * UNIT_SOLD ;
```

Note the use of parentheses to change the order of evaluation of the expression. For information on the order in which numeric operations are performed, see [Order of Evaluation](#) on page 437.

Before they are used in calculations, numeric values are generally converted to double-precision floating-point format. The result is then converted to the specified field format. In some cases the conversion may result in a difference in rounding. Note that environments that support native-mode arithmetic handle rounding differently. For details, see [Evaluating Numeric Expressions With Native-Mode Arithmetic](#) on page 438.

If a number is too large (greater than  $10^{75}$ ) or too small (less than  $10^{-75}$ ), you receive an Overflow or Underflow warning, and asterisks display for the field value.

**Note:** You can change the overflow character by issuing the SET OVERFLOWCHAR command.

For detailed information on rounding behavior for numeric data formats, see the *Describing Data With WebFOCUS Language* manual.

## **Syntax:** How to Express a Number in Scientific Notation

In an IF clause, use the following:

```
IF field op n[.nn]{E|D|e|d}{[+|-]p}
```

In a WHERE clause, use the following:

```
WHERE field op EXPN(n[.nn]{E|D|e|d}{[+|-]p});
```

In a COMPUTE command, use the following:

```
COMPUTE field[/format] = EXPN(n[.nn]{E|D|e|d}{[+|-]p});
```

In a DEFINE command, use the following:

```
DEFINE FILE filename  
field[/format] = EXPN(n[.nn]{E|D|e|d}[±|-]p);  
END
```

In a DEFINE in the Master File, use the following:

```
DEFINE field[/format] = EXPN(n[.nn]{E|D|e|d}[±|-]p);
```

where:

*field*

Is a field in a request.

*/format*

Is the optional format of the field. For information on formats, see the *Describing Data With WebFOCUS Language* manual.

*op*

Is a relational operator in a request.

*n.nn*

Is a numeric constant that consists of a whole number component, followed by a decimal point, followed by a fractional component.

*E, D, e, d*

Denotes scientific notation. E, e, d, and D are interchangeable.

*±, -*

Indicates if *p* is positive or negative. Positive is the default.

*p*

Is the power of 10 to which to raise the number. The range of values for *p* is between -78 and +78 on z/OS, -99 to +99 elsewhere.

**Note:** EXPN is useful for calculations on fields with F and D formats. It is generally not useful for calculations on fields with P or I formats.

### **Example:** Evaluating a Number in Scientific Notation

You can use scientific notation in an IF or WHERE clause to express 8000 as 8E+03:

```
IF RCOST LT 8E+03  
  
WHERE RCOST LT EXPN(8E+03)
```



**Reference: Arithmetic Operators**

The following list shows the arithmetic operators you can use in an expression:

Addition	+
Subtraction	-
Multiplication	*
Division	/
Exponentiation	**

**Note:** If you attempt to divide by 0, the value of the expression is 0. Multiplication and exponentiation are not supported for date expressions of any type. To isolate part of a date, use a simple assignment command.

**Order of Evaluation**

Numeric expressions are evaluated in the following order:

1. Exponentiation.
2. Division and multiplication.
3. Addition and subtraction.

When operators are at the same level, they are evaluated from left to right. Because expressions in parentheses are evaluated before any other expression, you can use parentheses to change this predefined order. For example, the following expressions yield different results because of parentheses:

```
COMPUTE PROFIT/D12.2 = RETAIL_PRICE - UNIT_COST * UNIT_SOLD ;
COMPUTE PROFIT/D12.2 = (RETAIL_PRICE - UNIT_COST) * UNIT_SOLD ;
```

In the first expression, UNIT\_SOLD is first multiplied by UNIT\_COST, and the result is subtracted from RETAIL\_PRICE. In the second expression, UNIT\_COST is first subtracted from RETAIL\_PRICE, and that result is multiplied by UNIT\_SOLD.

**Note:** Two operators cannot appear consecutively. The following expression is invalid:

```
a * -1
```

To make it valid, you must add parentheses:

```
a * (-1)
```

**Example: Controlling the Order of Evaluation**

The order of evaluation can affect the result of an expression. Suppose you want to determine the dollar loss in retail sales attributed to the return of damaged items. You could issue the following request:

```
TABLE FILE SALES
PRINT RETAIL_PRICE RETURNS DAMAGED
COMPUTE RETAIL_LOSS/D12.2 = RETAIL_PRICE * RETURNS + DAMAGED;
BY PROD_CODE
WHERE PROD_CODE IS 'E1';
END
```

The calculation

```
COMPUTE RETAIL_LOSS/D12.2 = RETAIL_PRICE * RETURNS + DAMAGED;
```

gives an incorrect result because RETAIL\_PRICE is first multiplied by RETURNS, and then the result is added to DAMAGED. The correct result is achieved by adding RETURNS to DAMAGED, then multiplying the result by RETAIL\_PRICE.

You can change the order of evaluation by enclosing expressions in parentheses. An expression in parentheses is evaluated before any other expression. You may also use parentheses to improve readability.

Using parentheses, the correct syntax for the preceding calculation is:

```
COMPUTE RETAIL_LOSS/D12.2 = RETAIL_PRICE * (RETURNS + DAMAGED);
```

The output is:

PROD_CODE	RETAIL_PRICE	RETURNS	DAMAGED	RETAIL_LOSS
-----	-----	-----	-----	-----
E1	\$ .89	4	7	9.79

**Evaluating Numeric Expressions With Native-Mode Arithmetic**

When native-mode arithmetic is used, a specific evaluation path is followed for each numeric expression based on the format of the operands and the operators. If the operands all have the same format, most operations are carried out in that format. If the operands have different formats, the operands are converted to a common format in a specific order of format precedence. Regardless of operand formats, some operators require conversion to specific formats so that all operands are in the appropriate format.

## Using Identical Operand Formats With Native-Mode Arithmetic

If all operands of a numeric operator are of the same format, you can use the following table to determine whether or not the operations are performed in that native format or if the operands are converted before and after executing the operation. In each case requiring conversion, operands are converted to the operational format and the intermediate result is returned in the operational format. If the format of the result differs from the format of the target variable, the result is converted to the format of the target variable.

Operation		Operational Format
Addition	+	Native
Subtraction	-	Native
Multiplication	*	Native
Full Division	/	Accepts single or double-precision floating point, converts all others to double-precision floating point
Exponentiation	**	Double-precision floating point

### *Example:* Using Identical Operand Formats (Native-mode Arithmetic)

The following variables are defined as integers in Maintain Data:

```
COMPUTE OPERANDONE/I4 ;
      OPERANDTWO/I4 ;
      RESULT/I4 ;
```

The required multiplication is done in native-mode arithmetic (integer arithmetic):

```
COMPUTE RESULT/I4 = OPERANDONE * OPERANDTWO ;
```

## Using Different Operand Formats With Native-Mode Arithmetic

If operands of a numeric operator have different formats, you can use the following table to determine what the common format is after they are converted. The lower operand is converted to the format of the higher operand before performing the operation.

Order	Format
1	16-byte packed decimal

Order	Format
2	Double-precision floating point
3	8-byte packed-decimal
4	Single-precision floating point
5	Integer
6	Character (alphanumeric and text)

For example, if a 16-byte packed-decimal operand is used in an expression, all other operands are converted to 16-byte packed-decimal format for evaluation. On the other hand, if an expression includes only integer and alphanumeric operands, all alphanumeric operands are converted to integer format.

A character (that is, alphanumeric or text) value can be used in a computation if it is a numeric string. An attempt is made to convert the character operand to the format of the other operand in the expression. If both operands are character, an attempt is made to convert them to double-precision. If the conversion is not possible, an error message is generated.

If you assign a decimal value to an integer, the fractional value is truncated.

### Creating a Date Expression

A date expression performs a numeric calculation that involves dates.

A date expression returns a date, a date component, or an integer that represents the number of days, months, quarters, or years between two dates. You can write a date expression directly that consists of:

- ❑ A date constant. For example:

```
COMPUTE END_DATE/MDYY = 'FEB 29 2000';
```

This requires single quotation marks around the date constant.

- ❑ A date field. For example:

```
COMPUTE NEWDATE/YMD = START_DATE;
```

- ❑ An alphanumeric, integer, or packed decimal format field, with date edit options. For example, in the second COMPUTE command, OLDDATE is a date expression:

```
COMPUTE OLDDATE/I6YMD = 980307;  
COMPUTE NEWDATE/YMD DFC 19 YRT 10 = OLDDATE;
```

- ❑ A calculation that uses an arithmetic operator or date function to return a date. Use a numeric operator only with date formats (formerly called Smart dates). The following example first converts the integer date HIRE\_DATE (format I6YMD) to the date format CONVERTED\_HDT (format YMD). It then adds 30 days to CONVERTED\_HDT:

```
COMPUTE CONVERTED_HDT/YMD = HIRE_DATE;
HIRE_DATE_PLUS_THIRTY/YMD = CONVERTED_HDT + 30;
```

- ❑ A calculation that uses a numeric operator or date function to return an integer that represents the number of days, months, quarters, or years between two dates. The following example uses the date function YMD to calculate the difference (number of days) between an employee hire date and the date of his first salary increase:

```
COMPUTE DIFF/I4 = YMD (HIRE_DATE,FST.DAT_INC);
```

## Formats for Date Values

You can work with dates in one of two ways:

- ❑ **In date format.** The value is treated as an integer that represents the number of days between the date value and a base date. There are two base dates for date formats:
  - ❑ YMD and YYMD formats have a base date of December 31, 1900.
  - ❑ YM and YYM formats have a base date of January, 1901 on z/OS and a base date of December 31, 1900 on Windows and UNIX.

When displayed, the integer value is converted to the corresponding date in the format specified for the field. The format can be specified in either the Master File or in the command that uses an expression to assign a value to the field. These were previously referred to as smart date formatted fields.

- ❑ **In integer, packed decimal, or alphanumeric format with date edit options.** The value is treated as an integer, a packed decimal, or an alphanumeric string. When displayed, the value is formatted as a date. These were previously referred to as old date formatted fields.

You can convert a date in one format to a date in another format simply by assigning one to the other. For example, the following assignments take a date stored as an alphanumeric field, formatted with date edit options, and convert it to a date stored as a temporary date field:

```
COMPUTE ALPHADATE/A6MDY = '120599' ;
REALDATE/MDY = ALPHADATE;
```

**Reference: Base Dates for Date Formats**

The following table shows the base date for each supported date format:

Format	Base Date
YMD, YYMD, MDYY, DMY, MDY, and DMY	1900/12/31
YM, YYM, MY, and MY	1901/01 on z/OS 1900/12/31 on Windows and UNIX
YQ, YYQ, QYY, and QY	1901 Q1
JUL and YYJUL	1900/365
D M Y, YY Q W	There is no base date for these formats; these are just numbers, not dates.

Note that the base date used for the functions DA and DT is December 31, 1899. For details on date functions, see the *Using Functions* manual.

**Reference: Impact of Date Formats on Storage and Display**

The following table illustrates how the field format affects storage and display:

Date Format (For example: MDYY)		Integer, Packed, Decimal, or Alphanumeric Format (For example: A8MDYY)		
February 28, 1999	35853	02/28/1999	02281999	02/28/1999
March 1, 1999	35854	03/01/1999	03011999	03/01/1999

## Performing Calculations on Dates

The format of a field determines how you can use it in a date expression. Calculations on dates in date format can incorporate numeric operators as well as numeric functions. Calculations on dates in integer, packed, decimal, or alphanumeric format require the use of date functions. Numeric operators return an error message or an incorrect result.

A full set of functions is supplied with your software, enabling you to manipulate dates in integer, packed decimal, and alphanumeric format. For details on date functions, see the *Using Functions* manual.

### **Example:** Calculating Dates

Assume that your company maintains a SHIPPING database. The following example calculates how many days it takes the shipping department to fill an order by subtracting the date on which an item is ordered, the ORDER\_DATE, from the date on which it is shipped, the SHIPDATE:

```
COMPUTE TURNAROUND/I4 = SHIP_DATE - ORDER_DATE;
```

An item ordered on February 28, 1999, and shipped on March 1, 1999, results in a difference of one day. However, if the SHIP\_DATE and ORDER\_DATE fields have an integer format, the result of the calculation (730000) is incorrect, since you cannot use the numeric operator minus (-) with that format.

The following table shows how the field format affects the result:

	Value in Date Format	Value in Integer Format
SHIP_DATE = March 1, 1999	35854	03011999
ORDER_DATE = February 28, 1999	35853	02281999
TURNAROUND	1	730000

To obtain the correct result using fields in integer, packed, decimal, or alphanumeric format, use the date function MDY, which returns the difference between two dates in the form month-day-year. Using the function MDY, you can calculate TURNAROUND as follows:

```
COMPUTE TURNAROUND/I4 = MDY(ORDER_DATE, SHIP_DATE);
```

## Cross-Century Dates With DEFINE and COMPUTE

You can use an expression in a DEFINE or COMPUTE command, or in a DEFINE attribute in a Master File, that implements the sliding window technique for cross-century date processing. The parameters DEFCENT and YRTHRESH provide a means of interpreting the century if the first two digits of the year are not provided elsewhere. If the first two digits are provided, they are simply accepted.

## Returned Field Format Selection

A date expression always returns a number. That number may represent a date, or the number of days, months, quarters, or years between two dates. When you use a date expression to assign a value to a field, the format selected for the field determines how the result is returned.

### *Example:*    **Selecting the Format of a Returned Field**

Consider the following commands, assuming that SHIP\_DATE and ORDER\_DATE are date-formatted fields. The first command calculates how many days it takes a shipping department to fill an order by subtracting the date on which an item is ordered, ORDER\_DATE, from the date on which it is shipped, SHIP\_DATE. The second command calculates a delivery date by adding five days to the date on which the order is shipped.

```
COMPUTE TURNAROUND/I4 = SHIP_DATE - ORDER_DATE;  
COMPUTE DELIVERY/MDY = SHIP_DATE + 5;
```

In the first command, the date expression returns the number of days it takes to fill an order; therefore, the associated field, TURNAROUND, must have an integer format. In the second command, the date expression returns the date on which the item will be delivered; therefore, the associated field, DELIVERY, must have a date format.

## Using a Date Constant in an Expression

When you use a date constant in a calculation with a field in date format, you must enclose it in single quotation marks; otherwise, it is interpreted as the number of days between the constant and the base date (December 31, 1900, or January 1, 1901). For example, if 022899 were not enclosed in quotation marks, the value would be interpreted as the 22,899th day after 12/31/1900, rather than as February 28, 1999.

### *Example:*    **Initializing a Field With a Date Constant**

The following command initializes START\_DATE with the date constant 02/28/99:

```
COMPUTE START_DATE/MDY = '022899';
```



The following command calculates the number of days elapsed since January 1, 1999:

```
COMPUTE YEAR_TO_DATE/I4 = CURR_DATE - 'JAN 1 1999' ;
```

## Extracting a Date Component

Date components include days, months, quarters, or years. You can write an expression that extracts a component from a field in date format. However, you cannot write an expression that extracts days, months, or quarters from a date that does not have these components. For example, you cannot extract a month from a date in YY format, which represents only the number of years.

### *Example:* Extracting the Month Component From a Date

The following example extracts the month component from SHIP\_DATE, which has the format MDYY:

```
COMPUTE SHIP_MONTH/M = SHIP_DATE;
```

If SHIP\_DATE has the value March 1, 1999, the above expression returns the value 03 for SHIP\_MONTH.

A calculation on a date component automatically produces a valid value for the desired component. For example, if the current value of SHIP\_MONTH is 03, the following expression correctly returns the value 06:

```
COMPUTE ADD_THREE/M = SHIPMONTH + 3;
```

If the addition of months results in an answer greater than 12, the months are adjusted correctly (for example, 11 + 3 is 2, not 14).

## Combining Fields With Different Formats in an Expression

When using fields in date format, you can combine fields with a different order of components within the same expression. In addition, you can assign the result of a date expression to a field with a different order of components from the fields in the expression.

You cannot, however, write an expression that combines dates in date format with dates in integer, packed, decimal or character format.

### **Example:** Combining Fields With Format YYMD and MDY

Consider the two fields DATE\_PAID and DUE\_DATE. DATE\_PAID has the format YYMD, and DUE\_DATE has the format MDY. You can combine these two fields in an expression to calculate the number of days that a payment is late:

```
COMPUTE DAYS_LATE/I4 = DATE_PAID - DUE_DATE;
```

### **Example:** Assigning a Different Order of Components to a Returned Field

Consider the field DATE\_SOLD. This field contains the date on which an item is sold, in YYMD format. The following expression adds seven days to DATE\_SOLD to determine the last date on which the item can be returned. It then assigns the result to a field with DMY format:

```
COMPUTE RETURN_BY/DMY = DATE_SOLD + 7;
```

## Creating a Date-Time Expression

A *date-time* expression returns date and time components. You can create these expressions using a variety of supplied date-time functions. For details about date-time functions, see the *Using Functions* manual.

### **Reference:** Automatic Conversion Between Date and Date-Time Formats

In early releases of date-time fields, you were required to use date-time functions for all conversions between date and date-time formats. While these functions are still supported for conversions, the requirement to use them has been eliminated in certain operations.

The following automatic direct operations are supported between date and date-time formats:

#### **Assignment.**

**Assignment of a date field or a date constant to a date-time field.** The time component is set to zero (midnight). The date can be a full component date such as YYMD or a partial component date such as YYM. It cannot be a single component date such as Q, as this type of date, although displayed as a date in reports, is stored as an integer value and is used as an integer value in expressions.

**Assignment of a date-time field or date-time constant to a date field.** The time components are removed.

#### **Comparison and Subtraction.**

When a date-time value is compared with or subtracted from a date value, or a date value is compared with or subtracted from a date-time value, the date is converted to date-time with the time component set to midnight. They are then compared or subtracted as date-time values.

#### ❑ **Function parameters.**

Simplified date functions can use either date or date-time values as their date parameters. Legacy user functions do not support this new functionality. The date-time functions (H functions) use date-time parameters and the new date functions use new dates, which are stored as offsets from a base date.

#### **Recognition and use of date or date-time constants.**

- ❑ Constants can be expressed as strings, without the DT operator.
- ❑ Constants are converted to or from date or date-time values in accordance with the field format they are compared with, subtracted from, or assigned to.
- ❑ Unless it is expressed in a non-ambiguous translated or formatted string format with proper delimiters (not as a numeric string or number), the recognition of a constant as a date depends on the format of its field counterpart.

In this case, the size in terms of number of digits is strictly limited to at least six for a full component date or date-time value, (eight for a four-digit year), three for a partial component date, and one for a single component date.

- ❑ When numeric constants are used as function parameters and, therefore, do not have a field counterpart, they are recognized according to YYMD or YMD format. The only exception is a string with a single blank or the number zero which, in reports, will be presented as a blank. Date offset constants are no longer allowed. Blank separators between digits in a string are also not supported.

For additional information about date and date-time formats, see the *Describing Data With WebFOCUS Language* manual.

**Example:**     **Assigning Date and Date-Time Values**

The following request generates a date-time value using the DT\_CURRENT\_DATETIME function. It then assigns this value to a date field and assigns that date field to a date-time field.

```
TABLE FILE WF_RETAIL_LITE
PRINT QUANTITY_SOLD NOPRINT AND COMPUTE
    DATETIME1/HYYMDm = DT_CURRENT_DATETIME(MILLISECOND);
    AS 'Date-Time 1'
COMPUTE
    DATE1/YYMD      = DATETIME1;
    AS 'Date'
COMPUTE
    DATETIME2/HYYMDm = DATE1;
    AS 'Date-Time 2'
WHERE RECORDLIMIT EQ 20
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

The output is shown in the following image. The original date-time field has a non-zero time component. When assigned to the date field, the time component is removed. When that date is assigned to the second date-time field, a zero time component is added.

<u>Date-Time 1</u>	<u>Date</u>	<u>Date-Time 2</u>
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.855000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.856000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.856000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.856000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.856000	2017/08/25	2017/08/25 00:00:00.000000
2017/08/25 09:10:06.856000	2017/08/25	2017/08/25 00:00:00.000000

**Example: Comparing Date and Date-Time Values**

The following request creates one date-time field and one date field. When QUANTITY\_SOLD is 1, they have the same date value and the date-time field has a zero time component. When QUANTITY\_SOLD is 2, they have different date values, and the date-time field has a zero time component. In all other cases, the date-time field has the current date with a non-zero time component, and the date field has the current date. The EQUAL1 field compares them to see if they compare as equal.

```
TABLE FILE WF_RETAIL_LITE
PRINT QUANTITY_SOLD AS Quantity AND COMPUTE
    DATETIME1/HYYMDm = IF QUANTITY_SOLD EQ 1 THEN '2017/06/05'
                        ELSE IF QUANTITY_SOLD EQ 2 THEN '2016/02/29'
                        ELSE DT_CURRENT_DATETIME(MILLISECOND);
                        AS 'Date-Time'
COMPUTE
    DATE1/YYMD      = IF QUANTITY_SOLD EQ 1 THEN '2017/06/05'
                        ELSE IF QUANTITY_SOLD EQ 2 THEN '2015/12/30'
                        ELSE DT_CURRENT_DATE();
                        AS 'Date'
COMPUTE
    EQUAL1/A1 = IF DATETIME1 EQ DATE1 THEN 'Y' ELSE 'N';
              AS 'Equal?'
WHERE RECORDLIMIT EQ 12
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

The output is shown in the following image. When a date value is compared to a date-time value, the date is converted to a date-time value with the time component set to zero, and then the values are compared. Therefore, when QUANTITY\_SOLD is 1, both the date components are equal and the time component of the date-time field is set to zero, so when the date is converted to a date-time value, they are equal. When QUANTITY\_SOLD is 2, the date components are different, so they are not equal. When QUANTITY\_SOLD is 3, the date components are the same, but the date-time field has a non-zero time component. Therefore, when the date field is converted to a date-time value with a zero time component and they are compared, they are not equal.

<u>Quantity</u>	<u>Date-Time</u>	<u>Date</u>	<u>Equal?</u>
1	2017/06/05 00:00:00.000000	2017/06/05	Y
1	2017/06/05 00:00:00.000000	2017/06/05	Y
1	2017/06/05 00:00:00.000000	2017/06/05	Y
2	2016/02/29 00:00:00.000000	2015/12/30	N
1	2017/06/05 00:00:00.000000	2017/06/05	Y
1	2017/06/05 00:00:00.000000	2017/06/05	Y
2	2016/02/29 00:00:00.000000	2015/12/30	N
1	2017/06/05 00:00:00.000000	2017/06/05	Y
1	2017/06/05 00:00:00.000000	2017/06/05	Y
1	2017/06/05 00:00:00.000000	2017/06/05	Y
3	2017/08/25 09:24:45.203000	2017/08/25	N
1	2017/06/05 00:00:00.000000	2017/06/05	Y

**Syntax:**      **How to Specify the Order of Date Components in a Date-Time Field**

`SET DATEFORMAT = option`

where:

*option*

Can be one of the following: MDY, DMY, YMD, or MYD. MDY is the default value for the U.S. English format.

## Specifying a Date-Time Value

An external date-time value is a constant in character format from one of the following sources:

- ☐ A sequential data source.
- ☐ Used in an expression in a WHERE, IF, DEFINE, or a COMPUTE.

A date-time constant or a date-time value as it appears in a character file has one of the following formats:

```
time_string [date_string]  
date_string [time_string]
```

A date-time constant in a COMPUTE, DEFINE, or WHERE expression must have one of the following formats:

```
DT(time_string [date_string])  
DT(date_string [time_string])
```

A date-time constant in an IF expression has one of the following formats:

```
'time_string [date_string]'  
'date_string [time_string]'
```

If the value contains no blanks or special characters, the single quotation marks are not necessary. Note that the DT prefix is not supported in IF criteria.

where:

*time\_string*

Cannot contain blanks. Time components are separated by colons, and may be followed by AM, PM, am, or pm. For example:

```
14:30:20:99      (99 milliseconds)  
14:30  
14:30:20.99      (99/100 seconds)  
14:30:20.999999  (999999 microseconds)  
02:30:20:500pm
```

Note that the second can be expressed with a decimal point or be followed by a colon:

- ☐ If there is a colon after the second, the value following it represents the millisecond. There is no way to express the microsecond or nanosecond using this notation.
- ☐ A decimal point in the second value indicates the decimal fraction of a second. A microsecond can be represented using six decimal digits. A nanosecond can be represented using nine decimal digits.



*date\_string*

Can have one of the following three formats:

- ❑ **Numeric string format.** Is exactly four, six, or eight digits. Four-digit strings are considered to be a year (century must be specified). The month and day are set to January 1. Six and eight-digit strings contain two or four digits for the year, followed by two for the month, and then two for the day. Because the component order is fixed with this format, the DATEFORMAT setting described in [How to Specify the Order of Date Components in a Date-Time Field](#) on page 451 is ignored.

If a numeric-string format longer than eight digits is encountered, it is treated as a combined date-time string in the Hn format. The following are examples of numeric string date constants:

```
99
1999
19990201
```

- ❑ **Formatted-string format.** Contains a one or two-digit day, a one or two-digit month, and a two or four-digit year separated by spaces, slashes, hyphens, or periods. All three parts must be present and follow the DATEFORMAT setting described in [How to Specify the Order of Date Components in a Date-Time Field](#) on page 451. If any of the three fields is four digits, it is interpreted as the year, and the other two fields must follow the order given by the DATEFORMAT setting. The following are examples of formatted-string date constants:

```
1999/05/20
5 20 1999
99.05.20
1999-05-20
```

- ❑ **Translated-string format.** Contains the full or abbreviated month name. The year must also be present in four-digit or two-digit form. If the day is missing, day 1 of the month is assumed; if present, it can have one or two digits. If the string contains both a two-digit year and a two-digit day, they must be in the order given by the DATEFORMAT setting. For example:

```
January 6 2000
```

**Note:**

- ❑ The date and time strings must be separated by at least one blank space. Blank spaces are also permitted at the beginning and end of the date-time string or immediately before an am/pm indicator.

- ❑ In each date format, two-digit years are interpreted using the [F]DEFCENT and [F]YRTHRESH settings.

**Example: Assigning Date-Time Literals**

The DT prefix can be used, although it is no longer required, in a COMPUTE, DEFINE, or WHERE expression to assign a date-time literal to a date-time field. For example:

```
DT2/HYYMDS = DT(20051226 05:45);

DT3/HYYMDS = DT(2005 DEC 26 05:45);

DT4/HYYMDS = DT(December 26 2005 05:45);
```

**Example: Specifying the Order of Date Components for a Date-Time Field**

The following request sets DATEFORMAT to MYD:

```
SET DATEFORMAT = MYD
DEFINE FILE EMPLOYEE
DTFLDYMD/HYYMDI = DT(APR 04 05);
END

TABLE FILE EMPLOYEE
PRINT CURR_SAL DTFLDYMD
END
```

The output shows that the natural date literal 'APR 04 05' is interpreted as April 5, 1904:

CURR_SAL	DTFLDYMD
-----	-----
\$11,000.00	1904/04/05 00:00
\$13,200.00	1904/04/05 00:00
\$18,480.00	1904/04/05 00:00
\$9,500.00	1904/04/05 00:00
\$29,700.00	1904/04/05 00:00
\$26,862.00	1904/04/05 00:00
\$21,120.00	1904/04/05 00:00
\$18,480.00	1904/04/05 00:00
\$21,780.00	1904/04/05 00:00
\$16,100.00	1904/04/05 00:00
\$9,000.00	1904/04/05 00:00
\$27,062.00	1904/04/05 00:00

**Example: Reading Date-Time Values From a Transaction File**

The DTTRANS comma-delimited transaction file has an ID field and a date-time field that contains both the date (as eight characters) and time (in the format hour:minute:second):

```
01, 20000101 02:57:25,$
02, 19991231 14:05:35,$
```

Because the transaction file contains the dates in numeric string format, the DATEFORMAT setting is not used, and the dates are entered in YMD order.

The following transaction file is also valid. It contains formatted string dates that comply with the default DATEFORMAT setting, MDY:

```
01, 01/01/2000 02:57:25,$
02, 12/31/1999 14:05:35,$
```

The following Master File describes the FOCUS data source named DATETIME, which receives these values:

```
FILE=DATETIME,      SUFFIX=FOC      ,$
SEGNAME=DATETIME,  SEGTYPE=S0      ,$
FIELD=ID, ID,      USAGE = I2      ,$
FIELD=DT1, DT1,    USAGE=HYYMDS    ,$
```

### **Example:** Using a Date-Time Value in a COMPUTE Command

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME AND COMPUTE
NEWSAL/D12.2M = CURR_SAL + (0.1 * CURR_SAL);
RAISETIME/HYYMDIA = DT(20000101 09:00AM);
WHERE CURR_JOBCODE LIKE 'B%'
END
```

The output is:

LAST_NAME	FIRST_NAME	NEWSAL	RAISETIME
-----	-----	-----	-----
SMITH	MARY	\$14,520.00	2000/01/01 9:00AM
JONES	DIANE	\$20,328.00	2000/01/01 9:00AM
ROMANS	ANTHONY	\$23,232.00	2000/01/01 9:00AM
MCCOY	JOHN	\$20,328.00	2000/01/01 9:00AM
BLACKWOOD	ROSEMARIE	\$23,958.00	2000/01/01 9:00AM
MCKNIGHT	ROGER	\$17,710.00	2000/01/01 9:00AM

### **Example:** Using a Date-Time Value in WHERE Criteria

In a WHERE clause, a date-time constant must use the DT( ) format:

```
TABLE FILE VIDEOTR2
PRINT CUSTID TRANSDATE
WHERE TRANSDATE GT DT(2000/01/01 02:57:25)
END
```

The output is:

CUSTID	TRANSDATE
-----	-----
1118	2000/06/26 05:45
1237	2000/02/05 03:30

**Example:**    Using a Date-Time Value in IF Criteria

In an IF clause, a date-time constant must be enclosed in single quotation marks if it contains any blanks:

```
TABLE FILE VIDEOTR2
PRINT CUSTID TRANSDATE
IF TRANSDATE GT '2000/01/01 02:57:25'
END
```

**Note:** The DT prefix for a date-time constant is not supported in an IF clause.

The output is:

CUSTID	TRANSDATE
-----	-----
1118	2000/06/26 05:45
1237	2000/02/05 03:30

**Example:**    Specifying Universal Date-Time Input Values

With DTSTANDARD settings of STANDARD and STANDARDU, the following date-time values can be read as input:

Input Value	Description
14:30[:20,99]	Comma separates time components instead of period
14:30[:20.99]Z	Universal time
15:30[:20,99]+01 15:30[:20,99]+0100 15:30[:20,99]+01:00	Each of these is the same as above in Central European Time
09:30[:20.99]-05	Same as above in Eastern Standard Time

Note that these values are stored identically internally with the STANDARDU setting. With the STANDARD setting, everything following the Z, +, or - is ignored.

**Manipulating Date-Time Values**

Any two date-time values can be compared, even if their lengths do not match.

If a date-time field supports missing values, fields that contain the missing value have a greater value than any date-time field can have. Therefore, in order to exclude missing values from the report output when using a GT or GE operator in a selection test, it is recommended that you add the additional constraint field NE MISSING to the selection test:

```
date_time_field {GT|GE} date_time_value AND date_time_field NE MISSING
```

Assignments are permitted between date-time formats of equal or different lengths. Assigning a 10-byte date-time value to an 8-byte date-time value truncates the microsecond portion (no rounding takes place). Assigning a short value to a long one sets the low-order three digits of the microseconds to zero.

Other operations, including concatenation, EDIT, and LIKE on date-time operands are not supported. Prefix operators that work with alphanumeric fields are supported.

### **Example:** Testing for Missing Date-Time Values

Consider the DATETIM2 Master File:

```
FILE=DATETIM2, SUFFIX=FOC , $
SEGNAME=DATETIME, SEGTYPE=S0 , $
FIELD=ID, ID, USAGE = I2 , $
FIELD=DT1, DT1, USAGE=HYYMDS, MISSING=ON, $
```

Field DT1 supports missing values. Consider the following request:

```
TABLE FILE DATETIM2
PRINT ID DT1
END
```

The resulting report output shows that in the instance with ID=3, the field DT1 has a missing value:

```
ID  DT1
--  ---
1   2000/01/01 02:57:25
2   1999/12/31 00:00:00
3   .
```

The following request selects values of DT1 that are greater than 2000/01/01 00:00:00 and are not missing:

```
TABLE FILE DATETIM2
PRINT ID DT1
WHERE DT1 NE MISSING AND DT1 GT DT(2000/01/01 00:00:00);
END
```

The missing value is not included in the report output:

ID	DT1
--	---
1	2000/01/01 02:57:25

**Example: Assigning a Different Usage Format to a Date-Time Column**

Consider the following request using the VIDEOTR2 data source:

```
TABLE FILE VIDEOTR2
PRINT CUSTID TRANSDATE AND COMPUTE
  DT2/HYMDH = TRANSDATE;
  T1/HHIS = TRANSDATE;
WHERE DATE EQ 2000
END
```

The output is:

CUSTID	TRANSDATE	DT2	T1
-----	-----	---	--
1118	2000/06/26 05:45	2000/06/26 05	05:45:00
1237	2000/02/05 03:30	2000/02/05 03	03:30:00

**Creating a Character Expression**

A character expression uses alphanumeric constants, fields, concatenation operators, or functions to derive an alphanumeric value.

Both text and alphanumeric fields can be assigned values stored in text fields or alphanumeric expressions in TABLE COMPUTE, MODIFY COMPUTE, and DEFINE commands. If an alphanumeric field is assigned the value of a text field that is too long for the alphanumeric field, the value is truncated before being assigned to the alphanumeric field.

A character expression can consist of:

- ❑ An alphanumeric constant (character string) enclosed in single quotation marks. For example:

```
COMPUTE STATE/A2 = 'NY';
```

- ❑ A combination of alphanumeric fields and/or constants joined by the concatenation operator. For example:

```
DEFINE FILE EMPLOYEE TITLE/A19 = 'DR. ' | LAST_NAME;
END
```

- ❑ An alphanumeric function. For example:

```
DEFINE FILE EMPLOYEE INITIAL/A1 = EDIT(FIRST_NAME, '9$$$$$$$$$');
END
```

- ☐ A text field.

**Note:**

- ☐ Non-printable characters are not supported in an alphanumeric constant.
- ☐ Two consecutive single quotation marks represent a null value with format A1V and an actual length of 0 (zero), when the field has MISSING ON.

## Embedding a Quotation Mark in a Quote-Delimited Literal String

Under certain conditions, you can use quote-delimited strings containing embedded quotation marks. Within the string, you can use either one single quotation mark or two contiguous single quotation marks to represent the single quotation mark. Both are interpreted as a single quotation mark.

You can use quote-delimited strings in the following instances:

- ☐ WHERE and IF criteria containing multiple quotes.
- ☐ WHERE criteria containing: *fieldname* {IS, IS-NOT, IN, IN FILE, or NOT IN FILE}.
- ☐ EDIT.
- ☐ WHEN *fieldname* EQ an embedded quote in a literal.
- ☐ DEFINE commands.
- ☐ DEFINE attributes in Master Files.
- ☐ Database Administrator (DBA) attributes in Master Files (for example, VALUE = *fieldname* EQ an embedded quote in a literal).
- ☐ ACCEPT=, DESCRIPTION=, TITLE= attributes in Master Files.
- ☐ AS.
- ☐ DECODE.

**Example: Specifying the Data Value O'BRIEN in a Quote-Delimited Literal String**

The following example illustrates the use of quotation marks for the correct interpretation of the data value O'BRIEN:

```
TABLE FILE VIDEOTRK
PRINT LASTNAME
WHERE LASTNAME IS 'O'BRIEN'
END
```

**Concatenating Character Strings**

You can write an expression that concatenates two or more alphanumeric constants and/or fields into a single character string. This concatenation operator has two forms, as shown in the following table:

Symbol	Represents	Description
	Weak concatenation	Preserves trailing blanks.
	Strong concatenation	Moves trailing blanks to the end of a concatenated string.

**Example: Concatenating Character Strings**

The following example uses the EDIT function to extract the first initial from a first name. It then uses both strong and weak concatenation to produce the last name, followed by a comma, followed by the first initial, followed by a period:

```
DEFINE FILE EMPLOYEE
FIRST_INIT/A1 = EDIT(FIRST_NAME, '9$$$$$$$$$');
NAME/A19 = LAST_NAME ||(' , ' | FIRST_INIT | '.');
END

TABLE FILE EMPLOYEE
PRINT NAME WHERE LAST_NAME IS 'BANNING'
END
```

The output is:

```
NAME
----
BANNING, J.
```



The request evaluates the expressions as follows:

1. The EDIT function extracts the initial J from FIRST\_NAME.
2. The expression in parentheses returns the value:  
 `, J.`
3. LAST\_NAME is concatenated to the string derived in step 2 to produce:  
`Banning, J.`

While LAST\_NAME has the format A15 in the EMPLOYEE Master File, strong concatenation suppresses the trailing blanks. Regardless of the suppression or inclusion of blanks, the resulting field name, NAME, has a length of 19 characters (A19).

## Creating a Variable Length Character Expression

As an alphanumeric type, an AnV field can be used in arithmetic and logical expressions in the same way that the An type is used.

- ☐ An expression that contains AnV type fields can be of either the AnV or An type.
- ☐ The type that results from the expression depends on the specific type of operation, as described in subsequent sections.

**Note:** Because AnV fields have two bytes of overhead and there is additional processing required to strip them, AnV format is not recommended for use in non-relational data sources.

## Using Concatenation With AnV Fields

If either of the operands in a concatenation between two fields is an AnV field, variable length alphanumeric rules are used to perform the concatenation:

- ☐ The size of the concatenated string is the sum of the sizes of the operands.
- ☐ For weak concatenation, the actual length of the concatenated string is the sum of the two actual lengths of the input strings.
- ☐ For strong concatenation, the actual length stored in an AnV field of the concatenated string is the sum of the actual length of the first input string minus its number of trailing blanks plus the actual length of the second string.
- ☐ For any An field in the concatenation, the size and length are equal.

- ❑ Two consecutive single quotation marks represent a null value with format A1V and an actual length of 0 (zero), when the field has MISSING ON.

Using the EDIT Function With AnV Fields

The following expression results in an AnV format only when x has AnV format.

```
EDIT(x,mask)
```

The actual length of the result is the number of characters in *mask* other than '\$'.

Note that an actual length of zero may result.

EDIT(x) can be used to convert an AnV field to an integer value when x has AnV format.

Using CONTAINS and OMITS With AnV Fields

The only difference in evaluation of the CONTAINS and OMITS operators with AnV fields occurs when one of the operands has an actual length of zero.

In the following examples, the field Z has an actual length of zero, but X and Y do not:

Expression	Result
Z CONTAINS Y	FALSE
X CONTAINS Z	TRUE
Z CONTAINS Z	TRUE
Z OMITS Y	TRUE
X OMITS Z	FALSE
Z OMITS Z	FALSE

Using LIKE With AnV Fields

The only difference in evaluation of the following expression occurs when x has an actual length of zero:

```
x LIKE mask ...
```

In the following example, the field instance Z has an actual length of zero:

```
Z LIKE mask ...
```

This expression evaluates to TRUE only when the mask consists exclusively of percent ('%') signs.

Note that no other mask can evaluate to an empty string. Even the mask in the following expression has a length of one, and therefore the expression evaluates as FALSE:

```
Z LIKE ''
```

### Using the EQ, NE, LT, GT, LE, and GE Operators With AnV Fields

As with An type fields, operations are evaluated on the assumption that the shorter operand is padded with blanks.

Therefore, even an empty AnV field, Z, is compared as a field consisting of all blanks.

In the following examples, Z is an empty AnV field instance and X is an AnV field instance that is not empty and contains non-blank characters:

Expression	Result
<pre>Z EQ Z Z GE Z Z LE Z</pre>	TRUE
<pre>Z NE Z Z LT Z Z GT Z</pre>	FALSE
<pre>Z EQ X</pre>	FALSE
<pre>Z NE X</pre>	TRUE
<pre>Z LT X</pre>	TRUE
<pre>Z GT X</pre>	FALSE
<pre>Z LE X</pre>	TRUE
<pre>Z GE X</pre>	FALSE

Expression	Result
X EQ Z	FALSE
X NE Z	TRUE
X LT Z	FALSE
X GT Z	TRUE
X LE Z	FALSE
X GE Z	TRUE

Using the DECODE Function With AnV Fields

```
DECODE alphafield (value 'result'...
```

The use of either an An or AnV field with DECODE causes a result of type An as long as the result part of the value-result pairs is provided as a constant. (Constants are type An.)

Using the Assignment Operator With AnV Fields

There are three situations to consider when using the assignment operator with the AnV format: AnV data type on the right hand side only, AnV data type on both sides, and AnV data type on the left side only.

```
fld/An = AnV_type_expression;
```

- ☐ The actual length of the evaluated expression is lost on assignment to the An field.
- ☐ The size of the AnV result does not prevent assignment to a shorter An format field:
  - ☐ If the result of the expression has an actual length that is shorter than the length of the field on the left side of the assignment operator, the result is padded with blanks.
  - ☐ If the result of the expression has an actual length that is longer than the length of the field on the left side of the assignment operator, the result is truncated.

*fld/AnV = AnV\_type\_expression;*

- ❑ The length of the result is assigned as the length of the field on the left of the assignment operator unless it exceeds the field's declared size. In this case, the length assigned is the declared size (*n*).
- ❑ The size of the AnV evaluation result does not prevent assignment to a shorter AnV field:
  - ❑ If the length of the result of the expression is shorter than the size of the field on the left side of the assignment operator, the result is padded with blanks.
  - ❑ If the result of the expression has an actual length that is longer than the size of the field on the left side of the assignment operator, the result is truncated.

*fld/AnV = An\_type\_expression;*

- ❑ The length of the field on the left side of the assignment operator is assigned equal to its size (*n*).
- ❑ The actual length of the result is verified against the size *n* declared for the AnV field. An error is generated if the result is longer than *n*.

## Creating a Logical Expression

A logical expression determines whether a particular condition is true. There are two kinds of logical expressions: relational and Boolean. The entities to be compared determine the kind of expression used:

- ❑ A relational expression returns TRUE or FALSE based on a comparison of two individual values (either field values or constants).
- ❑ A Boolean expression returns TRUE or FALSE based on the outcome of two or more relational expressions.

You can use a logical expression to assign a value to a numeric field. If the expression is true, the field receives the value 1. If the expression is false, the field receives the value 0.

**Reference: Logical Operators**

The following is a list of common operators used in logical expressions. For information on relational operators and additional operators available for record selection using WHERE and IF, see [Selecting Records for Your Report](#) on page 219.

Operator	Description
EQ	Returns the value TRUE if the value on the left is equal to the value on the right.
NE	Returns the value TRUE if the value on the left is not equal to the value on the right.
GE	Returns the value TRUE if the value on the left is greater than or equal to the value on the right.
GT	Returns the value TRUE if the value on the left is greater than the value on the right.
LE	Returns the value TRUE if the value on the left is less than or equal to the value on the right.
LT	Returns the value TRUE if the value on the left is less than the value on the right.
AND	Returns the value TRUE if both operands are true.
OR	Returns the value TRUE if either operand is true.
NOT	Returns the value TRUE if the operand is false.
CONTAINS	Contains the specified character strings.
OMITS	Omits the specified character strings.
IS MISSING	Returns the value TRUE if the field is missing.
IS-NOT MISSING	Returns the value TRUE if the field is not missing.

**Syntax: How to Write a Relational Expression**

Any of the following are valid for a relational expression:

```
value {EQ|NE} value value {LE|LT} value value {GE|GT}
valuecharacter_value {CONTAINS|OMITS} character_value
```

where:

*value*

Is a field value or constant.

*character\_value*

Is a character string. If it contains blanks, the string must be enclosed in single quotation marks.

**Syntax: How to Write a Boolean Expression**

Either of the following is valid for a Boolean expression:

```
(relational_expression) {AND|OR} (relational_expression)
NOT (logical_expression)
```

where:

*relational\_expression*

Is an expression based on a comparison of two individual values (either field values or constants).

*logical\_expression*

Is an expression that evaluates to the value TRUE or FALSE. If the expression is true, the field receives the value 1. If the expression is false, the field receives the value 0. The expression must be enclosed in parentheses.

**Creating a Conditional Expression**

A conditional expression assigns a value based on the result of a logical expression. The assigned value can be numeric or alphanumeric.

**Note:** Unlike selection criteria using IF, all alphanumeric values in conditional expressions must be enclosed in single quotation marks. For example, IF COUNTRY EQ 'ENGLAND'.

**Syntax:**      **How to Write a Conditional Expression**

```
IF expression1 THEN expression2 [ELSE expression3]
```

where:

*expression1*

Is the expression that is evaluated to determine whether the field is assigned the value of *expression2* or of *expression3*.

*expression2*

Is an expression that results in a format compatible with the format assigned to the field. It may be a conditional expression, in which case you must enclose it in parentheses.

*expression3*

Is an expression that results in a format compatible with the format assigned to the field. Enclosure of the expression in parentheses is optional.

ELSE

Is optional, along with *expression3*. However, if you do not specify an ELSE condition and the IF condition is not met, the value is taken from the last evaluated condition. Therefore, the results may not be what you expect if you do not include an ELSE condition.

Note that the final sorted report may display mixed values. This depends on whether a DEFINE or a COMPUTE is used, and if a data record is evaluated before or after aggregation.

The expressions following THEN and ELSE must result in a format that is compatible with the format assigned to the field. Each of these expressions may itself be a conditional expression. However, the expression following IF may not be an IF ... THEN ... ELSE expression (for example, IF ... IF ...).

**Example:**      **Supplying a Value With a Conditional Expression**

The following example uses a conditional expression to assign the value NONE to the field BANK\_NAME when it is missing a data value (that is, when the field has no data in the data source):

```
DEFINE FILE EMPLOYEE
BANK_NAME/A20 = IF BANK_NAME EQ ' ' THEN 'NONE'
ELSE BANK_NAME;
END
```

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL AND BANK_NAME
BY EMP_ID BY BANK_ACCT
END
```



The output is:

EMP_ID	BANK_ACCT	CURR_SAL	BANK_NAME
-----	-----	-----	-----
071382660		\$11,000.00	NONE
112847612		\$13,200.00	NONE
117593129	40950036	\$18,480.00	STATE
119265415		\$9,500.00	NONE
119329144	160633	\$29,700.00	BEST BANK
123764317	819000702	\$26,862.00	ASSOCIATED
126724188		\$21,120.00	NONE
219984371		\$18,480.00	NONE
326179357	122850108	\$21,780.00	ASSOCIATED
451123478	136500120	\$16,100.00	ASSOCIATED
543729165		\$9,000.00	NONE
818692173	163800144	\$27,062.00	BANK ASSOCIATION

### **Example:** Defining a True or False Condition

You can define a true or false condition and then test it to control report output. The following example assigns the value TRUE to the field MYTEST if either of the relational expressions in parentheses is true. It then tests the value of MYTEST:

```
DEFINE FILE EMPLOYEE
MYTEST= (CURR_SAL GE 11000) OR (DEPARTMENT EQ 'MIS');
END

TABLE FILE EMPLOYEE
PRINT CURR_SAL AND DEPARTMENT
BY EMP_ID
IF MYTEST IS TRUE
END
```

The output is:

EMP_ID	CURR_SAL	DEPARTMENT
-----	-----	-----
071382660	\$11,000.00	PRODUCTION
112847612	\$13,200.00	MIS
117593129	\$18,480.00	MIS
119329144	\$29,700.00	PRODUCTION
123764317	\$26,862.00	PRODUCTION
126724188	\$21,120.00	PRODUCTION
219984371	\$18,480.00	MIS
326179357	\$21,780.00	MIS
451123478	\$16,100.00	PRODUCTION
543729165	\$9,000.00	MIS
818692173	\$27,062.00	MIS

**Note:** Testing for a TRUE or FALSE condition is valid only with the IF command. It is not valid with WHERE.





# Chapter 8

## Saving and Reusing Your Report Output

---

When you run a report request, by default the data values are collected and presented in a viewable form, complete with column titles and formatting features. Instead of viewing the data values, you can save them to a special data file to:

- ☐ Display as a webpage, as a printed document, or in a text document.
- ☐ Process in another application, such as a spreadsheet, a database, a word processor, or a 3GL program.
- ☐ Send to another location, such as a browser or PC.
- ☐ Extract a subset of the original data source in order to generate multi-step reports.
- ☐ Extract a data source to a structured extract file that retains information about the segment relationships in order to facilitate migration of data sources and reports between operating environments.

### In this chapter:

- |  |   |
|--|---|
| <input type="checkbox"/> <a href="#">Saving Your Report Output</a>                   | <input type="checkbox"/> <a href="#">Creating a PCHOLD File</a>   |
| <input type="checkbox"/> <a href="#">Creating a HOLD File</a>                        | <input type="checkbox"/> <a href="#">Choosing Output File Formats</a>                                       |
| <input type="checkbox"/> <a href="#">Holding Report Output in FOCUS Format</a>       | <input type="checkbox"/> <a href="#">Using Text Fields in Output Files</a>                                  |
| <input type="checkbox"/> <a href="#">Controlling Attributes in HOLD Master Files</a> | <input type="checkbox"/> <a href="#">Creating a Delimited Sequential File</a>                               |
| <input type="checkbox"/> <a href="#">Keyed Retrieval From HOLD Files</a>             | <input type="checkbox"/> <a href="#">Saving Report Output in INTERNAL Format</a>                            |
| <input type="checkbox"/> <a href="#">Saving and Retrieving HOLD Files</a>            | <input type="checkbox"/> <a href="#">Creating A Subquery or Sequential File With HOLD FORMAT SQL_SCRIPT</a> |
| <input type="checkbox"/> <a href="#">Using DBMS Temporary Tables as HOLD Files</a>   | <input type="checkbox"/> <a href="#">Creating a Structured HOLD File</a>                                    |
| <input type="checkbox"/> <a href="#">Creating SAVE and SAVB Files</a>                | <input type="checkbox"/> <a href="#">Specifying MIME Types for WebFOCUS Reports</a>                         |
-

## Saving Your Report Output

The following commands extract and save report output in a variety of file formats:

- ❑ **HOLD.** The HOLD command creates a data source containing the output of a report request. By default, the data is stored in binary format, but you can specify a different format, such as FOCUS, HTML, or Excel. For some formats, the HOLD command also creates a corresponding Master File. You can then write other report requests that in turn extract or save data from the HOLD file. See [Creating a HOLD File](#) on page 473.
- ❑ **SAVE and SAVB.** The SAVE command is identical to a HOLD command, except that it does not create a Master File, and ALPHA is the default format. If you wish to create a SAVE file in BINARY format, use a variation of the SAVE command called SAVB.  
  
As with a HOLD file, you can specify a variety of formats suitable for use with other software products. See [Creating SAVE and SAVB Files](#) on page 506.
- ❑ **PCHOLD.** The PCHOLD command creates a data source containing the output of a report request, and downloads the HOLD data source and the optional Master File to a client computer or browser. As with a HOLD file, you can specify a variety of file formats. See [Creating a PCHOLD File](#) on page 509.

**Tip:** When saving or holding output files, it is recommended to have an Allocation in place for the file. This is not applicable for PCHOLD files.

## Naming and Storing Report Output Files

During a session, a report output file remains usable until it is erased or overwritten. A subsequent output file created during the same session replaces the initial version, unless you give it another name by using the AS phrase.

A FILEDEF command is automatically issued when you create an output file. The ddname used to identify the file is the same as the name of the report output file (HOLD, SAVE, or SAVB, or the name in the AS phrase), if not already allocated.

By default, report output files created with HOLD, SAVE, or SAVB are written to temporary space. When the session ends, these files are no longer available unless you save the output to a specific location.

To save report output files to a specific location, use an ALLOCATE or FILEDEF command. For details, see [Saving and Retrieving HOLD Files](#) on page 498. You can dynamically allocate an output file using the DYNAM ALLOCATE or TSO ALLOCATE command in z/OS.

For details, see the *Developing Reporting Applications* manual.

When you create a HOLD file using the syntax `ON TABLE HOLD AS name`, the name can contain up to the maximum number of characters supported by your operating system. For information, see the section *Naming a Master File* in the *Describing Data With WebFOCUS Language* manual.

While a numeric name or a name beginning with a number is valid on many operating systems, such names are discouraged because they:

- ❑ Cause errors (FOC14069) when a request attempts to access data using a `SUFFIX=EDA` synonym that points to a file starting with a number. (Note that a `SUFFIX=EDA` synonym is created by the Adapter for Remote Servers.)
- ❑ May cause problems when an external application (such as an API application) that does not accept files starting with numbers interacts with a `SUFFIX=EDA` synonym that points to a file starting with a number.
- ❑ May be problematic to a third party application that does not work with numeric file names or with file names that begin with numbers.

## Creating a HOLD File

You can use the HOLD command to create report output files for a range of purposes:

- ❑ As a tool for data extraction, the HOLD command enables you to retrieve and process data, then extract the results for further processing. Your report request can create a new data source, complete with a corresponding Master File, from which you can generate new reports.

The output Master File contains only the fields in the report request. The fields in a HOLD file have the original names specified in the Master File that are retrieved if the report is displayed or printed. You can alter the field names in the output Master File using the `AS` phrase in conjunction with the command `SET ASNAMES`. See [Controlling Field Names in a HOLD Master File](#) on page 485.

- ❑ The HOLD command enables you to specify the appropriate formats for displaying or processing report output files in other software applications. See [Choosing Output File Formats](#) on page 511.
- ❑ When an application requires a data format that is not among the HOLD options, you can use a subroutine to process each output record as it is written to the HOLD data source.

If your environment supports the `SET` parameter `SAVEMATRIX`, you can preserve the internal matrix of your last report in order to keep it available for subsequent HOLD, SAVE, and SAVB commands when the request is followed by Dialogue Manager commands. For details on `SAVEMATRIX`, see the *Developing Reporting Applications* manual.

## **Syntax:** How to Create a HOLD File

From a report request, use

```
ON TABLE HOLD [AS filename] [FORMAT fmt] [DATASET dataset]
    [MISSING {ON|OFF}]
    [PERSISTENCE {STAGE|PERMANENT}]
```

or

```
hold_field HOLD [AS filename] [FORMAT fmt] [DATASET dataset]
    [MISSING {ON|OFF}]
    [PERSISTENCE {STAGE|PERMANENT}]
```

where:

### **HOLD**

Extracts and saves report output. BINARY is the default format used when the HOLD command is issued without an explicit format. The output is saved with an associated Master File.

**Note:** Change the default output format to ALPHA by issuing the SET HOLDFORMAT command.

### ***hold\_field***

Is the name of the last display field in the request.

### **AS *filename***

Specifies a name for the HOLD file. If you do not specify a file name, HOLD becomes the default. Since each subsequent HOLD command overwrites the previous HOLD file, it is advisable to code a distinct file name in each request to direct the extracted data to a separate file, thereby preventing it from being overwritten by the next HOLD command.

The name can contain up to the maximum number of characters supported by your operating system. For information, see the section *Naming a Master File* in the *Describing Data With WebFOCUS Language* manual.

### **FORMAT *fmt***

Specifies the format of the HOLD output file. BINARY is the default format for reporting servers.

- ☐ To display as a webpage, choose: HTML, HTMTABLE, DHTML
- ☐ To display as a printed document, choose: PDF, PS
- ☐ To use in a text document, choose: ALPHA, DOC, WP
- ☐ To use in a spreadsheet application, choose: DIF, EXCEL, EXL97, EXL2K [PIVOT], LOTUS, SYLK

- ☐ To use in a database application, choose: COMMA, COM, COMT, DB2, DATREC, DFIX, FOCUS, INGRES, REDBRICK, SQL, SQLDBC, SQLORA, SQLINF, SQLMSS, SQLSYB, SQLODBC, TAB, TABT, XFOCUS
- ☐ To use with a 3-GL program, choose: INTERNAL
- ☐ To use for additional reporting, choose: ALPHA, BINARY, FOCUS
- ☐ To use as a transaction file for modifying a data source, choose: ALPHA, BINARY
- ☐ To use for interactive analysis without connection to a server, choose: AHTML, APDF, FLEX

For details about all available formats, see [Choosing Output File Formats](#) on page 511.

#### *dataset*

Can be a fully-qualified data set or file name or an n-part name (app/.../filename.ext).

You can specify a data set or file to contain the report output within the request itself, rather than relying on an external or default specification. This allows you to place a permanent hold file in any folder, directory or PDS that you can write to, whether or not that location is included in your APP PATH. The accompanying HOLD Master File will have the DATASET attribute pointing to the file that was generated.

The case in which the data set name is added in the Master File depends on the value of the FILECASE parameter. By default, lowercase is used. The actual data set is created with its name in the case that conforms to the standards of your operating environment.

#### **Note:**

- ☐ On z/OS, the file cannot already exist or be allocated when the HOLD command is issued. Therefore, If the file already exists you must free the allocation and then delete the file before running the request.
- ☐ You can use a USS naming convention for the DATASET attribute to store the file in the USS environment (for text output types).

#### **MISSING**

Controls whether fields with the attribute MISSING=ON in the Master File are carried over into the HOLD file. MISSING ON is the default attribute. If the HOLD command specifies MISSING OFF, fields with the MISSING attribute are not carried over. For related information, see [Handling Records With Missing Field Values](#) on page 971. Also see the *Developing Reporting Applications* manual for the SET HOLDMISS, SET NULL, and SET HNODATA parameters, which control how missing values are propagated to alphanumeric and comma-delimited files.

### PERSISTENCE

Applies only to Relational HOLD files (FORMAT *sqlengine*). Specifies how to create intermediate tables that will be used only during UPLOAD and EBL requests to accelerate performance by keeping all processing on the DBMS server instead of downloading data into a HOLD file. The actual type of the intermediate table will be determined at run time, based on specific DBMS-supported features and the data-populating mechanisms being used. Valid values are:

- ❑ **STAGE.** Will create either a Volatile or GLOBAL TEMPORARY table, for a DBMS that supports that functionality. For a DBMS that does not support that functionality, a message will display and the table will not be created.
- ❑ **PERMANENT.** Will create a regular SQL table with a uniquely-generated name that will be used in the request and will be available for further use after the request ends, but will be dropped at the end of the session. This is the default value for PERSISTENCE for HOLD FORMAT *sqlengine*.

### **Syntax:** How to Set the Default HOLD Format

```
SET HOLDFORMAT = {BINARY|ALPHA}
```

or

```
ON TABLE SET HOLDFORMAT {BINARY|ALPHA}
```

where:

#### [BINARY](#)

Sets the default HOLD file format to BINARY.

#### [ALPHA](#)

Sets the default HOLD file format to ALPHA.

### **Example:** Extracting Data to a HOLD File

The following request extracts data from the EMPLOYEE data source and creates a HOLD file.

```
TABLE FILE EMPLOYEE
SUM CURR_SAL AND ED_HRS
BY DEPARTMENT
LIST CURR_SAL AND ED_HRS AND BANK_ACCT
BY DEPARTMENT
BY LAST_NAME BY FIRST_NAME
ON TABLE HOLD
END
```

The following message appears:



```
NUMBER OF RECORDS IN TABLE= 12 LINES= 12
```

To display the report generated by this request issue a report request against the HOLD file.

**Tip:** If you wish to view the information in the HOLD Master File before reporting against it, you can issue the query command ? HOLD.

### **Syntax:** How to Query a HOLD Master File

If the HOLD format option you select creates a Master File, you can issue the following command to display the fields, aliases, and formats in the HOLD Master File:

```
? HOLD
```

This command shows field names up to 32 characters. If a field name exceeds 32 characters, a caret (>) in the 32nd position indicates a longer field name.

If you have renamed the HOLD file using AS *filename*, use the following syntax:

```
? HOLD filename
```

**Tip:** You must issue the ? HOLD query in the same session in which the HOLD file is created.

### **Example:** Reporting Against a HOLD Master File

In the following HOLD file, the formats shown are the values of the FORMAT attribute. You can see the values of the ACTUAL attribute by displaying the HOLD Master File using TED or any text editor. USAGE and ACTUAL formats for text fields specify only the length of the first line of each logical record in the HOLD file. The USAGE format is the same as the field format in the original Master File. The ACTUAL format is rounded up to a full (internal) word boundary, as is done for alphanumeric fields.

The following request contains the query command ? HOLD, which displays the fields, aliases, and formats in the associated Master File and creates a HOLD file.

```
TABLE FILE EMPLOYEE
SUM CURR_SAL AND ED_HRS
BY DEPARTMENT
LIST CURR_SAL AND ED_HRS AND BANK_ACCT
BY DEPARTMENT
BY LAST_NAME BY FIRST_NAME
ON TABLE HOLD
END

? HOLD
```

The output is:

```
NUMBER OF RECORDS IN TABLE=      12      LINES=      12

DEFINITION OF HOLD FILE: HOLD

FIELDNAME                                ALIAS      FORMAT

DEPARTMENT                                E01        A10
CURR_SAL                                  E02        D12.2M
ED_HRS                                    E03        F6.2
LAST_NAME                                E04        A15
FIRST_NAME                                E05        A10
LIST                                      E06        I5
CURR_SAL                                  E07        D12.2M
ED_HRS                                    E08        F6.2
BANK_ACCT                                E09        I9S
```

You can now issue the following report request against the HOLD file:

```
TABLE FILE HOLD
PRINT E07 AS 'SALARY OF,EMPLOYEE' AND LAST_NAME AND FIRST_NAME
BY HIGHEST E03 AS 'TOTAL,DEPT,ED_HRS'
BY E01
BY HIGHEST E08 AS 'EMPLOYEE,ED_HRS'
END
```

The output is:

TOTAL				
DEPT OF		EMPLOYEE	SALARY	
ED_HRS	DEPARTMENT	ED_HRS	EMPLOYEE	LAST_NAME
FIRST_NAME				
-----	-----	-----	-----	-----
231.00	MIS	75.00	\$21,780.00	BLACKWOOD
ROSEMARIE		50.00	\$18,480.00	JONES
DIANE		45.00	\$27,062.00	CROSS
BARBARA		36.00	\$13,200.00	SMITH
MARY		25.00	\$9,000.00	GREENSPAN
MARY		.00	\$18,480.00	MCCOY
JOHN	PRODUCTION	50.00	\$16,100.00	MCKNIGHT
120.00		30.00	\$26,862.00	IRVING
ROGER		25.00	\$11,000.00	STEVENS
JOAN		10.00	\$9,500.00	SMITH
ALFRED		5.00	\$21,120.00	ROMANS
RICHARD		.00	\$29,700.00	BANNING
ANTHONY				JOHN

## Holding Report Output in FOCUS Format

Whether issued within a request or after the request has been executed, the HOLD command can create a FOCUS data source and a corresponding Master File from the data extracted by the report request. This feature enables you to create:

- ☐ A FOCUS data source from any other supported data source type.
- ☐ A subset of an existing FOCUS data source.

**Tip:** If you are working in an environment that supports SCAN, FSCAN, MODIFY, or Maintain, and you create a HOLD file in FOCUS format, you can update, as well as report against, the HOLD file. See your documentation on these facilities for details.

**Note:** Holding a file in FOCUS format may generate the (FOC441) warning: *The file exists already. Create will write over it.* Issuing the SET WARNING=OFF command suppresses this message.

**Syntax:**      **How to Create HOLD Files in FOCUS Format**

In a report request, use

```
ON TABLE HOLD [AS filename] FORMAT FOCUS [INDEX field1 field2 ...]
```

where:

*AS filename*

Specifies a name for the HOLD file. If you do not specify a file name, HOLD becomes the default. Since each subsequent HOLD command overwrites the previous HOLD file, it is advisable to provide a distinct file name in each request to direct the extracted data to a separate file, thereby preventing it from being overwritten by the next HOLD command.

The name can be up to 64 characters long.

**Note:** If you use a name longer than eight characters on z/OS, an eight-character member name is generated as described in the *Describing Data With WebFOCUS Language* manual. To relate the long name to the short member name, the \$ VIRT attribute is generated on the top line in the Master File. The resulting HOLD file is a temporary data file. To allocate the long Master File name to a permanent data file, issue the DYNAM ALLOCATE command with the LONGNAME option prior to the HOLD request. The ddname in the command must refer to an existing member of the MASTER PDS.

*INDEX field1...*

Enables you to index FOCUS fields. All fields specified after INDEX are specified as FIELDTYPE=I in the Master File. Up to four fields can be indexed.

**Note:** Since the number of index field names is variable, a command name that follows the HOLD command and starts with the same characters as a field name may be counted as another index field, generating an error. For example, if the command following HOLD starts with ON TABLE and a field name starts with the characters 'ON', the ON in the command will be considered a truncated field name to add to the index. To avoid this issue, either set the FIELDNAME parameter to NOTRUNC, so that command names will not be confused with truncated field names, or move the HOLD command to the end of the procedure right before the END command.

Note that once you use this format from Hot Screen, you cannot issue another HOLD command while in the same Hot Screen session.

**Reference: Operating System Notes for HOLD Files in FOCUS Format**

The HOLD file is dynamically allocated if it is not currently allocated in z/OS. This means the system may delete the file at the end of the session, even if you have not done so. Since HOLD files are usually deleted, this is the desired default. However, if you want to save the Master File, allocate it to ddname HOLDMAST as a permanent data set. The allocation can be performed in the standard FOCUS CLIST. For example:

```
ALLOC F(HOLDMAST) DA('qualif.HOLDMAST') SHR REUSE
```

Note that ddname HOLDMAST must not refer to the same PDS referred to by the MASTER and FOCEXEC ddnames.

**Reference: Controlling the FOCUS File Structure**

The structure of the FOCUS data source varies according to the report request. The rules are as follows:

- ❑ Each aggregation command (SUM, COUNT, WRITE) creates a segment, with each new BY field in the request becoming a key. In a request that uses multiple display commands, the key to any newly created segment does not contain keys that are in the parent segment.
- ❑ If a PRINT or LIST command is used to create a segment, all the BY fields, together with the internal FOCLIST field, form the key.
- ❑ All fields specified after INDEX are indexed; that is, FIELDTYPE=I is specified in the Master File. Up to four fields may be indexed.
- ❑ If the data in the HOLD file is longer than a page (4K for FOCUS data sources or 16K for XFOCUS data sources), it cannot be stored in a single segment. Data that is too long to become a single segment will become a parent segment with unique child segments. For a FOCUS data source, the fields will be grouped into normal FOCUS page size segments and added as unique segments up to the total maximum of 32K of data. For an XFOCUS data source, the root segment can hold the first 16K of data, and additional data up to the 32K total, will be placed in a single unique segment. BY fields must all occur in the portion of the data assigned to the root segment.

To control whether the ACCEPT and TITLE attributes are propagated to the Master File associated with the HOLD file, use the SET HOLDATTR command. To control the FIELDNAME attribute in the Master File of the HOLD file, use the SET ASNAMES command. For more information on how to control the TITLE, ACCEPT, and FIELDNAME attributes in a HOLD Master File, see [Controlling Attributes in HOLD Master Files](#) on page 484.

**Example: Creating a HOLD File in FOCUS Format**

The following example creates a subset of the CAR data source.

```
TABLE FILE CAR
SUM SALES BY COUNTRY BY CAR BY MODEL
ON TABLE HOLD AS X1 FORMAT FOCUS
END
```

This request creates a single-segment FOCUS data source with a SEGTYPE of S3 (because it has three BY fields) named X1.

The X1 Master File is created by the request:

```
FILE=X1, SUFFIX=FOC
SEGMENT=SEG01 ,SEGTYPE=S03
  FIELDNAME=COUNTRY      ,ALIAS=E01      ,USAGE=A10      , $
  FIELDNAME=CAR          ,ALIAS=E02      ,USAGE=A16      , $
  FIELDNAME=MODEL        ,ALIAS=E03      ,USAGE=A24      , $
  FIELDNAME=SALES        ,ALIAS=E04      ,USAGE=I6       , $
```

**Example: Using PRINT to Create a FOCUS Data Source With a FOCLIST Field**

This example creates a single-segment FOCUS data source with a SEGTYPE of S4 because of the three BY fields and the FOCLIST FIELD.

```
TABLE FILE CAR
PRINT SALES BY COUNTRY BY CAR BY MODEL
ON TABLE HOLD AS X2 FORMAT FOCUS INDEX MODEL
END
```

The Master File created by this request is:

```
FILE=X2, SUFFIX=FOC
SEGMENT=SEG01, SEGTYPE=S04
  FIELDNAME=COUNTRY      ,ALIAS=E01      ,USAGE=A10      , $
  FIELDNAME=CAR          ,ALIAS=E02      ,USAGE=A16      , $
  FIELDNAME=MODEL        ,ALIAS=E03      ,USAGE=A24      , FIELDTYPE=I , $
  FIELDNAME=FOCLIST      ,ALIAS=E04      ,USAGE=I5       , $
  FIELDNAME=SALES        ,ALIAS=E05      ,USAGE=I6       , $
```

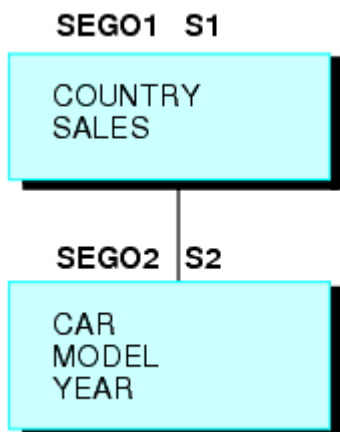
**Example: Creating a Two-Segment FOCUS Data Source**

The following request contains two SUM commands. The first, SUM SALES BY COUNTRY, creates a segment with COUNTRY as the key and the summed values of SALES as a data field. The second, SUM SALES BY COUNTRY BY CAR BY MODEL, creates a descendant segment, with CAR and MODEL as the keys and SALES as a non-key field.

The COUNTRY field does not form part of the key to the second segment. COUNTRY is a key in the path to the second segment. Any repetition of this value is redundant.

```
TABLE FILE CAR
SUM SALES BY COUNTRY
SUM SALES BY COUNTRY BY CAR BY MODEL
ON TABLE HOLD AS X3 FORMAT FOCUS
END
```

This creates a two-segment FOCUS data source:



The Master File for this newly-created FOCUS data source is:

```
FILE=X3, SUFFIX=FOC
SEGMENT=SEG01, SEGTYPE=S01
  FIELDNAME=COUNTRY      ,ALIAS=E01      ,USAGE=A10      , $
  FIELDNAME=SALES        ,ALIAS=E02      ,USAGE=I6        , $
SEGMENT=SEG02, SEGTYPE=S02,PARENT=SEG01
  FIELDNAME=CAR          ,ALIAS=E03      ,USAGE=A16      , $
  FIELDNAME=MODEL        ,ALIAS=E04      ,USAGE=A24      , $
  FIELDNAME=SALES        ,ALIAS=E05      ,USAGE=I6        , $
```

**Example: Creating a Three-Segment FOCUS Data Source**

In this example, each display command creates one segment.

The key to the root segment is the BY field, COUNTRY, while the keys to the descendant segments are the new BY fields. The last segment uses the internal FOCLIST field as part of the key, since the display command is PRINT.

```
TABLE FILE CAR
SUM SALES BY COUNTRY BY CAR
SUM SALES BY COUNTRY BY CAR BY MODEL
PRINT SALES BY COUNTRY BY CAR BY MODEL BY BODY
ON TABLE HOLD AS X4 FORMAT FOCUS INDEX COUNTRY MODEL
END
```

The Master File is:

```
FILE=X4, SUFFIX=FOC
SEGMENT=SEG01, SEGTYPE =S02
  FIELDNAME=COUNTRY ,ALIAS=E01 ,USAGE=A10 ,FIELDTYPE=I,$
  FIELDNAME=CAR ,ALIAS=E02 ,USAGE=A16 ,,$
  FIELDNAME=SALES ,ALIAS=E03 ,USAGE=I6 ,,$
SEGMENT=SEG02, SEGTYPE =S01 ,PARENT=SEG01
  FIELDNAME=MODEL ,ALIAS=E04 ,USAGE=A24 ,FIELDTYPE=I,$
  FIELDNAME=SALES ,ALIAS=E05 ,USAGE=I6 ,,$
SEGMENT=SEG03, SEGTYPE =S02 ,PARENT=SEG02
  FIELDNAME=BODYTYPE ,ALIAS=E06 ,USAGE=A12 ,,$
  FIELDNAME=FOCLIST ,ALIAS=E07 ,USAGE=I5 ,,$
  FIELDNAME=SALES ,ALIAS=E08 ,USAGE=I6 ,,$
```

**Controlling Attributes in HOLD Master Files**

The commands SET ASNAMES, SET HOLDLIST, and SET HOLDATTR enable you to control the FIELDNAME, TITLE, and ACCEPT attributes in HOLD Master Files. These commands are issued prior to the report request and remain in effect for the duration of the session, unless you change them.

- ❑ The SET ASNAMES command designates text specified in an AS phrase as the field name in the HOLD Master File, and concatenates it to the beginning of the first field name specified in an ACROSS phrase. See [Controlling Field Names in a HOLD Master File](#) on page 485.
- ❑ The SET HOLDLIST command restricts fields in HOLD and PCHOLD files to those appearing in a request. That is, non-displaying fields in a request (those designated as NOPRINT fields) are not included in the HOLD file. You can also distinguish between implicitly and explicitly non-displaying fields. See [Controlling Fields in a HOLD Master File](#) on page 490.



- ❑ The SET HOLDATTR command propagates TITLE and ACCEPT attributes used in the original Master File to the HOLD Master File. See [Controlling the TITLE and ACCEPT Attributes in the HOLD Master File](#) on page 495.

In addition, the SET HOLDSTAT command enables you to include comments and DBA information in the HOLD Master File. For more information about SET HOLDSTAT, see the *Describing Data With WebFOCUS Language* manual. For details about SET commands, see the *Developing Reporting Applications* manual.

## Controlling Field Names in a HOLD Master File

When SET ASNAMES is set to ON, MIXED or FOCUS, the literal specified in an AS phrase in a report request is used as the field name in a HOLD Master File. This command also controls how ACROSS fields are named in HOLD files.

### **Syntax:** How to Control Field Names in a HOLD Master File

```
SET ASNAMES = [ON|OFF|MIXED|FOCUS|FLIP]
```

where:

#### OFF

Does not use the literal specified in an AS phrase as a field name in HOLD files, and does not affect the way ACROSS fields are named.

#### ON

Uppercases the literal specified in an AS phrase and propagates it as the field name in the HOLD Master File. Creates names for ACROSS fields that consist of the AS name value concatenated to the beginning of the ACROSS field value and controls the way ACROSS fields are named in HOLD files of any format.

#### MIXED

Uses the literal specified in an AS phrase for the field name, retaining the case of the AS name, and creates names for ACROSS fields that consist of the AS name value concatenated to the beginning of the ACROSS field value.

#### FOCUS

Uses the literal specified in an AS phrase as the field name and controls the way ACROSS fields are named only in HOLD files in FOCUS format. FOCUS is the default value.

#### FLIP

Propagates the field names in the original Master File to the alias names in the HOLD Master File and the alias names in the original Master File to the field names in the HOLD Master File.

**Reference: Usage Notes for Controlling Field Names in HOLD Files**

- ❑ If no AS phrase is specified for a field, the field name from the original Master File is used. The TITLE attribute specified in the Master File is not used unless SET HOLDATTRS was previously issued.
- ❑ To ensure that fields referenced more than once in a request have unique names in the HOLD Master File, use SET ASNAMES.
- ❑ Special characters and blanks used in the AS phrase are preserved in the field name that is created when SET ASNAMES is used. Use single quotation marks around the non-standard field names when referring to them in the newly created Master File.
- ❑ Text specified in an AS phrase that contains more than 66 characters is truncated to 66 characters in the Master File.
- ❑ When generating field names and aliases for a HOLD file with the default setting for the ASNAMES parameter, if the HOLD file is a relational data source, the field names and aliases from the original Master File are propagated to the HOLD Master File. The alias names become the column names in the generated relational table. The AS name also becomes the TITLE attribute in the HOLD Master File.

When you set the ASNAMES parameter to FLIP, for relational HOLD files, the field names from the original Master File or the AS names specified in the request become the alias names in the HOLD Master File as well as the column names in the generated relational table and the TITLE attributes in the HOLD Master File. The alias names in the original Master File become the field names in the HOLD Master File, except when there is an AS name, in which case the original field name becomes the HOLD field name.

- ❑ If the HOLD file is not relational, field names from the original Master File are propagated to the HOLD Master File, but alias name are not propagated, and default aliases of the form E01, E02, and so on, are generated in the HOLD Master File.
- ❑ For SET ASNAMES=FLIP, for non-relational HOLD files, the field names from the original Master File or the AS names from the request become the alias names in the HOLD Master File, and default field names are generated in the form F01, F02, and so on.
- ❑ Duplicate field names may occur in the newly created Master File as a result of truncation or the way AS phrases have been specified. In this case, refer to the fields by their aliases (E01, E02, and so forth).
- ❑ When commas are used as delimiters to break lines in the column heading, only the literal up to the first comma is used as the field name in the Master File. For example, the following produces the field name PLACE in the HOLD Master File:

```
PRINT COUNTRY AS 'PLACE,OF,ORIGIN'
```

- When ACROSS is used in a report request and the results are extracted to a HOLD file, the columns generated by the ACROSS phrase all have the same field name in the HOLD Master File. If SET ASNAMES is issued, each new column may have a unique field name. This unique field name consists of the ASNAME value specified in the request's display command, concatenated to the beginning of the value of the field used in the ACROSS phrase. If several field names have the same letters, this approach does not work.

If an AS phrase is used for the fields in the ACROSS phrase, each new column has a field name composed of the literal in the AS phrase concatenated to the beginning of the value of the first field used in the ACROSS phrase.

### **Example:** Controlling Field Names in the HOLD Master File

In the following example, SET ASNAMES=ON causes the text in the AS phrase to be used as field names in the HOLD1 Master File. The two fields in the HOLD1 Master File, NATION and AUTOMOBILE, contain the data for COUNTRY and CAR.

```
SET ASNAMES=ON
TABLE FILE CAR
PRINT CAR AS 'AUTOMOBILE'
BY COUNTRY AS 'NATION'
ON TABLE HOLD AS HOLD1
END
```

The request produces the following Master File:

```
FILE=HOLD1, SUFFIX=FIX
SEGMENT=HOLD1, SEGTYPE=S01,$
  FIELDNAME=NATION      ,ALIAS=E01      ,USAGE=A10      ,ACTUAL=A12      , $
  FIELDNAME=AUTOMOBILE  ,ALIAS=E02      ,USAGE=A16      ,ACTUAL=A16      , $
```

### **Example:** Providing Unique Field Names With SET ASNAMES

The following request generates a HOLD Master File with one unique field name for SALES and one for AVE.SALES. Both SALES and AVE.SALES would be named SALES, if SET ASNAMES had not been used.

```
SET ASNAMES=ON
TABLE FILE CAR
SUM SALES AND AVE.SALES AS 'AVERAGESALES'
BY CAR
ON TABLE HOLD AS HOLD2
END
```

The request produces the following Master File:

```
FILE=HOLD2, SUFFIX=FIX
SEGMENT=HOLD2, SEGTYPE=S01,$
  FIELDNAME=CAR ,ALIAS=E01 ,USAGE=A16 ,ACTUAL=A16 ,,$
  FIELDNAME=SALES ,ALIAS=E02 ,USAGE=I6 ,ACTUAL=I04 ,,$
  FIELDNAME=AVERAGESALES ,ALIAS=E03 ,USAGE=I6 ,ACTUAL=I04 ,,$
```

### **Example:** Using SET ASNAMES With the ACROSS Phrase

The following request produces a HOLD Master File with the literal CASH concatenated to each value of COUNTRY.

```
SET ASNAMES=ON
TABLE FILE CAR
SUM SALES AS 'CASH'
ACROSS COUNTRY
ON TABLE HOLD AS HOLD3
END
```

The request produces the following Master File:

```
FILE=HOLD3, SUFFIX=FIX
SEGMENT=HOLD3, SEGTYPE=S01,$
  FIELDNAME=CASHENGLAND ,ALIAS=E01 ,USAGE=I6 ,ACTUAL=I04 ,,$
  FIELDNAME=CASHFRANCE ,ALIAS=E02 ,USAGE=I6 ,ACTUAL=I04 ,,$
  FIELDNAME=CASHITALY ,ALIAS=E03 ,USAGE=I6 ,ACTUAL=I04 ,,$
  FIELDNAME=CASHJAPAN ,ALIAS=E04 ,USAGE=I6 ,ACTUAL=I04 ,,$
  FIELDNAME=CASHW GERMANY ,ALIAS=E05 ,USAGE=I6 ,ACTUAL=I04 ,,$
```

Without the SET ASNAMES command, every field in the HOLD FILE is named COUNTRY.

To generate field names for ACROSS values that include only the field value, use the AS phrase followed by two single quotation marks, as follows:

```
SET ASNAMES=ON
TABLE FILE CAR
SUM SALES AS ''
ACROSS COUNTRY
ON TABLE HOLD AS HOLD4
END
```

The resulting Master File looks like this:

```
FILE=HOLD4, SUFFIX=FIX
SEGMENT=HOLD4, SEGTYPE=S0,$
  FIELDNAME=ENGLAND ,ALIAS=E01 ,USAGE=I6 ,ACTUAL=I04 ,,$
  FIELDNAME=FRANCE ,ALIAS=E02 ,USAGE=I6 ,ACTUAL=I04 ,,$
  FIELDNAME=ITALY ,ALIAS=E03 ,USAGE=I6 ,ACTUAL=I04 ,,$
  FIELDNAME=JAPAN ,ALIAS=E04 ,USAGE=I6 ,ACTUAL=I04 ,,$
  FIELDNAME=W GERMANY ,ALIAS=E05 ,USAGE=I6 ,ACTUAL=I04 ,,$
```

**Example: Generating a HOLD File With SET ASNAMES=FLIP**

The following request generates a HOLD file in ALPHA format using the OFF value for SET ASNAMES. The field CURR\_SAL has the AS name SALARY in the request:

```
SET ASNAMES=OFF
TABLE FILE EMPLOYEE
SUM CURR_SAL AS SALARY PCT_INC
BY DEPARTMENT
ON TABLE HOLD FORMAT ALPHA
END
```

In the HOLD Master File, AS names have not been propagated, the field names are from the original Master File, and default alias names are generated:

```
FILENAME=HOLD      , SUFFIX=FIX      , IOTYPE=STREAM, $
SEGMENT=HOLD, SEGTYPE=S1, $
  FIELDNAME=DEPARTMENT, ALIAS=E01, USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CURR_SAL, ALIAS=E02, USAGE=D12.2M, ACTUAL=A12, $
  FIELDNAME=PCT_INC, ALIAS=E03, USAGE=F6.2, ACTUAL=A06, $
```

The following version of the request generates a relational table:

```
SET ASNAMES=OFF
TABLE FILE EMPLOYEE
SUM CURR_SAL AS SALARY PCT_INC
BY DEPARTMENT
ON TABLE HOLD FORMAT SQLMSS
END
```

The field names from the original Master File have been propagated to the field names in the HOLD Master File, and the alias names from the original Master File have been propagated to the HOLD Master File. The AS name for CURR\_SAL has become the TITLE in the HOLD Master File:

```
FILENAME=HOLD      , SUFFIX=SQLMSS      , $
SEGMENT=SEG01, SEGTYPE=S0, $
  FIELDNAME=DEPARTMENT, ALIAS=DPT, USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CURR_SAL, ALIAS=CSAL, USAGE=D12.2M, ACTUAL=D8, $
  TITLE='SALARY', $
  FIELDNAME=PCT_INC, ALIAS=PI, USAGE=F6.2, ACTUAL=F4, $
```

Changing SET ASNAMES to ON propagates the AS name SALARY to the field name in the HOLD Master File. The following is the Master File for the HOLD file in ALPHA format:

```
FILENAME=HOLD      , SUFFIX=FIX      , IOTYPE=STREAM, $
SEGMENT=HOLD, SEGTYPE=S1, $
  FIELDNAME=DEPARTMENT, ALIAS=E01, USAGE=A10, ACTUAL=A10, $
  FIELDNAME=SALARY, ALIAS=E02, USAGE=D12.2M, ACTUAL=A12, $
  FIELDNAME=PCT_INC, ALIAS=E03, USAGE=F6.2, ACTUAL=A06, $
```

The following is the Master File for the HOLD file in relational format:

```
FILENAME=HOLD      , SUFFIX=SQLMSS      , $
SEGMENT=SEG01, SEGTYPE=S0, $
  FIELDNAME=DEPARTMENT, ALIAS=DPT, USAGE=A10, ACTUAL=A10, $
  FIELDNAME=SALARY, ALIAS=CURR_SAL, USAGE=D12.2M, ACTUAL=D8,
  TITLE='SALARY', $
  FIELDNAME=PCT_INC, ALIAS=PI, USAGE=F6.2, ACTUAL=F4, $
```

Changing SET ASNAMES to FLIP propagates the AS name SALARY to the alias name in the HOLD Master File. In the ALPHA HOLD file, the other field names have been propagated to the alias names in the HOLD Master File, and default field names have been generated:

```
FILENAME=HOLD      , SUFFIX=FIX          , IOTYPE=STREAM, $
SEGMENT=HOLD, SEGTYPE=S1, $
  FIELDNAME=F01, ALIAS=DEPARTMENT, USAGE=A10, ACTUAL=A10, $
  FIELDNAME=F02, ALIAS=SALARY, USAGE=D12.2M, ACTUAL=A12, $
  FIELDNAME=F03, ALIAS=PCT_INC, USAGE=F6.2, ACTUAL=A06, $
```

In the relational HOLD file, changing SET ASNAMES to FLIP propagates the AS name SALARY to the alias name in the HOLD Master File. For that field, the field name from the original Master File becomes the field name in the HOLD Master File and the TITLE attribute. The other field names have been propagated to the alias names in the HOLD Master File, and the corresponding alias names from the original Master File have been propagated to the field names in the HOLD Master File:

```
FILENAME=HOLD      , SUFFIX=SQLMSS      , $
SEGMENT=SEG01, SEGTYPE=S0, $
  FIELDNAME=DPT, ALIAS=DEPARTMENT, USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CURR_SAL, ALIAS=SALARY, USAGE=D12.2M, ACTUAL=D8,
  TITLE='SALARY', $
  FIELDNAME=PI, ALIAS=PCT_INC, USAGE=F6.2, ACTUAL=F4, $
```

## Controlling Fields in a HOLD Master File

You can use the SET HOLDLIST command to restrict fields in HOLD Master Files to those appearing in a request.

### *Syntax:* How to Control Fields in a HOLD File

```
SET HOLDLIST = {PRINTONLY|ALL|ALLKEYS|EXPLICIT}
```

where:

#### **PRINTONLY**

Specifies that only those fields that would appear in the report are included in the generated HOLD file. Non-displaying fields in a request (those designated as NOPRINT fields explicitly or implicitly) are not included in the HOLD file.

**ALL**

Specifies that all display fields referenced in a request appear in a HOLD file, including calculated values. ALL is the default value. OLD may be used as a synonym for ALL.

**Note:** Vertical sort (BY) fields specified in the request with the NOPRINT option are not included in the HOLD file even if HOLDLIST=ALL.

**ALLKEYS**

Propagates all fields, including NOPRINTed BY fields.

The ALLKEYS setting enables caching of all of the data necessary for manipulating an active report.

**EXPLICIT**

Includes fields in the HOLD or PCHOLD file that are explicitly omitted from the report output using the NOPRINT option in the request, but does not include fields that are implicitly NOPRINTed. For example, if a field is reformatted in the request, two versions of the field exist, the one with the new format and the one with the original format, which is implicitly NOPRINTed

Note that SET HOLDLIST may also be issued from within a TABLE request. When used with MATCH, SET HOLDLIST always behaves as if HOLDLIST is set to ALL.

**Example: Using HOLDLIST=ALL**

When HOLDLIST is set to ALL, the following TABLE request produces a HOLD file containing all specified fields, including NOPRINT fields and values calculated with the COMPUTE command.

```
SET HOLDLIST=ALL

TABLE FILE CAR
PRINT CAR MODEL NOPRINT
COMPUTE TEMPSEATS=SEATS+1;
BY COUNTRY
ON TABLE HOLD
END

? HOLD
```

The output is:

```
NUMBER OF RECORDS IN TABLE=      18      LINE=      18

DEFINITION OF HOLD FILE: HOLD

FIELDNAME                                ALIAS                                FORMAT
```

COUNTRY	E01	A10
CAR	E02	A16
MODEL	E03	A24
SEATS	E04	I3
TEMPSEATS	E05	D12.2

**Example:**    **Using HOLDLIST= PRINTONLY**

When HOLDLIST is set to PRINTONLY, the following TABLE request produces a HOLD file containing only fields that would appear in report output:

```
SET HOLDLIST=PRINTONLY

TABLE FILE CAR
PRINT CAR MODEL NOPRINT
COMPUTE TEMPSEATS=SEATS+1;
BY COUNTRY
ON TABLE HOLD
END

? HOLD
```

The output is:

NUMBER OF RECORDS IN TABLE=	18	LINES=	18
DEFINITION OF HOLD FILE: HOLD			
FIELDNAME	ALIAS	FORMAT	
COUNTRY	E01	A10	
CAR	E02	A16	
TEMPSEATS	E03	D12.2	



**Example: Comparing Master Files Created Using Different HOLDLIST Settings**

The following request against the GGSALES data source has two reformatted display fields (DOLLARS, UNITS). The DOLLARS field is also an explicit NOPRINT field. The BY field named CATEGORY is also an explicit NOPRINT field:

```
SET HOLDLIST=ALL
TABLE FILE GGSALES
SUM UNITS/I5 DOLLARS/D12.2 NOPRINT
BY REGION BY CATEGORY NOPRINT
ON TABLE HOLD FORMAT FOCUS
END
```

Running the request with SET HOLDLIST=ALL generates the following HOLD Master File. Note that the DOLLARS and UNITS fields are included twice, once with the original format (which would have been implicitly NOPRINTed if the report had been printed rather than held) and once with the new format. However the NOPRINTed BY field (CATEGORY) is not included:

```
FILENAME=HOLD, SUFFIX=FOC, $
SEGMENT=SEG01, SEGTYPE=S1, $
  FIELDNAME=REGION, ALIAS=E01, USAGE=A11,
    TITLE='Region', DESCRIPTION='Region code', $
  FIELDNAME=UNITS, ALIAS=E02, USAGE=I08,
    TITLE='Unit Sales', DESCRIPTION='Number of units sold', $
  FIELDNAME=UNITS, ALIAS=E03, USAGE=I5,
    TITLE='Unit Sales', $
  FIELDNAME=DOLLARS, ALIAS=E04, USAGE=I08,
    TITLE='Dollar Sales', DESCRIPTION='Total dollar amount of reported
sales', $
  FIELDNAME=DOLLARS, ALIAS=E05, USAGE=D12.2,
    TITLE='Dollar Sales', $
```

Running the request with SET HOLDLIST=ALLKEYS generates the following HOLD Master File. Note that the DOLLARS and UNITS fields are included twice, once with the original format, which would have been implicitly NOPRINTed if the report had been printed rather than held, and once with the new format. The NOPRINTed BY field (CATEGORY) is included:

```
FILENAME=HOLD, SUFFIX=FOC      , $
  SEGMENT=SEG01, SEGTYPE=S2, $
    FIELDNAME=REGION, ALIAS=E01, USAGE=A11,
      TITLE='Region', DESCRIPTION='Region code', $
    FIELDNAME=CATEGORY, ALIAS=E02, USAGE=A11,
      TITLE='Category', DESCRIPTION='Product category', $
    FIELDNAME=UNITS, ALIAS=E03, USAGE=I08,
      TITLE='Unit Sales', DESCRIPTION='Number of units sold', $
    FIELDNAME=UNITS, ALIAS=E04, USAGE=I5,
      TITLE='Unit Sales', $
    FIELDNAME=DOLLARS, ALIAS=E05, USAGE=I08,
      TITLE='Dollar Sales', DESCRIPTION='Total dollar amount of reported
sales', $
    FIELDNAME=DOLLARS, ALIAS=E06, USAGE=D12.2,
      TITLE='Dollar Sales', $
```

Running the request with SET HOLDLIST=PRINTONLY generates the following HOLD Master File. Only the fields that would have actually printed on the report output are included: REGION and UNITS with the new format (I5). All explicitly and implicitly NOPRINTed fields are excluded, including the NOPRINTed BY field (CATEGORY):

```
FILENAME=HOLD , SUFFIX=FOC      , $
  SEGMENT=SEG01, SEGTYPE=S1, $
    FIELDNAME=REGION, ALIAS=E01, USAGE=A11,
      TITLE='Region', DESCRIPTION='Region code', $
    FIELDNAME=UNITS, ALIAS=E02, USAGE=I5,
      TITLE='Unit Sales', $
```

Running the request with SET HOLDLIST=EXPLICIT generates the following HOLD Master File. The fields that would have actually printed on the report output are included and so are the explicitly NOPRINTed fields (the display field DOLLARS and the BY field CATEGORY). The implicitly NOPRINTed fields (DOLLARS and UNITS with their original formats) are omitted:

```
FILENAME=HOLD, SUFFIX=FOC      , $
  SEGMENT=SEG01, SEGTYPE=S2, $
    FIELDNAME=REGION, ALIAS=E01, USAGE=A11,
      TITLE='Region', DESCRIPTION='Region code', $
    FIELDNAME=CATEGORY, ALIAS=E02, USAGE=A11,
      TITLE='Category', DESCRIPTION='Product category', $
    FIELDNAME=UNITS, ALIAS=E03, USAGE=I5,
      TITLE='Unit Sales', $
    FIELDNAME=DOLLARS, ALIAS=E04, USAGE=D12.2,
      TITLE='Dollar Sales', $
```

## Controlling the TITLE and ACCEPT Attributes in the HOLD Master File

The SET HOLDATTR command controls whether the TITLE and ACCEPT attributes in the original Master File are propagated to the HOLD Master File. SET HOLDATTR does not affect the way fields are named in the HOLD Master File.

Note that if a field in a data source does not have the TITLE attribute specified in the Master File, but there is an AS phrase specified for the field in a report request, the corresponding field in the HOLD file is named according to the AS phrase.

### *Syntax:* How to Control TITLE and ACCEPT Attributes

```
SET HOLDATTR =[ON|OFF|FOCUS]
```

where:

#### ON

Uses the TITLE attribute as specified in the original Master File in HOLD files in any format. The ACCEPT attribute is propagated to the HOLD Master File only for HOLD files in FOCUS format.

#### OFF

Does not use the TITLE or ACCEPT attributes from the original Master File in the HOLD Master File.

#### FOCUS

Uses the TITLE and ACCEPT attributes only for HOLD files in FOCUS format. FOCUS is the default value.

### *Example:* Controlling TITLE and ACCEPT Attributes in a HOLD Master File

In this example, the Master File for the CAR data source specifies TITLE and ACCEPT attributes:

```
FILENAME=CAR2, SUFFIX=FOC
SEGNAME=ORIGIN, SEGTYPE=S1
    FIELDNAME =COUNTRY, COUNTRY, A10, TITLE='COUNTRY OF ORIGIN',
        ACCEPT='CANADA' OR 'ENGLAND' OR 'FRANCE' OR 'ITALY' OR
            'JAPAN' OR 'W GERMANY',
        FIELDTYPE=I,$
SEGNAME=COMP, SEGTYPE=S1, PARENT=ORIGIN
    FIELDNAME=CAR, CARS, A16, TITLE='NAME OF CAR',$
.
.
.
```

Using SET HOLDATTR=FOCUS, the following request

```
SET HOLDATTR = FOCUS
TABLE FILE CAR2
PRINT CAR
BY COUNTRY ON TABLE HOLD FORMAT FOCUS AS HOLD5
END
```

produces this HOLD Master File:

```
FILE=HOLD5, SUFFIX=FOC
SEGMENT=SEG01, SEGTYPE=S02
  FIELDNAME=COUNTRY ,USAGE=E01 ,ACTUAL=A10
    TITLE='COUNTRY OF ORIGIN' ,
    ACCEPT=CANADA ENGLAND FRANCE ITALY JAPAN 'W GERMANY' , $
  FIELDNAME=FOCLIST ,USAGE=E02 ,ACTUAL=I5 , $
  FIELDNAME=CAR ,USAGE=E03 ,ACTUAL=A16 ,
    TITLE='NAME OF CAR' , $
```

## Keyed Retrieval From HOLD Files

Keyed retrieval is supported with any single-segment SUFFIX=FIX data source or HOLD file that is sorted based on the key. Keyed retrieval can reduce the IOs incurred in reading extract files, by using the SEGTYPE parameter in the Master File to identify which fields comprise the logical key for sequential files. When FIXRETRIEVE is:

- ☐ ON, the retrieval process stops when an equality or range test on the key holds true.
- ☐ OFF, all of the records from the sequential file are read and screening conditions are applied when creating the final report.

The ON TABLE HOLD command enables you to read one of the many supported data sources and create extract files. You can then join these fixed-format sequential files to other data sources to narrow your view of the data. The concept of a logical key in a fixed-format file enables qualified keyed searches for all records that match IF/WHERE tests for the first *n* KEY fields identified by the SEGTYPE attribute. Retrieval stops when the screening test detects values greater than those specified in the IF/WHERE test.

When a Master File is created for the extract file, a SEGTYPE of either *Sn* or *SHn* is added, based on the BY fields in the request. For example, PRINT *field* BY *field* creates a HOLD Master File with SEGTYPE=S1. Using BY HIGHEST *field* creates a Master with SEGTYPE=SH1.

**Syntax:**      **How to Control Keyed Retrieval for a HOLD File**

```
SET FIXRET[RIEVE] = {ON|OFF}
```

where:

ON

Enables keyed retrieval. ON is the default setting.

OFF

Disables keyed retrieval.

**Example:**      **Master File for Keyed Retrieval From a HOLD File**

The following Master File describes a fixed-format sequential file with sorted values of SEQ\_NO, in ascending order from 1 to 100,000.

```
FILE=SORTED,SUFFIX=FIX,$
SEGMAME=ONE,SEGTYPE=S1,$
  FIELD=MYKEY,MK,I8,I8,$
  FIELD=MFIELD,MF,A10,A10,$
```

```
TABLE FILE SORTED
  PRINT MFIELD
  WHERE MYKEY EQ 100
END
```

In this instance, with FIXRETRIEVE=ON, retrieval stops when MYKEY reaches 101, vastly reducing the potential number of IOs, as only 101 records are read out of a possible 100,000.

**Example:**      **Selection Criteria for Keyed Retrieval From an Extract File**

Selection criteria that include lists of equality values use keyed retrieval. For example,

```
{IF|WHERE} MYKEY EQ x OR y OR z
```

IF and WHERE tests can also include range tests. For example,

```
{IF|WHERE} MYKEY IS-FROM x TO y
```

The maximum number of vertical (BY) sort fields remains 32.

In using this feature, keep in mind that when adding unsorted records to a sorted HOLD file, records that are out of sequence are not retrieved. For example, suppose that a sorted file contains the following three records:

Key

1 1200

2 2340

3 4875

and you add the following record at the bottom of the file:

1 1620

With FIXRETRIEVE=ON, the new record with a key value of 1 is omitted, as retrieval stops as soon as a key value of 2 is encountered.

## Saving and Retrieving HOLD Files

In WebFOCUS, HOLD files are saved to a temporary directory during processing and deleted after the connection to the server is broken. If you wish to retain these files for later use, you can save the HOLD data source and its associated Master File to a specific location using APP commands.

To report against the HOLD Master File at a later time, you can add the application to your APP path, if it is not already there, or you can use a two-part name (*appname/mastername*) in the TABLE FILE command.

To report against the saved HOLD data file, you can issue a FILEDEF command that specifies where to find the file before you issue the TABLE FILE command.

### **Syntax:** How to Specify a Storage Location for a HOLD Master File (Windows, UNIX, OpenVMS)

```
APP MAP appname path_to_directory APP HOLDDATA appname APP HOLDMETA  
appname
```

where:

APP MAP

Associates a directory location with an application name.

APP HOLDDATA

Specifies the application name for HOLD data files.

APP HOLDMETA

Specifies the application name for HOLD Master Files.

*path\_to\_directory*

Specifies the directory in which to store the files.

*appname*

Is an application name.

You should add the APP commands to a supported profile so that they will be available to all of your procedures rather than issuing them in every procedure that needs to access the application directory. In addition, you should add the application to your path so that it will be found by your procedures.

### **Example:** Specifying a Storage Location for HOLD Data and Master Files

The following example for WebFOCUS on UNIX illustrates how to create an application named *holdapp* and store HOLD data files and Master Files in the `\ggtmp` directory that is mapped to that application.

Place the following commands in any supported profile:

```
APP MAP holdapp /ggtmp
APP HOLDDATA holdapp
APP HOLDMETA holdapp
```

The following request creates a HOLD Master file named `sales.mas` and a HOLD data file named `sales.ftm` that will be stored in the `/ggtmp` directory:

```
TABLE FILE GGSales
PRINT SEQ_NO CATEGORY PRODUCT
ON TABLE HOLD AS SALES
END
```

To issue a TABLE request against the HOLD file, you must first issue a FILEDEF command that points to the HOLD data file. The DDNAME for the FILEDEF is the AS name specified in the HOLD command. If no AS name was specified, the DDNAME is HOLD:

```
FILEDEF SALES DIR \ggtmp\sales.ftm
```

You must also make sure that WebFOCUS can find the Master File. One way to do this is to make sure that the application is in your application path. You can add it to the path with the following command:

```
APP APPENDPATH holdapp
```

Then you can issue a request against the HOLD file:

```
TABLE FILE SALES
PRINT *
END
```

Alternatively, you can issue the TABLE request against the HOLD file using a two-part name (application name and Master File name):

```
TABLE FILE holdapp/sales
PRINT *
END
```

### **Reference:** Allocating HOLD Files on z/OS

The HOLD file is dynamically allocated if it is not currently allocated on z/OS. This means the system may delete the file at the end of the session, even if you have not. Since HOLD files are usually deleted, this is a desired default. However, if you want to save the file, we recommend that you allocate the HOLD Master File to the ddname HOLDMAST as a permanent data set, and allocate the HOLD data file to a permanent data set as well. The allocations can be performed within the standard Reporting Server CLIST or batch JCL or in a profile or procedure. For example, if your procedure had the following command:

```
ON TABLE HOLD AS SALES
```

you could use allocations similar to the following:

```
ALLOC F(HOLDMAST) DA('qualif.HOLDMAST.DATA') SHR REUSE
ALLOC F(SALES) DA('qualif.SALES.DATA') SHR REUSE
```

Note that ddname HOLDMAST must not refer to the same PDS referred to by the MASTER and FOCEXEC ddnames.

## Using DBMS Temporary Tables as HOLD Files

You can create a report output file (that is, a HOLD file), as a native DBMS temporary table. This increases performance by keeping the entire reporting operation on the DBMS server, instead of downloading data to your computer and then back to the DBMS server.

For example, if you temporarily store report output for immediate use by another procedure, storing it as a temporary table instead of creating a standard HOLD file avoids the overhead of transmitting the interim data to your computer.

The temporary table columns are created from the following report elements

- ☐ Display columns
- ☐ Sort (BY) columns
- ☐ COMPUTE columns

except for those for which NOPRINT is specified.



The temporary table that you create from your report will be the same data source type (that is, the same DBMS) as the data source from which you reported. If the data source from which you reported contains multiple tables, all must be of the same data source type and reside on the same instance of the DBMS server.

You can choose between several types of table persistence.

You can create extract files as native DBMS tables with the following adapters:

- ☐ Db2 (on z/OS, UNIX, and Windows)
- ☐ Informix
- ☐ Microsoft SQL Server
- ☐ MySQL
- ☐ Oracle
- ☐ Teradata

### **Syntax:** How to Save Report Output as a Native Temporary Table Using Commands

The syntax to save report output as a native DBMS temporary table is

```
ON TABLE HOLD [AS filename] FORMAT SAME_DB [PERSISTENCE persistValue]
```

where:

*filename*

Specifies the name of the HOLD file. If you omit AS *filename*, the name of the temporary table defaults to "HOLD".

Because each subsequent HOLD command overwrites the previous HOLD file, it is recommended to specify a name in each request to direct the extracted data to a separate file, thereby preventing an earlier file from being overwritten by a later one.

PERSISTENCE

Specifies the type of table persistence and related table properties. This is optional for DBMSs that support volatile tables, and required otherwise. For information about support for volatile tables for a particular DBMS, see [Temporary Table Properties for SAME\\_DB Persistence Values](#) on page 503, and consult your DBMS vendor documentation.

*[persistValue](#)*

Is one of the following:

[VOLATILE](#)

Specifies that the table is local to the DBMS session. A temporary synonym (a Master File and Access File), is generated automatically. It expires when the server session ends.

This is the default persistence setting for all DBMSs that support volatile tables.

For information about support for the volatile setting, and about persistence and other table properties, for a particular DBMS, see [Temporary Table Properties for SAME\\_DB Persistence Values](#) on page 503, and consult your DBMS vendor documentation.

[GLOBAL\\_TEMPORARY](#)

Specifies that while the table exists, its definition will be visible to other database sessions and users though its data will not be. A permanent synonym (a Master File and Access File), is generated automatically.

For information about support for the global temporary setting, and about persistence and other table properties, for a particular DBMS, see [Temporary Table Properties for SAME\\_DB Persistence Values](#) on page 503, and consult your DBMS vendor documentation.

[PERMANENT](#)

Specifies that a regular permanent table will be created. A permanent synonym (a Master File and Access File), is generated automatically.

**Reference: Temporary Table Properties for SAME\_DB Persistence Values**

The following chart provides additional detail about persistence and other properties of temporary tables of different data source types that are supported for use with HOLD format SAME\_DB.

DBMS	VOLATILE	GLOBAL_TEMPORARY
Db2	<p>Db2 on Linux, UNIX, and Windows, and Db2 for z/OS: a volatile table is created using the DECLARE GLOBAL TEMPORARY TABLE command with the ON COMMIT PRESERVE ROWS option. Declared global temporary tables persist and are visible only within the current session (connection). SESSION is the schema name for all declared global temporary tables.</p>	<p>Db2 on Linux, UNIX, and Windows, and Db2 for z/OS: a global temporary table is created using the CREATE GLOBAL TEMPORARY TABLE command. The definition of a created global temporary table is visible to other sessions, but the data is not. The data is deleted at the end of each transaction (COMMIT or ROLLBACK command). The table definition persists after the session ends.</p> <p>Global tables require the following setting to be in effect:</p> <p><code>ENGINE DB2 SET AUTOCOMMIT ON FIN</code></p> <p>For information on creating user-defined tablespaces on Linux, UNIX, and Windows for volatile and global temporary tables, see the <i>Adapter Administration</i> manual.</p>
Informix	<p>A volatile table is created using the CREATE TEMP TABLE command with the WITH NO LOG option. The definition and the data persist, and are visible, only within the current session.</p>	<p>This type of table is not supported by Informix.</p>

DBMS	VOLATILE	GLOBAL_TEMPORARY
Microsoft SQL Server	A volatile table is created as a local temporary table whose name is prefixed with a single number sign (#). Therefore, the name of a volatile table used as a HOLD file is the name specified by the HOLD phrase, prefixed with a number sign (#). The table definition and the data persist, and are visible, only within the current session.	The name of a global temporary table is prefixed with two number signs (##). Therefore, the name of a global temporary table used as a HOLD file is the name specified by the HOLD phrase, prefixed with two number signs (##). The table is dropped automatically when the session that created the table ends and all other tasks have stopped referencing it. The table definition and data are visible to other sessions.
MySQL	A volatile table is created using the CREATE TEMPORARY TABLE command. A temporary table persists and is visible only within the current session (connection). If a temporary table has the same name as a permanent table, the permanent table becomes invisible.	This type of table is not supported by MySQL.
Oracle	This type of table is not supported by Oracle.	The table definition is visible to all sessions. Its data is visible only to the session that inserts data into it. The table definition persists for the same period as the definition of a regular table.

DBMS	VOLATILE	GLOBAL_TEMPORARY
Teradata	A volatile table definition and data are visible only within the session that created the table and inserted the data. The volatile table is created with the ON COMMIT PRESERVE ROWS option.	A global temporary table persists for the same duration as a permanent table. The definition is visible to all sessions, but the data is visible only to the session that inserted the data. The global temporary table is created with the ON COMMIT PRESERVE ROWS option.

## Column Names in the HOLD File

Each HOLD file column is assigned its name:

1. From the AS name specified for the column in the report request.
2. If there is no AS name specified, the name is assigned from the alias specified in the synonym. (The alias is identical to the column name in the original relational table.)
3. In all other cases, the name is assigned from the field name as it is specified in the synonym.

## Primary Keys and Indexes in the HOLD File

A primary key or an index is created for the HOLD table. The key or index definition is generated from the sort (BY) keys of the TABLE command, except for undisplayed sort keys (that is, sort keys for which NOPRINT is specified). To determine whether a primary key or an index will be created:

1. If these sort keys provide uniqueness and do not allow nulls (that is, if in the synonym, the MISSING attribute column is unselected or OFF), and if the DBMS supports primary keys on the type of table being created, a primary key is created.
2. If these sort keys provide uniqueness but either
  - a. some of the columns allow nulls.
  - b. the DBMS does not support primary keys on the type of table being created then a unique index is created.
3. If these sort keys do not provide uniqueness, a non-unique index is created.
4. If there are no displayed sort keys (that is, no sort keys for which NOPRINT has not been specified), no primary key or index is created.

## Creating SAVE and SAVB Files

The SAVE command, by default, captures report output in ALPHA format as a simple sequential data source, without headings or subtotals. However, you can specify a variety of other formats for SAVE files, which are compatible with many software products. For example, you can specify SAVE formats to display report output in a webpage, a text document, a spreadsheet or word processing application, or to be used as input to other programming languages. For a list of supported formats, see [Choosing Output File Formats](#) on page 511.

Regardless of format, the SAVE command does not create a Master File.

The SAVB command is a variation on the SAVE command. SAVB creates a data source without a Master File, but numeric fields are stored in BINARY format. You can use the SAVB file as input to a variety of applications. SAVB output is the same as the default output created by the HOLD command.

### **Syntax:** How to Create a SAVE File

From a report request, use

```
ON TABLE SAVE [AS filename] [FORMAT fmt] [MISSING {ON|OFF}]
```

or

```
save_field SAVE [AS filename] [FORMAT fmt] [MISSING {ON|OFF}]
```

where:

*save\_field*

Is the name of the last field in the request, excluding BY or ACROSS fields.

AS *filename*

Specifies a name for the SAVE file. If you do not specify a file name, SAVE is used as the default. Since each subsequent SAVE command overwrites the previous SAVE file, it is advisable to code a distinct file name in each request to direct the extracted data to a separate file, thereby preventing it from being overwritten by the next SAVE command.

You can also include a path, enclosed in single quotation marks, indicating where to store the SAVE file. For example:

```
ON TABLE SAVE FILENAME 'install_dir:\dir\filename.ext' FORMAT fmt
```

FORMAT *fmt*

Specifies the format of the SAVE file. ALPHA is the default format.

- ☐ To display as or in a webpage:

HTML, HTMTABLE, DHTML

- ☐ To use in a text document:

ALPHA, DOC, PDF, WP, Text

- ☐ To use in a spreadsheet application:

DIF, EXCEL, EXL2K, LOTUS, (WK1), SYLK

- ☐ To use in a database application:

COMMA, COM, COMT

For details about all available formats, see [Choosing Output File Formats](#) on page 511.

#### MISSING

Ensures that fields with the MISSING attribute set to ON are carried over into the SAVE file. MISSING OFF is the default attribute. See [Handling Records With Missing Field Values](#) on page 971.

### **Example:** Creating a SAVE File

The following request extracts data from the EMPLOYEE data source and creates a SAVE file.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME AND FIRST_NAME
BY DEPARTMENT
ON TABLE SAVE
END
```

A description of the ALPHA (default SAVE format) file layout appears after the records are retrieved.

The output is:

<b>NUMBER OF RECORDS IN TABLE=</b>	<b>12</b>	<b>LINES=</b>	<b>12</b>
<b>ALPHANUMERIC RECORD NAMED SAVE</b>			
<b>FIELDNAME</b>	<b>ALIAS</b>	<b>FORMAT</b>	<b>LENGTH</b>
<b>DEPARTMENT</b>	<b>DPT</b>	<b>A10</b>	<b>10</b>
<b>LAST_NAME</b>	<b>LN</b>	<b>A15</b>	<b>15</b>
<b>FIRST_NAME</b>	<b>FN</b>	<b>A10</b>	<b>10</b>
<b>TOTAL</b>			<b>35</b>

**Syntax:**      **How to Create a SAVB File**

From a request, use

```
ON TABLE SAVB [AS filename] [MISSING {ON|OFF}]
```

or

```
save_field SAVB [AS filename] [MISSING {ON|OFF}]
```

where:

*save\_field*

Is the name of the last field in the request, excluding BY and ACROSS fields.

*AS filename*

Specifies a name for the SAVB file. If you do not specify a file name, SAVB is used as the default. Since each subsequent SAVB command overwrites the previous SAVB file, it is advisable to code a distinct file name in each request to direct the extracted data to a separate file, thereby preventing it from being overwritten by the next SAVB command.

You can also include a path, enclosed in single quotation marks, indicating where you wish to store the SAVB file. For example:

```
ON TABLE SAVB FILENAME 'c:\dir\filename.ext '
```

**MISSING**

Ensures that fields with the MISSING attribute set to ON are carried over into the SAVB file. The default is MISSING OFF. See [Handling Records With Missing Field Values](#) on page 971.

**Example:**      **Creating a SAVB File**

The following request extracts data from the SALES data source and creates a SAVB file.

```
TABLE FILE SALES
PRINT PROD_CODE AND AREA
BY DATE
WHERE CITY IS 'STAMFORD' OR 'UNIONDALE'
ON TABLE SAVB
END
```

A description of the BINARY file is appears after the records are retrieved.



The output is:

<b>NUMBER OF RECORDS IN TABLE=</b>		<b>10</b>	<b>LINES=</b>	<b>10</b>
 <b>INTERNAL RECORD NAMED SAVB</b>				
<b>FIELDNAME</b>	<b>ALIAS</b>	<b>FORMAT</b>	<b>LENGTH</b>	
<b>DATE</b>	<b>DTE</b>	<b>A4MD</b>	<b>4</b>	
<b>PROD_CODE</b>	<b>PCODE</b>	<b>A3</b>	<b>4</b>	
<b>AREA</b>	<b>LOC</b>	<b>A1</b>	<b>4</b>	
<b>TOTAL</b>			<b>12</b>	

## Creating a PCHOLD File

The PCHOLD command enables you to extract data from the WebFOCUS Reporting Server by way of the WebFOCUS client, and automatically display the data in HTML format in your browser.

In addition, if you have established a helper application, you can use the command ON TABLE PCHOLD to display the data in the helper application's viewer. For example, if a procedure contains the ON TABLE PCHOLD FORMAT EXCEL command, data is not returned to the browser in HTML format. Instead, data is returned and imported into an Excel spreadsheet, or other spreadsheet program you specify to your browser.

In contrast, when data access is handled directly by the Reporting Server (without intervention by the WebFOCUS Client), then the data is extracted to a PCHOLD file and automatically delivered to your PC for local reporting.

**Note:** If your environment supports the SET parameter SAVEMATRIX, you can preserve the internal matrix of your last report in order to keep it available for subsequent HOLD, SAVE, and SAVB commands when the request is followed by Dialogue Manager commands. For details on SAVEMATRIX, see the *Developing Reporting Applications* manual.

### **Syntax:** How to Create a PCHOLD File

The syntax for PCHOLD in a report request is

```
ON TABLE {PCHOLD|HOLD AT CLIENT} [AS filename] [FORMAT fmt]
```

where:

**PCHOLD|HOLD AT CLIENT**

Enables you to extract and automatically display data in HTML format in your browser.

HOLD AT CLIENT is a synonym for PCHOLD. The PCHOLD command does not have a default format. You must specify a format when using PCHOLD. The output is saved

with a Master File. For details about the behavior of PCHOLD, see [Creating a HOLD File](#) on page 473.

If you specify an ON TABLE PCHOLD command without a FORMAT, XML/HTML code is returned to the browser.

**AS** *filename*

Specifies a name for the PCHOLD file. If you do not specify a file name, PCHOLD becomes the default. Since each subsequent PCHOLD command overwrites the previous PCHOLD file, it is advisable to code a distinct file name in each request to direct the extracted data to a separate file, thereby preventing it from being overwritten by the next PCHOLD command.

**FORMAT** *fmt*

Specifies the format of the PCHOLD file.

☐ To display as or in a webpage, choose:

[HTML](#), [HTMTABLE](#), [DHTML](#), [VISDIS](#)

☐ To display as a printed document, choose:

[PDF](#), [PS](#)

☐ To use in a text document, choose:

[ALPHA](#), [DOC](#), [WP](#)

☐ To use in a spreadsheet application, choose:

[DIF](#), [XLSX](#), [EXL2K \[PIVOT\]](#), [LOTUS](#)

☐ To use for additional reporting, choose:

[ALPHA](#), [DFIX](#), [COM](#), [COMT](#), [TAB](#), [TABT](#)

☐ To use in another application choose:

[XML](#)

☐ To generate active output, choose:

[AHTML](#), [APDF](#), [AFLEX](#)

For details about all available formats, see [Choosing Output File Formats](#) on page 511.

## Choosing Output File Formats

You can select from a wide range of output formats to preserve your report output for use in any of the following ways:

- ☐ To display as or in a webpage, as a printed document, or in a text document.
- ☐ To process in another application, such as a spreadsheet, a database, a word processor, or a 3GL program.
- ☐ To send to another location, such as a browser or PC.
- ☐ To extract a subset of the original data source in order to generate multi-step reports.

For details on each of the supported formats, including the commands that support them (HOLD, PCHOLD, SAVE) and the operating environments in which they are available, see the reference topics for the following formats.

AHTML	DFIX	GIF	POSTSCRIPT	SQLORA
AHTMLTAB	DHTML	HTML	(PS)	SQLPSTGR
ALPHA	DIF	HTMTABLE	PPT	SQLSYB
APDF	DOC	INGRES	PPTX	SYLK
BINARY	EXCEL	INTERNAL	REDBRICK	TAB
COMMA	EXL2K	JPEG	SQL	TABT
COM	EXL2K	JSCHART	SQL_SCRIPT	VISDIS
COMT	FORMULA	LOTUS	SQLDBC	WK1
DATREC	EXL2K PIVOT	PDF	SQLINF	WP
DB2	EXL97		SQLMAC	XFOCUS
DBASE	FLEX		SQLMSS	XLSX
	FOCUS		SQLODBC	

**Tip:** To ensure that you will be able to open a format type in your browser, see [Specifying MIME Types for WebFOCUS Reports](#) on page 561.

**Reference:** FORMAT AHTML

**Description:** Saves report output as an active report (HTML file that can be used for offline analysis and interactive functions without any connection to a server). All of the data and JavaScript code are stored within the HTML file, which also makes the output highly compressible for email and transparent to security systems. For more information about active reports, see the *Active Technologies User's Guide*.

**Use:** For offline analysis of data.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

**Available in:** WebFOCUS, FOCUS, App Studio.

**Reference:** FORMAT AHTMLTAB

**Description:** Creates an output file that contains only data and parameters used in an HTML active report. The output produced is not a complete HTML active report. However, the file can be included in another HTML document using the Dialogue Manager command -HTMLFORM. For details, see the documentation on Dialogue Manager in the *Developing Reporting Applications* manual.

**Note:** When issuing HOLD AS *name* FORMAT AHTMLTAB to embed an HTML active report into another HTML document, you must include Active Technologies JavaScript code into the HTML BODY using:

```
<BODY>  
!IBI.OBJ.ACTIVEREPORTJS;
```

**Use:** For embedding HTML active reports into an existing HTML document.

**Supported with the commands:** HOLD, SAVE.

**Available in:** WebFOCUS, App Studio.

**Reference:** FORMAT ALPHA

**Description:** Saves report output as fixed-format character data and can be created as a HOLD file.

ALPHA is the default SAVE format. The output file contains data only.

Text fields are supported in ALPHA-formatted files. See [Using Text Fields in Output Files](#) on page 535.

To control missing data characters that are propagated to fields with the MISSING=ON attribute, use the SET HNODATA command. For more information, see the *Developing Reporting Applications* manual.

**Use:** For display in a text document. For further reporting in FOCUS, WebFOCUS, or App Studio. As a transaction file for modifying a data source.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** **FORMAT APDF**

**Description:** Saves report output as an Adobe® Flex® report (Adobe® Flash® file) embedded in a PDF file. The resulting PDF active report can be used for offline analysis and interactive functions without any connection to a server.

In order to run a report created using the APDF format, Adobe Reader 9 or higher is required so the Flash run-time code included in the Acrobat client can render the Flash content.

**Use:** For offline analysis of data.

**Supported with the command:** HOLD, PCHOLD, SAVE.

**Available in:** FOCUS, WebFOCUS, App Studio, InfoAssist.

**Reference:** **FORMAT BINARY**

**Description:** Saves report data and stores numeric fields as binary numbers. When created as a HOLD file, also creates a Master File.

BINARY is the default format for HOLD files. When created in BINARY format:

- ❑ The HOLD file is a sequential single-segment data source. The HOLD Master File is a subset of the original Master File, and may also contain fields that have been created using the COMPUTE or DEFINE commands or generated in an ACROSS phrase.
- ❑ By default, fields with format I remain four-byte binary integers. Format F fields remain in four-byte floating-point format. Format D fields remain in eight-byte double-precision floating-point, and format P fields remain in packed decimal notation and occupy eight bytes (for fields less than or equal to eight bytes long) or 16 bytes (for packed decimal fields longer than eight bytes). Alphanumeric fields (format A) are stored in character format.

Every data field in the sequential extract record is aligned on the start of a full four-byte word. Therefore, if the format is A1, the field is padded with three bytes of blanks on the right. This alignment makes it easier for user-coded subroutines to process these data fields. (Under some circumstances, you may wish to prevent the padding of integer and packed decimal fields. Do so with HOLD FORMAT INTERNAL. See [Saving Report Output in INTERNAL Format](#) on page 546.)

The output file contains data only.

**Use:** For further reporting in FOCUS, WebFOCUS, or App Studio. As a transaction file for modifying a data source.

**Supported with the commands:** HOLD

**Available in:** WebFOCUS, App Studio, FOCUS.

### **Reference:** FORMAT COMMA

**Description:** Saves the data values as a variable-length text file, with fields separated by commas and with character values enclosed in double quotation marks. All blanks within fields are retained. This format is the industry standard comma-delimited format.

This format does not have the safety feature of the double quote added within a text field containing a double quote.

The extension for this format is PRN. This format type does not create a Master File.

**Note:**

- ☐ Smart date fields and dates formatted as I or P fields with date format options are treated as numeric, and are not enclosed in double quotation marks in the output file. Dates formatted as alphanumeric fields with date format options are treated as alphanumeric, and enclosed in double quotation marks.
- ☐ Continental decimal notation (CDN=ON|SPACE|QUOTE|QUOTEP) is not supported. A comma within a number is interpreted as two separate columns by a destination application such as Microsoft Access.

The output file contains data only.

**Use:** For further processing in a database application. This format type can be imported into applications such as Excel or Lotus.

**Supported with the commands:** HOLD, SAVE.

**Available in:** WebFOCUS, App Studio, FOCUS.

### **Reference:** FORMAT COM

**Description:** Saves the data values as a variable-length text file with fields separated by commas and with character values enclosed in double quotation marks. Leading blanks are removed from numeric fields, and trailing blanks are removed from character fields. To issue a request against this data source, the setting PCOMMA=ON is required.

This format also includes a built-in safety feature, which allows embedded quotes within character fields. A second double quote (") is inserted adjacent to the existing one. For example, if you input Joe "Smitty" Smith, the output is Joe ""Smitty"" Smith.

The extension for this format is CSV. A Master File is created for this format type when the command used to create the output file is HOLD. The SUFFIX in the generated Master File is COM.

**Note:**

- ❑ Smart date fields and dates formatted as I or P fields with date format options are treated as numeric, and are not enclosed in double quotation marks in the output file. Dates formatted as alphanumeric fields with date format options are treated as alphanumeric, and enclosed in double quotation marks.
- ❑ Continental decimal notation (CDN=ON|SPACE|QUOTE|QUOTEP) is not supported. A comma within a number is interpreted as two separate columns by a destination application such as Microsoft Access.
- ❑ To create a variable-length comma- or tab-delimited HOLD file that differentiates between a missing value and a blank string or zero value, use the SET NULL=ON command. For more information, see the *Developing Reporting Applications* manual.

**Use:** For further processing in a database application. This format type can be imported into applications such as Excel or Lotus.

**Supported with the commands:** HOLD, SAVE, PCHOLD.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** FORMAT COMT

**Description:** Saves the column headings in the first row of the output file. It produces a variable-length text file with fields separated by commas, and with character values enclosed in double quotation marks. Leading blanks are removed from numeric fields, and trailing blanks are removed from character fields. This format is required by certain software packages such as Microsoft Access.

This format also includes a built-in safety feature, which allows embedded quotes within character fields. A second double quote (") is inserted adjacent to the existing one. For example, if you input Joe "Smitty" Smith, the output is Joe ""Smitty"" Smith.

The extension for this format is CSV. A Master File is created for this format type when the command used to create the output file is HOLD. The SUFFIX in the generated Master File is COMT.

**Note:**

- ❑ Smart date fields and dates formatted as I or P fields with date format options are treated as numeric, and are not enclosed in double quotation marks in the output file. Dates formatted as alphanumeric fields with date format options are treated as alphanumeric, and enclosed in double quotation marks.
- ❑ Continental decimal notation (CDN=ON|SPACE|QUOTE|QUOTEP) is not supported. A comma within a number is interpreted as two separate columns by a destination application such as Microsoft Access.
- ❑ To create a variable-length comma- or tab-delimited HOLD file that differentiates between a missing value and a blank string or zero value, use the SET NULL=ON command. For more information, see the *Developing Reporting Applications* manual.

**Use:** For further processing in a database application. This format type can be imported into applications such as Excel or Lotus.

**Supported with the commands:** HOLD, SAVE, PCHOLD.

**Available in:** FOCUS, App Studio, WebFOCUS.

**Reference:** **FORMAT DATREC**

**Description:** Saves report output as a sequential file with a Master File, and stores numeric fields as binary numbers without aligning them on fullword boundaries. The last field consists of one byte for each of the other fields in the Master File that indicates whether the corresponding field is missing.

**Use:** For further reporting in FOCUS, WebFOCUS, or App Studio. As a transaction file for modifying a data source.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** **FORMAT DB2**

**Description:** Creates a Db2 table, if you have the Adapter for Db2 and permission to create tables.

**Use:** For further processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio, FOCUS.



**Reference: FORMAT DBASE**

**Description:** Creates an extract file in dBase format that includes column headings in the first row.

**Note:** Blank field names display as blank column titles. This may result in an error when attempting to use the file as input to various applications.

**Use:** For importing data to Windows-based applications such as MS Access and Excel.

**Supported with the command:** HOLD.

**Available in:** App Studio.

**Reference: FORMAT DFIX**

**Description:** Creates a delimited output file. You can specify the delimiter, whether alphanumeric fields should be enclosed within a special character such as a double quotation mark, and whether the file should be generated with a header record containing the field names.

For more information, see [Creating a Delimited Sequential File](#) on page 537.

**Use:** For importing data to Windows-based applications such as MS Access and Excel.

**Supported with the command:** HOLD, PCHOLD.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference: FORMAT DHTML**

**Description:** Provides HTML output that has most of the features normally associated with output formatted for printing such as PDF or PostScript output. You can create an HTML file (.htm) or a Web Archive file (.mht). The type of output file produced is controlled by the value of the HTMLARCHIVE parameter. HTMLARCHIVE=ON creates a Web Archive file.

Some of the features supported by format DHTML are:

- ☐ **Absolute positioning.** DHTML precisely places text and images inside an HTML report, allowing you to use the same StyleSheet syntax to lay out HTML as you use for PDF or PS output.
- ☐ **On demand paging.** On demand paging is available with SET HTMLARCHIVE=OFF.
- ☐ **PDF StyleSheet features.** For example, the following features are supported: grids, background colors, OVER, bursting, coordinated compound reports.

**Note:**

- ☐ The font map file for DHTML reports is dhtml.fmp.
- ☐ Legacy compound reports are not supported.
- ☐ Drillthrough reports are not supported.

**Use:** For display as a webpage.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** **FORMAT DIF**

**Description:** Captures the entire report output, excluding headings, footings, subheads, and subfoots, and creates a character file that can be easily incorporated into most spreadsheet packages.

For example, running a TABLE request with HEADING/FOOTING and ON TABLE PCHOLD FORMAT DIF does not display the report output with headings and footings. As a workaround, use another format (such as HTML, PDF, or EXL2K).

**Note:** Microsoft Excel SR-1 is no longer supported for HOLD FORMAT DIF. To open these reports, use either Microsoft Excel SR-2 or Microsoft Excel 2000.

**Use:** For display or processing in a spreadsheet application.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

The PCHOLD variation transfers the data from a web server to a browser.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** **FORMAT DOC**

**Description:** Captures the entire report output, including headings, footings, and subtotals, and creates a text file with layout and line breaks that can be easily incorporated into most word processing packages. DOC format uses a form-feed character to indicate page control information.

**Note:** A request that contains ON TABLE PCHOLD FORMAT DOC results in a blank first page in the report when displayed in Microsoft XP Office. To eliminate this, include SET PAGE=NOPAGE in your request.

**Use:** For display in a text document.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

The PCHOLD variation transfers the data from a web server to a browser.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** **FORMAT EXCEL**

**Description:** Captures report output as a Microsoft Excel spreadsheet file, including data and column titles, but without report headings, footings, subheadings, or subfootings. If the report request contains an ACROSS phrase and specifies FORMAT EXCEL, column titles are not included in the output. If you wish to have the ACROSS column titles appear, use HOLD/PCHOLD FORMAT EXL2K.

Text and varchar (AnV) fields are not supported with FORMAT EXCEL. To include them, use HOLD FORMAT EXL2K. To use FORMAT EXL2K, you must have Excel 2000 installed.

Leading zeros do not appear for FORMAT EXCEL.

Extended currency symbols are not supported with the FORMAT EXCEL output option.

Since only single-line (single-cell) column titles are supported in format EXCEL reports, any additional column title rows are treated as data. For example, if you have a report with a multi-line (multi-cell) column title and you sort the column, the second (and so on) column title rows are sorted with the data. To avoid this, only select the data instead of the entire column when you select sorting options in Excel.

**Note:**

- ☐ Microsoft Excel SR-1 is no longer supported for HOLD FORMAT EXCEL. To open these reports, use either Microsoft Excel SR-2 or Microsoft Excel 2000.
- ☐ You can only have one drill-down link per EXCEL spreadsheet cell. HTML reports can have multiple drill-down links within a HEADING. However, if you change the output format to EXCEL, the HEADING becomes one cell in the spreadsheet. WebFOCUS does not support multiple drill-downs in a cell.

**Use:** For display or processing in a spreadsheet application.

**Supported with the commands:** HOLD, SAVE.

**Available in:** WebFOCUS, App Studio, FOCUS.

In FOCUS, the recommended transfer mechanism is FTP in binary mode. On a PC, the extension of the resulting file should be .xls.

**Note:** If drill-downs are included in a WebFOCUS report in Excel format, and the drill-down reports fail to open, you must check the "Browse in same window" option for XLS file types. If this setting is not selected, a WebFOCUS report in Excel format will be unable to pass cookies to drill-down reports, and the reports will fail to open. To check the "Browse in same window" option for XLS file types, perform the following steps:

1. Access Windows Explorer.
2. From the Tools menu, click *Folder Options*.
3. Select the *File Types* tab.
4. Select the XLS extension and then click *Advanced*.
5. Select the *Browse in same window* check box and then click *OK*.
6. Click *Close* to exit the Folder Options window.

### **Reference:** FORMAT XLSX

**Description:** Generates fully styled reports in Excel 2007 XML format. You must have Excel 2007 or higher installed to use this output format. For details, see [Choosing a Display Format](#) on page 565.

An Excel 2007 worksheet can contain 1,048,576 rows by 16,384 columns. WebFOCUS will generate worksheets larger than these defined limits, but Excel 2007 will have difficulty opening the resulting workbook, and the data that exceeds the Excel 2007 limits will be truncated.

For information about creating overflow worksheets for the additional rows when the number of rows becomes too high for a single worksheet, see [Overcoming the Excel 2007/2010 Row Limit Using Overflow Worksheets](#) on page 662.

**Use:** For display or processing in a spreadsheet application.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

For Internet Explorer, the PCHOLD variation launches Excel 2007 in the browser. For details, and for information about working with XLSX files, see [Choosing a Display Format](#) on page 565.

**Available in:** WebFOCUS, App Studio.

### **Reference:** FORMAT EXL2K

**Description:** Generates fully styled reports in Excel 2000 HTML format. You must have Excel 2000 installed to use this output format. For details, see [Choosing a Display Format](#) on page 565.

ACROSS column titles are supported for EXL2K output format.

For EXL2K output format, a report can include 65,536 rows and/or 256 columns. Rows and columns in excess of these limits are truncated from the worksheet when opened in Excel.

For information about creating overflow worksheets for the additional rows when the number of rows becomes too high for a single worksheet, see [Overcoming the Excel 2007/2010 Row Limit Using Overflow Worksheets](#) on page 662.

**Use:** For display or processing in a spreadsheet application.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

For Internet Explorer, the PCHOLD variation launches Excel 2000 in the browser. For details, and for information about working with EXL2K files, see [Choosing a Display Format](#) on page 565.

**Available in:** WebFOCUS, App Studio.

#### **Reference:** FORMAT EXL2K FORMULA

**Description:** Specifies that the report will be displayed as an Excel 2000 spreadsheet, with WebFOCUS totals and other calculated values translated to active Excel formulas. For details, see [Choosing a Display Format](#) on page 565.

**Use:** For display or processing in a spreadsheet application.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

**Available in:** WebFOCUS, FOCUS, App Studio.

#### **Reference:** FORMAT EXL2K PIVOT

**Description:** Generates fully styled reports in Excel 2000 HTML format, with added pivoting capabilities. Requires Excel 2000 on your PC. For details, see [Choosing a Display Format](#) on page 565.

**Use:** For display or processing in a spreadsheet application.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

For Internet Explorer, the PCHOLD variation transfers the data from a web server to a browser, which launches Excel 2000.

**Available in:** WebFOCUS, FOCUS, App Studio.

**Reference:** FORMAT EXL97

**Description:** Enables you to view and save reports in Excel 97 that include full styling. For details on working with Excel formats, see [Choosing a Display Format](#) on page 565.

Leading zeros do not display for FORMAT EXL97.

**Use:** For display or processing in a spreadsheet application.

**Supported with the command:** HOLD, PCHOLD, SAVE.

**Available in:** FOCUS, WebFOCUS, App Studio.

**Reference:** FORMAT FLEX

**Description:** Saves report output as an active report in an Adobe Flash player compatible (.swf) file that can be used for offline analysis and interactive functions without any connection to a server. Internet Explorer, Mozilla Firefox, and Opera internet browsers recognize an active report in the Adobe Flex format as a Shockwave Flash Object.

In order to run a report created using the Active Technologies for Adobe Flash player (FLEX) format, Adobe Flash Player 9.0.28 or higher is required.

**Use:** For offline analysis of data.

**Supported with the command:** HOLD, SAVE.

**Available in:** FOCUS, WebFOCUS, App Studio.

**Reference:** FORMAT FOCUS

**Description:** Creates a FOCUS data source. Four files result: a HOLD data file, a HOLD Master File, and two work files. See [Holding Report Output in FOCUS Format](#) on page 479.

Text fields are supported for FOCUS output files. See [Using Text Fields in Output Files](#) on page 535.

**Use:** For further processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** FORMAT GIF

**Description:** Saves the output of a graph request as a GIF file.

**Use:** The resulting GIF file can be embedded in the heading/footer or body of a PDF or HTML report. This technique is useful when you want to create a single PDF or HTML report that contains multiple outputs, such as output from a TABLE request and a GRAPH request. You can distribute this type of report using ReportCaster.

When running graphs in GIF format on a UNIX platform with the HEADLESS configuration, the graph may not display properly. You may see a red X instead of your image. This is due to Sun's methodology of creating images without a head, which does not currently support GIF graphics.

**Supported with the commands:** HOLD.

**Available in:** WebFOCUS, App Studio.

For details see [Creating a Graph](#) on page 1657.

### **Reference:** FORMAT HTML

**Description:** Creates a complete HTML document that can be viewed in a web browser. The PCHOLD variation transfers the data from a web server to a browser.

The following StyleSheet features are supported with HTML: JAVASCRIPT (drill down to JavaScript), TARGET, COLSPAN, HEADALIGN, IMAGEALIGN, IMAGEBREAK, GUTTER, BACKIMAGE, BACKCOLOR, IMAGE, GRIDS.

**Use:** For display as a webpage.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

**Available in:** WebFOCUS, App Studio, FOCUS.

### **Reference:** FORMAT HTMTABLE

**Description:** Creates an output file that contains only an HTML table. The output produced is not a complete HTML document.

However, the file can be included in another HTML document using the Dialogue Manager command -HTMLFORM. For details see the documentation on Dialogue Manager in the *Developing Reporting Applications* manual.

**Note:** When issuing HOLD AS *name* FORMAT HTMTABLE, you must specify a different name than the .htm filename used in the -HTMLFORM name.

The following StyleSheet features are supported with HTML: JAVASCRIPT (drill down to JavaScript), TARGET, COLSPAN, HEADALIGN, IMAGEALIGN, IMAGEBREAK, BACKCOLOR, IMAGE, GRIDS.

Internal cascading style sheets (CSS) are supported for FORMAT HTMTABLE. The CSS code is placed immediately before the TABLE command.

**Use:** For embedding reports and graphs in an existing HTML document.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

The PCHOLD variation also transfers the data from a web server to a browser.

**Available in:** WebFOCUS, App Studio, FOCUS.

### **Reference:** FORMAT INGRES

**Description:** Creates an Ingres table, if you have the Adapter for Ingres and permission to create tables.

**Use:** For further processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio, FOCUS when used as a client to the WebFOCUS Reporting Server.

### **Reference:** FORMAT INTERNAL

**Description:** Saves report output without padding the values of integer and packed fields. See [Saving Report Output in INTERNAL Format](#) on page 546.

**Use:** For accurate processing by 3GL programs.

**Supported with the command:** HOLD, SAVB.

**Available in:** WebFOCUS, App Studio, FOCUS.

### **Reference:** FORMAT JPEG

**Description:** Saves the output of a graph request as a JPEG file.

**Use:** The resulting JPEG file can be embedded in the heading/footer or body of a PDF or HTML report. This technique is useful when you want to create a single PDF or HTML report that contains multiple outputs, such as output from a TABLE request and a GRAPH request. You can distribute this type of report using ReportCaster. JPEG (.jpg) files behave the same as GIF files.

**Note:** For JPEG files, only the .jpg extension is supported. The .jpeg extension is not supported.



**Supported with the commands:** HOLD.

**Available in:** WebFOCUS, App Studio.

For details, see [Linking From a Graphic Image](#) on page 809 and [Adding an Image to a Report](#) on page 1376.

**Reference:** **FORMAT JSCHART**

**Description:** Saves the output of a graph request as an HTML5 graph.

**Use:** The charts are rendered in the browser as high quality interactive vector graphics using a built-in JavaScript engine. Note that older browsers do not support all of the features of the HTML5 standard. You must include the following command to create an HTML5 graph:

`ON GRAPH PCHOLD FORMAT JSCHART`

**Supported with the commands:** ON GRAPH PCHOLD, ON GRAPH HOLD.

**Available in:** WebFOCUS, App Studio.

For details, see [Creating an HTML5 Graph](#) on page 1661.

**Reference:** **FORMAT LOTUS**

**Description:** Captures all the columns of the report in a character file that LOTUS 1-2-3 can then import. All alphanumeric fields are enclosed in quotation marks. Columns are separated by commas.

**Use:** For display and processing in a spreadsheet application.

**Supported with the commands:** HOLD, PCHOLD, SAVE (WebFOCUS only).

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** **FORMAT PDF**

**Description:** Saves the report output in Adobe Portable Document Format (PDF), which enables precise placement of output (all formatting options such as headings, footings, and titles) correctly aligned on the physical page, so the report looks exactly as it does when printed.

In WebFOCUS, PDF format also supports StyleSheets that contain drill down parameters, links to arbitrary URLs, and embedded GIF or JPEG images in report, page, and sort headings and footings.

PDF format does not support OLAP reports since Java applets cannot be embedded inside PDF documents. However, an OLAP report can drill down to a PDF-formatted report. PDF does not support drill downs when sorting across column. When you drill down from one report to another report, do not use the following characters: + (plus sign); # (number sign); & (ampersand); \ (backslash).

A PDF object is a page, hyperlink, or image. The Portable Document Format (PDF) limits the number of objects that a PDF document can contain. WebFOCUS imposes the following object limits for each PDF report:

Object	Limit
Pages	10,000
Images	900
Hyperlinks per page	500
Total pages with hyperlinks	100
Total hyperlinks	44,500

PDF format retains all formatting options, such as a headings, footings, and titles.

The following fonts are supported: Courier (fixed width), Times (proportional width), and Helvetica (proportional width). PDF format maps all fonts to Courier, Helvetica, or Times. The font styles that can be used are Normal (default), Bold, Italic, Underline, and combinations of these.

The following StyleSheet features are supported with PDF: PAGESIZE, ORIENTATION, UNITS, TOPMARGIN, BOTTOMMARGIN, LEFTMARGIN, RIGHTMARGIN, POSITION, SQUEEZE, FOCEXEC (drill down to FOCEXEC), URL (drill down to URL), HGRID, VGRID, BACKCOLOR. Note when you use BACKCOLOR with PDF reports, extra space is added to the top, bottom, right, and left of each cell of data in the report. This is for readability and to prevent data truncation.

**Use:** For display as a printed document.

**Supported with the commands:** HOLD, PCHOLD, SAVE (WebFOCUS only).

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** **FORMAT PDF OPEN/CLOSE**

**Description:** Saves multiple reports into one PDF report.

**Use:** For combining multiple reports into a single PDF file, also known as a compound report. For complete details, see [Laying Out the Report Page](#) on page 1249.

**Supported with the command:** PCHOLD.

**Available in:** WebFOCUS, App Studio, FOCUS

**Reference:** **FORMAT POSTSCRIPT (PS)**

**Description:** Creates an output file in PostScript format, which supports headings, footings, and totals.

PS is an abbreviation for POSTSCRIPT.

PostScript format supports headings, footings, and totals. PS supports ISO Latin font encoding.

**Use:** For display as a printed document.

**Supported with the command:** HOLD, PCHOLD.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** **FORMAT PPT**

**Description:** Creates an output file in PowerPoint® format in which each page of report output becomes a separate slide in the file with all styling applied.

**Use:** For use in a slide presentation.

**Supported with the command:** HOLD, SAVE.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** **FORMAT PPTX**

**Description:** Creates an output file in PowerPoint format in which each page of report output becomes a separate slide in the file with all styling applied.

**Use:** For use in a slide presentation.

**Supported with the command:** HOLD, SAVE.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** **FORMAT REDBRICK**

**Description:** Creates a Red Brick table, if you have the Adapter for Red Brick and permission to create tables.

**Use:** For further processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio, FOCUS when used as a client to the WebFOCUS Reporting Server.

**Reference:** **FORMAT SQL\_SCRIPT**

**Description:** Creates an SQL subquery file or a file of data values with a corresponding synonym.

When used in a request against a relational data source, the HOLD FORMAT SQL\_SCRIPT command generates the SQL SELECT statement needed to execute the current query. It then stores it in the application folder as a file with a .sql extension, along with the Master and Access File pair that describes the SQL answer set.

When used in a request against any other type of data source, the HOLD FORMAT SQL\_SCRIPT command executes the current query and stores the retrieved values in the application folder as a sequential file with a .ftm extension, along with the Master File that describes the retrieved data.

**Use:** You can use the output from HOLD FORMAT SQL\_SCRIPT as the target file for the DB\_INFILE function. For information about the DB\_INFILE function, see the *Using Functions* manual.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** **FORMAT SQLDBC**

**Description:** Creates a Teradata table, if you have the Adapter for Teradata and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** **FORMAT SQLINF**

**Description:** Creates an Informix table, if you have the Adapter for Informix and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio, FOCUS when used as a client to the WebFOCUS Reporting Server.

**Reference:** **FORMAT SQLMAC**

**Description:** Creates a Microsoft Access table, if you have the Adapter for Microsoft Access and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio.

**Reference:** **FORMAT SQLMSS**

**Description:** Creates a Microsoft SQL Server table, if you have the Adapter for Microsoft SQL and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio, FOCUS when used as a client to the WebFOCUS Reporting Server.

**Reference:** **FORMAT SQLODBC**

**Description:** Creates an SQLODBC table if you have the current Adapter for ODBC and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio, FOCUS when used as a client to the WebFOCUS Reporting Server.

**Reference:** **FORMAT SQLORA**

**Description:** Creates an Oracle table, if you have the Adapter for Oracle and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** **FORMAT SQLPSTGR**

**Description:** Creates a PostgreSQL table, if you have the Adapter for PostgreSQL and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio, FOCUS when used as a client to the WebFOCUS Reporting Server.

**Reference:** **FORMAT SQLSYB**

**Description:** Creates a Sybase table, if you have the Adapter for Sybase and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio, FOCUS when used as a client to the WebFOCUS Reporting Server.

**Reference:** **FORMAT SYLK**

**Description:** Captures all the columns of the report request in a character file for Microsoft's spreadsheet program Multiplan. The generated file cannot have more than 9,999 rows.

**Use:** For display and processing in a spreadsheet application.

**Supported with the command:** HOLD, SAVE.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference: FORMAT TAB**

**Description:** Creates an output file in tab-delimited format. The TAB format includes a built-in safety feature, which allows embedded quotes within character fields. A second double quote (") is inserted adjacent to the existing one. For example, if you input Joe "Smitty" Smith, the output is Joe ""Smitty"" Smith. The TAB format also includes the following features:

- ☐ All trailing blanks are stripped from alpha [An] fields.
- ☐ All leading blanks are stripped from numeric [/Dx.y, /Fx.y, /Px.y, and /In] fields.
- ☐ There is a 32K record length limit in the output file.
- ☐ A Master File is created when the HOLD command is used to create the output file. The Master File behaves exactly as in FORMAT ALPHA, except for the inclusion of double quotes.

**Note:** To create a variable-length comma- or tab-delimited HOLD file that differentiates between a missing value and a blank string or zero value, use the SET NULL=ON command. For more information, see the *Developing Reporting Applications* manual.

**Use:** For importing data to Windows-based applications such as MS Access and Excel.

**Supported with the command:** HOLD, SAVE, PCHOLD.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference: FORMAT TABT**

**Description:** Creates an output file in tab-delimited format that includes column headings in the first row. The TABT format includes a built-in safety feature, which allows embedded quotes within character fields. A second double quote (") is inserted adjacent to the existing one. For example, if you input Joe "Smitty" Smith, the output is Joe ""Smitty"" Smith. The TABT format also includes the following features:

- ☐ The first row contains field names.
- ☐ All trailing blanks are stripped from alpha [An] fields.
- ☐ All leading blanks are stripped from numeric [/Dx.y, /Fx.y, /Px.y, and /In] fields.
- ☐ There is a 32K record length limit in the output file.
- ☐ A Master File is created when the HOLD command is used to create the output file. The Master File behaves exactly as in FORMAT ALPHA, except for the inclusion of double quotes.

**Note:**

- ❑ Blank field names display as blank column titles. This may result in an error when attempting to use the file as input to various applications.
- ❑ To create a variable-length comma- or tab-delimited HOLD file that differentiates between a missing value and a blank string or zero value, use the SET NULL=ON command. For more information, see the *Developing Reporting Applications* manual.

**Use:** For importing data to Windows-based applications such as MS Access and Excel.

**Supported with the command:** HOLD, SAVE, PCHOLD.

**Available in:** WebFOCUS, App Studio, FOCUS.

**Reference:** **FORMAT VISDIS**

**Description:** WebFOCUS has a data visualization tool called Visual Discovery that creates graphs using a tab-delimited data file as input. The first record of the data file contains the column titles for the data values. The next record can contain the Visual Discovery field formats. If this record is not present, Visual Discovery attempts to determine the formats of the data fields by reading the first 50 records from the data file, a process that is not guaranteed to create accurate representations of the WebFOCUS formats.

The following table identifies format conversions for HOLD FORMAT VISDIS:

FOCUS Format	Visual Discovery Format
Integer	I
Decimal/Packed	R
Alphanumeric	S
Date format (smart date)	D%format%format%format (for example, D %Y%M%D). If the year is not a four-digit year, the format returned is S.
Other	S

**Note:** No Master File is created for format VISDIS.

**Use:** HOLD FORMAT VISDIS creates a tab-delimited output file with the extension .txt in which the first record has the column titles and the second record contains Visual Discovery formats based on the FOCUS field formats of the data.



**Supported with the command:** HOLD, SAVE, PCHOLD

**Available in:** WebFOCUS, FOCUS.

**Reference:** **FORMAT WK1**

**Description:** Captures all the columns of the report in a character file that LOTUS 1-2-3 Release 2 can then import.

**Use:** For display and processing in a spreadsheet application.

**Supported with the commands:** HOLD, PCHOLD.

**Available in:** App Studio.

**Reference:** **FORMAT WP**

**Description:** Captures the entire report output, including headings, footings, and subtotals, and creates a text file that can easily be incorporated into most word processing packages.

Text fields are supported in WP format. See [Using Text Fields in Output Files](#) on page 535.

To control whether a carriage control character is included in column 1 of each page of the report output, use:

```
[ON TABLE] HOLD AS filename FORMAT WP [CC|NOCC]
```

NOCC excludes carriage control characters. The position reserved for those characters remains in the file, but is blank. CC includes carriage control characters and, in z/OS, creates the HOLD file with RECFM VBA. To include page control information in the WP file, you can also specify the TABPAGENO option in a heading or the SET PAGE=OFF command. The character 1 in the column 1 indicates the start of a new page.

The following rules summarize FORMAT WP carriage control options:

- ☐ The CC option always inserts the carriage control character.
- ☐ The NOCC option always omits the carriage control character.
- ☐ When you issue HOLD FORMAT WP without the CC or NOCC option:
  - ☐ SET PAGE NUM=OFF and SET PAGE NUM=TOP always insert the carriage control character.
  - ☐ SET PAGE NUM=NOPAGE always omits the carriage control character.
  - ☐ SET PAGE NUM=ON inserts the carriage control character if TABPAGENO is included in the heading and omits the carriage control character if TABPAGENO is not included in the heading.

**Tip:** HOLD FORMAT WP does not change the number of lines per page. In order to do so, issue one or a combination of the commands SET PRINT=OFFLINE, SET SCREEN=PAPER, or SET SCREEN=OFF.

The WP file is created with a record format of VB in z/OS when the carriage control character is omitted and with a record format of VBA when the carriage control character is inserted.

The maximum record length for HOLD FORMAT WP is 358 characters, 356 of which can represent data. For larger output, use PCHOLD FORMAT WP.

If you need the report width to remain fixed across releases for later processing of the output file, you can set the width you need using the SET WPMINWIDTH command. This parameter specifies the minimum width of the output file. It will be automatically increased if the width you set cannot accommodate the fields propagated to the output file in the request. The LRECL of the output file will be four bytes more than the report width on z/OS because the file is variable length and needs an additional four bytes to hold the actual length of each record instance. In other operating environments, the length of the record is the value of WPMIDWIDTH.

FORMAT WP retains headings, footings, and subtotals.

**Use:** For display in a text document.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

**Available in:** WebFOCUS, App Studio, FOCUS.

### **Reference:** FORMAT XFOCUS

**Description:** Creates an XFOCUS data source.

**Use:** For further processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, App Studio, FOCUS.

### **Reference:** FORMAT XML

**Description:** Creates an XML output file based on an internal DTD that includes the rows that are displayed in the final report output. It does not honor the HOLDLIST setting. It does not honor the ASNAMES setting. The parameter HOLDATTRS ON is automatically invoked. The XML file generated is based on the structure and layout of the report request. The metadata included in the DTD for each column is FIELDNAME, ALIAS, data type, width, FOCUS format, DESCRIPTION, ACCEPTS, HELP\_MESSAGE, TITLE, WITHIN, PROPERTY, REFERENCE, VALIGN, and column title. FORMAT XML does not create a Master File.

**Note:** When using an HTML Autoprompt page (select Run in new Window option), BI Portal, or an HTML page created with HTML canvas, browsers running in Standards Mode displaying output in an iframe do not display XML. When running a WebFOCUS request with PCHOLD FORMAT XML, the result will display in a new window.

**Use:** For further processing.

**Supported with the command:** HOLD, PCHOLD.

**Available in:** WebFOCUS, FOCUS.

## Using Text Fields in Output Files

Text fields can be propagated to HOLD and SAVE files that have the following formats: ALPHA, WP, and FOCUS or XFOCUS. However, although a Master File is generated for format ALPHA, you cannot issue a TABLE request against a HOLD file of format ALPHA that contains text fields.

### *Reference:* Rules for Text Fields in Output Files

- ☐ You can include as many text fields in the file as needed. However, you must specify text fields after non-text fields in the display list (PRINT..., SUM..., and so forth).
- ☐ You can specify only one text field in the display list, and no non-text fields, in a request that includes an ACROSS phrase.

The following rules apply to missing data for text fields in HOLD and SAVE files:

- ☐ A blank line is valid data. An end-of-text mark indicates the end of the field.
- ☐ If there is no text for a field, a single period (.) followed by blanks appears in the HOLD or SAVE file.
- ☐ If MISSING=ON during data extraction, a period (.) is written out to the HOLD or SAVE file.
- ☐ If MISSING=OFF during data extraction, a blank is written out to the HOLD or SAVE file.

See [Handling Records With Missing Field Values](#) on page 971.

In environments that support FIXFORM, due to limitations in the use of text fields with FIXFORM, the following restriction applies:

- ☐ When you use the command FIXFORM FROM HOLD, the HOLD file may not contain more than one text field, and the text field must be the last field in the Master File.

When HOLD files are read using FIXFORM, the interpretation of missing text depends on whether the field's designation is MISSING=ON in the Master File, conditional ©) in the FIXFORM format description, or some combination of the two.

**Example:**    **Applying Text Field Rules in HOLD Files**

The following request extracts data to a HOLD file named CRSEHOLD:

```
TABLE FILE COURSE
PRINT COURSECODE DESCRIPTN1
ON TABLE HOLD AS CRSEHOLD
END
```

The following request prints the data from the HOLD file:

```
TABLE FILE CRSEHOLD
PRINT *
END
```

The partial output is:

<u>COURSECODE</u>	<u>DESCRIPTN1</u>
AMA130	FOR PROJECT LEADERS AND SYS ANALYSTS.
AMA680	FOR INDUSTRIAL MARKETERS.
AMA800	FOR MANAGERS AND HUMAN RESOURCE STAFF.
BIT420	FOR SENIOR MANAGERS. ANALYZE AND IMPROVE
BIT640	FOR ADMINISTRATOR OF LABOR CONTRACTS.
BIT650	FOR PROGRAMMERS, PROJECT LEADERS AND
EDP090	FOR PRODUCTION MANAGERS. TO DELEGATE AND
EDP130	FOR PROGRAMMERS, PROJECT LEADERS AND
EDP390	FOR MANAGERS. HELP APPLY MANAGERIAL
EDP690	FOR EXEC MANAGERS AND MKTG RESEARCH
EDP750	FOR MARKETING MANAGERS. ENABLE TO
MC230	FOR PROGRAMMERS, PROJECT LEADERS AND
MC90	FOR VICE PRES OF SALES AND MARKETING.
NAMA40	FOR MARKETING VICE PRES AND MANAGERS.
NAMA730	FOR MANAGERS OF SALESPeOPLE. TO ENHANCE
NAMA930	FOR FINANCIAL MANAGERS. EXPLORE CONCEPTS
PDR330	FOR EXECUTIVE SECRETARIES. TO KNOW THE

```

PDR740          FOR EXPERIENCED MARKETING MANAGERS.  HELP
PDR870          FOR DESIGNERS AND TRAINING COORDINATORS.
PU168           FOR MARKETING, PRODUCT, ADVERTISING
PU440           FOR GENERAL MANAGERS.  TO EXPLORE
SFC280          FOR MANAGERS AND SECRETARIES.  HELP
.
.
.

```

## Creating a Delimited Sequential File

You can use the HOLD FORMAT DFIX command to create an alphanumeric sequential file delimited by any character or combination of characters. You can also specify whether to enclose alphanumeric values in quotation marks or some other enclosure, whether to include a header record that lists the names of the fields, whether to preserve leading and trailing blank spaces in alphanumeric data, and whether to insert a delimiter between records in the resulting file. (Note that when RDELIMITER is included, the RECFM is UB).

A Master File and an Access File are created to describe the delimited sequential file that is generated. The SUFFIX value in the Master File is DFIX. The Access File specifies the delimiter, the enclosure character (if any), whether to preserve leading and trailing blank spaces in alphanumeric data, whether there is a header record, and the record delimiter, if there is to be one. The Master and Access Files are useful if you will later read the sequential file using WebFOCUS.

### ***Syntax:*** How to Create a Delimited Sequential File

```

ON TABLE {HOLD|PCHOLD} [AS filename] FORMAT DFIX
DELIMITER delimiter
[ENCLOSURE enclosure] [HEADER {YES|NO}]
[PRESERVE SPACE {YES|NO}] [RDELIMITER rdelimiter]

```

where:

*filename*

Is the name of the file to be created. If you do not specify a name, the default name is HOLD.

### *delimiter*

Is the delimiter sequence consisting of up to 30 printable or non-printable non-null characters. This represents character semantics. For example, if you are using DBCS characters, the delimiter can be up to 60 bytes. Characters may also be represented by their 0x hex values which is the required specification method for non-printable characters. If you use a mixture of printable and non-printable characters, you must enter them all as hexadecimal values. To create a tab delimited file, you can specify the DELIMITER value as TAB or as its hexadecimal equivalent (0x09 on ASCII platforms or 0x05 on EBCDIC platforms). To create a single-quote delimited file, you must specify the single quote DELIMITER value as its hexadecimal equivalent (0x27 on ASCII platforms or 0x7D on EBCDIC platforms), otherwise the request will be mis-interpreted and result in an unusable HOLD file.

### *enclosure*

Is the enclosure sequence. It can be up to four printable or non-printable characters used to enclose each alphanumeric value in the file. This represents character semantics. For example, if you are using DBCS characters, the enclosure can be up to 8 bytes. Most alphanumeric characters can be used as all or part of the enclosure sequence. However, numeric digits and symbols used in numbers, such as a period (.), plus sign (+), or minus sign (-) cannot be used in the enclosure sequence. Also note that, in order to specify a single quotation mark as the enclosure character, you must enter four consecutive single quotation marks. The most common enclosure is one double quotation mark.

If you use a mixture of printable and non-printable characters, you must enter them all as hexadecimal values. For printable characters, you can either use the characters themselves or their hexadecimal equivalents (for example, the ampersand character may be interpreted as the beginning of a variable name rather than as part of the enclosure.)

### HEADER {YES|NO}

Specifies whether to include a header record that contains the names of the fields in the delimited sequential file generated by the request.

### PRESERVE SPACE {YES|NO}

Specifies whether to retain leading and trailing blanks in alphanumeric data. YES preserves leading and trailing blanks. NO only preserves leading and trailing blanks that are included within the enclosure characters. NO is the default value.

**Note:** PRESERVE SPACE is overridden by the ENCLOSURE option. Therefore, exclude the enclosure option in order to have the PRESERVE SPACE setting respected.

*rdelimiter*

Is the record delimiter sequence consisting of up to 30 printable or non-printable non-null characters. This represents character semantics. For example, if you are using DBCS characters, the delimiter can be up to 60 bytes. For a non-printable character, enter the hexadecimal value that represents the character. If you use a mixture of printable and non-printable characters, you must enter them all as hexadecimal values. For printable characters you can either use the characters themselves or their hexadecimal equivalents (for example, the ampersand character may be interpreted as the beginning of a variable name rather than as part of the delimiter). To use a tab character as the record delimiter, you can specify the delimiter value as *TAB* or as its hexadecimal equivalent (0x09 on ASCII platforms or 0x05 on EBCDIC platforms). The comma (,) is not supported as a record delimiter.

Note that numeric digits and symbols used in numbers, such as a period (.), plus sign (+), or minus sign (-) cannot be used in the delimiter sequence. When RDELIMITER is included, the RECFM is UB.

**Reference: Usage Notes for HOLD FORMAT DFIX**

- ☐ Missing data is indicated by no data. So, with enclosure, a missing alphanumeric field is indicated by two enclosure characters, while a missing numeric field is indicated by two delimiters.
- ☐ Text fields are not supported with HOLD FORMAT DFIX.
- ☐ While HOLD FORMAT DFIX creates a single segment file, you can manually add segments to the resulting Master and Access File. In the Access File, you can specify a separate delimiter and/or enclosure for each segment.
- ☐ The extension of the generated sequential file is *ftm*.
- ☐ HOLD FORMAT DFIX with the PRESERVE SPACE YES option creates a file in which leading and trailing blank spaces are preserved in alphanumeric data. It also adds the attribute PRESERVE SPACE=YES in the Access File. This attribute causes leading and trailing blank spaces to be preserved when reading a FORMAT DFIX file.

### **Example:** Creating a Pipe-Delimited File

The following request against the CENTORD data source creates a sequential file named PIPE1 with fields separated by the pipe character (|). Alphanumeric values are not enclosed in quotation marks, and there is no header record:

```
TABLE FILE CENTORD
SUM QUANTITY LINEPRICE BY REGION BY YEAR
ON TABLE HOLD AS PIPE1 FORMAT DFIX DELIMITER |
END
```

The PIPE1 Master File specifies the SUFFIX value as DFIX:

```
FILENAME=PIPE1      , SUFFIX=DFIX      , $
SEGMENT=PIPE1, SEGTYPE=S2, $
  FIELDNAME=REGION, ALIAS=E01, USAGE=A5, ACTUAL=A05, $
  FIELDNAME=YEAR, ALIAS=E02, USAGE=YY, ACTUAL=A04, $
  FIELDNAME=QUANTITY, ALIAS=E03, USAGE=I8C, ACTUAL=A08, $
  FIELDNAME=LINEPRICE, ALIAS=E04, USAGE=D12.2MC, ACTUAL=A12, $
```

The PIPE1 Access File specifies the delimiter:

```
SEGNAME=PIPE1, DELIMITER=|, HEADER=NO, $
```

The PIPE1 sequential file contains the following data. Each data value is separated from the next value by a pipe character:

```
EAST|2000|3907|1145655.77
EAST|2001|495922|127004359.88
EAST|2002|543678|137470917.05
NORTH|2001|337168|85750735.54
NORTH|2002|370031|92609802.80
SOUTH|2000|3141|852550.45
SOUTH|2001|393155|99822662.88
SOUTH|2002|431575|107858412.0
WEST|2001|155252|39167974.18
WEST|2002|170421|42339953.45
```

The following version of the HOLD command specifies both the delimiter and an enclosure character (" "):

```
ON TABLE HOLD AS PIPE1 FORMAT DFIX DELIMITER | ENCLOSURE " "
```

The Master File remains the same, but the Access File now specifies the enclosure character:

```
SEGNAME=PIPE1, DELIMITER=|, ENCLOSURE=" ", HEADER=NO, $
```



In the delimited file that is created, each data value is separated from the next by a pipe character, and alphanumeric values are enclosed within double quotation marks:

```
"EAST" | 2000 | 3907 | 1145655.77
"EAST" | 2001 | 495922 | 127004359.88
"EAST" | 2002 | 543678 | 137470917.05
"NORTH" | 2001 | 337168 | 85750735.54
"NORTH" | 2002 | 370031 | 92609802.80
"SOUTH" | 2000 | 3141 | 852550.45
"SOUTH" | 2001 | 393155 | 99822662.88
"SOUTH" | 2002 | 431575 | 107858412.01
"WEST" | 2001 | 155252 | 39167974.18
"WEST" | 2002 | 170421 | 42339953.45
```

This version of the HOLD command adds a header record to the generated file:

```
ON TABLE HOLD AS PIPE1 FORMAT DFIX DELIMITER | ENCLOSURE " HEADER YES
```

The Master File remains the same, but the Access File now specifies that the generated sequential file should contain a header record with column names as its first record:

```
SEGNAME=PIPE1, DELIMITER=|, ENCLOSURE=", HEADER=YES, $
```

In the delimited file that is created, each data value is separated from the next by a pipe character, and alphanumeric values are enclosed within double quotation marks. The first record contains the column names:

```
"REGION" | "YEAR" | "QUANTITY" | "LINEPRICE"
"EAST" | 2000 | 3907 | 1145655.77
"EAST" | 2001 | 495922 | 127004359.88
"EAST" | 2002 | 543678 | 137470917.05
"NORTH" | 2001 | 337168 | 85750735.54
"NORTH" | 2002 | 370031 | 92609802.80
"SOUTH" | 2000 | 3141 | 852550.45
"SOUTH" | 2001 | 393155 | 99822662.88
"SOUTH" | 2002 | 431575 | 107858412.01
"WEST" | 2001 | 155252 | 39167974.18
"WEST" | 2002 | 170421 | 42339953.45
```

### **Example:** Creating a Tab-Delimited File

The following request against the CENTORD data source creates a sequential file named TAB1 with fields separated by a tab character:

```
TABLE FILE CENTORD
SUM QUANTITY LINEPRICE BY REGION BY YEAR
ON TABLE HOLD AS TAB1 FORMAT DFIX DELIMITER TAB
END
```

As the tab character is not printable, the TAB1 Access File specifies the delimiter using its hexadecimal value.

The following is the Access File in an EBCDIC environment:

```
SEGNAME=TAB1, DELIMITER=0x05, HEADER=NO, $
```

The following is the Access File in an ASCII environment:

```
SEGNAME=TAB1, DELIMITER=0x09, HEADER=NO, $
```

### ***Example:*** Creating a Delimited File With Blank Spaces Preserved

The following request against the GGSALES data source creates a comma-delimited file. The original alphanumeric data has trailing blank spaces. The PRESERVE SPACE YES option in the HOLD command preserves these trailing blank spaces:

```
APP HOLDDATA APP1
APP HOLDMETA APP1
TABLE FILE GGSALES
SUM DOLLARS UNITS
BY REGION
BY CATEGORY
BY PRODUCT
ON TABLE HOLD AS DFIX1 FORMAT DFIX DELIMITER , PRESERVE SPACE YES
END
```

The following Master File is generated:

```
FILENAME=DFIX1, SUFFIX=DFIX, $
SEGMENT=DFIX1, SEGTYPE=S3, $
  FIELDNAME=REGION, ALIAS=E01, USAGE=A11, ACTUAL=A11, $
  FIELDNAME=CATEGORY, ALIAS=E02, USAGE=A11, ACTUAL=A11, $
  FIELDNAME=PRODUCT, ALIAS=E03, USAGE=A16, ACTUAL=A16, $
  FIELDNAME=DOLLARS, ALIAS=E04, USAGE=I08, ACTUAL=A08, $
  FIELDNAME=UNITS, ALIAS=E05, USAGE=I08, ACTUAL=A08, $
```

The following Access File is generated:

```
SEGNAME=DFIX1, DELIMITER=',', HEADER=NO, PRESERVE SPACE=YES, $
```

In the DFIX1 file, the alphanumeric fields contain all of the blank spaces that existed in the original file:

Midwest	,Coffee	,Espresso	,1294947,101154
Midwest	,Coffee	,Latte	,2883566,231623
Midwest	,Food	,Biscotti	,1091727,86105
Midwest	,Food	,Croissant	,1751124,139182
Midwest	,Food	,Scone	,1495420,116127
Midwest	,Gifts	,Coffee Grinder	,619154,50393
Midwest	,Gifts	,Coffee Pot	,599878,47156
Midwest	,Gifts	,Mug	,1086943,86718
Midwest	,Gifts	,Thermos	,577906,46587
Northeast	,Coffee	,Capuccino	,542095,44785
Northeast	,Coffee	,Espresso	,850107,68127
Northeast	,Coffee	,Latte	,2771815,222866
Northeast	,Food	,Biscotti	,1802005,145242
Northeast	,Food	,Croissant	,1670818,137394
Northeast	,Food	,Scone	,907171,70732
Northeast	,Gifts	,Coffee Grinder	,509200,40977
Northeast	,Gifts	,Coffee Pot	,590780,46185
Northeast	,Gifts	,Mug	,1144211,91497
Northeast	,Gifts	,Thermos	,604098,48870
Southeast	,Coffee	,Capuccino	,944000,73264
Southeast	,Coffee	,Espresso	,853572,68030
Southeast	,Coffee	,Latte	,2617836,209654
Southeast	,Food	,Biscotti	,1505717,119594
Southeast	,Food	,Croissant	,1902359,156456
Southeast	,Food	,Scone	,900655,73779
Southeast	,Gifts	,Coffee Grinder	,605777,47083
Southeast	,Gifts	,Coffee Pot	,645303,49922
Southeast	,Gifts	,Mug	,1102703,88474
Southeast	,Gifts	,Thermos	,632457,48976
West	,Coffee	,Capuccino	,895495,71168
West	,Coffee	,Espresso	,907617,71675
West	,Coffee	,Latte	,2670405,213920
West	,Food	,Biscotti	,863868,70436
West	,Food	,Croissant	,2425601,197022
West	,Food	,Scone	,912868,72776
West	,Gifts	,Coffee Grinder	,603436,48081
West	,Gifts	,Coffee Pot	,613624,47432
West	,Gifts	,Mug	,1188664,93881
West	,Gifts	,Thermos	,571368,45648

Creating the same file with PRESERVE SPACE NO removes the trailing blank spaces:

```
Midwest,Coffee,Espresso,1294947,101154
Midwest,Coffee,Latte,2883566,231623
Midwest,Food,Biscotti,1091727,86105
Midwest,Food,Croissant,1751124,139182
Midwest,Food,Scone,1495420,116127
Midwest,Gifts,Coffee Grinder,619154,50393
Midwest,Gifts,Coffee Pot,599878,47156
Midwest,Gifts,Mug,1086943,86718
Midwest,Gifts,Thermos,577906,46587
Northeast,Coffee,Capuccino,542095,44785
Northeast,Coffee,Espresso,850107,68127
Northeast,Coffee,Latte,2771815,222866
Northeast,Food,Biscotti,1802005,145242
Northeast,Food,Croissant,1670818,137394
Northeast,Food,Scone,907171,70732
Northeast,Gifts,Coffee Grinder,509200,40977
Northeast,Gifts,Coffee Pot,590780,46185
Northeast,Gifts,Mug,1144211,91497
Northeast,Gifts,Thermos,604098,48870
Southeast,Coffee,Capuccino,944000,73264
Southeast,Coffee,Espresso,853572,68030
Southeast,Coffee,Latte,2617836,209654
Southeast,Food,Biscotti,1505717,119594
Southeast,Food,Croissant,1902359,156456
Southeast,Food,Scone,900655,73779
Southeast,Gifts,Coffee Grinder,605777,47083
Southeast,Gifts,Coffee Pot,645303,49922
Southeast,Gifts,Mug,1102703,88474
Southeast,Gifts,Thermos,632457,48976
West,Coffee,Capuccino,895495,71168
West,Coffee,Espresso,907617,71675
West,Coffee,Latte,2670405,213920
West,Food,Biscotti,863868,70436
West,Food,Croissant,2425601,197022
West,Food,Scone,912868,72776
West,Gifts,Coffee Grinder,603436,48081
West,Gifts,Coffee Pot,613624,47432
West,Gifts,Mug,1188664,93881
West,Gifts,Thermos,571368,45648
```

### **Example:** Specifying a Record Delimiter

The following request against the GGSALES data source, the field delimiter is a comma, the enclosure character is a single quotation mark, and the record delimiter consists of both printable and non-printable characters, so it is specified as the following hexadecimal sequence:

- ☐ 0x: character sequence identifying the delimiter as hexadecimal character codes.
- ☐ 2C: hexadecimal value for comma (,).

- ☐ 24: hexadecimal value for dollar sign (\$).
- ☐ 0D: hexadecimal value for carriage return.
- ☐ 0A: hexadecimal value for new line.

```
TABLE FILE GGSALES
PRINT DOLLARS UNITS CATEGORY REGION
ON TABLE HOLD AS RDELIM1 FORMAT DFIX DELIMITER , ENCLOSURE '''
HEADER NO RDELIMITER 0x2C240D0A
END
```

The generated Master File follows:

```
FILENAME=RDELIM1 , SUFFIX=DFIX , $
SEGMENT=RDELIM1, SEGTYPE=S0, $
  FIELDNAME=DOLLARS, ALIAS=E01, USAGE=I08, ACTUAL=A08, $
  FIELDNAME=UNITS, ALIAS=E02, USAGE=I08, ACTUAL=A08, $
  FIELDNAME=CATEGORY, ALIAS=E03, USAGE=A11, ACTUAL=A11, $
  FIELDNAME=REGION, ALIAS=E04, USAGE=A11, ACTUAL=A11, $
```

The Access File contains the delimiters and enclosure characters:

```
SEGNAME=RDELIM1,
  DELIMITER=',',
  ENCLOSURE='''',
  HEADER=NO,
  RDELIMITER=0x2C240D0A,
  PRESERVE SPACE=NO, $
```

Each row of the resulting DFIX file ends with the comma-dollar combination and a carriage return and line space. a partial listing follows:

```
20805,1387,'Coffee','Northeast',,$
20748,1729,'Coffee','Northeast',,$
20376,1698,'Coffee','Northeast',,$
20028,1669,'Coffee','Northeast',,$
19905,1327,'Coffee','Northeast',,$
19470,1770,'Coffee','Northeast',,$
19118,1738,'Coffee','Northeast',,$
18720,1560,'Coffee','Northeast',,$
18432,1536,'Coffee','Northeast',,$
17985,1199,'Coffee','Northeast',,$
17630,1763,'Coffee','Northeast',,$
16646,1189,'Coffee','Northeast',,$
15650,1565,'Coffee','Northeast',,$
15450,1545,'Coffee','Northeast',,$
15435,1029,'Coffee','Northeast',,$
14270,1427,'Coffee','Northeast',,$
```

**Example:**    **Missing Data in the HOLD File**

The following request against the CENTORD data source creates missing alphanumeric and numeric values in the resulting comma-delimited HOLD file:

```
DEFINE FILE CENTORD
AREA/A5 MISSING ON = IF REGION EQ 'EAST' THEN MISSING ELSE REGION;
MQANTITY/I9 MISSING ON = IF REGION EQ 'WEST' THEN MISSING ELSE 200;
END

TABLE FILE CENTORD
SUM QUANTITY MQANTITY LINEPRICE BY AREA BY YEAR
WHERE AREA NE 'NORTH' OR 'SOUTH'
      ON TABLE HOLD AS MISS1 FORMAT DFIX DELIMITER , ENCLOSURE "
END
```

In the MISS1 HOLD file, the missing alphanumeric values are indicated by two enclosure characters in a row ("" ) and the missing numeric values are indicated by two delimiters in a row (,,):

```
" " ,2000,3907,600,1145655.77
" " ,2001,495922,343000,127004359.88
" " ,2002,543678,343000,137470917.05
"WEST",2001,155252,,39167974.18
"WEST",2002,170421,,42339953.45
```

**Saving Report Output in INTERNAL Format**

HOLD files pad binary integer and packed decimal data values to a full word boundary. For example, a three-digit integer field (I3), is stored as four bytes in a HOLD file. In order for third generation programs, such as COBOL, to be able to read HOLD files in an exact manner, you may need to save the fields in the HOLD file without any padding.

To suppress field padding in the HOLD file, you must reformat the fields in the request in order to override the default ACTUAL formats that correspond to the USAGE formats in the Master File:

- ☐ Reformat the integer and packed fields that you do not want to be padded in the HOLD file to the correct display lengths.
- ☐ Specify HOLD FORMAT INTERNAL for the report output.

**Syntax:**      **How to Suppress Field Padding in HOLD Files**

```

SET HOLDLIST = PRINTONLY
TABLE FILE filename
display_command fieldname/[In|Pn.d]
.
.
ON TABLE HOLD AS name FORMAT INTERNAL
END

```

where:

**PRINTONLY**

Causes your report request to propagate the HOLD file with only the specified fields displaying in the report output. If you do not issue this setting, an extra field containing the padded field length is included in the HOLD file. See [Controlling Attributes in HOLD Master Files](#) on page 484.

***fieldname*/[In|Pn.d]**

Specify correct lengths in the formats for integer and packed fields where you wish to suppress padding. These formats override the ACTUAL formats used for the display formats in the Master File. See [Usage Notes for Suppressing Padded Fields in HOLD Files](#) on page 548.

Note that floating point double-precision (D) and floating point single-precision (F) are not affected by HOLD FORMAT INTERNAL.

**FORMAT INTERNAL**

Saves the HOLD file without padding for specified integer and packed decimal fields.

**Reference:** Usage Notes for Suppressing Padded Fields in HOLD Files

- ❑ Integer fields (I) of one, two, three, or four bytes produce four-byte integers without HOLD FORMAT INTERNAL.
- ❑ For packed decimal fields (Pn.d), *n* is the total number of digits and *d* is the number of digits to the right of the decimal point. The number of bytes is derived by dividing *n* by 2 and adding 1.

The syntax is

```
bytes = INT (n/2) + 1
```

where:

```
INT (n/2)
```

Is the greatest integer after dividing by 2.

- ❑ HOLD FORMAT INTERNAL does not affect floating point double-precision (D) and floating point single-precision (F) fields. D remains at eight bytes, and F at four bytes.
- ❑ Alphanumeric fields automatically inherit their length from their source Master File, and are not padded to a full word boundary.
- ❑ If a format override is not large enough to contain the data values, the values are truncated. Truncation may cause the data in the HOLD file to be incorrect in the case of an integer. For packed data and integers, truncation occurs for the high order digits so the remaining low order digits resemble the digits from the correct values.  
  
To avoid incorrect results, be sure that the format you specify is large enough to contain the data values.
- ❑ If you use the HOLDMISS=ON setting to propagate missing values to the HOLD file, short packed fields and fields with formats I1, I2, and I3 are not large enough to hold the missing value.

**Example:** Creating a HOLD File Without HOLD FORMAT INTERNAL

In this example, the values of ACTUAL for RETAIL\_COST, DEALER\_COST, and SEATS are all padded to a full word. Alphanumeric fields also occupy full words.

```
TABLE FILE CAR
PRINT CAR COUNTRY RETAIL_COST DEALER_COST SEATS
ON TABLE HOLD AS DJG
END
```



The request creates the following Master File:

```
FILE=DJG, SUFFIX=FIX
SEGMENT=DJG, SEGTYPE=S0
  FIELDNAME=CAR           ,ALIAS=E01   ,USAGE=A16   ,ACTUAL=A16   , $
  FIELDNAME=COUNTRY       ,ALIAS=E02   ,USAGE=A10   ,ACTUAL=A12   , $
  FIELDNAME=RETAIL_COST   ,ALIAS=E03   ,USAGE=D7    ,ACTUAL=D08   , $
  FIELDNAME=DEALER_COST   ,ALIAS=E04   ,USAGE=D7    ,ACTUAL=D08   , $
  FIELDNAME=SEATS         ,ALIAS=E05   ,USAGE=I3    ,ACTUAL=I04   , $
```

### **Example:** Creating a HOLD File With HOLD FORMAT INTERNAL

In this example, DEALER\_COST and RETAIL\_COST are defined in the Master File as D fields, but the request overrides RETAIL\_COST as an I2 field and DEALER\_COST as a P3 field.

```
SET HOLDLIST=PRINTONLY
TABLE FILE CAR
PRINT CAR COUNTRY RETAIL_COST/I2 DEALER_COST/P3 SEATS/I1
ON TABLE HOLD AS HINT3 FORMAT INTERNAL
END
```

This creates the following Master File:

```
FILE=HINT3, SUFFIX=FIX
SEGMENT=HINT3, SEGTYPE=S0
  FIELDNAME=CAR           ,ALIAS=E01   ,USAGE=A16   ,ACTUAL=A16   , $
  FIELDNAME=COUNTRY       ,ALIAS=E02   ,USAGE=A10   ,ACTUAL=A10   , $
  FIELDNAME=RETAIL_COST   ,ALIAS=E03   ,USAGE=I6    ,ACTUAL=I02   , $
  FIELDNAME=DEALER_COST   ,ALIAS=E04   ,USAGE=P4    ,ACTUAL=P02   , $
  FIELDNAME=SEATS         ,ALIAS=E05   ,USAGE=I4    ,ACTUAL=I01   , $
```

The ACTUAL formats for the overridden fields are I2, P2, and I1. DEALER\_COST has an ACTUAL of P2 because P3, the format override, means 3 display digits that can be stored in 2 actual digits. Note that the alphanumeric field is also not padded.

### **Creating A Subquery or Sequential File With HOLD FORMAT SQL\_SCRIPT**

When used in a request against a relational data source, the HOLD FORMAT SQL\_SCRIPT command generates the SQL SELECT statement needed to execute the current query and stores it in the application folder as a file with a .sql extension along with the Master and Access File pair that describes the SQL answer set.

When used in a request against any other type of data source, the HOLD FORMAT SQL\_SCRIPT command executes the current query and stores the retrieved values in the application folder as a sequential file with a .ftm extension along with the Master File that describes the retrieved data.

You can use the output from HOLD FORMAT SQL\_SCRIPT as the target file for the DB\_INFILE function. For information about the DB\_INFILE function, see the *Using Functions* manual.

**Note:** Once you have the .sql file and its accompanying Master File, you can customize the .sql file using global Dialogue Manager variables. You must declare these global variables in the Master File. For information about parameterizing Master Files with global variables, see the *Describing Data With WebFOCUS Language* manual.

**Syntax:**      **How to Create an SQL Script or Sequential File Using HOLD FORMAT SQL\_SCRIPT**

```
ON TABLE HOLD AS script_name FORMAT SQL_SCRIPT
```

where:

*script\_name*

Is the name of the .sql file or the .ftm file created as a result of the HOLD FORMAT SQL\_SCRIPT command.

**Example:**      **Creating an SQL Script File Using HOLD FORMAT SQL\_SCRIPT**

The following request against the WF\_RETAIL relational data source creates an SQL Script file in the baseapp application:

```
APP HOLD baseapp
TABLE FILE wf_retail_lite
SUM BUSINESS_REGION STATE_PROV_CODE_ISO_3166_2
BY BUSINESS_REGION NOPRINT BY STATE_PROV_CODE_ISO_3166_2 NOPRINT
WHERE BUSINESS_REGION EQ 'North America' OR 'EMEA'
WHERE STATE_PROV_CODE_ISO_3166_2 EQ 'AR' OR 'IA' OR 'KS' OR 'KY' OR 'WY' OR
'CT' OR 'MA' OR '04' OR '11' OR '14'
OR 'NJ' OR 'NY' OR 'RI'
ON TABLE HOLD AS RETAIL_SCRIPT FORMAT SQL_SCRIPT
END
```

WF\_RETAIL is a sample data source you can create by right-clicking an application on the Reporting Server Web Console and selecting *New* and then *Samples* from the context menu.

The result of this request is a script file named retail\_script.sql and a corresponding Master and Access File.

The retail\_script.sql file contains the following SQL SELECT statement:

```
SELECT          MAX(T3."BUSINESS_REGION") AS "VB001_MAX_BUSINESS_REGION",
MAX(T3."STATE_PROV_CODE_ISO_3166_2")
AS "VB002_MAX_STATE_PROV_CODE_ISO_"      FROM      wrd_wf_retail_geography
T3
WHERE           (T3."STATE_PROV_CODE_ISO_3166_2" IN('AR', 'IA', 'KS', 'KY', 'WY',
'CT', 'MA', '04', '11', '14', 'NJ', 'NY', 'RI'))
AND            (T3."BUSINESS_REGION" IN('North America', 'EMEA'))      GROUP BY
T3."BUSINESS_REGION",      T3."STATE_PROV_CODE_ISO_3166_2"
```

The retail\_script.mas Master File follows:

```
FILENAME=RETAIL_SCRIPT, SUFFIX=MSODBC , $
  SEGMENT=RETAIL_SCRIPT, SEGTYPE=S0, $
    FIELDNAME=BUSINESS_REGION, ALIAS=VB001_MAX_BUSINESS_REGION, USAGE=A15V,
    ACTUAL=A15V,
    MISSING=ON,
    TITLE='Customer,Business,Region', $
    FIELDNAME=STATE_PROV_CODE_ISO_3166_2,
    ALIAS=VB002_MAX_STATE_PROV_CODE_ISO_, USAGE=A5V, ACTUAL=A5V,
    MISSING=ON,
    TITLE='Customer,State,Province,ISO-3166-2,Code', $
```

The retail\_script.acx Access File follows:

```
SEGNAME=RETAIL_SCRIPT,
  CONNECTION=CON01,
  DATASET=RETAIL_SCRIPT.SQL,
  SUBQUERY=Y, $
```

### **Example:** Creating a Sequential File Using HOLD FORMAT SQL\_SCRIPT

The following request against the EMPLOYEE data source creates a sequential file containing the values retrieved by the request along with a corresponding Master File:

```
APP HOLD baseapp
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME
WHERE DEPARTMENT EQ 'MIS'
ON TABLE HOLD AS EMPVALUES FORMAT SQL_SCRIPT
END
```

The sequential file empvalues.ftm contains the following data:

SMITH	MARY	JONES	DIANE	MCCOY
JOHN	BLACKWOOD	ROSEMARIE	GREENSPAN	MARY
CROSS	BARBARA			

The empvalues.mas Master File follows:

```
FILENAME=EMPVALUES, SUFFIX=DATREC , IOTYPE=BINARY, $
  SEGMENT=EMPVALUE, SEGTYPE=S0, $
    FIELDNAME=LAST_NAME, ALIAS=E01, USAGE=A15, ACTUAL=A15, $
    FIELDNAME=FIRST_NAME, ALIAS=E02, USAGE=A10, ACTUAL=A10, $
    FIELDNAME=NULLFLAG, ALIAS=__NULLFLAG__, USAGE=A2, ACTUAL=A2B,
    ACCESS_PROPERTY=(INTERNAL), $
```

## Creating a Structured HOLD File

Structured HOLD Files facilitate migration of data sources and reports between operating environments.

Other HOLD formats capture data from the original sources and may retain some implicit structural elements from the request itself. However, they do not propagate most of the information about the original data sources accessed and their inter-relationships to the HOLD Master File or data source. Structured HOLD files, however, extract the data to a structure that parallels the original data sources. Subsequent requests against the HOLD file can use these retained data relationships to recreate the same types of relationships in other environments or in other types of data sources.

A Structured HOLD File can be created in ALPHA, BINARY, or FOCUS format:

- ❑ A Structured HOLD file created in either ALPHA or BINARY format is a flat file that saves the segment instances that contain the data that satisfy the conditions of the TABLE request. Multiple segments are generated based on the original structure read by the TABLE request. Segments are identified by assigning a RECTYPE for differentiation. Child segments in the original data source become a unique segment in the HOLD file
- ❑ A Structured HOLD file in FOCUS format uses normal FOCUS segments to retain the original structure.

In all cases the HOLD file contains all of the original segment instances required to provide the complete report based on the TABLE request itself. Regardless of the display command used in the original request (PRINT, LIST, SUM, COUNT), the Structured HOLD File is created as if the request used PRINT. Aggregation is ignored.

The HOLD file contains either all of the fields in the structure identified by the request that are used to satisfy the request, or all of the display fields and BY fields. The file does not contain DEFINE fields not specifically referenced in the request. It does contain all fields needed to evaluate any DEFINE fields referenced in the request.

Structured HOLD files are only supported for TABLE and TABLEF commands. They can be created anywhere a HOLD file is supported. You must activate Structured HOLD files in a specific request by issuing the ON TABLE SET EXTRACT command in the request prior to creating the Structured HOLD File.

### ***Syntax:***      **How to Activate Structured HOLD Files for a Request**

```
ON TABLE SET EXTRACT {ON|*|OFF}
```

where:

ON

Activates Structured HOLD Files for this request and extracts all fields mentioned in the request.

\*

Activates Structured HOLD Files for this request and indicates that a block of extract options follows. For example, you can exclude specific fields from the Structured HOLD File. For information, see [How to Specify Options for Generating Structured HOLD Files](#) on page 553.

[OFF](#)

Deactivates Structured HOLD files for this request. OFF is the default value.

### **Syntax:** How to Create a Structured HOLD File

Before issuing the HOLD command, activate Structured HOLD Files for the request by issuing the ON TABLE SET EXTRACT command described in [How to Activate Structured HOLD Files for a Request](#) on page 552. Then issue the HOLD command to create the Structured HOLD File:

```
[ON TABLE] {HOLD|PCHOLD} [AS name] FORMAT {ALPHA|BINARY|FOCUS}
```

where:

*name*

Is the name of the HOLD file. If omitted, the name becomes HOLD by default.

[FORMAT](#)

Is ALPHA, BINARY or FOCUS.

**Note:** You can issue a SET command to set the default HOLD format to either ALPHA or BINARY:

```
SET HOLDFORMAT=ALPHA
SET HOLDFORMAT=BINARY
```

### **Syntax:** How to Specify Options for Generating Structured HOLD Files

To specify options for creating the extract, such as excluding specific fields, use the \* option of the SET EXTRACT command:

```
ON TABLE SET EXTRACT *
EXCLUDE = (fieldname1, fieldname2, fieldname3 , ..., fieldnamen), $
FIELDS={ALL|EXPLICIT}, $
ENDEXTRACT
ON TABLE HOLD AS name FORMAT {ALPHA|BINARY|FOCUS}
```

where:

[EXCLUDE](#)=(*fieldname1*, *fieldname2*, *fieldname3*, ..., *fieldnamen*)

Excludes the specified fields from the HOLD file.

, \$

Is required syntax for delimiting elements in the extract block.

### ALL

Includes all real fields and all DEFINE fields that are used in running the request.

### EXPLICIT

Includes only those real fields and DEFINE fields that are in the display list or the BY sort field listing. DEFINE fields that are not explicitly referenced, and fields that are used to evaluate DEFINES, are not included.

### ENDEXTRACT

Ends the extract block.

### *Example:* Creating a Structured HOLD File in ALPHA Format

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME JOBCODE ED_HRS
BY DEPARTMENT
BY HIGHEST SALARY
ON TABLE SET EXTRACT ON
ON TABLE HOLD FORMAT ALPHA
END
```

This request produces the following HOLD Master File:

```
FILENAME=HOLD      , SUFFIX=FIX      , $
SEGMENT=EMPINFO, SEGTYPE=S0, $
  FIELDNAME=RECTYPE, ALIAS=R, USAGE=A3, ACTUAL=A3, $
  FIELDNAME=LAST_NAME, ALIAS='LN', USAGE=A15, ACTUAL=A15, $
  FIELDNAME=FIRST_NAME, ALIAS='FN', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=DEPARTMENT, ALIAS='DPT', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=ED_HRS, ALIAS='OJT', USAGE=F6.2, ACTUAL=A06, $
SEGMENT=PAYINFO, SEGTYPE=S0, PARENT=EMPINFO, $
  FIELDNAME=RECTYPE, ALIAS=1, USAGE=A3, ACTUAL=A3, $
  FIELDNAME=SALARY, ALIAS='SAL', USAGE=D12.2M, ACTUAL=A12, $
  FIELDNAME=JOBCODE, ALIAS='JBC', USAGE=A3, ACTUAL=A03, $
```

Note the RECTYPE field generated for ALPHA or BINARY Structured HOLD files. Each record in the HOLD file begins with the RECTYPE to indicate the segment to which it belonged in the original structure. The root segment has RECTYPE=R. The RECTYPES for other segments are sequential numbers assigned in top to bottom, left to right order.

Following are the first several records in the HOLD file:

```

R  STEVENS      ALFRED      PRODUCTION  25.00
1    11000.00A07
1    10000.00A07
R  SMITH        MARY        MIS          36.00
1    13200.00B14
R  JONES        DIANE       MIS          50.00
1    18480.00B03
1    17750.00B02
R  SMITH        RICHARD     PRODUCTION  10.00
1    9500.00A01
1    9050.00B01

```

**Example:** Creating a Structured HOLD File in FOCUS Format

```

TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME JOBCODE ED_HRS
BY DEPARTMENT
BY HIGHEST SALARY
ON TABLE SET EXTRACT ON
ON TABLE HOLD FORMAT FOCUS
END

```

This request produces the following HOLD Master File:

```

FILENAME=HOLD      , SUFFIX=FOC      , $
SEGMENT=EMPINFO, SEGTYPE=S0, $
  FIELDNAME=LAST_NAME, ALIAS='LN', USAGE=A15, $
  FIELDNAME=FIRST_NAME, ALIAS='FN', USAGE=A10, $
  FIELDNAME=DEPARTMENT, ALIAS='DPT', USAGE=A10, $
  FIELDNAME=ED_HRS, ALIAS='OJT', USAGE=F6.2, $
SEGMENT=PAYINFO, SEGTYPE=S0, PARENT=EMPINFO, $
  FIELDNAME=SALARY, ALIAS='SAL', USAGE=D12.2M, $
  FIELDNAME=JOBCODE, ALIAS='JBC', USAGE=A3, $

```

**Example:** Reconstituting a Structured HOLD File

The following request reconstitutes the original FOCUS data source from the Structured HOLD File created in the example named *Creating a Structured HOLD File in ALPHA Format*:

```

TABLE FILE HOLD
PRINT LAST_NAME FIRST_NAME JOBCODE ED_HRS
BY DEPARTMENT
BY HIGHEST SALARY
ON TABLE SET EXTRACT ON
ON TABLE HOLD AS RECONST FORMAT FOCUS
END

```

This request produces the following Master File:

```

FILENAME=RECONST      , SUFFIX=FOC      , $
  SEGMENT=EMPINFO, SEGTYPE=S0, $
    FIELDNAME=LAST_NAME, ALIAS='LN', USAGE=A15, $
    FIELDNAME=FIRST_NAME, ALIAS='FN', USAGE=A10, $
    FIELDNAME=DEPARTMENT, ALIAS='DPT', USAGE=A10,
    FIELDNAME=ED_HRS, ALIAS='OJT', USAGE=F6.2, $
  SEGMENT=PAYINFO, SEGTYPE=S0, PARENT=EMPINFO, $
    FIELDNAME=SALARY, ALIAS='SAL', USAGE=D12.2M, $
    FIELDNAME=JOBCODE, ALIAS='JBC', USAGE=A3, $

```

The following request prints the report output:

```

TABLE FILE RECONST
PRINT LAST_NAME FIRST_NAME JOBCODE ED_HRS
BY DEPARTMENT
BY HIGHEST SALARY
END

```

The output is:

DEPARTMENT	SALARY	LAST_NAME	FIRST_NAME	JOBCODE	ED_HRS
-----	-----	-----	-----	-----	-----
MIS	\$27,062.00	CROSS	BARBARA	A17	45.00
	\$25,775.00	CROSS	BARBARA	A16	45.00
	\$21,780.00	BLACKWOOD	ROSEMARIE	B04	75.00
	\$18,480.00	JONES	DIANE	B03	50.00
		MCCOY	JOHN	B02	.00
	\$17,750.00	JONES	DIANE	B02	50.00
	\$13,200.00	SMITH	MARY	B14	36.00
	\$9,000.00	GREENSPAN	MARY	A07	25.00
	\$8,650.00	GREENSPAN	MARY	B01	25.00
	\$29,700.00	BANNING	JOHN	A17	.00
PRODUCTION	\$26,862.00	IRVING	JOAN	A15	30.00
	\$24,420.00	IRVING	JOAN	A14	30.00
	\$21,120.00	ROMANS	ANTHONY	B04	5.00
	\$16,100.00	MCKNIGHT	ROGER	B02	50.00
	\$15,000.00	MCKNIGHT	ROGER	B02	50.00
	\$11,000.00	STEVENS	ALFRED	A07	25.00
	\$10,000.00	STEVENS	ALFRED	A07	25.00
	\$9,500.00	SMITH	RICHARD	A01	10.00
	\$9,050.00	SMITH	RICHARD	B01	10.00



**Example:** Excluding Fields From Structured HOLD Files

This request excludes the SALARY field used for sequencing.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME JOBCODE ED_HRS
BY DEPARTMENT
BY HIGHEST SALARY
ON TABLE SET EXTRACT *
EXCLUDE=(SALARY), $
ENDEXTRACT
ON TABLE HOLD FORMAT FOCUS
END
```

This request produces the following HOLD Master File:

```
FILENAME=HOLD      , SUFFIX=FOC      , $
  SEGMENT=EMPINFO, SEGTYPE=S0, $
    FIELDNAME=LAST_NAME, ALIAS='LN', USAGE=A15, $
    FIELDNAME=FIRST_NAME, ALIAS='FN', USAGE=A10, $
    FIELDNAME=DEPARTMENT, ALIAS='DPT', USAGE=A10, $
    FIELDNAME=ED_HRS, ALIAS='OJT', USAGE=F6.2, $
  SEGMENT=PAYINFO, SEGTYPE=S0, PARENT=EMPINFO, $
    FIELDNAME=JOBCODE, ALIAS='JBC', USAGE=A3, $
```

**Reference:** Elements Included in a Structured HOLD File

Structured HOLD files contain all original segment instances required to complete the TABLE or TABLEF request. Regardless of the display command used in the original request (PRINT, LIST, SUM, or COUNT), the structured HOLD file will be created as if the command was PRINT.

Specifically, the extract file contains the following elements:

- ☐ All real fields named in the request such as display objects, sort fields, and fields used in selection criteria (WHERE/IF tests).  
  
Note that fields referenced multiple times in a request are included only once in the HOLD file.
- ☐ Fields used in FILTER FILE condition.
- ☐ Prefix operators are ignored except for ALL. (which just affects the amount of data collected and does not imply a calculation).
- ☐ Field based reformatting (FIELD1/FIELD2=) causes the original field and the format field to be included.

- ❑ A GROUP field if referenced explicitly or when all of its members are referenced in the request.

**Note:** If a group member is specifically excluded (EXCLUDE) or not referenced, its GROUP is not added to the extract Master File (this applies to nested and overlapping groups, as well). If a GROUP and its elements are all named in a request, the GROUP is not added as a real field in the extract HOLD file.

- ❑ For FIELDS=ALL, all DEFINE fields used in the request become real fields in the structured HOLD File and are included along with other fields used in the DEFINE expression (including other DEFINE fields). Use EXCLUDE to reduce the number of fields included in the EXTRACT output.
- ❑ For FIELDS=EXPLICIT, display objects and sort fields are included. DEFINE fields become real fields if referenced in the request, but fields used to create them will not be included unless referenced explicitly. This reduces the number of fields returned in the request.

### **Reference:** Elements Not Included in a Structured HOLD File

- ❑ Prefix operators on WHERE fields are evaluated in data selection but not included in the extract output.
- ❑ Prefix operators on display objects are ignored (except ALL).
- ❑ Using Structured HOLD File syntax in MATCH, MORE, and GRAPH requests produces error messages and exits the procedure.
- ❑ WHERE/IF TOTAL tests are not supported in Structured HOLD File requests and result in cancellation of the request.
- ❑ Reformatting of real fields is ignored (only the real field is included).
- ❑ Computed fields are not included, but fields used in COMPUTE expressions are included in the extract file.

### **Reference:** Structural and Behavioral Notes

- ❑ Structured HOLD File requests are subject to the same limitations on number and size of fields as any other TABLE request.

**Structural Notes**

- ☐ The following SET parameters are turned off or ignored in Structured HOLD File requests:
  - ☐ AUTOINDEX
  - ☐ AUTOPATH
  - ☐ AUTOSTRATEGY
  - ☐ EXTHOLD
  - ☐ EXTSORT
  - ☐ HOLDATTR
- ☐ All SET and ON TABLE SET commands used to control output format are ignored in creating the extract file.
- ☐ Alternate views are respected and reflected in the structure of the extract file.
- ☐ Indexed views specified in the request are respected and reflected in the structure of the output file.
- ☐ If a request would generate a file containing two independent orphan segments because the parent segment is specifically excluded, a dummy system segment is created in the to act as parent of the two unrelated segments. There is only one instance of data for that segment. Both orphan segments refer to that system segment as parent. If the parent is missing because it was not mentioned in the request, it is activated during the request and included as the parent the segments.
- ☐ In the event that two unique (U) segments are included without the parent segment, the unique segments are converted to segments with SEGTYPE S0 that reference the system segment as parent.
- ☐ JOIN and JOIN WHERE structures are supported.

**SQL Optimization Notes**

- ☐ SQL optimization for aggregation must be turned off for EXTRACT requests.

**BY/ACROSS/FOR Notes**

- ☐ BY and ACROSS sort fields become additional display objects.
- ☐ BY. . .ROWS and ACROSS . . .COLUMNS function only as implicit WHEREs to limit field values included.

- ☐ FOR fields are included.
- ☐ RECAP fields are excluded (like COMPUTEs).
- ☐ Summarization fields referencing previously identified fields are ignored in creating Structured HOLD Files. These include: SUMMARIZE, RECOMPUTE, SUBTOTAL, SUB-TOTAL ACROSS-TOTAL, ROW-TOTAL and COLUMN-TOTAL.

### Formatting Notes

- ☐ Structured HOLD File processing ignores all formatting elements, including: IN, OVER, NOPRINT, SUP-PRINT, FOLD-LINE, SKIP-LINE, UNDER-LINE, PAGE-BREAK, TABPAGENO, AS, and column title justification. However, fields referenced within formatting commands, such as HEADING, FOOTING, SUBHEAD, and SUBFOOT, and any WHEN expressions associated with them, are included.
- ☐ All STYLE and STYLESHEET commands are ignored in producing extract output.
- ☐ AnV and AnW fields are supported. TX fields are exported in FOCUS files only.
- ☐ In the event that the FIELDNAME and ALIAS are the same for a real field and that field is redefined as itself (possibly with a different format), two fields are created in the HOLD Master File with identical field names and aliases. In this situation, the second version of the field can never be accessed if referenced by name. You can use FIELDS=EXPLICIT to include only the second version of the field. The following DEFINE illustrates an example of creating a duplicate field name and alias:

```
DEFINE FILE CAR  
COUNTRY/A25=COUNTRY;  
END
```

### DBA Notes

- ☐ DBA controls on source files are respected when running Structured HOLD File requests with the exception of RESTRICT=NOPRINT, where fields named in a request are not displayed (such fields cannot be exported and should be specifically EXCLUDED).
- ☐ DBA restrictions do not carry over to the HOLD Master File.

### Reconstituting Extract Files

- ☐ To reconstitute a FOCUS or flat file from a Structured HOLD file, you use the same syntax used to generate the Structured HOLD File. The ON TABLE SET EXTRACT syntax must be used to preserve multipath structures.

- ❑ All reconstituted FOCUS segments are SEGTYPE=SO, as neither KEY nor INDEX information is retained. An INDEX can be reinserted using REBUILD INDEX.

## Specifying MIME Types for WebFOCUS Reports

In addition to creating reports in HTML format for display in a web browser, you can generate reports that can be returned to the browser and opened in a desktop application or in a helper application.

In order for the browser to recognize and call the correct desktop application, you must associate the MIME (Multipurpose Internet Mail Extension) type of the report with a specific application. After the association for a MIME type has been established, you can take advantage of the ability to exchange different kinds of data files on the Internet or through email. These data files can include audio, video, and images, as well as application programs. For example, assuming that the required MIME types have been defined, you can receive a report as an email attachment in PDF or DOC format and launch the report from your browser in Adobe Reader or in Microsoft Word, respectively.

### **Reference:** Supported MIME Types and Applications

WebFOCUS supports the following MIME types and desktop applications:

MIME Type	Application	File Extension
application/PostScript	GhostView.  GhostView is a third-party product and is available at the following website:  <a href="http://www.cs.wisc.edu/~ghost/gsview/index.htm">www.cs.wisc.edu/~ghost/gsview/index.htm</a> .	.ps
application/x-prn	Microsoft Excel or equivalent spreadsheet application.	.prn
application/x-dif	Microsoft Excel or equivalent spreadsheet application.	.dif
application/ms word	Microsoft Word.	.doc
application/vnd.ms-excel	Microsoft Excel.	.xls

<b>MIME Type</b>	<b>Application</b>	<b>File Extension</b>
application/x-msexcel	Microsoft Excel 2000.	.htx
application/pdf	Adobe Reader.	.pdf
text/plain	Microsoft Notepad or equivalent text processor.	.wp or .hts
text/html	Microsoft Notepad or equivalent HTML editor.	.html
text/htm	Microsoft Notepad or equivalent HTML editor.	.htm
image/gif	Microsoft Paint or equivalent graphic editor.	.gif
image/jpg	Microsoft Paint or equivalent graphic editor.	.jpg

**Procedure: How to Specify MIME Types in Windows**

To specify a MIME type in Windows:

1. Open the Windows Explorer.
2. Select *Folder Options* from the View menu.
3. Select the *File Types* tab.
4. Select a file type in the Registered File Types list box, then click the *New Type* button.

The Add New File Type dialog box opens.

5. Type the following information. If necessary, refer to the table on the previous page.
  - ☐ In the Description of Type field, type a description of your choice.
  - ☐ In the Associated Extension field, type the extension relevant to the file type you specified.
  - ☐ In the Content Type (MIME) field, type the MIME type relevant to the file type and extension you specified.
6. Click *OK* to save your changes.





## Choosing a Display Format

---

You can choose from several different display formats when you display a report on the screen. Some display formats are best suited for particular kinds of uses. For example, you can choose to display the report as:

- ☐ An HTML page, which is optimized for display in a web browser.
- ☐ A PDF document, which is useful when you want the report to look the same whether displayed on a screen or printed.
- ☐ A DHTML file, which is HTML output that has most of the features normally associated with output formatted for printing, such as PDF or PostScript output.
- ☐ An Excel 2007 worksheet, where you can work with the data in Excel 2007 or higher.
- ☐ An Excel 2000 worksheet, where you can work with the data in Excel 2000 or 2003.

You can learn which display formats are available in [Report Display Formats](#) on page 566.

If you wish to send a report to a file instead of to the screen, you can learn about the file formats that are available in [Saving and Reusing Your Report Output](#) on page 471.

### **In this chapter:**

- ☐ [Report Display Formats](#)
  - ☐ [Preserving Leading and Internal Blanks in Report Output](#)
  - ☐ [Using Web Display Format: HTML](#)
  - ☐ [Using Print Display Formats: PDF, PS](#)
  - ☐ [Using Word Processing Display Formats: DOC, WP](#)
  - ☐ [Saving Report Output in Excel XLSX Format](#)
  - ☐ [Using PowerPoint PPT Display Format](#)
  - ☐ [Saving Report Output in PPTX Format](#)
-

## Report Display Formats

You can choose from among several display formats for your report:

- ❑ **Web format:** HTML. For more information, see [Using Web Display Format: HTML](#) on page 571.
- ❑ **Print formats:** PDF (Adobe Acrobat Portable Document Format) and PostScript (PS). For more information, see [Using Print Display Formats: PDF, PS](#) on page 574.
- ❑ **Word-processing formats:** WP and DOC. For more information, see [Using Word Processing Display Formats: DOC, WP](#) on page 602.
- ❑ **Worksheet formats:** Excel 2007/2010 XML-based format, Excel 2000/2003 HTML-based format, with variations for Excel 2000 PivotTable and Excel 2000 FORMULA, Excel 97 HTML-based format, and Excel binary format. For more information, see [Saving Report Output in Excel XLSX Format](#) on page 603.

**A note about DHTML and HTML:** DHTML is the absolute positioning version of HTML. As architected, format HTML generates output in a table-based format that leaves the exact positioning to the browser that is presenting the report. Format DHTML on the other hand is designed to render with the user-defined positioning in the same way as PDF. This means things should position on the page precisely as defined in the report procedure. PDF, DHTML, PPT, PPTX, and PS are position-based. HTML and EXL2K are table or cell based. Therefore, DHTML output looks more like PDF rather than HTML.

For information about which file formats are available for saving and reusing (as opposed to displaying) report data, see [Saving and Reusing Your Report Output](#) on page 471.

### **Syntax:**

#### **How to Choose a Display Format Using PCHOLD**

You can display a report on screen using the ON TABLE PCHOLD command in a report request.

ON TABLE PCHOLD FORMAT *formatname*

where:

*formatname*

Can be one of the following:

DOC	Specifies that the report will be displayed as a plain-text word processing document, with page breaks, in Microsoft Word within your web browser. See <a href="#">ok</a>  <a href="#">Using Word Processing Display Formats: DOC, WP</a> on page 602.
EXCEL	Specifies that the report will be displayed as an Excel spreadsheet. See <a href="#">Saving Report Output in Excel XLSX Format</a> on page 603.
XLSX	Specifies that the report will be displayed as an Excel 2007/2010 worksheet. See <a href="#">Saving Report Output in Excel XLSX Format</a> on page 603.
EXL2K	Specifies that the report will be displayed as an Excel 2000/2003 worksheet.
EXL2K FORMULA	Specifies that the report will be displayed as an Excel 2000/2003 worksheet, with WebFOCUS totals and other calculated values translated to active Excel formulas. (If your report does not contain any formulas, consider using EXL2K format since EXL2K FORMULA requires additional processing time.)
EXL2K PIVOT	Specifies that the report will be displayed as an Excel 2000/2003 PivotTable.
EXL97	Specifies that the report will be displayed as an Excel 97 worksheet.
HTML	Specifies that the report will be displayed as an HTML page. See <a href="#">Using Web Display Format: HTML</a> on page 571.
PDF	Specifies that the report will be displayed as a PDF document (Adobe Acrobat's Portable Document Format). See <a href="#">Using PDF Display Format</a> on page 575.
PostScript (PS)	Specifies that the report will be displayed as a PostScript document. You must have installed a third party tool capable of displaying PS. See <a href="#">Using PostScript (PS) Display Format</a> on page 580.

WP	Specifies that the report will be displayed as a plain-text word processing document in the web browser. See <a href="#">Using Word Processing Display Formats: DOC, WP</a> on page 602.
----	--

**Syntax:**      **How to Choose a Display Format Using SET ONLINE-FMT**

For a limited set of formats, you can display a report on screen using the SET command ONLINE-FMT parameter.

Outside of a report request, use the following syntax to specify a format for all report requests within the procedure

```
SET ONLINE-FMT = formatname
```

Within a report request, use the following syntax to specify a format for that request only

```
ON TABLE SET ONLINE-FMT formatname
```

where:

*formatname*

Can be one of the following:

HTML (default)	Specifies that the report will be displayed as an HTML page. See <a href="#">Using Web Display Format: HTML</a> on page 571.
PDF	Specifies that the report will be displayed as a PDF document (Adobe Acrobat Portable Document Format). See <a href="#">Using PDF Display Format</a> on page 575.
XLSX	Specifies that the report will be displayed as an Excel 2007/2010 worksheet. See <a href="#">Saving Report Output in Excel XLSX Format</a> on page 603.
EXL2K	Specifies that the report will be displayed as an Excel 2000/2003 worksheet.
EXL97	Specifies that the report will be displayed as an Excel 97 worksheet.

PostScript (PS)	Specifies that the report will be displayed as a PostScript document. You must have installed a third party tool capable of displaying PS. See <a href="#">Using PostScript (PS) Display Format</a> on page 580.
STANDARD	Specifies that the report will be displayed using a legacy character-based and line-based layout and a monospaced font.

**Tip:** ONLINE-FMT syntax will be superseded by PCHOLD syntax in future releases of WebFOCUS (see [How to Choose a Display Format Using PCHOLD](#) on page 566). At present, they can be used interchangeably.

### **Reference:** Specifying MIME Types for WebFOCUS Reports

In addition to creating reports in HTML format for display in a web browser, you can generate reports that can be returned to the browser and opened in a desktop application or in a helper application. In order for the browser to recognize and call the correct desktop application, you must associate the MIME (Multipurpose Internet Mail Extension) type of the report with a specific application.

For details, see the *Developing Reporting Applications* manual.

## **Preserving Leading and Internal Blanks in Report Output**

By default, HTML browsers and Excel remove leading and trailing blanks from text and compress multiple internal blanks to a single blank.

If you want to preserve leading and internal blanks in HTML and EXL2K report output, you can issue the SET SHOWBLANKS=ON command.

Even if you issue this command, trailing blanks will not be preserved except in heading, subheading, footing, and subfooting lines that use the default heading or footing alignment.

**Syntax:**      **How to Preserve Leading and Internal Blanks in HTML and EXL2K Reports**

In a FOCEXEC or in a profile, use the following syntax:

```
SET SHOWBLANKS = {OFF|ON}
```

In a request, use the following syntax

```
ON TABLE SET SHOWBLANKS {OFF|ON}
```

where:

OFF

Removes leading blanks and compresses internal blanks in HTML and EXL2K report output.

ON

Preserves leading blanks and internal blanks in HTML and EXL2K report output. Also preserves trailing blanks in heading, subheading, footing, and subfooting lines that use the default heading or footing alignment.

**Example:**      **Preserving Leading and Internal Blanks in HTML and EXL2K Report Output**

The following request creates a virtual field that adds leading blanks to the value ACTION and both leading and internal blanks to the values TRAIN/EX and SCI/FI in the CATEGORY field. It also adds trailing blanks to the value COMEDY:

```
SET SHOWBLANKS = OFF
DEFINE FILE MOVIES
NEWCAT/A30 = IF CATEGORY EQ 'ACTION' THEN '  ACTION'
             ELSE IF CATEGORY EQ 'SCI/FI' THEN 'SCIENCE  FICTION'
             ELSE IF CATEGORY EQ 'TRAIN/EX' THEN '  TRAINING    EXERCISE'
             ELSE IF CATEGORY EQ 'COMEDY' THEN 'COMEDY      '
             ELSE                'GENERAL';
END
TABLE FILE MOVIES
SUM CATEGORY LISTPR/D12.2 COPIES
BY NEWCAT

ON TABLE SET STYLE *
GRID=OFF,$
TYPE=REPORT, FONT=COURIER NEW,$
ENDSTYLE
END
```

With SHOWBLANKS OFF, these additional blanks are removed:

<u>NEWCAT</u>	<u>CATEGORY</u>	<u>LISTPR</u>	<u>COPIES</u>
TRANING EXERCISE	TRAIN/EX	119.87	10
ACTION	ACTION	94.82	14
COMEDY	COMEDY	154.80	19
GENERAL	MYSTERY	1,216.82	67
SCIENCE FICTION	SCI/FI	114.84	7

With SHOWBLANKS ON, the additional leading and internal blanks are preserved. Note that trailing blanks are not preserved:

<u>NEWCAT</u>	<u>CATEGORY</u>	<u>LISTPR</u>	<u>COPIES</u>
TRANING EXERCISE	TRAIN/EX	119.87	10
ACTION	ACTION	94.82	14
COMEDY	COMEDY	154.80	19
GENERAL	MYSTERY	1,216.82	67
SCIENCE FICTION	SCI/FI	114.84	7

## Using Web Display Format: HTML

You can display a report as an HTML page. HTML supports most style sheet options (especially when used with an internal cascading style sheet), allowing for full report formatting.

By default, leading and internal blanks are compressed on the report output. For information on preserving them, see [Preserving Leading and Internal Blanks in Report Output](#) on page 569.

For more information, see [Controlling Report Formatting](#) on page 1139.

HTML is the default display format when WebFOCUS is installed. An HTML report opens in your web browser.

If you do not wish to rely on the default, you can specify that a report display as an HTML page when you run the report. You can use either:

- ☐ **PCHOLD command.** For more information, see [How to Choose a Display Format Using PCHOLD](#) on page 566.
- ☐ **ONLINE-FMT parameter** of the SET command. For more information, see [How to Choose a Display Format Using SET ONLINE-FMT](#) on page 568.

HTML display format requires that the SET command's STYLESHEET parameter be set to any value *except* OFF. Appropriate values include ON (the default), the name of a StyleSheet file, or an inline StyleSheet (\*).

Reporting and formatting options that are supported for HTML are described and illustrated extensively throughout the WebFOCUS language documentation.

You can additionally customize the display of HTML reports with any JavaScript or VBScript function using the JSURL SET parameter. For details, see the *Developing Reporting Applications* manual.

### **Example:** Customizing the Display of an HTML Report

The following example illustrates how you can customize the display of an HTML report by calling your own JavaScript function in addition to the Information Builders default JavaScript functions. Use the JSURL SET parameter to accomplish this.

The JavaScript function shown here disables the right-click menu when you run a report.

1. Create a js file and save it in a location accessible by the web server.

For example, the following disables the right-click menu in an HTML report:

```
function setnocontextclick () {  
    if (document.body != null) {  
        document.body.oncontextmenu=new Function("return false");  
    }  
    else  
        window.setTimeout("setnocontextclick()",100);  
}  
function killmenuOnLoadFunc(arrayofonloads,currentindex) {  
    setnocontextclick();  
}
```

This file is saved as killmenu.js in the ibi\_apps/ibi\_html directory.

**Note:** The onload function must be named in the format.

*customfunctionnameOnLoadFunc*

where:

*customfunctionname*

Is the name of the JavaScript file that contains the function code.

2. Add the JSURL parameter to your TABLE request. You can add the command to the edasprof.prf file if you want the js file to run with every HTML report that is run on that server.
3. Run the report.



```

SET JSURL=/ibi_apps/ibi_html/killmenu.js
TABLE FILE CENTORD
SUM QUANTITY
BY PLANTLNG
END

```

The right-click menu option is not available in the report output.

### **Example:** Disabling Default WebFOCUS JavaScript Functions

You can disable or modify default WebFOCUS JavaScript functions using the JSURL SET parameter. The following example illustrates how all WebFOCUS default functions can be displayed in an alert box and disabled.

1. Create a js file and save it in a location accessible by the web server.

This file is saved as `disable.js` in the `ibi_apps/ibi_html` directory. The `arrayofonloads` array consists of two string parameters, `str1` and `str2`. `str1` is the name of the function to call on load. `str2` is a Boolean (`true/false`) that indicates whether or not to perform the action described by `str1`. The `currentindex` parameter is a sequence number that defines the order in which the function is loaded when the page is displayed.

```

function disableOnLoadFunc(arrayofonloads,currentindex) {
    buffer = "";
    for (var index=0;index<arrayofonloads.length;index++) {
        buffer += arrayofonloads[index].str1+"\n" ;
        arrayofonloads[index].str2=false;
    }
    alert(buffer);
}

```

2. Add the JSURL parameter to your TABLE request.
3. Run the report.

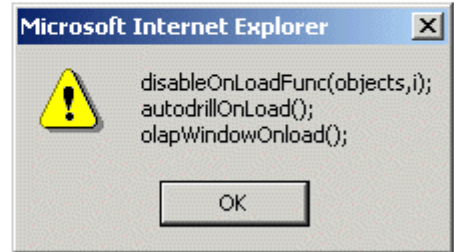
```

-OLAP ON
SET AUTODRILL = ON
SET JSURL=/ibi_apps/ibi_html/disable.js
TABLE FILE CENTORD
SUM QUANTITY
BY PLANTLNG
END

```

The output looks like this:

PAGE 1	
PLANTLNG	Quantity:
Boston	1,033,818
Dallas	390,844
Los Angeles	229,256
Orlando	386,909
Seattle	86,680
St Louis	776,743



**Reference: Usage Notes for HTML Report Output**

- ❑ The default behavior for HTML format when borders are turned on is to display column titles without the underline. To display column titles with underlines when borders are on, set GRID OFF.
- ❑ The AUTOFIT parameter automatically resizes HTML report output to fit the container (window or frame). For procedures that contain multiple report output, if AUTOFIT is set to ON in any of the report output procedures, the setting will apply to all report output on the page.

AUTOFIT is supported using the Accordion, On Demand Paging, HTML TOC, and HFREEZE interactive reporting features.

For more information on the AUTOFIT parameter, see the *Developing Reporting Applications* manual.

**Using Print Display Formats: PDF, PS**

PDF (Adobe Acrobat Portable Document Format) is most often used to distribute and share electronic documents through the web. It is especially useful if you want a report to maintain its presentation and layout regardless of a browser or printer type. For details, see [Using PDF Display Format](#) on page 575.

PS (PostScript format), a print-oriented page description language, is most often used to send a report directly to a printer. While used less frequently as an online display format, you can display PS report output on your monitor before printing it. For details, see [Using PostScript \(PS\) Display Format](#) on page 580.

With the exception of drill-downs, all of the report formatting features that are supported for PDF are also supported for PostScript output.

You can specify that a report display as a PDF or PS document when you run the report. You can use either:

- ❑ **PCHOLD command.** For more information, see [How to Choose a Display Format Using PCHOLD](#) on page 566.
- ❑ **ONLINE-FMT parameter** of the SET command. For more information, see [How to Choose a Display Format Using SET ONLINE-FMT](#) on page 568.

You can combine multiple styled reports into a single PDF or PS file. For details, see [Laying Out the Report Page](#) on page 1249.

PDF and PS reports, including compound reports, can be distributed using ReportCaster. See the ReportCaster documentation for details.

## Using PDF Display Format

You can display a report as a PDF document. PDF (Adobe Acrobat Portable Document Format) supports most StyleSheet attributes, allowing for full report formatting. The wide range of StyleSheet features supported for PDF are described throughout this documentation.

PDF prints and displays a document consistently, regardless of the application software, hardware, and operating system used to create or display the document.

The report opens in Adobe Acrobat or Acrobat Reader within a web browser. To display a PDF report, a computer must have Adobe Acrobat Reader installed. For free downloads of Acrobat Reader, go to <http://www.adobe.com>.

**Limit:** Adobe Acrobat PDF format limits the number of pages, hyperlinks, and images in a document. For information about what limits this creates for a WebFOCUS report in PDF format, see [Saving and Reusing Your Report Output](#) on page 471.

**Other print-oriented display formats.** You can also display a report as a PostScript document. For more information, see [Using PostScript \(PS\) Display Format](#) on page 580.

**Syntax:**      **How to Compress a PDF Output File**

File compression can be used to minimize the physical size of the PDF output file. Using this PDF-specific feature, you can generate smaller PDF files, making them easier to store and distribute, while having no visible effect on the formatting or content of the reports they contain.

```
SET FILECOMPRESS = {ON|OFF}
```

where:

ON

Compresses PDF output files.

OFF

Does not compress PDF output files. OFF is the default value.

This command applies to PDF output only. It is ignored by all other output formats, such as HTML and Excel.

**Displaying Watermarks in PDF Output**

Watermarks are images or text strings that are placed on the bottom layer of a document and displayed through the transparent layered content.

WebFOCUS backcolor does not support transparency. Therefore, standard images placed below it on the page may be obscured. To resolve this, in PDF reports, WebFOCUS mirrors the approach taken by standard printers, and places an opaque image on the top of the document layers. With this approach, the layers of the document will be visible beneath the transparent watermark image.

Watermark images are provided by the report developer. When creating a transparent image, the image needs to be created in GIF format with a transparent background.

**Reference:**      **Inserting Images in PDF Reports With Backcolor**

Watermarks are supported for PDF output in compound reports and in single TABLE requests. Each document supports a single active watermark image. This image is designated as the watermark image, by defining the placement order within the Z-INDEX attribute.

The first image with a Z-INDEX value will be considered the active watermark for the current document. Any subsequent images, defined with style sheet attributes for Z-INDEX or OPACITY, will be displayed as standard WebFOCUS images.

Watermark images are designated by defining the following attributes in the style sheet for the transparent GIF.

Z-INDEX=TOP	Designates that the image is to be handled as a watermark image and should always be placed on top of all other objects on the page. This value will be respected as the topmost layer and will be supported with other layers in future releases.
OPACITY= <i>n</i>	Where <i>n</i> represents the percent (%) of OPACITY to be applied to the image. The greater the OPACITY, the less transparent the image. Less of the underlying report will be visible below the image. The value for <i>n</i> can be any number from 0 through 100. If a value is not specified, it defaults to 100%, presenting a fully opaque image.

Within a single TABLE request:

```
TYPE=<REPORT|HEADING>, OBJECT=IMAGE, IMAGE=<image.gif>,
Z-INDEX=TOP, OPACITY=15, POSITION=(.25 .25), DIMENSION=(8 10.5),$
```

Within compound syntax:

#### On Page Master

```
PAGELAYOUT=ALL, NAME='Page Master', $
OBJECT=IMAGE, IMAGE= internalonlyport.GIF, Z-INDEX=TOP, OPACITY=15,
POSITION=(.25 .25), DIMENSION=(8 10.5),$
```

#### On Page Layout

```
PAGELAYOUT=1, NAME='Page Layout1', $
OBJECT=IMAGE, IMAGE= internalonlyport.GIF, Z-INDEX=TOP, OPACITY=15,
POSITION=(.25 .25), DIMENSION=(8 10.5),$
```

### **Example:** Inserting Transparent Images Into a PDF Report

The following request against the GGSALES data source places the coffee image (coffee.gif) on the page and layers the watermark image (internalonlyport.gif) on top. These images are displayed on every page of the report.

```
TABLE FILE GGSALES
SUM
    GGSALES.SALES01.DOLLARS/D12CM
    GGSALES.SALES01.UNITS/D12C
    GGSALES.SALES01.BUDDOLLARS/D12CM
    GGSALES.SALES01.BUDUNITS/D12C
BY GGSALES.SALES01.REGION
BY GGSALES.SALES01.CATEGORY
BY GGSALES.SALES01.PRODUCT
HEADING
"Gotham Grinds"
"Product Sales By Region"
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
    INCLUDE = endeflt,
    TOPMARGIN=.5,
    BOTTOMMARGIN=.5,
    LEFTMARGIN=1,
    RIGHTMARGIN=1,
$
TYPE=REPORT,
    OBJECT=IMAGE,
    IMAGE=internalonlyport.gif,
    POSITION=(+0.70000 +0.70000),
    SIZE=(7 7.5),
    Z-INDEX=TOP, OPACITY=15,
$
TYPE=REPORT,
    OBJECT=IMAGE,
    IMAGE=coffee.gif,
    POSITION=(+1.0 +0.5),
    SIZE=(.5 .5),
$ENDSTYLE
END
```

The output is:



**Gotham Grinds**  
**Product Sales By Region**

Region	Category	Product	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
Midwest	Coffee	Espresso	\$1,294,947	101,154	\$1,258,232	101,869
		Latte	\$2,883,566	231,623	\$2,827,800	233,657
	Food	Biscotti	\$1,091,727	86,105	\$1,067,629	85,839
		Croissant	\$1,751,124	139,182	\$1,708,733	139,648
		Scone	\$1,495,420	116,127	\$1,444,359	113,776
	Gifts	Coffee Grinder	\$619,154	50,393	\$613,453	50,628
		Coffee Pot	\$599,878	47,156	\$614,007	47,779
		Mug	\$1,086,943	86,718	\$1,096,150	87,092
		Thermos	\$577,906	46,587	\$564,010	46,819
Northeast	Coffee	Capuccino	\$542,095	44,785	\$561,491	44,432
		Espresso	\$850,107	68,127	\$872,902	69,776
		Latte	\$2,771,815	222,866	\$2,818,069	221,712
	Food	Biscotti	\$1,802,005	145,242	\$1,848,682	145,152
		Croissant	\$1,670,818	137,394	\$1,739,522	137,864
		Scone	\$907,171	70,732	\$865,703	68,415
	Gifts	Coffee Grinder	\$509,200	40,977	\$511,642	41,297
		Coffee Pot	\$590,780	46,185	\$573,349	45,404
		Mug	\$1,144,211	91,497	\$1,170,314	90,540
Southeast	Coffee	Capuccino	\$604,098	48,870	\$615,247	49,767
		Espresso	\$944,000	73,264	\$956,661	75,353
		Latte	\$853,572	68,030	\$849,465	66,785
	Food	Biscotti	\$2,617,836	209,654	\$2,625,303	213,555
		Croissant	\$1,505,717	119,594	\$1,512,019	120,549
		Scone	\$1,902,359	156,456	\$1,969,906	157,148
	Gifts	Coffee Grinder	\$900,655	73,779	\$927,363	73,812
		Coffee Pot	\$605,777	47,083	\$569,585	46,784
		Coffee Pot	\$645,303	49,922	\$654,579	50,637
West	Coffee	Mug	\$1,102,703	88,474	\$1,124,345	89,371
		Thermos	\$632,457	48,976	\$618,745	48,253
		Capuccino	\$895,495	71,168	\$877,304	70,585
	Food	Espresso	\$907,617	71,675	\$923,941	72,927
		Latte	\$2,670,405	213,920	\$2,722,718	215,272
		Biscotti	\$863,868	70,436	\$861,804	67,780
	Gifts	Croissant	\$2,425,601	197,022	\$2,406,554	195,329
		Scone	\$912,868	72,776	\$914,886	72,252
		Coffee Grinder	\$603,436	48,081	\$571,316	47,397
		Coffee Pot	\$613,624	47,432	\$630,196	49,208
		Mug	\$1,188,664	93,881	\$1,156,976	93,629
		Thermos	\$571,368	45,648	\$575,818	46,402

## Features Supported

The following core PDF features are supported with watermarks:

- ☐ Standard TABLE requests, including reports with paneling
- ☐ Compound report (MERGE=OFF)
- ☐ Coordinated compound report (MERGE=ON)
- ☐ Old compound syntax (OPEN/CLOSE)
- ☐ Drilldown
- ☐ Drillthrough
- ☐ Bookmarks
- ☐ Borders/backcolor

## Limits

- ☐ OPACITY must be between 0 and 100, inclusive.
- ☐ A single watermark image is supported for a single document.

## Usage Notes

- ☐ For new compound syntax, the first watermark image found in the syntax will be used for the report. Any other watermark images found in the code will be ignored or displayed as standard WebFOCUS images.
- ☐ For old compound syntax, the watermark image must be in the first report. If it is not in the first report, a FOC3362 message is generated.
- ☐ The embedded PDF viewer for a browser may not display NLS characters correctly. If NLS characters do not display correctly, use font embedding, as described in [Adding PostScript Type 1 Fonts for PS and PDF Formats](#) on page 585, or configure your browser to use Adobe Reader.

## Using PostScript (PS) Display Format

You can display a report as a PostScript document. PostScript (format PS) is a print-oriented page description language. As a display format, it may be helpful if you wish to see report output on your monitor before printing it using PostScript.



To display a PostScript report, a computer must have a third-party PostScript application installed, such as GSview (a graphical interface for Ghostscript).

If you are sending a PS report to a printer from WebFOCUS or from ReportCaster, you can select the size of the paper on which to print the output. The PostScript code that is generated works on PS printers that support Language Level 2 or above. It is ignored, without harmful effects, on Level 1 printers. For details, see [How to Select Paper Size in a PostScript \(PS\) Report](#) on page 581. This capability is only supported for the PostScript format.

**Other print-oriented display formats.** You can also display a report as a PDF document. For more information, see [Using PDF Display Format](#) on page 575.

### **Procedure:** How to Select Paper Size in a PostScript (PS) Report

PAGESIZE and ORIENTATION are two WebFOCUS StyleSheet options that are used to display a report. You can enable these features in printed documents using the SET parameter PSPAGESETUP. You can then select the size of the paper on which to print a PostScript report by selecting a PAGESIZE option.

Complete these steps:

1. Place the following SET parameter before or within your request

```
SET PSPAGESETUP= ON
```

or

```
ON TABLE SET PSPAGESETUP ON
```

OFF is the default setting.

2. Include your PAGESIZE specification in a SET command or StyleSheet declaration in the request

```
SET PAGESIZE= option
```

or

```
ON TABLE SET PAGESIZE option
```

or

```
PAGESIZE=option, $
```

where:

*option*

Can be any paper size supported for your printer. LETTER is the default setting.

**Note:** If you send a job to a printer that does not have the requested paper size loaded, the printer may stop and instruct its operator to load the specified paper. To ensure control over your printing, it is best to set paper size in individual requests (rather than as an installation-wide default) so that you can load paper as required.

**Example:**    **Selecting Paper Size Using SET Command in a PostScript Report**

This example automatically prints a document on legal paper as specified in the two SET commands that precede the report request. The referenced printer is a PostScript Level 2 or higher printer, which will automatically select legal paper and print the document in landscape mode.

```
SET PSPAGESETUP=ON
SET PAGESIZE=LEGAL
SET ORIENTATION=LANDSCAPE
SET PAGE-NUM=OFF
TABLE FILE CENTORD
HEADING
"Sales Report"
" "
SUM LINEPRICE
BY PRODCAT
ON TABLE SET STYLE *
TYPE=HEADING, SIZE=18, $
ENDSTYLE
ON TABLE HOLD FORMAT PS
END

-RUN
DOS COPY HOLD.PS \\IBIPRINTA\28C2
```

**Selecting Paper Size Using a SET Command and a StyleSheet**

This request automatically prints a document on legal paper based on the paper size and orientation specifications in the StyleSheet declaration. The referenced printer is a PostScript Level 2 or higher printer, which will automatically select legal paper and print the document in landscape mode.

```

SET PSPAGESETUP=ON
TABLE FILE CAR
SUM RC DC SALES BY COUNTRY BY CAR BY MODEL
ON TABLE HOLD FORMAT PS
ON TABLE SET STYLE *
UNITS=IN,
    PAGESIZE='Legal',
    LEFTMARGIN=1.000000,
    RIGHTMARGIN=0.500000,
    TOPMARGIN=1.500000,
    BOTTOMMARGIN=0.500000,
    SQUEEZE=ON,
    ORIENTATION=LANDSCAPE,$
GRAPHTYPE=DATA, COLUMN=RC, GRAPHPATTERN=EMPTY, $
GRAPHTYPE=DATA, COLUMN=DC, GRAPHPATTERN=SLANT, $
GRAPHTYPE=DATA, COLUMN=SALES,
GRAPHPATTERN=HORIZONTAL, $
ENDSTYLE
END
-RUN
DOS COPY HOLD.PS \\IBIPRINTA\28C2

```

## WebFOCUS Font Support

You can add and configure PostScript Type 1 fonts to significantly expand your options for displaying and printing PS and PDF reports, beyond those provided by the basic set of fonts distributed with Adobe Reader. Thousands of PostScript fonts are available to make your reports more stylish and useful, including some that support symbols and bar codes.

The font map files for Type 1 fonts are stored as XML files (fontmap.xml and fontuser.xml). All font mappings in these files are used for both PDF and PostScript report output.

The font definitions for DHTML also cover PowerPoint (PPT and PPTX). For XLSX, you can define a new default font in the fontuser.xml file. This allows ease in customizing the look and feel of XLSX workbooks. WebFOCUS uses Arial as the default font. However, you can change the default font to match the Microsoft Office standard font, Calibri, or your corporate standard.

You can also add and configure a set of TrueType or OpenType fonts to be embedded in PDF output files.

**Note:** You can use OpenType font files (OTF) only with content in Compact Font Format (CFF).

### **Reference:** Support for the Symbol Font

To use the Symbol font, specify font=symbol in your WebFOCUS StyleSheet:

- ☐ Some versions of Firefox 3 do not support the Symbol font and will substitute it with another font. For information about Firefox support for the Symbol font, refer to Firefox sources.

- ☐ The Euro character displays in PDF output because the Adobe Symbol character set includes the Euro character.
- ☐ The Euro character does not display in DHTML, PPT, and PPTX report output because the Windows Symbol character set does not include the Euro character.
- ☐ The following style options can be rendered with the Symbol font:
  - ☐ DHTML, PPT, and PPTX support style=normal, bold, italic, and bold+italic.
  - ☐ PDF supports only style=normal. Any other style specified in the StyleSheet will be mapped to normal.

## How WebFOCUS Uses Type 1 Fonts

WebFOCUS generates a PDF or PS document from scratch. In order to do that, it must physically embed all the objects it displays or prints, including images and fonts, in the document itself.

When you execute a report and specify one of these formats as your display format, the WebFOCUS Reporting Server retrieves the data and begins to format the report. Fonts and images specified in the StyleSheet must be available to the Reporting Server to create the output file. It reads the font information from the font files and embeds that information into the document. The font itself is stored on the Reporting Server.

To ensure that the Reporting Server can locate the required information, you must define and map it in the following files:

- ☐ **Font file, usually a PFB (Printer Font Binary) file.** This file contains the information about the shape to draw for each character of the font. The information in the font file is scalable, which means that a single font file can be used to generate characters of any size. Note, however, that bold and italic variations of the typeface are separate fonts. An alternative ASCII format, PFA, can also be used by WebFOCUS. In addition to PFB and PFA font files, you can use OTF font files to enable more flexibility in customizing PDF files, such as support for expanded character sets and layout features, and cross-platform compatibility.
- ☐ **Adobe Font Metrics (AFM) file.** This file is distributed with all Adobe fonts. It contains information about the size of each character in each font. WebFOCUS uses this information to lay out the report on the page. Note that the three built-in fonts also have AFM files, which are distributed with WebFOCUS. However, these fonts do not require font files, since the fonts are built in to Adobe Reader.

**Note:** A Printer Font Metrics (PFM) file is also available. This file is used by applications such as Adobe Reader for laying out text, however it is not supported by WebFOCUS. You must use the AFM file.

- ❑ **WebFOCUS Font Map files.** These configuration files map the name of a font to the appropriate font metrics and font files (AFM, PFB (or PFA), or OTF). The mapping determines which actual font is used when you specify a font using the FONT attribute in a WebFOCUS StyleSheet. For example, if your StyleSheet contains the following declaration, WebFOCUS will search the font map for a font mapping with a matching name and style, and use the font specified by the mapping:

```
TYPE=REPORT, FONT=HELVETICA, STYLE=ITALIC, $
```

There are two files WebFOCUS uses for mapping fonts, both in an XML-based format:

- ❑ The default font map file, *fontmap.xml*, contains the font definitions for all output formats that are supported with WebFOCUS, as originally installed. Users should not modify this file.
- ❑ The user font map file, *fontuser.xml*, contains font definitions added by the user. The following sections describe how to add your fonts to this file.

The user font map is searched before the default font map, so font definitions in the user map will override definitions of the same font in the default map.

You can also use a variety of utilities to convert Windows True Type fonts (such as Arial and Tahoma) into Type 1 fonts. Verify that you are licensed for this type of font use. Then, after you convert them, you can define and map these fonts for use by WebFOCUS.

One such utility is TTF2PT1.

For information about the Windows version, go to:

<http://gnuwin32.sourceforge.net/packages/ttf2pt1.htm>

For information about UNIX versions, go to: <http://ttf2pt1.sourceforge.net/download.html>

## Adding PostScript Type 1 Fonts for PS and PDF Formats

This section describes how to add PostScript type 1 fonts to the *fontuser.xml* file.

### **Procedure:** How to Configure Type 1 PostScript Fonts on the Windows and UNIX Platforms

Locate the necessary font files (AFM, PFB (or PFA), or OTF). These files should be available in the location where the fonts were originally installed. You will be copying these files to a location from which they can be accessed by the WebFOCUS Reporting Server.

**Tip:**

- ❑ You may need to run an installer program to install these in a directory on your Windows or UNIX machine. Note that the fonts do not have to be installed on a client machine in order to be used, since they will be embedded in the PDF or PostScript files created by WebFOCUS. PDF files with these embedded fonts can be displayed on any machine that has Adobe Reader (or a similar PDF viewer), and PostScript files with these embedded fonts can be printed on any PostScript printer (or displayed in a PostScript viewer such as GhostView).
- ❑ Note that PFB files are binary, so if they are FTPed from another machine, they must be FTPed in BINARY mode.

After you have located the font files you wish to add, you can set up WebFOCUS to use one or more Type 1 fonts.

1. For each font you wish to add, copy the AFM, PFB (or PFA), and OTF files into the *etc* subdirectory of your WebFOCUS configuration directory. On a Windows machine, the location is usually:

*drive:\ibi\srv82\wfs\etc*

where:

*wfs*

Is your WebFOCUS configuration directory (it may have a different name depending on installation options, but should always be a directory directly under *drive:\ibi\srv82*). Note that *home* is the other directory directly under *drive:\ibi\srv82*.

Under Unix, the location is */ibi/srv82/wfs/etc*, assuming that the WebFOCUS server was installed in */ibi/srv82*.

Keeping user font files in this directory allows user font files to remain separate from the default font files (under *\ibi\srv82\home*) so they can be easily preserved if WebFOCUS is updated to a new release.

After you copy these files, you can rename them to any descriptive name.

2. The user font map file, *fontuser.xml*, is created by the installer in the same directory. Using a text editor, add your font definitions to this file using the syntax described in the following section, [How to Add Fonts to the Font Map](#) on page 588.

**Procedure: How to Configure Type 1 PostScript Fonts on z/OS Under PDS Deployment**

After you have located the font files you wish to add, you can configure WebFOCUS to use one or more Type 1 fonts.

1. Copy the AFM (font metrics) file into the PDS allocated to DDNAME EDACCFG in the Reporting Server JCL. You can copy this file from another machine using FTP in standard ASCII (text) mode. The member name of the AFM file in this PDS will match the *metricsfile* value in the font map file.

**Note:** If the Windows font file names contain underscore characters or are longer than eight characters, you must rename them, since these are not valid for z/OS member names.

2. You can use either PFB (binary) fonts or PFA (ASCII) fonts:

- ❑ If you are using PFB (binary) fonts, create a partitioned data set, put the PFB file in it (for example, using FTP in BINARY mode), and concatenate this data set to the data set already allocated to DDNAME EDAHBIN in the WebFOCUS Reporting Server JCL.

This PDS should be created with the following DCB attributes:

```
RECFM: VB      LRECL: 1028      BLKSIZE: 27998
```

The member name in this PDS should match the *fontfile* name in the font map file.

If you copy the PFB font file into the PDS using FTP, you must use BINARY mode. The member name of the PFB file in this PDS will match the *fontfile* value in the font map file.

- ❑ If you are using PFA (ASCII) font files, create a PDS (separate from the one you use for PFB fonts), put the PFA file in it (for example, using FTP in regular, ASCII mode), and concatenate this data set to the data set already allocated to DDNAME EDAHETC in the Reporting Server JCL. This PDS should be created with the following DCB attributes:

```
RECFM: VB      LRECL: 2044      BLKSIZE: 27998
```

The member name in this PDS should match the *fontfile* value in the font map file.

Note that you can use PFB and PFA files simultaneously. The *fonttype* attribute in the font map file (PFB or PFA) tells WebFOCUS which PDS to search for the specified member name.

3. The user font map file is in member FONTUSER in the data set allocated to DDNAME EDACCFG. Using a text editor, add your font definition to the user font map using the syntax described in [How to Add Fonts to the Font Map](#) on page 588.

**Syntax: How to Add Fonts to the Font Map**

The Type 1 PostScript fonts used with the PostScript and PDF output formats use separate font files for each variant of the font: normal, bold, italic, and bold-italic. This grouping of related fonts is called a *font family*.

The XML font map syntax uses two XML tags, `<family>` and `<font>`, to represent this structure. The example uses the family name *Garamond*. For example:

```
<family name="garamond">
  <font style="normal"
    metricsfile="gdrg" fontfile="gdrg" fonttype="PFB" />
  <font style="bold"
    metricsfile="gdb" fontfile="gdb" fonttype="PFB" />
  <font style="italic"
    metricsfile="gdi" fontfile="gdi" fonttype="PFB" />
  <font style="bold+italic"
    metricsfile="gdbi" fontfile="gdbi" fonttype="PFB" />
</family>
```

The following example uses the family name *otf*. For example:

```
<family name="otf">
  <font style="normal"
    metricsfile="otfn" fontfile="otfn" fonttype="OTF" />
  <font style="bold"
    metricsfile="otfb" fontfile="otfb" fonttype="OTF" />
  <font style="italic"
    metricsfile="otfi" fontfile="otfi" fonttype="OTF" />
  <font style="bold+italic"
    metricsfile="otfbi" fontfile="otfbi" fonttype="OTF" />
</family>
```

The basics of the XML syntax are:

- ☐ Tag names (such as *family* and *font*) and attribute names (such as *style* or *metricsfile*) must be in lowercase. Attribute values, such as font file names, are case-insensitive.
- ☐ Attribute values, which is the text after the equal sign (=), must be in double quotation marks (for example, *"bold"*)
- ☐ Elements that have no explicit end-tag must end with `/>`. (For example, the family tag has the closing tag `</family>`, but the font tag has no closing tag, so it ends with `/>`.)
- ☐ Comments are enclosed in special delimiters:

```
<!-- This is a comment -->
```

- ☐ Line breaks may be placed between attribute-value pairs.

A more complete description of XML syntax can be found here:



<http://en.wikipedia.org/wiki/Xml>

### The family element

The *family* element specifies the name of a font family. This family name, specified in the *name* attribute of the family element, is the name by which the font will be referenced in a StyleSheet. It corresponds to the value of the FONT attribute in the StyleSheet. The end-tag `</family>` closes the family element, and any number of additional family elements may follow.

Font family names should be composed of letters (A-Z, a-z), digits, and limited special characters, such as a minus sign (-), underscore (\_), and blank. Font family names should have a maximum length of 40 characters. Since the font name is only a reference to a mapping in the font map, it does not need to be related to the actual name of the font (which WebFOCUS obtains from the mapped AFM file) or the file name of the font.

### Font elements

Nested within each *family* element are one or more *font* elements that specify the font files for each font in the family. For example, there may be one font element for the font *Garamond Regular* (normal), one for *Garamond Italic* (italic). Since a font element has no child elements, it is closed with `</>`.

The actual name of the font as used in the PDF or PostScript document is taken from the font metric file.

Fonts defined in the user font file (fontuser.xml) can override default font definitions in fontmap.xml. Thus, you should be careful to choose family names that do not conflict with existing definitions, unless you actually wish to override these definitions (which should generally not be done).

Each font element contains the following attributes:

- ❑ **style:** This attribute specifies the style of the font and corresponds to the STYLE attribute in the StyleSheet. The allowed values are "normal", "bold", "italic", and "bold+italic". For example, the font defined in the following bold italic font element:

```
<font style="bold+italic" metricsfile="gdbi" fontfile="gdbi"
fonttype="PFB" />
```

could be referenced in the StyleSheet like this:

```
TYPE=REPORT, FONT=GARAMOND, STYLE=BOLD+ITALIC, $
```

Although most fonts have a font file for each of the four styles, some specialized fonts such as bar code fonts might only have a single style (usually "normal"). Only the styles that exist for a particular font need to be specified in the font map file.

The actual names of the fonts may vary. Some fonts may be called "oblique" rather than "italic", or "heavy" rather than "bold". However, the font map and StyleSheet always use the keywords "normal", "bold", "italic" and "bold+italic".

- ❑ **metricsfile:** This attribute specifies the name of the Adobe Font Metrics (AFM) file that provides the measurements of the font. You should only use the base name of the file (for example, "gdrgr", not "gdrgr.afm"). On Windows and UNIX systems, the file is assumed to have the extension *.afm* and reside in the *wfs/etc* directory. On z/OS with PDS deployment, the name refers to a member in the PDS allocated to EDACCFG. For information about file locations, see [How to Configure Type 1 PostScript Fonts on the Windows and UNIX Platforms](#) on page 585 or [How to Configure Type 1 PostScript Fonts on z/OS Under PDS Deployment](#) on page 587.

File names should be composed of letters and numbers, and should not contain blanks. On Windows and UNIX systems, the file names may also contain underscore characters. On UNIX systems, the file names should not contain uppercase letters. Since the files must be located in specific directories, no directory paths or drive letters are allowed.

- ❑ **fonttype:** This attribute specifies the type of the font file. The allowed values are "PFA", "PFB", or "OTF".
- ❑ **fontfile:** This attribute specifies the name of the PFB or PFA file that contains the font itself. As with *metricsfile*, the value specifies only the base file name (the *fonttype* attribute specifies the type). On Windows and UNIX, the file is assumed to have the extension *.pfb* for binary (PFB) font files or *.pfa* for ASCII (PFA) font files, and should reside in the same directory as the AFM files (*wfs/etc*). On z/OS with PDS deployment, the name refers to a member in the appropriate PDS.

Additional items of XML syntax include the XML header on the first line of the file and the <fontmap> and <when> elements that enclose all of the family elements. The <when> tag allows the same font mappings to be used for both PDF and PostScript reports across output formats. These can include PDF, PS, and DHTML. PPT and PPTX formats will use fonts specified for DHTML. If no <when> is specified, the font will be available for all formats.

The following is a complete example of a user font map:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Example of a user font map file with two font families. -->
<fontmap version="1">
  <when format="PDF PS">
    <family name="garamond">
      <font style="normal"
        metricsfile="gdrgr" fontfile="gdrgr" fonttype="PFB" />
      <font style="bold"
        metricsfile="gdb" fontfile="gdb" fonttype="PFB" />
      <font style="italic"
        metricsfile="gdi" fontfile="gdi" fonttype="PFB" />
      <font style="bold+italic"
        metricsfile="gdbi" fontfile="gdbi" fonttype="PFB" />
    </family>
    <!-- This font only has a "normal" style, others omitted. -->
    <family name="ocra">
      <font style="normal"
        metricsfile="ocra" fontfile="ocra" fonttype="PFB" />
    </family>
  </when>
</fontmap>
```

### **Example:** WebFOCUS StyleSheet Declaration

Once the font map files have been set up, the newly mapped fonts can be used in a WebFOCUS StyleSheet. For example, to use the Garamond fonts:

```
ON TABLE SET STYLE *
type=report, font=garamond, size=12, $
type=title, font=garamond, style=bold, color=blue, $
ENDSTYLE
```

Since the *style* attribute has been omitted for the report font in the StyleSheet, it defaults to normal. Attributes such as *size* and *color* can also be applied.

### **Reference:** Editing the Font Map File

There is a byte order mark (BOM) at the beginning of the user font map file (fontuser.xml), which must be preserved for this file to be read correctly.

If you are using a Unicode-aware editor, such as Notepad on Windows, to edit the file, the BOM will not be visible, but you can preserve it by making sure that you select an encoding of UTF-8 in the Save-As dialog. In most other editors, such as *vi* on UNIX or the ISPF editor under z/OS, the BOM will display as three or four strange-looking characters at the beginning of the file. As long as you do not delete or modify these characters, the BOM will be preserved.

### **Reference:** The WebFOCUS Default Font Map

Since the user font map is searched before the WebFOCUS default font map, font definitions in the user font map file will override mappings of the same font in the default font map file.

Since you usually would not want to override existing font mappings, you can check which font names are already used by WebFOCUS by examining the default font map file.

On Windows platforms, it can be found in

`drive:\ibi\srv77\home\etc\style\fontmap.xml`

On UNIX, it can be found in a similarly named directory.

On z/OS with PDS deployment, the default font map file is in the FONTMAP member of the *prefix.P.HOME.ERR* partitioned data set. Unlike the user font map file, this file has separate sections containing definitions for PS, PDF, and DHTML formats.

**Note:** The DHTML mappings are used for the DHTML and PowerPoint output formats, which do not support user-added fonts.

Since the font mappings in the default font map file are for fonts that are already assumed to exist on the user machines (for example, built-in Adobe Reader fonts, standard PostScript printer fonts, or standard Windows fonts), they do not reference font files, only font metrics files. Fonts provided by the user should reference both font files and metrics files.

AFM files for the default fonts can be found in `drive:\ibi\srv77\home\etc\style` (or members of *prefix.P.HOME.ERR* with z/OS under PDS deployment).

### **Procedure:** How to Define a Default Font in the Font Map

An individual default font can be set for each output type and/or language setting within an output type. This setting should be defined in the `fontuser.xml` file rather than the `fontmap.xml` file. `Fontmap.xml` may be updated by a future release installation, so customizations may be lost. Additionally, the settings in `fontuser.xml` override settings in `fontmap.xml`.

#### **Note:**

- ☐ `Fontmap.xml` can be found in `..\ibi\srvXX\home\etc\style`, where `XX` is your server release.
- ☐ `Fontuser.xml` can be found in `..\ibi\srvXX\wfs\etc`, where `XX` is your server release.

To designate the default font use the following steps:

1. Copy the selected font entry from `fontmap.xml` to `fontuser.xml`.
  - a. Within `fontmap.xml`, find the entry for the font family within the desired output format to be designated as the default.

- b. Copy the entire entry into the appropriate format area within fontuser.xml.
2. In fontuser.xml, within the entry for the font to be designated as the default font and style, add the following attribute:

```
default="yes"
```

For example, the following code defines the default fonts to be Helvetica bold for PDF, Calibri for XLSX, and Arial Italic for DHTML:

```
<fontmap version="1">
<when format="PDF PS">
  <family name="Helvetica" htmlfont="Arial">
    <font style="normal" metricsfile="pdherv" />
    <font style="bold" metricsfile="pdhervb" default="yes" />
    <font style="italic" metricsfile="pdhervi" />
    <font style="bold+italic" metricsfile="pdhervbi" />
  </family>
</when>
<when format="XLSX">
  <family name="Calibri" htmlfont="Calibri">
    <font style="normal" metricsfile="ttcali" default="yes" />
    <font style="bold" metricsfile="ttcalib" />
    <font style="italic" metricsfile="ttcalii" />
    <font style="bold+italic" metricsfile="ttcalibi" />
  </family>
</when>
<when format="DHTML">
  <family name="Arial">
    <font style="normal" metricsfile="ttarial" />
    <font style="bold" metricsfile="ttarialb" />
    <font style="italic" metricsfile="ttariali" default="yes" />
    <font style="bold+italic" metricsfile="ttariabi" />
  </family>
</when>
</fontmap>
```

If multiple fonts in a font map family, such as PDF, have the default="yes" attribute, the last font with that attribute becomes the default font. Fonts in fontuser.xml are processed after those in fontmap.xml, so a default font set in fontuser.xml can override the one set in fontmap.xml.

A default font set in the PDF section of the font map does not affect a default in the DHTML section, and a default for one specific language does not override the default for other languages.

## Embedding TrueType Fonts Into WebFOCUS PDF Reports Generated in Windows

You can have WebFOCUS embed the following TrueType fonts into PDF output files generated in Windows:

- ☐ Arial Unicode MS
- ☐ Courier New
- ☐ Lucida Sans Unicode
- ☐ Tahoma
- ☐ Times New Roman
- ☐ Trebuchet MS

**Note:** The addition of the font file and font type attributes activates the embedding feature. To use any of these fonts and font styles without embedding, do not add the font file and font style attributes into the font map definition for each individual style.

### ***Procedure:*** How to Add TrueType Fonts for Embedding Into PDF Output Files

1. Make the fonts available to the Reporting Server by copying the TrueType font files from the Windows font directory (C:\Windows\fonts) to the WebFOCUS server configuration directory:

*drive:\ibi\srv82\wfs\etc*

where:

*drive*

Is the drive on which the Reporting Server is installed.

*wfs*

Is your WebFOCUS configuration directory (it may have a different name depending on installation options, but should always be a directory directly under *drive:\ibi\srv82*).

Note that *home* is the other directory directly under *drive:\ibi\srv82*.

For each of the supported fonts, you will need to copy the following font files. You will also need to know the metrics file name associated with each font file:

### Arial Unicode MS

Style	Metrics File Name	Font File Name
Normal	pdarum.afm	arialuni.ttf
Bold	pdarumb.afm	arialuni.ttf
Italic	pdarumi.afm	arialuni.ttf
Bold Italic	pdarumbi.afm	arialuni.ttf

### Courier New

Style	Metrics File Name	Font File Name
Normal	pdconu.afm	cour.ttf
Bold	pdconub.afm	courbd.ttf
Italic	pdconui.afm	couri.ttf
Bold Italic	pdconubi.afm	courbi.ttf

### Lucida Sans Unicode

Style	Metrics File Name	Font File Name
Normal	pdlusu.afm	l_10646.ttf
Bold	pdlusub.afm	l_10646.ttf
Italic	pdlusui.afm	l_10646.ttf
Bold Italic	pdlusubi.afm	l_10646.ttf

**Tahoma**

<b>Style</b>	<b>Metrics File Name</b>	<b>Font File Name</b>
Normal	pdtaho.afm	tahoma.ttf
Bold	pdtahob.afm	tahomabd.ttf
Italic	pdtahoi.afm	tahoma.ttf
Bold italic	pdtahobi.afm	tahomabd.ttf

**Times New Roman**

<b>Style</b>	<b>Metrics File Name</b>	<b>Font File Name</b>
Normal	pdtimu.afm	times.ttf
Bold	pdtimub.afm	timesbd.ttf
Italic	pdtimui.afm	timesi.ttf
Bold Italic	pdtimubi.afm	timesbi.ttf

**Trebuchet MS**

<b>Style</b>	<b>Metrics File Name</b>	<b>Font File Name</b>
Normal	pdrbu.afm	trebuc.ttf
Bold	pdrbub.afm	trebucbd.ttf
Italic	pdrbui.afm	trebucit.ttf
Bold Italic	pdrbubi.afm	trebucbi.ttf

2. Add the fonts to the user font map file, fontuser.xml. This file is also located in the Reporting Server configuration directory *drive:\ibi\sr82\wfs\etc*.

The fontuser.xml file has a sample family tag. Copy the sample family tag between <when format="PDF PS"> and </when>, then edit it for the font you are adding. For more information on editing font map syntax, see [How to Add Fonts to the Font Map](#) on page 588.



The following shows a user font map file with the Arial Unicode MS font added:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Example of a user font map file defining Arial Unicode MS true
type fonts -->
<fontmap version="1">
  <when format="PDF">
    <family name = "Arial Unicode MS">
      <font style="normal" metricsfile="pdarum"
        fontfile="arialuni" fonttype="TTF" />
      <font style="bold" metricsfile="pdarumb"
        fontfile="arialuni" fonttype="TTF" />
      <font style="bold+italic" metricsfile="pdarumbi"
        fontfile="arialuni" fonttype="TTF" />
      <font style="italic" metricsfile="pdarumi"
        fontfile="arialuni" fonttype="TTF" />
    </family>
  </when>
</fontmap>
```

Note that the font file name does not include the extension. The extension, TTF, is entered as the fonttype attribute.

3. In a report request, specify the font family names and the style attributes in the stylesheet, and hold the report output in PDF format.

An example follows of the contents of a fontuser.xml file with all of the supported embedded fonts defined. You can select only the ones you need for your environment:

```
<fontmap version="1">
  <when format="PDF PS">

    <family name = "Arial Unicode MS">
      <font style="normal"
        metricsfile="pdarum"    fontfile="arialuni" fonttype="TTF" />
      <font style="bold"
        metricsfile="pdarumb"   fontfile="arialuni" fonttype="TTF" />
      <font style="bold+italic"
        metricsfile="pdarumbi"  fontfile="arialuni" fonttype="TTF" />
      <font style="italic"
        metricsfile="pdarumi"   fontfile="arialuni" fonttype="TTF" />
    </family>

    <family name="Trebuchet MS">
      <font style="normal"
        metricsfile="pdtrbu"    fontfile="trebuc"   fonttype="TTF" />
      <font style="bold"
        metricsfile="pdtrbub"   fontfile="trebuchd" fonttype="TTF" />
      <font style="italic"
        metricsfile="pdtrbui"   fontfile="trebucit" fonttype="TTF" />
      <font style="bold+italic"
        metricsfile="pdtrbubi"  fontfile="trebucbi" fonttype="TTF" />
    </family>
```

```
<family name="Times New Roman">
  <font style="normal"
    metricsfile="pdtimu"    fontfile="times"    fonttype="TTF" />
  <font style="bold"
    metricsfile="pdtimub"   fontfile="timesbd"   fonttype="TTF" />
  <font style="italic"
    metricsfile="pdtimui"   fontfile="timesi"    fonttype="TTF" />
  <font style="bold+italic"
    metricsfile="pdtimubi"  fontfile="timesbi"   fonttype="TTF" />
</family>

<family name="Lucida Sans Unicode">
  <font style="normal"
    metricsfile="pdlusu"    fontfile="L_10646"    fonttype="TTF" />
  <font style="bold"
    metricsfile="pdlusub"   fontfile="L_10646"    fonttype="TTF" />
  <font style="italic"
    metricsfile="pdlusui"   fontfile="L_10646"    fonttype="TTF" />
  <font style="bold+italic"
    metricsfile="pdlusubi"  fontfile="L_10646"    fonttype="TTF" />
</family>

<family name = "Courier New">
  <font style="normal"
    metricsfile="pdconu"    fontfile="cour"    fonttype="TTF" />
  <font style="bold"
    metricsfile="pdconub"   fontfile="courbd"   fonttype="TTF" />
  <font style="bold+italic"
    metricsfile="pdconubi"  fontfile="courbi"   fonttype="TTF" />
  <font style="italic"
    metricsfile="pdconui"   fontfile="couri"   fonttype="TTF" />
</family>

<family name = "Tahoma">
  <font style="normal"
    metricsfile="pdtaho"    fontfile="tahoma"    fonttype="TTF" />
  <font style="bold"
    metricsfile="pdtahob"   fontfile="tahomabd"   fonttype="TTF" />
  <font style="bold+italic"
    metricsfile="pdtahobi"  fontfile="tahomabd"   fonttype="TTF" />
  <font style="italic"
    metricsfile="pdtahoi"   fontfile="tahoma"    fonttype="TTF" />
</family>
</when>
</fontmap>
```

**Example: Embedding TrueType Fonts in a PDF Output File**

The font files `trebuc.ttf`, `trebudbd.ttf`, `trebucit.ttf`, `trebucbi.ttf`, `tahoma.ttf`, and `tahomabd.ttf` have been copied to the Reporting Server configuration directory, `drive:\ibi\srv82\wfs\etc`. In addition, the Trebuchet MS and Tahoma fonts have been added to the `fontuser.xml` file:

```
<fontmap version="1">
  <when format="PDF PS">
    <!-- family/font tags should be added here -->
    <family name="Trebuchet MS">
      <font style="normal"
        metricsfile="pdtrbu"   fontfile="trebuc"   fonttype="TTF" />
      <font style="bold"
        metricsfile="pdtrbub"  fontfile="trebudbd" fonttype="TTF" />
      <font style="italic"
        metricsfile="pdtrbui"  fontfile="trebucit"  fonttype="TTF" />
      <font style="bold+italic"
        metricsfile="pdtrbubi" fontfile="trebucbi"  fonttype="TTF" />
    </family>
    <family name="Tahoma">
      <font style="normal"
        metricsfile="pdtaho"   fontfile="tahoma"   fonttype="TTF" />
      <font style="bold"
        metricsfile="pdtahob"  fontfile="tahomabd"  fonttype="TTF" />
    </family>
  </when>
</fontmap>
```

The following request against the GGSALES data source specifies the Trebuchet MS font for the column headings and the bold style of Tahoma for the data in the PDF report output:

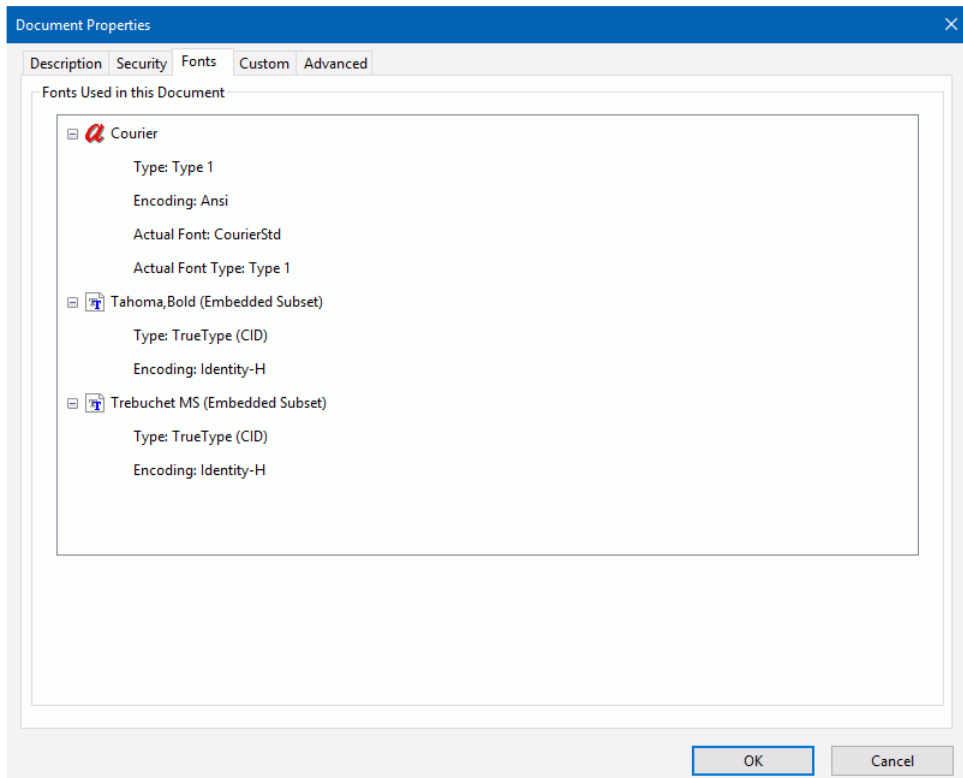
```
TABLE FILE GGSALES
SUM DOLLARS UNITS
BY REGION
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
TYPE = TITLE, FONT='Trebuchet MS',$
TYPE = DATA, FONT='Tahoma', style=bold, size=10, color=blue, $
END
```

The output is:

<u>Region</u>	<u>Dollar Sales</u>	<u>Unit Sales</u>
Midwest	11400665	905045
Northeast	11392300	916675
Southeast	11710379	935232
West	11652946	932039

**Note:** Except for OpenType fonts, in order to reduce the size of a generated PDF document, only the subset of used characters of a font are included into the PDF document.

To confirm that a subset of the font is embedded, save a copy of the PDF file and open it in Adobe Reader. On the File menu, select *Properties*. In the Font tab, look for the words Embedded Subset next to the font name, as shown in the following image for the Tahoma and Trebuchet MS fonts.



### Creating PDF Files on z/OS for Use With UNIX Systems

PDF files created with HOLD FORMAT PDF present a challenge if you work in a z/OS environment and use UNIX-based systems as the server for Adobe or as an intermediate transfer point.

The end of each PDF file has a table containing the byte offset, including line termination characters, of each PDF object in the file. The offsets indicate that each line is terminated by two characters, a carriage return and a line feed, which is the standard Windows text file format. However, records in a UNIX text file are terminated by one character, a line feed only. When using default settings, the offsets in a PDF file will be incorrect, causing an error when Acrobat attempts to open the file. If the file is then transferred in BINARY mode to Windows, it cannot be opened in Acrobat for Windows, as the carriage-return character was not inserted.

One solution has been to transfer the file to the UNIX system in text mode and then transfer in text mode to the Windows system, as the carriage return is added by the transfer facility when transferring to Windows.

If that is not possible or desirable, you can use the SET PDFLINETERM=SPACE command to facilitate binary transfer to Windows from an ASCII-based UNIX system. This command causes an extra space character to be appended to each record of the PDF output file. This extra space acts as a placeholder for the expected carriage return character and makes the object offsets in the file correct when it is transferred from z/OS to a UNIX system. This enables a UNIX server to open a PDF file in that environment.

**Note:** A text mode transfer is always required when transferring a text file from a mainframe to any other environment (Windows, ASCII Unix, or EBCDIC Unix).

### ***Syntax:***      **How to Specify Line Termination Characters When Creating a PDF File**

In a profile, a FOCEXEC, or from the command line, issue the following command:

```
SET PDFLINETERM={ STANDARD | SPACE }
```

In a TABLE request, issue the following command

```
ON TABLE SET PDFLINETERM { STANDARD | SPACE }
```

where:

[STANDARD](#)

Creates a PDF file without any extra characters. This file will be a valid PDF file if transferred in text mode to a Windows machine, but not to a UNIX machine. If subsequently transferred from a UNIX machine to a Windows machine in text mode, it will be a valid PDF file on the Windows machine.

SPACE

Creates a PDF file with an extra space character appended to each record. This file will be a valid PDF file if transferred in text mode to a UNIX machine, but not to a Windows machine. If subsequently transferred from an ASCII UNIX machine to a Windows machine in binary mode, it will be a valid PDF file on the Windows machine.

**Reference: Required PDFLINETERM Settings Based on Environment**

The following chart will assist you in determining the correct setting to use, based on your environment:

Transferring from z/OS to:	SET PDFLINETERM=
EBCDIC UNIX (text transfer)	SPACE
ASCII UNIX (text transfer)	SPACE
ASCII UNIX (text); then to Windows (binary)	SPACE
UNIX (text); then to Windows (text)	STANDARD
Directly to Windows (text)	STANDARD

Using Word Processing Display Formats: DOC, WP

You can display a report as a plain text word processing document using:

- ☐ **DOC format.** The report opens in Microsoft Word within your web browser. When Word opens the report, it may prompt you for information about converting text. If it does, accept the default selection. The computer on which the report is being displayed must have Word installed.  
  
DOC format includes page breaks, including an initial page break before the beginning of the report. If you wish to omit page breaks, issue the SET PAGE = NOPAGE command at the beginning of the procedure.  
  
DOC format does not support StyleSheets.
- ☐ **WP format.** The report opens as plain text within your web browser.

If you issue the SET PAGE = OFF command, or include TABPAGENO in a heading or footing, WP will indicate page breaks by including the character "1" in the first column at each break, which is recognized as a page break control in the S/390 environment. WP format does not include page breaks that are recognized by most browsers or word processing programs.

WP format does not support StyleSheets.

You can specify that a report display as a plain text word processing document via the PCHOLD command when you run the report in WebFOCUS. For more information, see [How to Choose a Display Format Using PCHOLD](#) on page 566.

## Saving Report Output in Excel XLSX Format

With Excel® 2007, Microsoft® introduced enhanced spreadsheet functionality in a new workbook file format. Using WebFOCUS, you can retrieve data from any WebFOCUS supported data source and generate a native XLSX format (Excel 2007, Excel 2010, and Excel 2013) workbook for data analysis and distribution.

The WebFOCUS XLSX/EXL07 format supports the following Microsoft Office software products:

- ☐ Microsoft Office 2013/2010/2007 and Microsoft Office 2000/2003 with the Microsoft Office Compatibility Pack.
- ☐ Open Office Support (FORMAT EXL07/XLSX). Core Excel functionality generated by the EXL07/XLSX format is supported for Open Office as of WebFOCUS 8. For details on Open Office, see <http://www.openoffice.org/>.
- ☐ MAC Office 2008 and 2011. FORMAT EXL07/XLSX is certified with WebFOCUS 8.

WebFOCUS generates XLSX workbooks based on the Microsoft XLSX standard. These workbooks are accessible through all browsers and mobile applications that support native Microsoft XLSX files.

**Note:** This applies to Excel 2007, Excel 2010, and Excel 2013, unless otherwise indicated.

## Overview of EXL07/XLSX Format

FORMAT EXL07 and FORMAT XLSX are synonyms and can be used interchangeably. The FILE SAVED message will always display "XLSX FILE SAVED", regardless of the syntax specified.

The WebFOCUS procedure generates a new workbook containing a single worksheet with the report output containing your defined report elements (headings and subtotals), as well as StyleSheet syntax (such as conditional styling and drill downs):

- ☐ You can define a new default font for XLSX in the fontuser.xml file. This allows ease in customizing the look and feel of XLSX workbooks. WebFOCUS uses Arial as the default font. However, you can change the default font to match the Microsoft Office standard font, Calibri, or your corporate standard.
- ☐ XLSX format accurately displays formatted numeric, character, and date formats.
- ☐ XLSX FORMULA enables you to convert summed information (such as column totals, row totals, and calculated values) into Excel formulas that will automatically update as you edit the Excel worksheet.
- ☐ ReportCaster supports distribution of XLSX workbooks and XLSX FORMULA workbooks.
- ☐ Within each generated worksheet, the columns in the report are automatically sized to fit the largest value in the column (SQUEEZE=ON). WebFOCUS calculates the width of each data column based on the font and size requirement of all cells in that column using font metrics developed for other styled formats, including PDF and DHTML. Calculations are based on the data and title elements of the report. Heading and footing elements are not used in the sizing calculation and will be sized based on the data column requirements.
- ☐ By default, there is a standard height for the data and Title rows. Heading, Footing, Subhead, and Subfoot rows are taller than the data rows to support wrapping and for a clearer distinction between headings and data.
- ☐ Using the TITLETEXT StyleSheet attribute, tab names within the workbook can be customized to provide better descriptions of the worksheet content.
- ☐ Unlike the HTML-based (EXL2K) format, which removes all blanks, XLSX, by default, retains leading, internal, and trailing blanks in cells within the worksheet. For more information on how to affect these blanks, see [Preserving Leading and Internal Blanks in Report Output](#) on page 636.
- ☐ An XLSX worksheet can contain 1,048,576 rows by 16,384 columns. WebFOCUS will generate worksheets larger than these defined limits, but Excel is not able to open the workbook. For more information on how to support overflow in worksheets, see [Overcoming the Excel 2007/2010 Row Limit Using Overflow Worksheets](#) on page 662.



- ❑ Because of the new format of the zipped XLSX files, native HTML symbols, such as a caret (<), cannot be supported as tag characters. For XLSX, unlike other output formats, HTMLENCODE defaults to ON. HTMLENCODE set to OFF will cause any data containing HTML tag characters to be omitted from the cell. For more information on the SET HTMLENCODE command, see the *Developing Reporting Applications* manual.

## Building the .xlsx Workbook File

Microsoft changed the format and structure of the Excel workbook in Excel 2007. The new .xlsx file is a binary compilation of a group of xml files. Generating this new file format using WebFOCUS is a two-step process that consists of generating the xml files containing the report output and zipping the xml documents into the binary .xlsx format. The Reporting Server performs the xml generation process. The zipping process can be completed either by the client (WebFOCUS Servlet) or the server (JSCOM3):

- ❑ **WebFOCUS Servlet.** The WebFOCUS Client within the application server performs the zipping process. This can be done within the local client or through a remotely accessed client. The servlet method is the default approach defined for each WebFOCUS Client, with the client pointing to itself, by default.
- ❑ **JSCOM3.** The Java layer of the Reporting Server performs the zipping operation. This option should be used when the WebFOCUS Servlet is configured on a secured web or application server. This is because JSCOM3 does not require URL access to a remote WebFOCUS Client.

### *Syntax:*

## How to Select the Method for Zipping the .xlsx File

You designate the method and location where the zipping will occur by setting EXCELSERVURL to a URL (for the WebFOCUS Servlet) or to a blank (for JSCOM3). You can set this value for a specific procedure or for the entire environment:

- ❑ **For a procedure.** Issue the SET EXCELSERVURL command within the procedure.
- ❑ **For the entire environment.** Edit the IBIF\_excelservurl variable in the WebFOCUS Administration Console by selecting:

Configuration/Client Settings/General/IBIF\_excelservurl

For more information on accessing the WebFOCUS Administration Console and setting the IBIF\_excelservurl variable, see the *WebFOCUS Security and Administration* manual.

The value you assign to EXCELSERVURL determines whether the WebFOCUS Servlet or JSCOM3 performs the zipping operation:

- ❑ **Specifying the Servlet.** To specify that the WebFOCUS Servlet should be used, set the EXCELSERVURL parameter or the IBIF\_variable to the URL. For example,

In a procedure:

```
SET EXCELSERVURL = http://servername:8080/ibi_apps
```

In the WebFOCUS Administration Console:

```
IBIF_excelservurl = http://servername:8080/ibi_apps
```

- ❑ **Specifying JSCOM3.** To specify that JSCOM3 should be used within the current Reporting Server, set EXCELSERVURL to a blank or an empty string.

In a procedure:

```
SET EXCELSERVURL = ''
```

In the WebFOCUS Administration Console:

```
IBIF_excelservurl = ''
```

By default, each WebFOCUS Client contains the following URL definition that points to itself:

```
&URL_PROTOCOL://&servername:&server_port&IBIF_webapp
```

### **Syntax:** How to Generate an Excel XLSX Workbook

You can specify that a report should be saved to an XLSX workbook, displayed in the browser, or displayed in the Excel application.

```
ON TABLE {PCHOLD|HOLD} AS name FORMAT XLSX
```

where:

**PCHOLD**

Displays the generated workbook in either the browser or the Excel application, based on your desktop settings. For information, see [Viewing Excel Workbooks in the Browser vs. the Excel Application](#) on page 608.

**HOLD**

Saves a workbook with an .xlsx extension to the designated location.

*name*

Specifies a file name for the generated workbook.

**Note:** To assign a file name to the generated workbook, set the *Save Report* option to *YES* for the .xlsx file extension in the WebFOCUS Client Redirection Settings. When opened in the Excel application, the generated workbook will retain the designated AS name. For more information, see the *WebFOCUS Security and Administration* manual.

### Opening XLSX Report Output

To open XLSX workbooks, Excel 2013, 2010, or 2007 must be installed on the desktop.

#### **Reference:** Opening XLSX Report Output in Excel 2000/2003

Excel 2000 and Excel 2003 can be updated to read Excel XLSX workbooks using the Microsoft Office Compatibility Pack available from the Microsoft download site (<http://www.microsoft.com/downloads/en/default.aspx>). When the file extension of the file being opened is .xlsx (XLSX workbook), the Microsoft Office Compatibility Pack performs the necessary conversion to allow Excel 2000/2003 to read and open it.

In addition to the Microsoft Office Compatibility Pack, it is important to enable the WebFOCUS Client Redirection Settings *Save As* option so that Excel 2000/2003 will be able to open the XLSX report output without users first having to save it to their machine with the .xlsx file extension. The WebFOCUS Client processing Redirection Settings *Save As* option configures how the WebFOCUS Client sends each report output file type to the user machine. This option can be set as follows:

- ☐ **Save As Option disabled (NO).** The WebFOCUS Client Redirection Setting *Save As* is disabled by default. When the *Save As* option is disabled, the WebFOCUS Client sends report output to the user machine in memory with the application association specified for the report format in the WebFOCUS Client Redirection Settings configuration file (mime.wfs).

A user machine that does not have Excel 2007/2010 installed will not recognize the application association for Excel 2007/2010 and Excel will display a message.

The Excel 2000/2003 user can select *Save* and provide a file name with the .xlsx extension to save the report output to their machine. The user can then open the .xlsx file directly from Excel 2000/2003.

- ❑ **Save As Option enabled (YES).** When the WebFOCUS Redirection Save As option is enabled, the WebFOCUS Client sends the report output to the user as a file with the extension specified in the WebFOCUS Client Redirection Settings configuration file (mime.wfs).

Upon receiving the file, Windows will display the File Download prompt asking the user to Open or Save the file with the identified application type. The File Download prompt displays the Name with the .xlsx file extension for the report output that is recognized as an Excel XLSX file type.

**Note:** The download prompt will display for all users, including users who have Excel 2007/2010 installed on their machines.

If an Excel 2000/2003 user chooses to open the file, the Microsoft Office Compatibility Pack will recognize the .xlsx file extension and perform the necessary conversion to allow Excel 2000/2003 to read the Excel XLSX workbook.

If an Excel 2007/2010 user chooses to open the file, Excel will recognize the .xlsx file extension and read the Excel XLSX workbook.

For additional information on WebFOCUS Client Redirection Settings, see the *WebFOCUS Security and Administration Guide*.

### **Reference: Viewing Excel Workbooks in the Browser vs. the Excel Application**

Your Operating System and desktop settings determine whether Excel output sent to the client is displayed in an Internet Browser window or within the Excel application. When Excel output has been defined within the Windows environment to *Browse in same window*, the workbook generated by a WebFOCUS request is opened within an Internet Explorer® browser window. When the *Browse in same window* option is unchecked for the .xls file type, the browser window created by WebFOCUS is blank because the report output is displayed in the stand-alone Excel application window.

- ❑ In Windows XP and earlier, file type specific settings are managed on the desktop within Windows Explorer by selecting *Tools/Folder Options*, clicking the *File Types* tab, selecting the extension (.xls or .xlsx), clicking the *Advanced* button, and checking the *Browse in same window* box.

- ❑ In Windows 7, Microsoft removed the desktop settings that support opening worksheets in the browser. This means that to change this behavior, you can no longer simply navigate to the Folder Options dialog box, but that you must change a registry setting. This change is documented in the Microsoft Knowledge Base Article ID 927009 at the following web site:

<http://support.microsoft.com/kb/927009>

**Note:** This works the same for both EXL2K and XLSX formats. The only difference is the selection of file type based on the version of Excel output you will be generating.

## Formatting Values Within Cells in XLSX Report Output

WebFOCUS formats defined in Master Files or within a FOCEXEC will be represented in the resulting cells in an Excel XLSX worksheet. Where possible, the WebFOCUS formats are translated to custom Excel formats and applied to values passed as raw data. Each data value passed to a cell in Excel is defined with a value and a format mask pair. The data format is associated with the cell rather than embedded in the value. This technique provides enhanced support for editing worksheets generated by WebFOCUS. New values entered into existing cells will retain the cell formats and continue to display in the style defined for the column within the report.

The following types of data can be passed to Excel:

- ❑ **Numeric.** Where corresponding Excel format masks can be defined, numeric values are passed as raw values with associated format masks. In instances where an equivalent format mask cannot be defined, the numeric value is passed as a text string.
- ❑ **Alphanumeric.** Alphanumeric formats are passed to Excel as text strings, with General format defined. By default, General format presents all text fields as left-justified. Alignment and other styling attributes can be applied to these cells to override the default.
- ❑ **Date formats.** Data that contain sufficient elements to define a valid Excel date format are passed as raw date values with the WebFOCUS formats translated to Excel date format masks. In WebFOCUS formats that do not contain sufficient information to create valid Excel date values, the dates are converted to text strings.
- ❑ **Date-Time formats.** Date-time values are passed as raw date-time values with WebFOCUS formats translated to Excel date-time format masks using Custom formats.
- ❑ **Text.** Text values are passed as strings with General Format defined (as with alphanumeric data).

**Note:** This behavior is a change from EXL2K format, where cells containing dates and more complex numeric formats were passed as formatted text.

## Displaying Formatted Numeric Values in XLSX Report Output

Each numeric WebFOCUS format is translated to a custom numeric Excel format. The numeric value is displayed in the Excel formula bar for the selected cell. Within the actual cell, the value with the format mask applied displays.

The WebFOCUS formats for the following numeric data types are translated into Excel XLSX format masks supporting full editing within the resulting workbook:

- ☐ Data types: E, F, D, I, P
- ☐ Comma edit option (C)
- ☐ Zero suppression (S)
- ☐ Leading zero (L)
- ☐ Floating currency symbol (M)
- ☐ Comma suppression (c)
- ☐ Right-side minus sign (-)
- ☐ Credit negative (CR)
- ☐ Bracket negative (B)
- ☐ Fixed extended currency symbol (!d, !e, !l, !y)
- ☐ Floating extended currency symbol (!D, !E, !L, !Y)
- ☐ Percent (%)

### ***Example:*** Passing Numeric Formats to XLSX Report Output

In the following example, the DOLLARS field is assigned different numeric formats to demonstrate different available options. The column titles have been edited to display the WebFOCUS format options that have been applied:

```
TABLE FILE GGSales
SUM DOLLARS/D12.2 AS 'D12.2'
    DOLLARS/D12C AS 'D12C'
    DOLLARS/D12CM AS 'D12CM'
BY REGION
BY CATEGORY
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET BYDISPLAY ON
END
```

In the resulting worksheet, notice that cell C2 containing the DOLLAR value for *Midwest Coffee* presents the value with the WebFOCUS format D12.2, which presents the comma (,) and two decimal places. On the formula bar, the actual value is presented without any formatting. Examine each of the DOLLAR values in each row to see that the value as displayed in the formula bar remains the same, and only the display values presented in each cell change.

Also notice that with SET BYDISPLAY ON, the BY field values are repeated for every row on the worksheet. This creates fully qualified data rows that can be used with various data sorting, filtering, and table features in Excel without losing valuable information. This setting is recommended as a best practice for all worksheets.

A1      fx      Region						
	A	B	C	D	E	F
1	Region	Category	D12.2	D12C	D12CM	
2	Midwest	Coffee	4,178,513.00	4,178,513	\$4,178,513	
3	Midwest	Food	4,338,271.00	4,338,271	\$4,338,271	
4	Midwest	Gifts	2,883,881.00	2,883,881	\$2,883,881	
5	Northeast	Coffee	4,164,017.00	4,164,017	\$4,164,017	
6	Northeast	Food	4,379,994.00	4,379,994	\$4,379,994	
7	Northeast	Gifts	2,848,289.00	2,848,289	\$2,848,289	
8	Southeast	Coffee	4,415,408.00	4,415,408	\$4,415,408	
9	Southeast	Food	4,308,731.00	4,308,731	\$4,308,731	
10	Southeast	Gifts	2,986,240.00	2,986,240	\$2,986,240	
11	West	Coffee	4,473,517.00	4,473,517	\$4,473,517	
12	West	Food	4,202,337.00	4,202,337	\$4,202,337	
13	West	Gifts	2,977,092.00	2,977,092	\$2,977,092	
14						

The following example uses Fixed Dollar (N) format, as well as multiple combined format options. Each WebFOCUS format option is translated to the appropriate Excel XLSX format mask and applied to the cell value:

```
TABLE FILE GGSales
SUM BUDDOLLARS/D12N
DOLLARS/D12M
COMPUTE OVERBUDGET/D12BMc = BUDDOLLARS-DOLLARS; AS 'Over Budget'
BY REGION
BY CATEGORY
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET BYDISPLAY ON
END
```

Notice the fixed numeric format defined for the BUDDOLLARS column (Column C) presents the local currency symbol in a fixed position within each cell, regardless of the size of the data value. On the formula bar, the values in the *Over Budget* calculated field is passed as a negative value where appropriate. In the actual cells, the bracketed styling is applied to the negative values as part of the custom Excel XLSX format mask.

E2		fx -92481				
	A	B	C	D	E	F
1	Region	Category	Budget Dollars	Dollar Sales	Over Budget	
2	Midwest	Coffee	\$ 4,086,032	\$4,178,513	(\$92481)	
3	Midwest	Food	\$ 4,220,721	\$4,338,271	(\$117550)	
4	Midwest	Gifts	\$ 2,887,620	\$2,883,881	\$3739	
5	Northeast	Coffee	\$ 4,252,462	\$4,164,017	\$88445	
6	Northeast	Food	\$ 4,453,907	\$4,379,994	\$73913	
7	Northeast	Gifts	\$ 2,870,552	\$2,848,289	\$22263	
8	Southeast	Coffee	\$ 4,431,429	\$4,415,408	\$16021	
9	Southeast	Food	\$ 4,409,288	\$4,308,731	\$100557	
10	Southeast	Gifts	\$ 2,967,254	\$2,986,240	(\$18986)	
11	West	Coffee	\$ 4,523,963	\$4,473,517	\$50446	
12	West	Food	\$ 4,183,244	\$4,202,337	(\$19093)	
13	West	Gifts	\$ 2,934,306	\$2,977,092	(\$42786)	
14						
15						

Using Numeric Formats in Report Headings and Footings

By default, headings and footings are passed to Excel as a single character string. Spot markers are not supported for positioning within each line. Numeric fields and dates passed in headings and footings are passed as text strings within the overall heading or footing contents.

To display numeric fields and dates within headings and footings as numeric or date values, use HEADALIGN=BODY in the StyleSheet to define each of the items in the heading as an individual cell. Each cell containing numeric or date values will then be passed as the appropriate value with the associated format mask.

Using Numeric Format Punctuation in Headings and Footings

For data columns, all currency formats are translated using the Excel XLSX format masks that use the punctuation rules defined by the regional settings of the desktop.



In languages that use Continental Decimal Notation, the currency definitions designate that a comma (,) is used as the decimal separator, and a period (.) is used as the thousands separator, so D12.2CM may present the value as \$ 9.999,99 rather than the English (United States) value \$ 9,999.99. In headings and footings, you can designate that punctuation should be converted to Continental Decimal notation by issuing the SET CDN=ON command. With this setting in effect, the data embedded within heading and footing text strings will be formatted using the converted punctuation. Specify HEADALIGN=BODY to delineate items as individual cells and to retain the numeric formatting within the field, which will follow the same rules as the report data within the data columns.

**Reference: Usage Note for Sorting an XSLX Report That Contains a Footing**

In XLSX format, the report footer is included as a part of the data table in Excel. This is not the same behavior in EXL2K. In EXL2K format, the footer is not included as a part of the table.

For example, if you run the following procedure and sort the data table, the report footer is part of the data table, as shown in the image below the request:

```
TABLE FILE WF_RETAIL_LITE
PRINT COUNTRY_NAME AS Country
STATE_PROV_NAME AS State
PRODUCT_CATEGORY AS Category
WHERE RECORDLIMIT EQ 10
FOOTING
" "
"TEST FOOTING TEST FOOTING"
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET STYLE *
ENDSTYLE
END
```

The output is:

	A	B	C
1	Country	State	Category
2	Brazil	Mato Grosso	Video Production
3	Switzerland	Aargau	Media Player
4	Singapore	Central Singapore	Media Player
5	Canada	Ontario	Media Player
6	Germany	Thuringen	Media Player
7	Switzerland	Solothurn	Stereo Systems
8	Colombia	Distrito de Bogota	Stereo Systems
9	Switzerland	Zurich	Stereo Systems
10	Canada	Ontario	Media Player
11	Mexico	Sonora	Stereo Systems
12			
13	TEST FOOTING TEST FOOTING		

The workaround is to add a named data range to the procedure, as shown in the following procedure:

```
TABLE FILE WF_RETAIL_LITE
PRINT COUNTRY_NAME AS Country
STATE_PROV_NAME AS State
PRODUCT_CATEGORY AS Category
WHERE RECORDLIMIT EQ 10
FOOTING
" "
"TEST FOOTING TEST FOOTING"
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET STYLE *
TYPE=DATA, IN-RANGES=DATA, $
TYPE=TITLE, IN-RANGES=DATA, $
ENDSTYLE
END
```

The report footer is not part of the data table, as shown in the following image:

	A	B	C
1	Country	State	Category
2	Brazil	Mato Grosso	Video Production
3	Switzerland	Aargau	Media Player
4	Singapore	Central Singapore	Media Player
5	Canada	Ontario	Media Player
6	Germany	Thuringen	Media Player
7	Switzerland	Solothurn	Stereo Systems
8	Colombia	Distrito de Bogota	Stereo Systems
9	Switzerland	Zurich	Stereo Systems
10	Canada	Ontario	Media Player
11	Mexico	Sonora	Stereo Systems
12			
13	TEST FOOTING TEST FOOTING		

### Passing Dates to XLSX Report Output

Most translated and smart dates can be sent to Excel as standard date values with format masks, enabling Excel to use them in functions, formulas, and sort sequences.

Excel 2007 only supports mixed-case date text strings so all month and day names are displayed in mixed-case, regardless of how the case has been specified in the WebFOCUS format. For example, the WebFOCUS date format WRYMTRD presents the date text information in uppercase in all non-Excel formats. Excel transforms this value to mixed-case automatically.

In HTML, the date format displays as:

WRYMTRD
FRIDAY, 10 JANUARY 1

In XLSX, the date format displays as:

WRYMTRD
Friday, 10 January 1

**Example: Translating WebFOCUS Dates to Excel XLSX Dates**

The following request against the GGSALES data source creates the date January 1, 2010 and converts it to four date formats with translated text:

```
DEFINE FILE GGSALES
NEWDATE/MDYY = '01/01/2010';
WRMtrDY/WRMtrDY = NEWDATE;
wDMTY/wDMTY = NEWDATE;
wrDMTRY/wrDMTRY = NEWDATE;
wrYMtrD/wrYMtrD = NEWDATE;
END
TABLE FILE GGSALES
SUM DATE NOPRINT
NEWDATE WRMtrDY wDMTY wrDMTRY wrYMtrD
ON TABLE PCHOLD FORMAT XLSX
END
```

The following table shows how the dates should appear.

WebFOCUS Format	WebFOCUS Display	XLSX Display	XLSX Value
WRMtrDY	FRIDAY, January 1 10	Friday, January 1 10	1/1/2010
wDMTY	Fri, 1 JAN 10	Fri, 1 Jan 10	1/1/2010
wrDMTRY	Friday, 1 JANUARY 10	Friday, 1 January 10	1/1/2010
wrYMtrD	FRIDAY, 10 JANUARY 1	Friday, 10 January 1	1/1/2010

In Excel 2007/2010, all of the cells have a date value with format masks, and all month and day names are in mixed-case, regardless of how the case has been specified in the WebFOCUS format. The output is:

	A	B	C	D	E
1	NEWDATE	WRMtrDY	wDMTY	wrDMTRY	wrYMtrD
2	01/01/2010	Friday, January 1 10	Fri, 1 Jan 10	Friday, 1 January 10	Friday, 10 January 1

**Passing Dates Without a Day Component**

Date formats that do not specify the day value explicitly are defined as the date value of the first day of the month. Therefore, the value placed in the cell may be different from the day component value in the source data field and may produce unexpected results when used for sorting or date calculations in an Excel formula.

The following table shows how WebFOCUS date formats are represented in XLSX. The table shows how the value is preserved in the cell and how the display is generated using the format mask that corresponds to the WebFOCUS date format.

`DATEFLD/MDYY = '01/02/2010'`

WebFOCUS Format	XLSX Display	XLSX Value
DMYY	02/01/2010	1/2/2010
MY	01/10	1/1/2010
MTY	Jan, 10	1/1/2010
MTDY	Jan 2, 10	1/2/2010

**Example:** Passing WebFOCUS Dates With and Without a Day Component to XLSX Report Output

The following request against the GGSALES data source creates the date January 2, 2010 and passes it to Excel with formats MDYY, DMYY, MY, and MTDY:

```
DEFINE FILE GGSALES
NEWDATE/MDYY = '01/02/2010';
END
TABLE FILE GGSALES
SUM DATE NOPRINT
NEWDATE AS 'MDYY' NEWDATE/DMYY AS 'DMYY' NEWDATE/MY AS 'MY'
      NEWDATE/MTY AS 'MTY' NEWDATE/MTDY AS 'MTDY'
ON TABLE PCHOLD FORMAT XLSX
END
```

Columns D and E have actual date values with format masks, displayed by Excel 2007/2010 in mixed-case. Since the MTY format does not have a day component, the date value stored is the first of January 2010 (1/1/2010), not the second of January 2010 (1/2/2010):

	A	B	C	D	E
1	MDYY	DMYY	MY	MTY	MTDY
2	01/02/2010	02/01/2010	01/10	Jan, 10	Jan 2, 10

**Passing Date Components for Use in Excel Formulas**

Dates formatted as individual components (for example, D, Y, M, W) are passed to Excel as numeric values that can be used as parameters to Excel date functions. The values are passed as General format that are recognized by Excel as numbers.

**Example:**    **Passing Numeric Date Components to XLSX Report Output**

The following request against the GGSALES data source creates the date January 1, 2010 and extracts numeric date components, passing them to Excel 2007/2010:

```
DEFINE FILE GGSALES
NEWDATE/MDYY = '01/01/2010';
D/D = NEWDATE;
Y/Y = NEWDATE;
W/W = NEWDATE;
w/w = NEWDATE;
M/M = NEWDATE;
YY/YY = NEWDATE;
END
TABLE FILE GGSALES
SUM DATE NOPRINT
NEWDATE D Y W w M YY
ON TABLE PCHOLD FORMAT XLSX
END
```

The output is:

	A	B	C	D	E	F	G
1	NEWDATE	D	Y	W	w	M	YY
2	01/01/2010	01	10	5	5	01	2010

**Passing Quarter Formats**

Date formats that contain a Quarter component are always passed to Excel as text strings since Excel does not support Quarter formats.

**Example:**    **Passing Dates With a Quarter Component to XLSX Report Output**

The following request against the GGSALES data source creates the date January 1, 2010 and converts it to date formats that contain a Quarter component:

```
DEFINE FILE GGSALES
NEWDATE/MDYY = '01/01/2010';
Q/Q = NEWDATE;
QY/QY = NEWDATE;
YBQ/YBQ = NEWDATE;
END
TABLE FILE GGSALES
SUM DATE NOPRINT
NEWDATE Q QY YBQ
ON TABLE PCHOLD FORMAT XLSX
END
```

In XLSX, the cells containing dates with Quarter components have General format. To see this, open the Format Cells dialog box.

The output is:

	A	B	C	D
1	NEWDATE Q	QY	YBQ	
2	01/01/2010 Q1	Q1 10	10 Q1	

### Passing Date Components Defined as Translated Text

Date formats that do not contain sufficient information to present a valid date result in Excel are not translated to a value, including formats that do not contain year and/or month information. These dates will be sent to Excel as text. In the absence of complete information, the year defaults to the current year, so the value sent would be incorrect if this type of format was passed as a date value. The following formats will not be sent as values:

☐ MT, MTR, Mt, Mtr

☐ W, w, WR, wr

When date formats are passed to XLSX with format masks, all month and day names are in mixed-case, regardless of how the case has been specified in the WebFOCUS format. However, since the values in this example are always sent as text, the casing defined in the WebFOCUS format is applied in the resulting cell.

#### **Example:** Passing Date Components Defined as Translated Text to XLSX Report Output

The following request against the GGSALES data source creates the date January 1, 2010 and converts it to date formats that are defined as either month name or day name:

```
DEFINE FILE GGSALES
NEWDATE/MDYY = '01/01/2010';
MT/MT = NEWDATE;
MTR/MTR = NEWDATE;
Mtr/Mtr = NEWDATE;
WR/WR = NEWDATE;
wr/wr = NEWDATE;
END
TABLE FILE GGSALES
SUM DATE NOPRINT
NEWDATE MT MTR Mtr WR wr
ON TABLE PCHOLD FORMAT XLSX
END
```

In Excel 2007 or 2010, the cells containing the days have General format. To see this, open the Format Cells dialog box.

The output is:

	A	B	C	D	E	F
1	NEWDATE	MT	MTR	Mtr	WR	wr
2	01/01/2010	Jan	January	January	FRIDAY	Friday

**Reference: Usage Notes for Date Values in XLSX Report Output**

- ☐ The following date formats are not supported in XLSX. They will translate into Excel General format and possibly produce unpredictable results:
  - ☐ JUL, YYJUL, and I2MT.
  - ☐ Dates stored as a packed or alphanumeric field with date display options.

**Passing Date-Time to XLSX**

Most WebFOCUS date-time formats can be sent to XLSX as standard date/time values with format masks, enabling Excel to use them in functions, formulas, and sort sequences.

As with the Date formats, Excel only supports mixed-case to date-time fields, so if the date-time format contains text and is supported by Excel, the text will be in mixed-case, regardless of the casing defined within the WebFOCUS format.

**Example: Passing Date-Time to XLSX**

The following request shows an example against the GGSALES data source.

```

DEFINE FILE GGSALES
DT1/HYYMDm WITH REGION = DT(20100506 16:17:01.993876);
DPT1/HDMTTYm = DT1;
ALPHA_DATE1/A30 = HCNVRT(DT1, '(HYYMDm)', 30, 'A30');
END
TABLE FILE GGSALES
PRINT
ALPHA_DATE1
DT1 AS 'HYYMDm'
DPT1 AS 'HDMTTYm'
DT1/HdMTYYBS AS 'HdMTYYBS'
DT1/HdMTYYBs AS 'HdMTYYBs'
ON TABLE SET SPACES 1
IF RECORDLIMIT EQ 1
ON TABLE PCHOLD FORMAT XLSX
END
    
```



The output is:

C2 5/6/2010 4:17:02 PM				
A	B	C	D	E
ALPHA_DATE1	HYYYMDm	HDMTYYm	HdMTYYBS	HdMTYYBs
2010/05/06 16:17:01.993876	2010/05/06 16:17:01.993	06 May 2010 16:17:01.993	6 May 2010 16:17:01	6 May 2010 16:17:01.993

**Note:** Minutes by themselves are not supported in Excel and will be sent as an integer to XLSX with a Custom format.

Also, Excel time formats only support to the milliseconds. WebFOCUS formats that display microseconds will send the value to Excel, but the value will be rounded to milliseconds within the worksheet if the cell is edited.

The following table shows how the date-time values appear.

WebFOCUS Format	XLSX Displays	XLSX Value
HYYYMDm	2010/05/06 16:17:01.993	5/6/2010 4:17:02 PM
HDMTYYm	06 May 2010 16:17:01.993	5/6/2010 4:17:02 PM
HdMTYYBS	6 May 2010 16:17:01	5/6/2010 4:17:01 PM
HdMTYYBs	6 May 2010 16:17:01.993	5/6/2010 4:17:02 PM

## Generating Native Excel Formulas in XLSX Report Output

When you display or save a tabular report request using XLSX FORMULA, the resulting worksheet contains an Excel formula that computes and displays the results of any type of summed information, such as column totals, row totals, subtotals, and calculated values, rather than static numbers. A formula for a calculated value is generated by translating the internal form of the WebFOCUS expression into an Excel formula. Worksheets saved using the XLSX FORMULA format are interactive, allowing for "what if" scenarios that immediately reflect any additions or modifications made to the data.

## Understanding Formula Versus Value

The XLSX FORMULA format will generate formulas rather than values for the following WebFOCUS TABLE commands: ROW-TOTAL, COLUMN-TOTAL, SUB-TOTAL, SUBTOTAL, and SUMMARIZE, as well as for calculations performed by functions.

- ☐ A DEFINE field will always generate a constant value and not a formula.

- ❑ COMPUTE will generate the formula, except when the COMPUTE is equal to a single variable. In that case, the constant is placed and not the formula.
- ❑ If your report contains a calculated value (generated by the COMPUTE or RECOMPUTE command), all of the fields referenced by the calculated value must be displayed in the report in order for a cell reference to be included in the formula. If the referenced column is not displayed in the workbook, the data value will be placed in the formula, rather than a cell reference. Additionally, if the value cannot be reliably calculated based on the information passed to Excel, the value, rather than an expression, will be used. For example, using the LAST function in WebFOCUS cannot be translated correctly into Excel. In this instance, the LAST value is used in the expression, rather than a cell reference.

XLSX FORMULA is not supported with financial reports created with the Report canvas or the underlying Financial Modeling Language (FML).

For more information, see [Translation Support for FORMAT XLSX FORMULA](#) on page 622.

### **Reference:** Translation Support for FORMAT XLSX FORMULA

This topic describes translation support for FORMAT XLSX FORMULA. Use of unsupported WebFOCUS features may produce unreliable results.

- ❑ All standard operators are supported. These include arithmetic operators, relational operators, string operators, IF/THEN/ELSE, and logical operators. However, column notation is not supported.

The IS-PRESENT, IS-MISSING, IS-FROM, FROM, NOT-FROM, IS-MORE-THAN, IS-LESS-THAN, CONTAINS, and OMITS operators are not supported.

The logical operators AND and OR are not supported in conditional (IF-THEN-ELSE) or logical expressions.

- ❑ The following functions are supported:

ABS, ARGLEN, ATODBL, BYTVAL, CHARGET, CTRAN, DMOD, DOWK, DOWK, DOWKL, EXP, FMOD, HEXBYT, HHMMSS, IMOD, LCWORD, LOCASE, LOG, MAX, MIN, OVRLAY, POSIT, RDUNIF, SQRT, SUBSTR, TODAY, and UPCASE. The EDIT function is supported for converting formats (one argument variant). It is not supported for editing strings.

The functions CTRFLD, LJUST, and RJUST are not recommended for justifying data in Excel columns. With the use of Excel proportional fonts, the StyleSheet JUSTIFY attribute is more appropriate.

Be cautious when using functions that use decimal values as an argument (BYTVAL, CTRAN, HEXBYT). Based on whether the operating environment is EBCDIC or ASCII, the results may be different.

- ☐ XLSX FORMULA is not supported with the following WebFOCUS commands and phrases:
  - ☐ DEFINE
  - ☐ OVER
  - ☐ FOR
  - ☐ NOPRINT
  - ☐ Multiple display (PRINT, LIST, SUM, and COUNT) commands
  - ☐ SEQUENCE StyleSheet attribute
  - ☐ RECAP
  - ☐ SET HIDENULLACRS
  - ☐ SET SUBTOTALS = ABOVE
  - ☐ LAST
- ☐ The BYDISPLAY ON setting is recommended to allow the sort field value to be available on all rows for recalculations.
- ☐ If an expression requires more than 1024 characters, WebFOCUS will place the value into the cell, and not the formula.
- ☐ Conditional styling is based on the values in the original report. If the worksheet values are changed and the formulas are recomputed, the styling will not reflect the updated information.

**Syntax:**      **How to Save Reports as FORMAT XLSX FORMULA**

Add the following syntax to your request to take advantage of Excel formulas in your workbook:

```
ON TABLE {PCHOLD|HOLD} FORMAT XLSX FORMULA
```

where:

**PCHOLD**

Displays the output in an XLSX workbook.

**HOLD**

Saves the output for reuse in an Excel worksheet. For details, see [Saving and Reusing Your Report Output](#) on page 471.

**Example:**    **Generating Native Excel Formulas for Column Totals**

The following example illustrates how a column total in a report request is translated to an Excel formula when you use the XLSX FORMULA format. Notice that the formatting of the column total (TYPE=GRANDTOTAL) is retained in the Excel workbook. When you select the total in the report, the equation =SUM(B4:B10) displays in the formula bar, representing the column total as a sum of cell ranges.

```
TABLE FILE SHORT
HEADING
"Projected Return By Region"
" "
SUM PROJECTED_RETURN AS 'RETURN'
BY REGION AS 'REGION'
ON TABLE COLUMN-TOTAL
ON TABLE PCHOLD FORMAT XLSX FORMULA
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, FONT='ARIAL', SIZE=9, TITLETEXT='By Region',$
TYPE=TITLE, BACKCOLOR=RGB(102 102 102), COLOR=RGB(255 255 255),$
TYPE=HEADING, SIZE=12, STYLE=BOLD, JUSTIFY=CENTER,$
TYPE=GRANDTOTAL, BACKCOLOR=RGB(210 210 210), STYLE=BOLD,$
END
```

The output is:

	A	B
1	<b>Projected Return By Region</b>	
2		
3	<b>REGION</b>	<b>RETURN</b>
4	CENTRAL AMERICA	9.520
5	EASTERN EUROPE	16.150
6	FAR EAST	9.100
7	MIDDLE EAST	7.980
8	NORTH AMERICA	12.460
9	SOUTH AMERICA	8.400
10	WESTERN EUROPE	7.980
11	<b>TOTAL</b>	<b>71.590</b>
12		
13		
14		

WebFOCUS can translate any total (subtotal, row total, or column total) to an Excel formula. For related information, see [Translation Support for FORMAT XLSX FORMULA](#) on page 622.

### **Example:** Generating Native Excel Formulas for Row Totals

The following request calculates totals for returns and balances across continents. The row totals are represented as sums of cell ranges.

```
TABLE FILE SHORT
HEADING
"Projected Return Across Continent"
" "
SUM PROJECTED_RETURN AS 'Return' AND BALANCE AS 'Balance'
ACROSS CONTINENT AS 'CONTINENT'
BY REGION AS 'REGION'
ON CONTINENT ROW-TOTAL AS 'TOTAL'
ON TABLE COLUMN-TOTAL AS 'TOTAL'
ON TABLE PCHOLD FORMAT XLSX FORMULA
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, FONT='ARIAL', SIZE=9, TITLETEXT='Across Continent',$
TYPE=TITLE, BACKCOLOR=RGB(102 102 102), COLOR=RGB(255 255 255),$
TYPE=HEADING, SIZE=12, STYLE=BOLD, JUSTIFY=CENTER,$
TYPE=ACROSSTITLE, STYLE=BOLD,$
TYPE=GRANDTOTAL, BACKCOLOR=RGB(210 210 210), STYLE=BOLD,$
END
```

The following output highlights the formula that calculates the row total in cell I12=C12+E12+G12.

=C12+E12+G12									
	A	B	C	D	E	F	G	H	I
1	<b>Projected Return Across Continent</b>								
2									
3		<b>CONTINENT</b>							
4		<b>AMERICAS</b>		<b>ASIA</b>		<b>EUROPE</b>		<b>TOTAL</b>	
5	<b>REGION</b>	<b>Return</b>	<b>Balance</b>	<b>Return</b>	<b>Balance</b>	<b>Return</b>	<b>Balance</b>	<b>Return</b>	<b>Balance</b>
6	CENTRAL AMERICA	9.520	40,200,667	.	.	.	.	9.520	40,200,667
7	EASTERN EUROPE	.	.	.	.	16.150	121,977,597	16.150	121,977,597
8	FAR EAST	.	.	9.100	202,660,842	.	.	9.100	202,660,842
9	MIDDLE EAST	.	.	7.980	44,659,771	.	.	7.980	44,659,771
10	NORTH AMERICA	12.460	367,610,683	.	.	.	.	12.460	367,610,683
11	SOUTH AMERICA	8.400	87,661,381	.	.	.	.	8.400	87,661,381
12	WESTERN EUROPE	.	.	.	.	7.980	210,550,374	7.980	210,550,374
13	<b>TOTAL</b>	<b>30.380</b>	<b>495,472,731</b>	<b>17.080</b>	<b>247,320,613</b>	<b>24.130</b>	<b>332,527,971</b>	<b>71.590</b>	<b>1,075,321,315</b>
14									
15									

**Example: Generating Native Excel Formulas for Calculated Values**

The following request totals the columns for retail cost and dealer cost, and calculates the value of a field called PROFIT by subtracting the DOLLARS from the BUDDOLLARS.

The formula for the calculated values is generated by translating the internal form of the WebFOCUS expression (PROFIT/D12.2MC = BUDDOLLARS - DOLLARS;) into an Excel formula. In this example, the formulas appear in cells B8, C8, and D8.

All fields referenced in the calculation should be displayed in the report for a valid formula to be created using cell references. Otherwise, it may be created using values not in the report. If the fields used in the calculation are not present in the report and there is a subsequent RECOMPUTE, the formula created for the RECOMPUTE will not be correct.

```
TABLE FILE GGSALES
ON TABLE SET PAGE-NUM OFF
SUM BUDDOLLARS/I8MC AND DOLLARS/I8MC
COMPUTE PROFIT/D12.2MC = BUDDOLLARS - DOLLARS;
BY REGION
HEADING
"Profit By Region"
" "
ON TABLE COLUMN-TOTAL
ON TABLE PCHOLD FORMAT XLSX FORMULA
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, FONT='ARIAL', SIZE=9, TITLETEXT='By Region',$
TYPE=TITLE, BACKCOLOR=RGB(102 102 102), COLOR=RGB(255 255 255),$
TYPE=HEADING, SIZE=12, STYLE=BOLD, JUSTIFY=CENTER,$
TYPE=GRANDTOTAL, BACKCOLOR=RGB(210 210 210), STYLE=BOLD,$
END
```

The following output highlights the formula that calculates for the column total of PROFIT: D8=SUM(D4:D7).

	A	B	C	D
1	Profit By Region			
2				
3	Region	Budget Dollars	Dollar Sales	PROFIT
4	Midwest	\$11,194,373	\$11,400,665	-\$206,292.00
5	Northeast	\$11,576,921	\$11,392,300	\$184,621.00
6	Southeast	\$11,807,971	\$11,710,379	\$97,592.00
7	West	\$11,641,513	\$11,652,946	-\$11,433.00
8	TOTAL	\$46,220,778	\$46,156,290	\$64,488.00
9				

**Example: Generating a Native Excel Formula for a Function**

The following example illustrates how functions are translated to Excel reports. The function IMOD divides ACCTNUMBER by 1000 and returns the remainder to LAST3\_ACCT. The Excel formula corresponds to =TRUNC((MOD(\$C3,(1000)))). TRUNC is used when the answer returned from an equation is being placed into an Integer field, to be sure there are no decimals.

```
TABLE FILE EMPLOYEE
PRINT ACCTNUMBER AS 'Account Number'
COMPUTE LAST3_ACCT/I3L = IMOD(ACCTNUMBER, 1000, LAST3_ACCT);
BY LAST_NAME AS 'Last Name'
BY FIRST_NAME AS 'First Name'
WHERE (ACCTNUMBER NE 000000000) AND (DEPARTMENT EQ 'MIS');
ON TABLE PCHOLD FORMAT XLSX FORMULA
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, FONT='ARIAL', SIZE=9,$
TYPE=TITLE, BACKCOLOR=RGB(102 102 102), COLOR=RGB(255 255 255), STYLE=BOLD,$
END
```

The output is:

f_x =TRUNC((MOD(\$C3,1000)))				
	A	B	C	D
1	Last Name	First Name	Account Number	LAST3_ACCT
2	BLACKWOOD	ROSEMARIE	122850108	108
3	CROSS	BARBARA	163800144	144
4	GREENSPAN	MARY	150150302	302
5	JONES	DIANE	040950036	036
6	MCCOY	JOHN	109200096	096
7	SMITH	MARY	027300024	024
8				

**Reference: Generating a Formula With Recomputed Values**

If your report contains a calculated value (generated by the COMPUTE or RECOMPUTE command), all of the fields referenced by the calculated value must be displayed in the report in order for cell references to be included in the formula. If a referenced column is not displayed in the workbook, the data value will be placed in the formula, rather than a cell reference. In the case of RECOMPUTE, the value used may be an incorrect value from the last detail record of the sort break.

**Example: Generating a Formula With Recomputed Values**

The following request computes the difference (DIFF) by subtracting budgeted dollars from dollar sales. The budgeted dollars field used in the expression is not included in the SUM command. The value of DIFF is recomputed on the region level.

```
TABLE FILE GGSALES
HEADING
"Profit By Region"
" "
SUM DOLLARS/I8CM
COMPUTE DIFF/I8CM=DOLLARS - BUDDOLLARS;
BY REGION
BY CATEGORY
ON REGION RECOMPUTE
ON TABLE PCHOLD FORMAT XLSX FORMULA
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, FONT='ARIAL', SIZE=9, TITLETEXT='By Region',$
TYPE=TITLE, BACKCOLOR=RGB(102 102 102), COLOR=RGB(255 255 255),$
TYPE=HEADING, SIZE=12, STYLE=BOLD, JUSTIFY=CENTER,$
TYPE=SUBTOTAL, BACKCOLOR=RGB(210 210 210),$
TYPE=GRANDTOTAL, BACKCOLOR=RGB(166 166 166), STYLE=BOLD,$
END
```

The output shows that the formula is subtracting a data value that is not displayed on the worksheet. It is actually the BUDDOLLARS value from the current hardcoded value, since there is no cell reference.

fx =TRUNC(\$C7 -2887620)				
	A	B	C	D
1	<b>Profit By Region</b>			
2				
3	Region	Category	Dollar Sales	DIFF
4	Midwest	Coffee	\$4,178,513	\$92,481
5		Food	\$4,338,271	\$117,550
6		Gifts	\$2,883,881	-\$3,739
7	*TOTAL Midwest		\$11,400,665	\$8,513,045
8	Northeast	Coffee	\$4,164,017	-\$88,445
9		Food	\$4,379,994	-\$73,913
10		Gifts	\$2,848,289	-\$22,263
11	*TOTAL Northeast		\$11,392,300	\$8,521,748
12	Southeast	Coffee	\$4,415,408	-\$16,021
13		Food	\$4,308,731	-\$100,557
14		Gifts	\$2,986,240	\$18,986
15	*TOTAL Southeast		\$11,710,379	\$8,743,125
16	West	Coffee	\$4,473,517	-\$50,446
17		Food	\$4,202,337	\$19,093
18		Gifts	\$2,977,092	\$42,786
19	*TOTAL West		\$11,652,946	\$8,718,640
20	<b>TOTAL</b>		<b>\$46,156,290</b>	<b>-\$64,488</b>
21				

If you add the BUDDOLLARS column to the request, the formula can be recomputed correctly.

```
SUM DOLLARS/I8MC BUDDOLLARS/I8MC
```



The formula generated with the new SUM command contains cell references for both fields used in the calculation.

=TRUNC(\$C7 -\$D7 )					
	A	B	C	D	E
1	<b>Profit By Region</b>				
2					
3	<b>Region</b>	<b>Category</b>	<b>Dollar Sales</b>	<b>Budget Dollars</b>	<b>DIFF</b>
4	Midwest	Coffee	\$4,178,513	\$4,086,032	\$92,481
5		Food	\$4,338,271	\$4,220,721	\$117,550
6		Gifts	\$2,883,881	\$2,887,620	-\$3,739
7	*TOTAL Midwest		\$11,400,665	\$11,194,373	\$206,292
8	Northeast	Coffee	\$4,164,017	\$4,252,462	-\$88,445
9		Food	\$4,379,994	\$4,453,907	-\$73,913
10		Gifts	\$2,848,289	\$2,870,552	-\$22,263
11	*TOTAL Northeast		\$11,392,300	\$11,576,921	-\$184,621
12	Southeast	Coffee	\$4,415,408	\$4,431,429	-\$16,021
13		Food	\$4,308,731	\$4,409,288	-\$100,557
14		Gifts	\$2,986,240	\$2,967,254	\$18,986
15	*TOTAL Southeast		\$11,710,379	\$11,807,971	-\$97,592
16	West	Coffee	\$4,473,517	\$4,523,963	-\$50,446
17		Food	\$4,202,337	\$4,183,244	\$19,093
18		Gifts	\$2,977,092	\$2,934,306	\$42,786
19	*TOTAL West		\$11,652,946	\$11,641,513	\$11,433
20	<b>TOTAL</b>		<b>\$46,156,290</b>	<b>\$46,220,778</b>	<b>-\$64,488</b>
21					

### Using XLSX FORMULA With Prefix Operators

XLSX FORMULA output supports prefix operators that are used on summary lines generated by WebFOCUS commands, such as SUBTOTAL and RECOMPUTE. Where a corresponding formula exists in Excel, these prefix operators are translated into the equivalent Excel summarization formula. The results of prefix operators used directly against retrieved data continue to be passed to Excel as values, not formulas.

The following table identifies the prefix operators supported by XLSX FORMULA when used on summary lines, and the Excel formula equivalent placed in the generated worksheet.

Prefix Operator	Excel Formula Equivalent
SUM.	=SUM()
AVE.	=AVERAGE()

Prefix Operator	Excel Formula Equivalent
CNT.	=COUNT()
MIN.	=MIN()
MAX.	=MAX()

The following prefix operators are not translated to formulas when used on summary lines in XLSX FORMULA.

- ☐ ASQ.
- ☐ FST.
- ☐ LST.

**Note:**

- ☐ When using a prefix operator on a field specified directly against retrieved data, there is no space between the prefix operator and the field on which it operates.

For example, in the following aggregating display command, the AVE. prefix operator operates on the DOLLARS field.

```
SUM AVE.DOLLARS
```

- ☐ When using a prefix operator on a summary line, you must leave a space between the prefix operator and the aggregated field on which it operates.

In the following summary command, the MAX. prefix operator operates on the DOLLARS field at the REGION sort break. Note the required blank space between the prefix operator and the field name.

```
ON REGION RECOMPUTE MAX. DOLLARS
```

**Example: Using a Summary Prefix Operator With FORMAT XLSX FORMULA**

In the following request against the GGSales data source, the RECOMPUTE command for the REGION sort field calculates the maximum of the aggregated DOLLARS field and the minimum of the aggregated BUDDOLLARS field.

```

TABLE FILE GGSALES
SUM UNITS DOLLARS/I8MC BUDDOLLARS/I8MC
AND COMPUTE DIFF/I8MC= DOLLARS-BUDDOLLARS;
BY REGION
BY CATEGORY
WHERE CATEGORY EQ 'Food' OR 'Coffee'
WHERE REGION EQ 'West' OR 'Midwest'
ON REGION RECOMPUTE MAX. DOLLARS MIN. BUDDOLLARS DIFF
ON TABLE PCHOLD FORMAT XLSX FORMULA
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, FONT='ARIAL', SIZE=9,$
TYPE=TITLE, BACKCOLOR=RGB(102 102 102), COLOR=RGB(255 255 255),$
TYPE=SUBTOTAL, BACKCOLOR=RGB(210 210 210),$
TYPE=GRANDTOTAL, BACKCOLOR=RGB(166 166 166), STYLE=BOLD,$
END

```

In the output, shown in the following image, the cell that represents the recomputed DOLLARS for the Midwest region has been generated as the following formula.

```
=MIN(E2:E3)
```

	fx =MIN(E2:E3)					
	A	B	C	D	E	F
1	Region	Category	Unit Sales	Dollar Sales	Budget Dollars	DIFF/I10
2	Midwest	Coffee	332777	\$4,178,513	\$4,086,032	\$92,481
3		Food	341414	\$4,338,271	\$4,220,721	\$117,550
4	*TOTAL Midwest			\$4,338,271	\$4,086,032	\$252,239
5	West	Coffee	356763	\$4,473,517	\$4,523,963	-\$50,446
6		Food	340234	\$4,202,337	\$4,183,244	\$19,093
7	*TOTAL West			\$4,473,517	\$4,183,244	\$290,273
8	TOTAL			\$4,473,517	\$4,086,032	\$387,485
9						

### Example: Using a Prefix Operator on a Display Command With FORMAT XLSX FORMULA

In the following request against the GGSALES data source, the CNT., AVE., and PCT. Prefix operators are used in the SUM display command.

```

TABLE FILE GGSALES
SUM UNITS
CNT.UNITS
AVE.UNITS
PCT.UNITS
BY REGION
BY ST
ON TABLE PCHOLD FORMAT XLSX FORMULA
END

```

The output shows that the prefix operators were not passed to Excel as formulas. They were passed as data values.

		fx 360				
	A	B	C	D	E	F
1	Region	State	Unit Sales	Unit Sales COUNT	AVE Unit Sales	PCT Unit Sales
2	Midwest	IL	307581	360	854	8
3		MO	297727	359	829	8
4		TX	299737	360	832	8
5	Northeast	CT	302440	360	840	8
6		MA	301909	360	838	8
7		NY	312326	360	867	8
8	Southeast	FL	310302	360	861	8
9		GA	330283	360	917	8
10		TN	294647	360	818	7
11	West	CA	610570	719	849	16
12		WA	321469	359	895	8
13						

### NODATA With Formulas

Support for full Excel functionality requires that only valid numeric values are placed into cells that will be used for formula references.

The null value (NODATA='') is supported for calculations. When cells containing the default NODATA symbol (.) are used in a formula, they will cause a formula error.

For example:

```

SET NODATA=' '
TABLE FILE GGSales
SUM DOLLARS/D12CM UNITS/D12C AND ROW-TOTAL AND COLUMN-TOTAL
COMPUTE REVENUE/D12CM=DOLLARS*UNITS; AS 'Revenue'
BY LOWEST GGSales.SALES01.CATEGORY
BY GGSales.SALES01.PRODUCT
ACROSS REGION
ON TABLE PCHOLD FORMAT XLSX FORMULA
END
-----
SET NODATA=' '
DEFINE FILE GGSales
DOLLARMOD/D12CM MISSING ON=IF REGION GT 'V' THEN MISSING ELSE DOLLAR;
END
TABLE FILE GGSales
SUM DOLLARMOD/D12CM UNITS/D12C AND ROW-TOTAL AND COLUMN-TOTAL
COMPUTE REVENUE/D12CM=DOLLARMOD*UNITS; AS 'Revenue'
BY REGION
BY LOWEST GGSales.SALES01.CATEGORY
BY GGSales.SALES01.PRODUCT
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE PCHOLD FORMAT XLSX FORMULA
END

```

### **Reference:** Usage Notes for XLSX With Formulas

- ☐ Formulas are defined within a single worksheet. They will not be assigned across worksheets.
- ☐ BYTOC compound workbooks can contain formulas. A separate worksheet/tab is generated for each primary key value and formulas are defined with references within that sheet. In BYTOC compound workbooks, a separate tab is generated for overall grand totals. These will not contain formula references to the component worksheets.

### **Controlling Column Width and Wrapping in XLSX Report Output**

- ☐ Column width and data wrapping can be controlled in an Excel worksheet when using FORMAT XLSX.
- ☐ To size the column without wrapping and define the exact size width, use SQUEEZE=ON. If a data value is wider than the specified width of the column, a portion of the data will be hidden from view, but fully visible in the formula bar. You can adjust the column width in Excel after the worksheet has been generated.
- ☐ The default behavior is for all data to wrap within the defined column width. You can also specify the exact width of a column using WRAP=ON.
- ☐ WRAP is not supported for Date format fields.

**Syntax:**      **How to Set Column Width in XLSX Report Output**

`TYPE=REPORT, [COLUMN=column,] SQUEEZE=value, $`

where:

*column*

Identifies a particular column. If COLUMN is not included in the declaration, default SQUEEZE behavior is applied to the entire report.

*value*

Is one of the following:

`ON`

Automatically sizes the columns based on the largest data value in the column. This is the default behavior.

`OFF`

Sizes the columns based on the maximum size defined for the field in the Master File or Define.

*n*

Represents a specific numeric value for which the column width can be set. The value represents the measure specified with the UNITS parameter (the default is inches). This is the most commonly used SQUEEZE setting in an XLSX report. This turns off data wrapping.

**Note:**

- ☐ SQUEEZE can be applied to the entire report by using the ON TABLE SET SQUEEZE ON command.
- ☐ SQUEEZE is not supported for columns created with the OVER phrase or with TABLEF.

**Syntax:**      **How to Wrap Data in XLSX Report Output**

`TYPE=REPORT, [COLUMN=column,] WRAP=value, $`

where:

*column*

Designates a particular column to apply wrapping behavior to. If COLUMN is not included in the declaration, wrapping will be applied to the entire report.

*value*

Is one of the following:

**ON**

Turns on data wrapping. ON is the default value. With this setting, the column width is determined by the client (Excel). Data wraps if it exceeds the width of the column and the row height expands to meet the new height of the wrapped data.

**OFF**

Turns off data wrapping. Data will not wrap in any cell in the column.

*n*

Represents a specific numeric value that the column width can be set to. The value represents the measure specified with the UNITS parameter (the default is inches).

This setting implies ON. However, the column width is set to the specified width unless the data is wider than the column width, in which case, wrapping will occur as for ON.

**Note:** WRAP is not supported for Date format fields.

### **Example:** Controlling Column Width and Wrapping in XLSX Report Output

The following example illustrates how to turn on and turn off data wrapping in a column and how to set the column width for a particular column. The UNITS in this example are set to inches (the default).

```
DEFINE FILE GGSALES
PROFIT/D14.3 = BUDDOLLARS-DOLLARS;
DESCRIPTION/A80 = 'Subtract Total Sales Quota from Reported Sales to
calculate profit.';
END
```

```
TABLE FILE GGSALES
SUM
DESCRIPTION AS 'DEFAULT'
DESCRIPTION AS 'WRAP = 2'
DESCRIPTION AS 'WRAP = OFF'
DESCRIPTION AS 'SQUEEZE = 1.5'
PROFIT
BY REGION NOPRINT
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET STYLE *
TYPE=REPORT, COLUMN=DESCRIPTION(2), WRAP=2,$
TYPE=REPORT, COLUMN=DESCRIPTION(3), WRAP=OFF,$
TYPE=REPORT, COLUMN=DESCRIPTION(4), SQUEEZE=1.5,$
END
```

where:

- 1. The column titled "DEFAULT" illustrates the default column width and wrapping behavior.
- 2. The column titled "WRAP=2" sets the column width to 2 inches with data wrapping on.
- 3. The column titled "WRAP=OFF" turns off data wrapping for that column.
- 4. The column titled "SQUEEZE=1.5" sets the column width to 1.5 inches with data wrapping off.

Since the output spans two pages, the output is shown below in two separate images.

The following output displays the different behavior for the "DEFAULT" and "WRAP=2" columns.

A	B
DEFAULT	WRAP=2
Subtract Total Sales Quota from Reported Sales to calculate profit.	Subtract Total Sales Quota from Reported Sales to calculate profit.
Subtract Total Sales Quota from Reported Sales to calculate profit.	Subtract Total Sales Quota from Reported Sales to calculate profit.
Subtract Total Sales Quota from Reported Sales to calculate profit.	Subtract Total Sales Quota from Reported Sales to calculate profit.
Subtract Total Sales Quota from Reported Sales to calculate profit.	Subtract Total Sales Quota from Reported Sales to calculate profit.
Subtract Total Sales Quota from Reported Sales to calculate profit.	Subtract Total Sales Quota from Reported Sales to calculate profit.

The following output displays the output for the "WRAP=OFF" and "SQUEEZE=1.5" columns.

fx Subtract Total Sales Quota from Reported Sales to calculate profit.		
C	D	E
WRAP=OFF	SQUEEZE=1.5	PROFIT
Subtract Total Sales Quota from Reported Sales to calculate profit.	Subtract Total Sales Quota	-206,292.000
Subtract Total Sales Quota from Reported Sales to calculate profit.	Subtract Total Sales Quota	184,621.000
Subtract Total Sales Quota from Reported Sales to calculate profit.	Subtract Total Sales Quota	97,592.000
Subtract Total Sales Quota from Reported Sales to calculate profit.	Subtract Total Sales Quota	-11,433.000

Preserving Leading and Internal Blanks in Report Output

The SHOWBLANKS command allows you to preserve leading blanks in data cells in XLSX reports.

HEADALIGN=BODY can be used to support trailing blanks in a heading, subheading, footing, and subfooting.



**Note:** Since XLSX is not HTML-based like EXL2K, setting SHOWBLANKS OFF will not affect internal blanks. All internal blanks will be displayed. This means that by default, in XLSX, all leading blanks in data fields are removed. Because the spacing is different, wrapped fields may display differently in the spreadsheet.

SET SHOWBLANKS Command	XLSX (not HTML-based)	EXL2K (HTML-based)
SET SHOWBLANKS = ON	Leading and additional embedded blanks are preserved.	Leading and additional embedded blanks are preserved.
SET SHOWBLANKS = OFF	Leading blanks are removed, but embedded blanks are respected.	Leading and additional embedded blanks are removed.
HEADALIGN=BODY	If there are leading blanks and HEADALIGN=BODY is set, both leading and trailing blanks will be displayed.	If HEADALIGN=BODY is set, EXL2K will never display the trailing blanks.

### *Syntax:*

### How to Preserve Leading and Internal Blanks in XLSX Reports

In a FOCEXEC or in a profile, use the following syntax:

```
SET SHOWBLANKS = {OFF|ON}
```

In a request, use the following syntax

```
ON TABLE SET SHOWBLANKS {OFF|ON}
```

where:

#### OFF

Removes leading blanks and preserves internal blanks in XLSX report output. OFF is the default value.

#### ON

Preserves leading and internal blanks in XLSX report output. Also preserves trailing blanks in heading, subheading, footing, subfooting lines that use the default heading or footing alignment.

**Example: Preserving Leading and Internal Blanks in XLSX Report Output**

The following request creates a virtual field that adds leading blanks to the value ACTION, and both leading and internal blanks to the values TRAIN/EX and SCI/FI in the CATEGORY field.

```

SET SHOWBLANKS = OFF
DEFINE FILE MOVIES
NEWCAT/A30 = IF CATEGORY EQ 'ACTION' THEN ' ACTION'
ELSE IF CATEGORY EQ 'SCI/FI' THEN ' SCIENCE FICTION'
ELSE IF CATEGORY EQ 'TRAIN/EX' THEN ' TRAINING EXERCISE'
ELSE IF CATEGORY EQ 'COMEDY' THEN 'COMEDY '
ELSE 'GENERAL';
END
TABLE FILE MOVIES
HEADING
"Example of Excel produced using SET SHOWBLANKS=OFF and XLSX"
ON TABLE SUBHEAD
"<NEWCAT<NEWCAT"
SUM CATEGORY LISTPR/D12.2 COPIES
BY NEWCAT
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET STYLE *
GRID=OFF,$
TYPE=REPORT, FONT=COURIER NEW,$
TYPE=TABHEADING, HEADALIGN=BODY,$
ENDSTYLE
END

```

The following reports show the differences in Excel generated using SET SHOWBLANKS = OFF and SET SHOWBLANKS = ON.

**SET SHOWBLANKS = OFF with HEADALIGN=BODY**

	A	B	C	D
1	TRAINING EXERCISE	TRAINING EXERCISE		
2	Example of Excel produced using SET SHOWBLANKS=OFF and XLSX			
3	NEWCAT	CATEGORY	LISTPR	COPIES
4	ACTION	ACTION	94.82	14
5	SCIENCE FICTION	SCI/FI	114.84	7
6	TRAINING EXERCISE	TRAIN/EX	119.87	10
7	COMEDY	COMEDY	154.80	19
8	GENERAL	MYSTERY	1,216.82	67
9				

**SET SHOWBLANKS = OFF without HEADALIGN=BODY**

	A	B	C	D
1	TRAINING EXERCISE	TRAINING EXERCISE		
2	Example of Excel produced using SET SHOWBLANKS=OFF without HEADALIGN=BODY			
3	NEWCAT	CATEGORY	LISTPR	COPIES
4	ACTION	ACTION	94.82	14
5	SCIENCE FICTION	SCI/FI	114.84	7
6	TRAINING EXERCISE	TRAIN/EX	119.87	10
7	COMEDY	COMEDY	154.80	19
8	GENERAL	MYSTERY	1,216.82	67
9				

**SET SHOWBLANKS = ON with HEADALIGN=BODY**

	A	B	C	D
1	TRAINING EXERCISE	TRAINING EXERCISE		
2	Example of Excel produced using SET SHOWBLANKS=ON and XLSX			
3	NEWCAT	CATEGORY	LISTPR	COPIES
4	ACTION	ACTION	94.82	14
5	SCIENCE FICTION	SCI/FI	114.84	7
6	TRAINING EXERCISE	TRAIN/EX	119.87	10
7	COMEDY	COMEDY	154.80	19
8	GENERAL	MYSTERY	1,216.82	67
9				

**Support for Drill Downs With XLSX Report Output**

Drill downs are supported within the data elements in a report in XLSX format in the same manner as they are supported in EXL2K format. Hyperlink connections can be defined in the StyleSheet declaration of any data column to provide access to any external web source or to execute a FOCEXEC. Drill downs to FOCEXECs can contain data-driven parameters and can generate any of the supported output formats, including XLSX, PDF, HTML, DHTML, and PPT.

Drill downs within text embedded in headings, subheadings, subfootings, and footings will be implemented for XLSX format in a future release.

### Note:

- ❑ When the limit of 65530 hyperlinks for a worksheet is reached, a warning message displays and no further links can be inserted. For more information on drill downs, see [Linking Using StyleSheets](#) on page 763.
- ❑ The JAVASCRIPT and IMAGE drill-down options are not supported with FORMAT XLSX.

### Redirection and Excel Drill-Down Reports

The WebFOCUS Client can use redirection when passing the report output to the client application. When redirection is enabled, the WebFOCUS Client saves report output in a temporary directory when a request is executed. Then, an HTTP call is made from the browser to retrieve the temporary stored output for display. When redirection is disabled, the report output is sent directly to the browser without any buffering.

Redirection is disabled by default for the .xlsx file extension because this enables drill downs to run successfully whether the user machine is configured to launch Excel in the browser or as an application outside of the browser.

When redirection is enabled, drill downs within Excel reports will work differently depending on whether the workbook is opened in the browser (only applies to Windows XP) or in the Excel application. For information about launching Excel in the browser or as an application, see [Viewing Excel Workbooks in the Browser vs. the Excel Application](#) on page 608.

- ❑ **For workbooks opened outside the browser in the Excel application:** The current security context and any previously established session-related cookies are not retained, changing the user authorization, so drill-down reports will not have the information required to access the redirected files. The initial workbook will open within Excel, but the target drill-down workbook will not open and you will receive a message stating *You are not allowed to access this viewer file*. The drill-down feature in Microsoft Office products functioned in WebFOCUS Release 7.7.x because anonymous drill-down access was permitted.

The following options are available to allow the feature in WebFOCUS Release 8.x:

- ❑ Configure WebFOCUS authentication to allow anonymous access. For more information, see the *WebFOCUS Security and Administration* manual.
- ❑ Use SSO with IIS/Tomcat Integrated Windows Authentication. Renegotiation occurs automatically and the Excel and PowerPoint reports display correctly.
- ❑ As of WebFOCUS Release 8.x, the Remember Me feature can be enabled on the Sign-in page. If the end user uses the Remember Me feature, a persistent cookie is used.

For more information on how Microsoft Office products work with session related information, see the Microsoft Office support site at <http://support.microsoft.com/kb/218153>.

- ☐ **For workbooks opened in the browser (only applies to Windows XP):** Drill downs will work with redirection enabled because the browser session has access to the HTTP header and/or cookies that need to be sent with the HTTP request to the WebFOCUS Client in order to obtain the redirected target workbook file.

**Note:** For Windows 7, Excel applications no longer display in a browser window.

For additional information about redirection options, see WebFOCUS Administration Console Client Settings described in the *WebFOCUS Security and Administration* manual.

## Excel Page Settings

Excel page settings for the XLSX workbook default to the WebFOCUS standards:

- ☐ Orientation: Portrait
- ☐ Page Size: Letter
- ☐ .75 inches (Excel default)
- ☐ .75 inches (Excel default)

To customize these page settings, turn the XLSXPAGESETS attribute ON and define individual attributes.

If XLSXPAGESETS is turned on, but the page margin attributes are not defined within the procedure, the values will be set to the WebFOCUS default of .25 inches.

## Syntax: How to Define Excel Page Settings

```
[TYPE=REPORT, ] XLSXPAGESETS={ON|OFF} [ ,PAGESIZE={pagesize|LETTER} ]
[ ,ORIENTATION={PORTRAIT|LANDSCAPE} ] [ ,TOPMARGIN=n] [ ,BOTTOMMARGIN=m] , $
```

where:

`XLSXPAGESETS={ON|OFF}`

ON causes the page settings defined in the WebFOCUS request to be applied to the Excel worksheet page settings. OFF retains the default page settings defined in the standard Excel workbook. OFF is the default value.

*n*

Defines the top margin for the worksheet in the units identified by the UNITS parameter (inches, by default). The default value is .25.

*m*

Defines the bottom margin for the worksheet in the units identified by the UNITS parameter (inches, by default). The default value is .25.

*pagesize*

Is one of the PAGESIZE values supported in a WebFOCUS StyleSheet. LETTER is the default page size.

PORTRAIT | LANDSCAPE

PORTRAIT displays the report across the narrower dimension of a vertical page, producing a page that is longer than it is wide. PORTRAIT is the default value.

LANDSCAPE displays the report across the wider dimension of a horizontal page, producing a page that is wider than it is long.

### Adding an Image to a Report

WebFOCUS supports the placement of images within each area or node of the report on the worksheet. An image, such as a logo, gives corporate identity to a report, or provides visual appeal. Data specific images can be placed in headers, footers, and data columns to provide additional clarity and style.

The image must reside on the WebFOCUS Reporting Server in a directory named on EDAPATH or APPPATH. If the file is not on the search path, supply the full path name.

### Inserting Images Into Excel XLSX Reports

Images can be placed in any available WebFOCUS reporting node or element of a worksheet. Supported image formats include .gif and .jpg.

#### Usage Considerations

- ☐ All images will be placed in the top-left corner of the first cell of the defined area, based on the top and left gap. Defined explicit positioning and justification have not been implemented yet.
- ☐ Standard page setting keywords can be used in conjunction with XLSXPAGESETS to control the page layout in standard reports (not compound).
- ☐ Images placed within a report cell in a row or column is anchored to the top-left corner of the cell. The cell is automatically sized to the height and width to fit the largest image (SQUEEZE=ON).

- ❑ Additional lines may need to be added within a heading, footing, subhead, or subfoot to accommodate the placement of the image.

### **Syntax:** How to Insert Images Into WebFOCUS Report Elements in XLSX Reports

```
TYPE={REPORT|heading|data}, IMAGE={url|file|(column)} [,BY=byfield]
[,SIZE=(w h)] , $
```

where:

#### REPORT

Embeds an image in the body of a report. The image appears in the background of the report. REPORT is the default value.

#### *heading*

Embeds an image in a heading or footing. Valid values are TABHEADING, TABFOOTING, FOOTING, HEADING, SUBHEAD, and SUBFOOT. Provide sufficient blank space in the heading or footing so that the image does not overlap the heading or footing text. You may also want to place heading or footing text to the right of the image using spot markers.

#### *data*

Defines a cell within a data column to place the image. Must be used with COLUMNS= attributes to identify the specific report column where the image should be anchored.

#### *url*

Is the URL of the image file.

#### *file*

Is the name of the image file. It must reside on the WebFOCUS Reporting Server in a directory named on EDAPATH or APPPATH. If the file is not on the search path, supply the full path name. When specifying a GIF file, you can omit the file extension.

#### *column*

Is an alphanumeric field in the data source that contains the name of an image file. Enclose the column in parentheses ( ). The field containing the file name or image must be a display field or BY field referenced in the request. Note that the value of the field is interpreted exactly as if it were typed as the URL of the image in the StyleSheet. If you omit the suffix, .GIF is supplied, by default. You can use the SET BASEURL command for supplying the base URL of the images. This way, the value of the field does not have to include the complete URL. This syntax is useful, for example, if you want to embed an image in a SUBHEAD, and you want a different image for each value of the BY field on which the SUBHEAD occurs.

*byfield*

Is the sort field that generates the subhead or subfoot.

*SIZE*

Is the size of the image. By default, an image is added at its original size.

*w*

Is the width of the image, expressed in the unit of measurement specified by the UNITS parameter. Enclose the *w* and *h* values in parentheses. Do not include a comma between them.

*h*

Is the height of the image, expressed in the unit of measurement specified by the UNITS parameter.

**Example:** Adding a GIF Image to a Single Table Request

```
DEFINE FILE GGSALES
SHOWCAT/A100=CATEGORY || '.GIF';
END
TABLE FILE GGSALES
SUM DOLLARS/D12CM UNITS/D12C
BY LOWEST CATEGORY NOPRINT
BY SHOWCAT NOPRINT
BY PRODUCT
ACROSS REGION
WHERE CATEGORY NE 'Gifts'

ON CATEGORY SUBHEAD
" "
"Image in SUBHEAD for Category <CATEGORY "
" "
ON TABLE SUBHEAD
" "
" "
" Report Heading "
" "
ON CATEGORY SUBFOOT
"ON CATEGORY SUBFOOT"
```







```

ON TABLE SUBFOOT
"Report Footing"
" "
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE NOTOTAL
ON TABLE SET ACROSSTITLE SIDE
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, FONT='ARIAL', SIZE=9, TITLETEXT='Food and Coffee',$
TYPE=REPORT, COLUMN=PRODUCT, SQUEEZE=1,$
TYPE=TITLE, BACKCOLOR=RGB(90 90 90), COLOR=RGB(255 255 255), STYLE=BOLD,$
TYPE=ACROSSTITLE, STYLE=BOLD, BACKCOLOR=RGB(90 90 90),
COLOR=RGB(255 255 255),$
TYPE=ACROSSVALUE, BACKCOLOR=RGB(218 225 232), STYLE=BOLD, JUSTIFY=CENTER,$
TYPE=HEADING, STYLE=BOLD, COLOR=RGB(0 35 95), SIZE=12, JUSTIFY=Center,$
TYPE=FOOTING, BACKCOLOR=RGB(90 90 90), SIZE=12, COLOR=RGB(255 255 255),
STYLE=BOLD, JUSTIFY=CENTER,$
TYPE=SUBHEAD, SIZE=12, STYLE=BOLD, BACKCOLOR=RGB(218 225 232),
JUSTIFY=CENTER,$
TYPE=SUBHEAD, IMAGE=(SHOWCAT), SIZE=(.6 .6),$
TYPE=SUBFOOT, SIZE=10, STYLE=BOLD, JUSTIFY=CENTER,$
TYPE=TABHEADING, SIZE=12, STYLE=BOLD, JUSTIFY=CENTER,$
TYPE=TABHEADING, IMAGE=gglogo.gif, $
TYPE=TABFOOTING, SIZE=12, STYLE=BOLD, JUSTIFY=RIGHT,$
TYPE=TABFOOTING, IMAGE=logo.gif, SIZE=(1.67 .6),$
END

```

In the following request, since the referenced images are not part of the existing GGSales table, the image files (.gif) are being built in the DEFINE and then referenced in the TABLE request. You can NOPRINT fields if you do not want them to display as columns, but the fields must be referenced in the table to include them in the internal matrix. This will allow the images to be placed in the headings, footings, or data cells. The specific location is defined using StyleSheet definitions for attaching the image based on field value.

	A	B	C	D	E	F	G	H	I
1	 <b>Report Heading</b>								
2									
3									
4									
5	Region	Midwest		Northeast		Southeast		West	
6	Product	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales
7	 <b>Image in SUBHEAD for Category Coffee</b>								
8									
9									
10	Capuccino	.	.	\$542,095	44,785	\$944,000	73,264	\$895,495	71,168
11	Espresso	\$1,294,947	101,154	\$850,107	68,127	\$853,572	68,030	\$907,617	71,675
12	Latte	\$2,883,566	231,623	\$2,771,815	222,866	\$2,617,836	209,654	\$2,670,405	213,920
13	ON CATEGORY SUBFOOT								
14	 <b>Image in SUBHEAD for Category Food</b>								
15									
16									
17	Biscotti	\$1,091,727	86,105	\$1,802,005	145,242	\$1,505,717	119,594	\$863,868	70,436
18	Croissant	\$1,751,124	139,182	\$1,670,818	137,394	\$1,902,359	156,456	\$2,425,601	197,022
19	Scone	\$1,495,420	116,127	\$907,171	70,732	\$900,655	73,779	\$912,868	72,776
20	ON CATEGORY SUBFOOT								
21									<b>Report Footing</b>
22									
23									
24									

### Example: Adding a GIF Image to a Compound Request

**Note:** Compound Layout syntax cannot contain hidden carriage return or line feed characters. For purposes of presenting this example, line feed characters have been added so that the sample code wraps to fit within the printed page. To run this example in your environment, copy the code into a text editor and delete any line feed characters within the Compound Layout syntax by going to the end of each line and pressing *Delete*. In some instances, you may need to add a space to maintain the structure of the string.

```

APP PATH IBISAMP
SET HTMLARCHIVE=ON
*-HOLD_SOURCE
COMPOUND LAYOUT PCHOLD FORMAT XLSX
UNITS=IN,$
SECTION=section1, LAYOUT=ON, METADATA='prop_with_names, Margins_Left=0.5,
Margins_Top=0.5, Margins_Right=0.5, Margins_Bottom=0.5,
thumbnailscale=4', MERGE=OFF, ORIENTATION=LANDSCAPE, PAGESIZE=Legal,
SHOW_GLOBALFILTER=OFF,$
PAGELAYOUT=1, NAME='Page layout 1', text='Page layout 1', TOC-LEVEL=1,
BOTTOMMARGIN=0.5, TOPMARGIN=0.5, METADATA='BOTTOMMARGIN=0.5, TOPMARGIN=0.5,
LEFTMARGIN=0,RIGHTMARGIN=0,',,$
COMPONENT='report1', TEXT='report1', TOC-LEVEL=2, POSITION=(0.650 0.917),
DIMENSION=(7.250 3.000), BYTOC=0, ARREPORTSIZE=DIMENSION,
METADATA='left: 0.65in; top: 0.917in; width: 7.25in; height: 3in;
position: absolute; z-index: 1;',,$
COMPONENT='chart1', TEXT='chart1', TOC-LEVEL=2, POSITION=(0.735 4.332),
DIMENSION=(7.167 2.917), COMPONENT-TYPE=GRAPH, ARREPORTSIZE=DIMENSION,
METADATA='left: 0.735in; top: 4.332in; width: 7.167in; height: 2.917in;
position: absolute; z-index: 2;',,$
END

SET COMPONENT='report1'
-*component_type report
DEFINE FILE GGSales
SHOWCAT/A100=CATEGORY || '.GIF';
SHOWDATEQ/Q=DATE;
SHOWDATEY/YY=DATE;
SHOWDATEQY/YYQ=DATE;
END

```

```

TABLE FILE GGSALES
SUM DOLLARS/D12CM AS 'Dollars'
BY REGION AS ''
BY LOWEST CATEGORY
BY SHOWCAT AS 'Data Image'
ACROSS SHOWDATEY AS ''
ACROSS SHOWDATEQ AS ''
WHERE REGION NE 'Midwest' OR 'West'
ON TABLE SET HIDENULLACRS ON
HEADING
" "
"Image in Page Heading "
ON REGION SUBHEAD
" <+0> SUBHEAD: <REGION"
FOOTING
" "
"Image in Page Footing"
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=10, BACKCOLOR=NONE, GRID=OFF, FONT='ARIAL',
XLSXPAGESETS=ON, TOPMARGIN=1, BOTTOMMARGIN=1, ORIENTATION=LANDSCAPE,
PAGESIZE=LEGAL, TITLETEXT='With Images',$
TYPE=REPORT, COLUMN=REGION, SQUEEZE=1.5, JUSTIFY=CENTER,$
TYPE=DATA, BACKCOLOR=NONE,$
TYPE=DATA, COLUMN=SHOWCAT, IMAGE=(SHOWCAT), SIZE=(.5 .5),$
TYPE=TITLE, BACKCOLOR=RGB(218 225 232), BORDER=LIGHT,
STYLE=-UNDERLINE+BOLD,$
TYPE=HEADING, IMAGE=GGLOGO.GIF, SIZE=(.65 .65),$
TYPE=HEADING, SIZE=12, STYLE=BOLD, JUSTIFY=CENTER,$
TYPE=SUBHEAD, SIZE=10, STYLE=BOLD, BORDER-TOP=LIGHT,$
TYPE=SUBHEAD, BY=1, JUSTIFY=CENTER, BORDER-TOP=LIGHT,$
TYPE=SUBFOOT, STYLE=BOLD,$
TYPE=FOOTING, SIZE=12, STYLE=+BOLD, JUSTIFY=CENTER,$
TYPE=FOOTING, IMAGE=logo.gif, SIZE=(1.67 .6),$
TYPE=ACROSS, JUSTIFY=CENTER, BORDER=LIGHT,$
TYPE=ACROSSTITLE, STYLE=-UNDERLINE+BOLD,$
TYPE=ACROSSVALUE, BACKCOLOR=RGB(218 225 232), STYLE=-UNDERLINE+BOLD,$
END

```







```

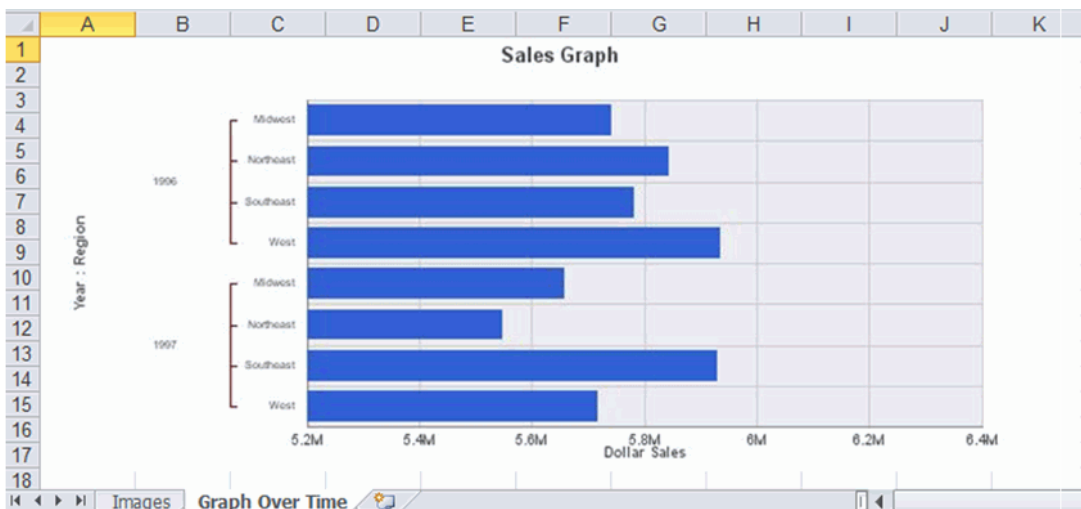
SET COMPONENT='chart1'
ENGINE INT CACHE SET ON
-DEFAULTH &WF_STYLE_UNITS='PIXELS';
-DEFAULTH &WF_STYLE_HEIGHT='1005.0';
-DEFAULTH &WF_STYLE_WIDTH='1070.0';
-DEFAULTH &WF_TITLE='WebFOCUS Report';
GRAPH FILE GGSales
HEADING
"Sales Graph"
SUM
GGSales.SALES01.DOLLARS
BY SHOWDATEY AS Year
BY GGSales.SALES01.REGION
ON GRAPH PCHOLD FORMAT XLSX
ON GRAPH SET VZERO OFF
ON GRAPH SET HTMLENCODE ON
ON GRAPH SET GRAPHDEFAULT OFF
ON GRAPH SET GRWIDTH 1
ON GRAPH SET UNITS &WF_STYLE_UNITS
ON GRAPH SET HAXIS 1000
ON GRAPH SET VAXIS 1000
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRLEGEND 0
ON GRAPH SET GRXAXIS 2
ON GRAPH SET LOOKGRAPH HBAR
ON GRAPH SET STYLE *

*GRAPH_SCRIPT
setPieDepth(0);
setPieTilt(0);
setDepthRadius(0);
setCurveFitEquationDisplay(false);
setPlace(true);
setPieFeelerTextDisplay(1);
setUseSeriesShapes(true);
setMarkerSizeDefault(50);
setScaleMustIncludeZero(getX1Axis(), false);
setScaleMustIncludeZero(getY1Axis(), false);
setScaleMustIncludeZero(getY2Axis(), false);
setMarkerSizeDefault(60);
*END
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/
ENIADefault_combine.sty,$
TYPE=REPORT, TITLETEXT='Graph Over Time',$
*GRAPH_SCRIPT
setReportParsingErrors(false);
setSelectionEnableMove(false);
*END
ENDSTYLE
END
COMPOUND END

```

The output is shown in the following images.

	A	B	C	D	E	F	G	H	I	J	K
1											
2	<b>Image in Page Heading</b>										
3											
4											
5											
6											
7	<b>SUBHEAD: Northeast</b>										
8	Northeast	Coffee		\$560,057	\$489,050	\$528,299	\$559,488	\$503,629	\$565,503	\$501,884	\$456,107
9		Food		\$529,237	\$535,075	\$635,881	\$546,154	\$492,300	\$523,784	\$566,931	\$550,632
10		Gifts		\$369,301	\$327,843	\$385,484	\$378,030	\$317,487	\$345,227	\$367,737	\$357,180
11	<b>SUBHEAD: Southeast</b>										
12	Southeast	Coffee		\$517,360	\$528,692	\$577,773	\$478,972	\$620,073	\$515,993	\$598,251	\$578,294
13		Food		\$539,876	\$540,733	\$523,482	\$596,651	\$556,701	\$528,073	\$499,708	\$523,507
14		Gifts		\$372,652	\$375,837	\$364,880	\$364,858	\$374,260	\$329,274	\$378,170	\$426,309
15											
16											
17	<b>Image in Page Footing</b>										
18											
19											




**Example: Adding a GIF Image as a BYTOC Compound Request**

The following syntax is a portion of the code from the previous example to show the COMPOUND BYTOC syntax. By adding the ON TABLE SET COMPOUND BYTOC command to the compound report above, you can turn the report into a Compound Table of Contents report. The BYTOC syntax can be added to a stand-alone request or to a component of a compound document.

```
TABLE FILE GGSALES
SUM DOLLARS/D12CM AS 'Dollars'
SHOWREG
NOPRINT
BY REGION AS ''
BY LOWEST CATEGORY
BY SHOWCAT AS 'Data Image'
WHERE REGION NE 'Midwest' OR 'West'
ACROSS SHOWDATEY AS '' ACROSS SHOWDATEQ AS ''
ON TABLE SET HIDENULLACRS ON HEADING
"Image in Page Heading" ON REGION SUBHEAD
"<+> Image in SUBHEAD:<REGION" FOOTING
" "
"Image in Page Footing"
ON TABLE SET PAGE-NUM NOLEAD ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT XLSX ON TABLE SET COMPOUND BYTOC
ON TABLE SET STYLE *
```

The output is:

	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3											
4											
5											
6		Category	Data Image								
7											
8		Northeast	Coffee								
9											
10											
11											
12											
13											
14											
15											

## Inserting Images Into XLSX Workbook Headers and Footers

WebFOCUS supports the insertion of images into Excel headers and footers and the definition of key page settings to support the placement of these images in relationship to the overall worksheet and the Excel generated page breaks. This new access to the Excel page functionality is designed to enhance overall usability of the worksheets for users who will be printing these reports. Page settings including orientation, page size, and page margins will directly affect the layout of each Excel page based on values defined within the FOCEXEC. Images can be included on headers and footers on every printed page, on the first page of the report only, or only on all subsequent pages. The WebFOCUS headings and footings continue to display within the worksheet. With this new feature, WebFOCUS can insert logos to be printed once at the top of a report and watermark images that need to be displayed on every printed page.

### **Syntax:** How to Insert Images Into Excel Headers and Footers

```
TYPE={PAGEHEADER|PAGEFOOTER},OBJECT=IMAGE,
IMAGE=imagename, JUSTIFY={LEFT|CENTER|RIGHT}
[,DISPLAYON={FIRST|NOT-FIRST}] [,SIZE=(w h)], $
```

where:

#### PAGEHEADER

Places the image in the worksheet header.

#### PAGEFOOTER

Places the image in the worksheet footer.

#### *imagename*

Is the name of a valid image file to be placed in the header or footer. The image must be located in the defined application path on the Reporting Server. The image types supported are GIF and JPEG.

#### JUSTIFY={LEFT|CENTER|RIGHT}

Identifies the area in the header or footer to contain the image and the justification or placement within that defined area.

#### DISPLAYON

Defines whether the image should be placed on the first page only or on all pages except the first. Omit this attribute to place the image on all pages.

Valid values are:

**FIRST** places the image only on the first page.



`NOT-FIRST` places the image on every page, except the first page.

`SIZE=( w h )`

Is the size of the image. By default, an image is added at its original size.

`w` is the width of the image, expressed in the unit of measurement specified by the `UNITS` parameter.

`h` is the height of the image, expressed in the unit of measurement specified by the `UNITS` parameter.

### **Example:** Inserting Images in Excel Headers and Footers and Defining Page Settings

The following request against the GGSALES data source places the image `ibi_logo.gif` on the left header area of the first page and the right header area of every subsequent page of the resulting worksheet. It places the image `webfocus1.gif` in the center area of the footer on every page.

```
TABLE FILE GGSALES
SUM DOLLARS UNITS BUDDOLLARS BUDUNITS
BY REGION
BY ST
BY CATEGORY
BY PRODUCT
ON TABLE SET BYDISPLAY ON
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET STYLE *
FONT=ARIAL,SIZE=12,
XLSXPAGESETS=ON,TOPMARGIN=1,BOTTOMMARGIN=1,ORIENTATION=LANDSCAPE,
PAGESIZE=LETTER,$
TYPE=TITLE, COLOR=WHITE, BACKCOLOR=GREY,$
TYPE=PAGEHEADER, OBJECT=IMAGE, JUSTIFY=LEFT, IMAGE=IBI_LOGO.GIF,
DISPLAYON=FIRST,$
TYPE=PAGEHEADER, OBJECT=IMAGE, JUSTIFY=RIGHT, IMAGE=IBI_LOGO.GIF,
DISPLAYON=NOT-FIRST,$
TYPE=PAGEFOOTER, OBJECT=IMAGE, JUSTIFY=CENTER, IMAGE=WEBFOCUS1.GIF,$
END
```

The first page of output has the image `ibilogo.gif` in the left area of the header and the image `webfocus1.gif` in the center area of the footer.



Region	State	Category	Product	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
Mid west	IL	Coffee	Espresso	420439	32237	401477	32416
Mid west	IL	Coffee	Latte	978340	77344	964787	79015
Mid west	IL	Food	Biscotti	378412	29413	385369	30001
Mid west	IL	Food	Croissant	549366	43300	528255	43271
Mid west	IL	Food	Scone	595069	45355	567231	45091
Mid west	IL	Gifts	Coffee Grinder	233292	19339	241711	19224
Mid west	IL	Gifts	Coffee Pot	204828	15785	208255	16035
Mid west	IL	Gifts	Mug	376754	30157	388612	30881
Mid west	IL	Gifts	Thermos	187901	14651	181159	14137
Mid west	MO	Coffee	Espresso	419143	32596	416875	32787
Mid west	MO	Coffee	Latte	966981	77347	921336	77141
Mid west	MO	Food	Biscotti	368077	29188	360403	28764
Mid west	MO	Food	Croissant	613871	48941	592609	49327
Mid west	MO	Food	Scone	481953	37602	478691	36573
Mid west	MO	Gifts	Coffee Grinder	181570	14614	171501	14779
Mid west	MO	Gifts	Coffee Pot	190153	14807	191451	14970
Mid west	MO	Gifts	Mug	343852	27040	324488	26837
Mid west	MO	Gifts	Thermos	195686	15592	189484	15903
Mid west	TX	Coffee	Espresso	455365	36321	439880	36666
Mid west	TX	Coffee	Latte	938245	76932	941677	77501
Mid west	TX	Food	Biscotti	345238	27504	321857	27074
Mid west	TX	Food	Croissant	587887	46941	587869	47050
Mid west	TX	Food	Scone	418398	33170	398437	32112
Mid west	TX	Gifts	Coffee Grinder	204292	16440	200241	16625
Mid west	TX	Gifts	Coffee Pot	204897	16564	214301	16774
Mid west	TX	Gifts	Mug	366337	29521	383050	29374
Mid west	TX	Gifts	Thermos	194319	16344	193367	16779
Northeast	CT	Coffee	Capuccino	158995	12386	141574	11098
Northeast	CT	Coffee	Espresso	279373	22482	299854	23676
Northeast	CT	Coffee	Latte	926052	74623	953855	74427
Northeast	CT	Food	Biscotti	589355	46214	587501	46404

powered by




The second page of output has the image `ibilogo.gif` in the right area of the header and the image `webfocus1.gif` in the center area of the footer.

Information Builders

The Standard for Enterprise Business Intelligence

Northeast	CT	Food	Croissant	551489	45847	580168	46335
Northeast	CT	Food	Scone	283874	22378	269221	21038
Northeast	CT	Gifts	Coffee Grinder	169908	13691	159620	13117
Northeast	CT	Gifts	Coffee Pot	208209	15523	197051	15190
Northeast	CT	Gifts	Mug	392967	31728	424333	32415
Northeast	CT	Gifts	Thermos	221827	17568	219025	17667
Northeast	MA	Coffee	Capuccino	174344	15358	192747	15672
Northeast	MA	Coffee	Espresso	248356	19698	254310	19888
Northeast	MA	Coffee	Latte	917737	74572	941438	73874
Northeast	MA	Food	Biscotti	570391	47064	616766	48246
Northeast	MA	Food	Croissant	497234	41029	519322	41351
Northeast	MA	Food	Scone	332486	25363	312004	23774
Northeast	MA	Gifts	Coffee Grinder	177940	14382	187686	15384
Northeast	MA	Gifts	Coffee Pot	184119	15349	184071	15171
Northeast	MA	Gifts	Mug	401944	32360	401617	31324
Northeast	MA	Gifts	Thermos	203435	16734	208436	16921
Northeast	NY	Coffee	Capuccino	208756	17041	227170	17662
Northeast	NY	Coffee	Espresso	322378	25947	318738	26212
Northeast	NY	Coffee	Latte	928026	73671	922776	73411
Northeast	NY	Food	Biscotti	642259	51964	644415	50502
Northeast	NY	Food	Croissant	622095	50518	640032	50178
Northeast	NY	Food	Scone	290811	22991	284478	23603
Northeast	NY	Gifts	Coffee Grinder	161352	12904	164336	12796
Northeast	NY	Gifts	Coffee Pot	198452	15313	192227	15043
Northeast	NY	Gifts	Mug	349300	27409	344364	26801
Northeast	NY	Gifts	Thermos	178836	14568	187786	15179
Southeast	FL	Coffee	Capuccino	317027	24143	285194	23092
Southeast	FL	Coffee	Espresso	256539	19730	236531	18690
Southeast	FL	Coffee	Latte	889887	71123	886465	72975
Southeast	FL	Food	Biscotti	511597	40606	516984	41242
Southeast	FL	Food	Croissant	602076	50175	644884	51437
Southeast	FL	Food	Scone	311836	24543	299547	24576

powered by



### **Reference:** Usage Notes for Inserting Images Into XLSX Worksheet Headers and Footers

- ☐ The Excel headers and footers are not automatically sized based on contents of the areas. Define page margins within the page settings (XLSPAGESETS) to account for the space required to display the images within each page of the report.
- ☐ The image sizing based on the specified height and width is not proportional. Sizing may cause image distortion.
- ☐ BLOB image fields are not supported in this release.

**Reference: Displaying Watermarks on XLSX Report Output**

Watermark images can be placed into the Excel headers to display on every printed page of the generated worksheet.

Excel places images on the page starting in the header from left to right and then the footer from left to right. Large images placed in the header may overlap images before them in the presentation order. For page layouts with a logo in the left area and watermark centered on the page, watermark image background must be transparent so it does not overlay the logo image.

In Excel, images are placed first on the page. All other contents of the worksheet are then placed on top of the images. Text in cells and styling, such as background color and drawing objects, are placed on top of the images. Excel supports transparency in drawing objects and images, but not in cell background color. BACKCOLOR will cover over images placed on the page.

**Example: Placing a Watermark in an XLSX Header**

The following request against the GGSALES data source uses the image internaluseonly.gif as a watermark to display in the background of every page of the worksheet. Although the image is placed in the center area of the header, it is large enough to span the entire worksheet page. It has a transparent background, so it does not cover the logo images placed at the left in the header and the center in the footer.

```
TABLE FILE GGSALES
SUM DOLLARS UNITS BUDDOLLARS BUDUNITS
BY REGION
BY ST
BY CATEGORY
BY PRODUCT
ON TABLE SET BYDISPLAY ON
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET STYLE *
XLSXPAGESETS=ON,
TOPMARGIN=1,BOTTOMMARGIN=1,LEFTMARGIN=1, RIGHTMARGIN=1,
ORIENTATION=LANDSCAPE,PAGESIZE=LETTER,$
TYPE=PAGEHEADER, OBJECT=IMAGE, JUSTIFY=LEFT, IMAGE=IBI_LOGO.GIF,
DISPLAYON=FIRST,$
TYPE=PAGEHEADER, OBJECT=IMAGE, JUSTIFY=CENTER, IMAGE=WFINTERNALUSEONLY.GIF,$
TYPE=PAGEFOOTER, OBJECT=IMAGE, JUSTIFY=RIGHT, IMAGE=WEBFOCUS1.GIF,$
END
```

The first page of the generated worksheet shows the watermark image beneath the data. This image is displayed on every page of the worksheet.

Information Builders		The Standard for Enterprise Business Intelligence					
Region	State	Category	Product	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
Midwest	IL	Coffee	Espresso	420439	32237	401477	32416
Midwest	IL	Coffee	Latte	978340	77344	964787	79015
Midwest	IL	Food	Biscotti	378412	29413	385369	30001
Midwest	IL	Food	Croissant	549366	43300	528255	43271
Midwest	IL	Food	Scone	595069	45355	567231	45091
Midwest	IL	Gifts	Coffee Grinder	233292	19339	241711	19224
Midwest	IL	Gifts	Coffee Pot	204828	15785	208255	16035
Midwest	IL	Gifts	Mug	376754	30157	368655	30881
Midwest	IL	Gifts	Thermos	187901	14651	18115	14137
Midwest	MO	Coffee	Espresso	419143	32596	416875	32787
Midwest	MO	Coffee	Latte	966981	77347	921336	77141
Midwest	MO	Food	Biscotti	368077	29185	360403	28764
Midwest	MO	Food	Croissant	613871	48941	592609	49327
Midwest	MO	Food	Scone	481953	37652	478691	36573
Midwest	MO	Gifts	Coffee Grinder	181575	14614	171501	14779
Midwest	MO	Gifts	Coffee Pot	191155	14807	191451	14970
Midwest	MO	Gifts	Mug	133852	27040	324488	26837
Midwest	MO	Gifts	Thermos	156686	15592	189484	15903
Midwest	TX	Coffee	Espresso	455365	36321	439880	36666
Midwest	TX	Coffee	Latte	938245	76932	941677	77501
Midwest	TX	Food	Biscotti	345238	27504	321857	27074
Midwest	TX	Food	Croissant	587887	46941	587869	47050
Midwest	TX	Food	Scone	418398	33170	398437	32112
Midwest	TX	Gifts	Coffee Grinder	204292	16440	200241	16625
Midwest	TX	Gifts	Coffee Pot	204897	16564	214301	16774
Midwest	TX	Gifts	Mug	366337	29521	383050	29374
Midwest	TX	Gifts	Thermos	194319	16344	193367	16779
Northeast	CT	Coffee	Capuccino	158995	12386	141574	11098
Northeast	CT	Coffee	Espresso	279373	22482	299854	23676
Northeast	CT	Coffee	Latte	926052	74623	953855	74427
Northeast	CT	Food	Biscotti	589355	46214	587501	46404
Northeast	CT	Food	Croissant	551489	45847	580168	46335
Northeast	CT	Food	Scone	283874	22378	269221	21038
Northeast	CT	Gifts	Coffee Grinder	169908	13691	159620	13117
Northeast	CT	Gifts	Coffee Pot	208209	15523	197051	15190
Northeast	CT	Gifts	Mug	392967	31728	424333	32415

powered by  
**WebFOCUS**

## Creating Excel XLSX Worksheets Using Templates

XLSX report output can be generated based on Excel templates. This feature allows for the integration of WebFOCUS reports into workbooks containing multiple worksheets. Any native Excel template can be used to generate a new workbook containing a WebFOCUS report.

The following Excel file types can be used as template files to generate XLSX workbooks.

Template File Type	Output Workbook Generated
Template (.xltx)	Workbook (.xlsx)
Macro-Enabled Template (.xlsm)	Macro-Enabled workbook (.xlsm)

Template File Type	Output Workbook Generated
Workbook (.xlsx)	Workbook (.xlsx)
Macro-Enabled workbook (.xlsm)	Macro-Enabled workbook (.xlsm)

WebFOCUS XLSX TEMPLATE format provides support for basic Excel templates (.xltx) files. These templates cannot contain macros or other content that Microsoft considers active, as well as templates with active content (XLTM/XLSM). Additionally, macro-enabled templates (.xlsm) allow for the inclusion of active content (macros and VB script) into templates.

A WebFOCUS EXL07 template procedure generates a native Excel workbook with the standard Excel extension, based on the defined template file. The WebFOCUS request will replace an existing worksheet within the template workbook, and any formulas or references defined in other worksheets to cells within the replaced worksheet will automatically update when the workbook is opened.

Since the template feature replaces existing worksheets, the designated worksheet must exist in the template workbook. Any content on the replaced worksheet within the template will not be retained. Content contained on any other worksheets will be retained and updated.

Named ranges can be defined within the procedure using the INRANGES attribute to designate cell groupings that can be referenced by other worksheets.

An Excel 2007/2010 template can be generated by saving any workbook with the .xltx extension. The template file should be stored within your application path (EDAPATH or APP PATH) rather than the default Excel template directory so that it can be accessed by the Reporting Server when the procedure is executed.

The EXL07 TEMPLATE feature is supported for basic EXL07 format reports. The following features are not supported with EXL07 TEMPLATE in this release: FORMULA, EXL97, EXCEL, and compound Excel reports.

In most cases, existing Excel 2003/2000 templates created as .mht files can easily be converted to Excel 2013/2010/2007 templates by opening the .mht file in Excel 2013/2010/2007 and resaving the file as either an Excel template (.xltx) or a macro enabled (.xlsm) file. Native Excel formulas and functionality should be retained within these templates. Use .xlsm to retain active content, including macros. This new XLTX template can be used with XLSX procedures.

**Syntax:**      **How to Create an XLSX Report Using Any Supported Template File Type**

To support the expanded template files types, the template file name attribute has been enhanced to allow for the inclusion of the file extension. If no extension is specified within the template name, the file extension will default to .XLTX.

```
ON TABLE PCHOLD FORMAT XLSX TEMPLATE template_name SHEETNUMBER n
```

where:

*template\_name*

Is the name of the Excel template file (workbook), up to 64 characters including the file name and extension, residing on the WebFOCUS Reporting Server application directory search path. For example, IPOLICY.XLTX, PRINTSHEETS.XLTM, or DASHBOARD.XLSM. If the extension is not provided, it defaults to .XLTX.

*n*

Is the number of the existing Excel worksheet being replaced in the template file (workbook).

**Reference:**      **Usage Notes for XLSX Templates**

The workbook template used by the WebFOCUS procedure must contain valid worksheets.

- ❑ The sheet that is updated must exist in the workbook, as WebFOCUS is replacing the worksheet rather than inserting a worksheet. If the sheet designated does not exist, the procedure will return an error.
- ❑ In any template file, at least one of the sheets in the workbook must contain a cell with a valid value (blank or any other value). To replace a worksheet in a template that contains only empty worksheets, replace one of the cells in any of the sheets with a space and save. This will instantiate the worksheets so they are accessible to WebFOCUS for updating.
- ❑ The supported file name length has been extended to 64 characters. Any procedure referencing a template with a longer file name produces a message.

**Creating Excel Table of Contents Reports**

Excel Table of Contents (BYTOC) enables you to generate a separate worksheet within an instance of the report for each value of the first BY field in the WebFOCUS report.

**Syntax:**      **How to Use the Excel Table of Contents Feature**

There are three different ways that BYTOC can be invoked:

```
ON TABLE {HOLD|PCHOLD} FORMAT XLSX BYTOC
```

```
SET COMPOUND=BYTOC
```

```
ON TABLE SET COMPOUND BYTOC
```

Since a BYTOC report generates separate worksheets according to the value of the first BY field in the report, the report must contain at least one BY field. The primary BY field may be a NOPRINT field.

The BYTOC feature is not supported with the XLSX TEMPLATE format.

**Example:**      **Creating a Simple BYTOC Report**

The following request against the GGSALES data source creates separate tabs based on the REGION sort field.

```
TABLE FILE GGSALES
SUM UNITS/D12C DOLLARS/D12CM
BY REGION NOPRINT
BY CATEGORY
BY PRODUCT
HEADING
"<REGION Region Sales"
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET BYDISPLAY ON
ON TABLE SET COMPOUND BYTOC
ON TABLE SET STYLE *
TYPE=REPORT, FONT=ARIAL, SIZE=9,$
TYPE=HEADING, SIZE=12,$
TYPE=TITLE, BACKCOLOR=GREY, COLOR=WHITE,$
ENDSTYLE
END
```



The output is:

	A	B	C	D
1	Midwest Region Sales			
2	Category	Product	Unit Sales	Dollar Sales
3	Coffee	Espresso	101,154	\$1,294,947
4	Coffee	Latte	231,623	\$2,883,566
5	Food	Biscotti	86,105	\$1,091,727
6	Food	Croissant	139,182	\$1,751,124
7	Food	Scone	116,127	\$1,495,420
8	Gifts	Coffee Grinder	50,393	\$619,154
9	Gifts	Coffee Pot	47,156	\$599,878
10	Gifts	Mug	86,718	\$1,086,943
11	Gifts	Thermos	46,587	\$577,906
12				

Midwest Northeast Southeast West

### Reference: How to Name Worksheets

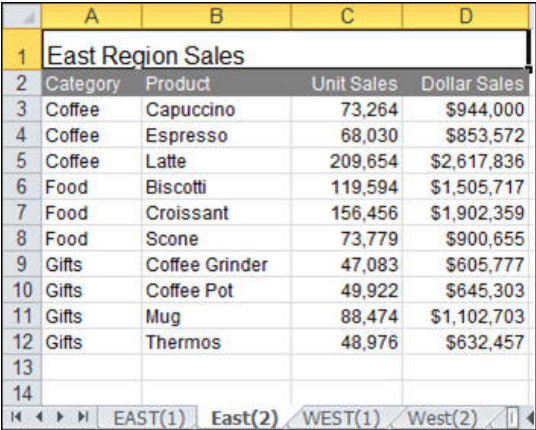
- ❑ The worksheet tab names are the BY field values that correspond to the data on the current worksheet. If the user specifies the TITLETEXT keyword in the StyleSheet, it will be ignored.
- ❑ Excel limits the length of worksheet titles to 31 characters. The following special characters cannot be used: ':', '?', '\*', and '/'.
- ❑ If you want to use date fields as the bursting BY field, you can include the - character instead of the / character. The - character is valid in an Excel tab title. However, if you do use the / character, WebFOCUS will substitute it with the - character.

### Naming XLSX Worksheets With Case Sensitive Data

Excel requires each sheet name to be unique. Excel is case insensitive meaning it evaluates two values as being the same when the values contain the same characters but have different casing. For example, Excel evaluates the values WEST and West to be the same value. WebFOCUS XLSX format identifies duplicate names and adds a unique number to the name to allow Excel to maintain both sheets.

By default, WebFOCUS sort processing is case-sensitive, so the same field value with different casing is considered to be two different values when used as a sort (BY) field. In an Excel BYTOC report, WebFOCUS will generate sheets with sheet names for each value of the primary sort (BY) key based on case sensitivity. To account for this, XLSX has been enhanced to add counters where duplicate tab names are found in the data to ensure the names are unique.

For example, if the report had EAST and East as the values for the Region, each worksheet would be displayed as EAST(1) and East(2), as shown in the following image.



	A	B	C	D
1	East Region Sales			
2	Category	Product	Unit Sales	Dollar Sales
3	Coffee	Capuccino	73,264	\$944,000
4	Coffee	Espresso	68,030	\$853,572
5	Coffee	Latte	209,654	\$2,617,836
6	Food	Biscotti	119,594	\$1,505,717
7	Food	Croissant	156,456	\$1,902,359
8	Food	Scone	73,779	\$900,655
9	Gifts	Coffee Grinder	47,083	\$605,777
10	Gifts	Coffee Pot	49,922	\$645,303
11	Gifts	Mug	88,474	\$1,102,703
12	Gifts	Thermos	48,976	\$632,457
13				
14				

Overcoming the Excel 2007/2010 Row Limit Using Overflow Worksheets

The maximum number of rows supported by Excel 2007/2010 on a worksheet is 1,048,576 (1MB). When you create an XLSX output file from a WebFOCUS report, the number of rows generated can be greater than this maximum.

To avoid creating an incomplete output file, you can have extra rows flow onto a new worksheet, called an *overflow worksheet*. The name of each overflow worksheet will be the name of the original worksheet appended with an increment number.

In addition, when the overflow worksheet feature is enabled, you can set a target value for the maximum number of rows to be included on a worksheet. By default, the row limit will be set to the default value for the LINES parameter (57).

**Note:** By default, when generating XLSX output, the WebFOCUS page heading and page footing commands generate only worksheet headings and worksheet footings.

Syntax: How to Enable Overflow Worksheets

Add the ROWOVERFLOW attribute to your WebFOCUS StyleSheet

```
TYPE=REPORT, ROWOVERFLOW={ON|OFF|PBON}, [ROWLIMIT={n|MAX}], $
```

where:

**ON**

Enables overflow worksheets.

**OFF**

Disables overflow worksheets. OFF is the default value.

**PBON**

Inserts WebFOCUS page breaks that display the page heading, footing, and column titles at the appropriate places within the worksheet rows. This option does not cause a new worksheet to start when a WebFOCUS page break occurs.

**ROWLIMIT=*n***

Sets a target value for the number of rows to be included on a worksheet to *n* rows. The default value is the LINES value (by default, 57).

**ROWLIMIT=MAX**

Sets a target value for the number of rows to be included on a worksheet to 1,048,000 rows for XLSX output.

### **Reference:** Usage Notes for XLSX Overflow Worksheets

- ☐ The report heading is placed once at the start of the first sheet. The report footing is placed once at the bottom of the last overflow sheet.
- ☐ Unless the PBON setting is used, worksheet headings and column titles are repeated at the top of the original sheet and each subsequent overflow sheet. worksheet footings are placed at the bottom of the original sheet and each subsequent overflow sheet. The data values are displayed on the top data row of each overflow sheet as they would be on a standard new page.
- ☐ Report total lines are displayed at the bottom of the last overflow sheet directly above the final page and table footings.
- ☐ Subheadings, subfootings, and subtotal lines display within the data flow as normal. No special consideration is made to retain groupings within a given sheet.
- ☐ If ROWOVERFLOW=PBON, the page headings and footings and column titles display within the worksheet when a WebFOCUS command causes a page break.
- ☐ For XLSX output, if the ROWOVERFLOW attribute is specified in the StyleSheet and ROWLIMIT is greater than 1MB, the following message is presented and no output file is generated:

(FOC3338) The row limit for EXCEL XLSX worksheets is 1048576.

- ❑ Output types that contain formula references (EXL2K PIVOT and EXL2K FORMULA) are not supported, as formula references are not automatically updated to reflect placement on new overflow worksheets.
- ❑ The overflow worksheet feature applies to rows only, not columns. A new worksheet will not automatically be created if a report generates more than the Excel 2007/2010 limit or 16,384 columns.
- ❑ ROWOVERFLOW is supported for BYTOC reports for XLSX.
- ❑ As named ranges in Excel cannot run across multiple worksheets, the IN-RANGES phrase that defines named ranges in the resulting workbook is not supported with the ROWOVERFLOW feature. When they exist together in the same request, ROWOVERFLOW takes precedence and the IN-RANGES phrase is ignored.

### ***Example:*** Creating Overflow Worksheets

The following request creates XLSX report output with overflow worksheets. The ROWOVERFLOW=ON attribute in the StyleSheet activates the overflow feature. Without this attribute, one worksheet would have been generated instead of three.

```
TABLE FILE GGSales
-* ****Report Heading****
ON TABLE SUBHEAD
"SALES BY REGION, CATEGORY, AND PRODUCT"
" "
-* ****Worksheet Heading****
HEADING
"SALES REPORT WORKSHEET <TABPAGENO"
" "
-* ****Worksheet Footing****
FOOTING
" "
"END OF WORKSHEET <TABPAGENO"
PRINT DOLLARS UNITS BUDDOLLARS BUDUNITS
BY REGION
BY CATEGORY
BY PRODUCT
BY DATE
```

```

-* ****Subfoot****
ON REGION SUBFOOT
" "
" End of Region <REGION>"
" "
-* ****Subhead****
ON REGION SUBHESD
" "
"Category <CATEGORY> for Region <REGION>"
" "
-* ****Report Footing****
ON TABLE SUBFOOT
" "
"END OF REPORT"
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET STYLE *
TYPE=REPORT, TITLETEXT=EXLOVER, ROWOVERFLOW=ON, ROWLIMIT=2000,$
ENDSTYLE
END

```

The report heading displays on the first worksheet only, the page heading and column titles display on each worksheet, and the subhead and subfoot display whenever the associated sort field changes value. The following image shows the top of the first worksheet, displaying the report heading, page heading, column titles, and first subhead.

SALES BY REGION, CATEGORY, AND PRODUCT										
1	A	B	C	D	E	F	G	H	I	J
2	SALES BY REGION, CATEGORY, AND PRODUCT									
3	SALES REPORT WORKSHEET 1									
4										
5	Region	Category	Product	Date	Dollar Sales	Unit Sales	Budget Dollars	Budget Units		
6										
7	Category Coffee for Region Midwest									
8										
9	Midwest	Coffee	Espresso	1996/01/01	19752	1646	24141	1857		
10					13416	1118	16068	1236		
11					13170	878	10000	1000		
12					10164	924	12552	1046		
13					6048	432	6565	505		
14					4693	361	8310	554		
15				1996/02/01	16968	1212	13622	973		
16					13420	1220	20415	1361		
17					13095	873	6910	691		
18					8340	834	8460	705		
19					6454	461	8540	610		
20					4355	335	5400	360		
21				1996/03/01	18466	1319	15120	1512		
22					18135	1395	17069	1313		
23					12690	846	7060	706		
24					11450	1145	14490	1035		
25					10802	982	9555	735		
26					10073	771	8180	765		

Note that the TITLETEXT attribute in the StyleSheet specified the name EXLOVER, so the three worksheets were generated with the names EXLOVER1, EXLOVER2, and EXLOVER3. If there had been no TITLETEXT attribute, the sheets would have been named SHEET1, SHEET2, and SHEET3.

The worksheet footing displays at the bottom of each worksheet and the report footing displays at the bottom of the last worksheet. The following image shows the bottom of the last worksheet, displaying the last subfoot, the page footing, and the report footing.

A1 SALES REPORT WORKSHEET 3										
	A	B	C	D	E	F	G	H	I	J
363					9194	657	7722	772		
364					8981	816	7983	665		
365				1997/08/01	14276	1020	17087	1139		
366					8679	868	7366	737		
367					2179	168	1911	127		
368				1997/09/01	4380	365	5869	451		
369					2152	215	3732	287		
370					1459	112	0	0		
371				1997/10/01	9798	754	8070	673		
372					6402	640	9880	659		
373					6302	450	6570	597		
374				1997/11/01	8448	845	7855	714		
375					6154	440	4966	451		
376					4552	414	4844	484		
377				1997/12/01	15092	1161	16549	1182		
378					11197	746	13464	898		
379					10346	690	10536	810		
380										
381	End of Region West									
382										
383										
384	END OF WORKSHEET 3									
385										
386	END OF REPORT									

**Example: Creating Overflow Worksheets With WebFOCUS Page Breaks**

The following request creates XLSX report output with overflow worksheets. The ROWOVERFLOW=PBON attribute in the StyleSheet activates the overflow feature, and the ROWLIMIT=250 sets the maximum number of rows in each worksheet to approximately 250. Without this attribute, one worksheet would have been generated. The PRODUCT sort phrase specifies a page break.

```
TABLE FILE GGSALES
- * ****Report Heading****
ON TABLE SUBHEAD
"SALES BY REGION, CATEGORY, AND PRODUCT"
" "
PRINT DOLLARS UNITS BUDDOLLARS BUDUNITS
BY REGION
BY HIGHEST CATEGORY
BY PRODUCT PAGE-BREAK
BY DATE
WHERE DATE GE '19971001'
- * ****Page Heading****
HEADING
" Product: <PRODUCT in Category: <CATEGORY for Region: <REGION"
- * ****Page Footing****
FOOTING
" "
- * ****Report Footing****
ON TABLE SUBFOOT
" "
"END OF REPORT"
ON TABLE SET BYDISPLAY ON
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/
ENIADefault_combine.sty,$
TITLETEXT=EXLOVER, ROWOVERFLOW=PBON, ROWLIMIT=250,
$
ENDSTYLE
END
```

The report heading displays on the first worksheet only, the page heading, footing, and column titles display on each worksheet and at each WebFOCUS page break (each time the product changes), and the subhead and subfoot display whenever the associated sort field changes value. The following image shows the top of the first worksheet.

SALES BY REGION, CATEGORY, AND PRODUCT							
Product: Coffee Grinder in Category: Gifts for Region: Midwest							
Region	Category	Product	Date	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
Midwest	Gifts	Coffee Grinder	1997/10/01	14494	95.6	1175.4	1069
Midwest	Gifts	Coffee Grinder	1997/10/01	5452	49.6	601.9	502
Midwest	Gifts	Coffee Grinder	1997/10/01	1750	13.5	217.5	148
Midwest	Gifts	Coffee Grinder	1997/11/01	12525	114.0	1009.7	1191
Midwest	Gifts	Coffee Grinder	1997/11/01	12190	110.0	1457.1	1041
Midwest	Gifts	Coffee Grinder	1997/11/01	9552	59.6	1016.1	547
Midwest	Gifts	Coffee Grinder	1997/12/01	47455	360.9	4305.6	3787
Midwest	Gifts	Coffee Grinder	1997/12/01	7356	49.0	650.1	650
Midwest	Gifts	Coffee Grinder	1997/12/01	4073	37.0	513.5	513
Midwest	Gifts	Coffee Grinder	1997/12/01	1967	17.9	354.0	325
Product: Coffee Pot in Category: Gifts for Region: Midwest							
Region	Category	Product	Date	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
Midwest	Gifts	Coffee Pot	1997/10/01	12525	107.0	1533.5	1032
Midwest	Gifts	Coffee Pot	1997/10/01	11609	116.1	1542.5	1216
Midwest	Gifts	Coffee Pot	1997/10/01	1375	11.5	112.4	35
Midwest	Gifts	Coffee Pot	1997/11/01	5345	59.6	1099.1	545
Midwest	Gifts	Coffee Pot	1997/11/01	5151	51.5	910.1	650
Midwest	Gifts	Coffee Pot	1997/11/01	3722	34.5	335.4	199
Midwest	Gifts	Coffee Pot	1997/12/01	10593	70.6	830.6	554
Midwest	Gifts	Coffee Pot	1997/12/01	9337	61.7	765.0	544
Midwest	Gifts	Coffee Pot	1997/12/01	5500	44.6	422.1	352
Product: Mug in Category: Gifts for Region: Midwest							
Region	Category	Product	Date	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
Midwest	Gifts	Mug	1997/10/01	14492	111.5	1566.2	1044
Midwest	Gifts	Mug	1997/10/01	14325	95.6	1264.2	542
Midwest	Gifts	Mug	1997/10/01	11370	112.7	1175.1	1175
Midwest	Gifts	Mug	1997/10/01	10745	75.5	1102.9	735
Midwest	Gifts	Mug	1997/10/01	7319	73.2	754.1	703
Midwest	Gifts	Mug	1997/10/01	4955	33.5	553.5	395
Midwest	Gifts	Mug	1997/11/01	11450	55.5	973.5	749
Midwest	Gifts	Mug	1997/11/01	9395	72.2	777.5	707
Midwest	Gifts	Mug	1997/11/01	5551	33.5	1107.5	739
Midwest	Gifts	Mug	1997/11/01	4917	49.2	593.6	496
Midwest	Gifts	Mug	1997/11/01	1960	13.1	352.1	335
Midwest	Gifts	Mug	1997/11/01	1956	15.0	390.1	330
Midwest	Gifts	Mug	1997/12/01	15355	109.0	1555.2	1039
Midwest	Gifts	Mug	1997/12/01	14912	114.7	1452.0	1215
Midwest	Gifts	Mug	1997/12/01	14655	112.7	1790.9	1194
Midwest	Gifts	Mug	1997/12/01	14337	110.0	1555.6	1135

## Excel Compound Reports Using XLSX

Excel compound reports generate compound workbooks that can contain multiple worksheet reports using the XLSX output format.



You can use standard Compound Layout syntax to generate XLSX compound workbooks. By default, each of the component reports from the compound report is placed in a new Excel worksheet (analogous to a new page in PDF).

The components of an Excel compound report can include standard tables, Table of Content (BYTOC), and ROWOVERFLOW reports.

Component graphs will be added to worksheets as images.

**Reference:** Usage Notes for Excel Compound Reports Using XLSX

- ☐ Images and graphs can be embedded within a component, but images and drawing objects (lines, boxes, strings) defined in Compound Layout syntax on a page layout will not be included in the generated workbook.
- ☐ Graphs and images are not supported in Excel headers and footers within XLSX compound workbooks.
- ☐ Coordinated compound reports that generate individual instances of the overall report for each unique primary key are not available in XLSX.

**Note:** Since multiple tables are generated, WebFOCUS will ensure that each tab name is unique.

**Example: Compound Excel Report including Table of Contents (BYTOC)**

```

SET PAGE-NUM=OFF
COMPOUND LAYOUT PCHOLD FORMAT XLSX
SECTION=Example, LAYOUT=ON, MERGE=OFF,$
PAGELAYOUT=1,$
COMPONENT=R1, TYPE=REPORT,TEXT='report1', POSITION=(0.833 0.729),
DIMENSION=(6.250 1.771),$
COMPONENT=R2, TYPE=REPORT,TEXT='report2', POSITION=(0.833 2.917),
DIMENSION=(6.250 1.875),$
COMPONENT=R3, TYPE=REPORT,TEXT='report3', POSITION=(0.938 5.313),
DIMENSION=(6.250 1.354),$
COMPONENT=R4, TYPE=REPORT,TEXT='report4', POSITION=(0.938 7.083),
DIMENSION=(6.042 1.146),$
END
SET COMPONENT=R1
TABLE FILE GGSales
HEADING CENTER
"Gotham Grinds Sales to Information Builders"
" "
"Report 1"
"Sales Summary by Region"
" "
SUM UNITS/D12C BUDUNITS/D12C DOLLARS/D12CM BUDDOLLARS/D12CM
BY REGION
ON TABLE HOLD FORMAT XLSX
ON TABLE SET STYLE *
TYPE=REPORT, TITLETEXT=Region Summary,$
TYPE=REPORT, TOPMARGIN=1.5, BOTTOMMARGIN=1, PAGESIZE=LETTER,$
TYPE=TITLE, COLOR=WHITE, BACKCOLOR=GREY,$
TYPE=HEADING, LINE=1, OBJECT=TEXT, COLOR=PURPLE, JUSTIFY=CENTER, STYLE=BOLD,
$
TYPE=HEADING, LINE=3, OBJECT=TEXT, COLOR=BLUE, JUSTIFY=CENTER, STYLE=BOLD,$
TYPE=HEADING, LINE=4, OBJECT=TEXT, COLOR=PURPLE, JUSTIFY=CENTER, STYLE=BOLD,
$
ENDSTYLE
END

```

```

SET COMPONENT=R2
TABLE FILE GGSales
SUM UNITS/D12C DOLLARS/D12CM
BY REGION BY CATEGORY BY PRODUCT
HEADING CENTER
"Gotham Grinds Sales to Information Builders"
" "
"Report 2"
"Sales Detail By Region"
ON REGION SUBHEAD
"<REGION Region Sales"
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET BYDISPLAY ON
ON TABLE SET COMPOUND BYTOC
ON TABLE SET STYLE *
TYPE=REPORT, TOPMARGIN=1.5, BOTTOMMARGIN=1, PAGESIZE=LETTER,$
TYPE=REPORT, TITLETEXT='Region-', $
TYPE=TITLE, COLOR=WHITE, BACKCOLOR=GREY,$
TYPE=HEADING, LINE=1, COLOR=PURPLE, JUSTIFY=CENTER, STYLE=BOLD,$
TYPE=HEADING, LINE=3, COLOR=BLUE, JUSTIFY=CENTER, STYLE=BOLD,$
TYPE=HEADING, LINE=4, COLOR=PURPLE, JUSTIFY=CENTER, STYLE=BOLD,$
ENDSTYLE
END

SET COMPONENT=R3
TABLE FILE GGSales
HEADING CENTER
"Gotham Grinds Sales to Information Builders"
" "
"Report 3"
"Sales Summary by Category"
" "
SUM UNITS/D12C BUDUNITS/D12C DOLLARS/D12CM BUDDOLLARS/D12CM
BY CATEGORY
ON TABLE HOLD FORMAT XLSX
ON TABLE SET STYLE *
TYPE=REPORT, TITLETEXT=Sales Summary,$
TYPE=REPORT, TOPMARGIN=1.5, BOTTOMMARGIN=1, PAGESIZE=LETTER,$
TYPE=TITLE, COLOR=WHITE, BACKCOLOR=GREY,$
TYPE=HEADING, LINE=1, COLOR=PURPLE, JUSTIFY=CENTER, STYLE=BOLD,$
TYPE=HEADING, LINE=3, OBJECT=TEXT, COLOR=BLUE, JUSTIFY=CENTER, STYLE=BOLD,$
TYPE=HEADING, LINE=4, OBJECT=TEXT, COLOR=PURPLE, JUSTIFY=CENTER, STYLE=BOLD,$
ENDSTYLE
END

```

```
SET COMPONENT=R4
TABLE FILE GGSales
HEADING CENTER
"Gotham Grinds Sales to Information Builders"
" "
"Report 4"
"Sales Detail Report By Category"
" "

SUM UNITS/D12C BUDUNITS/D12C DOLLARS/D12C BUDDOLLARS/D12C
BY CATEGORY BY PRODUCT BY REGION
ON TABLE SET BYDISPLAY ON
ON TABLE HOLD FORMAT XLSX
ON TABLE SET STYLE *
TYPE=REPORT, TITLETEXT=Sales Detail,$
TYPE=REPORT, TOPMARGIN=1.5, BOTTOMMARGIN=1, PAGESIZE=LETTER,$
TYPE=TITLE, COLOR=WHITE, BACKCOLOR=GREY,$
TYPE=HEADING,LINE=1,OBJECT=TEXT,COLOR=PURPLE, JUSTIFY=CENTER, STYLE=BOLD,$
TYPE=HEADING,LINE=3,OBJECT=TEXT,COLOR=BLUE, JUSTIFY=CENTER, STYLE=BOLD,$
TYPE=HEADING,LINE=4,OBJECT=TEXT,COLOR=PURPLE, JUSTIFY=CENTER, STYLE=BOLD,$
ENDSTYLE
END
COMPOUND END
```

The output is:

Report 1: Summary Report by Region

	A	B	C	D	E
1	Gotham Grinds Sales to Information Builders				
2					
3	Report 1				
4	Sales Summary by Region				
5					
6	Region	Unit Sales	Budget Units	Dollar Sales	Budget Dollars
7	Midwest	905,045	907,107	\$11,400,665	\$11,194,373
8	Northeast	916,675	914,359	\$11,392,300	\$11,576,921
9	Southeast	935,232	942,247	\$11,710,379	\$11,807,971
10	West	932,039	930,781	\$11,652,946	\$11,641,513
11					

Region Summary

Region-Midwest

Region-Northeast

Region-Southeast

Region-West

**Report 2: BYTOC Reports by Region**

The following image is tiled to show multiple worksheet reports, one for each region.

	A	B	C	D	E
1	Gotham Grinds Sales to Information Builders				
2					
3	Report 2				
4	Sales Detail By Region				
5	Region	Category	Product	Unit Sales	Dollar Sales
6	West Region Sales				
7	West	Coffee	Capuccino	71,168	\$895,495
8	West	Coffee	Espresso	71,675	\$907,617
9	West	Coffee	Latte	213,920	\$2,670,405
10	West	Food	Biscotti	70,436	\$863,868
11	West	Food	Croissant	197,022	\$2,425,601
12	West	Food	Scone	72,776	\$912,868
13	West	Gifts	Coffee Grinder	48,081	\$603,436
14	West	Gifts	Coffee Pot	47,432	\$613,624
15	West	Gifts	Mug	93,881	\$1,188,664
16	West	Gifts	Thermos	45,648	\$571,368
17					
Region-Southeast Region-West Sales Summary Sales					
Region Summary Region-Midwest Region-Northeast Re					
Region Summary Region-Midwest Region-Northeast Re					
Region-Midwest Region-Northeast Region-Southeast					

**Report 3: Sales Summary Report by Category**

	A	B	C	D	E
1	Gotham Grinds Sales to Information Builders				
2					
3	Report 3				
4	Sales Summary by Category				
5					
6	Category	Unit Sales	Budget Units	Dollar Sales	Budget Dollars
7	Coffee	1,376,266	1,385,923	\$17,231,455	\$17,293,886
8	Food	1,384,845	1,377,564	\$17,229,333	\$17,267,160
9	Gifts	927,880	931,007	\$11,695,502	\$11,659,732
10					
11					
Region-Southeast Region-West Sales Summary Sales Det					

### Report 4: Sales Detail Report by Category

	A	B	C	D	E	F	G
2							
3	Report 4						
4	Sales Detail Report By Category						
5							
6	Category	Product	Region	Unit Sales	Budget Units	Dollar Sales	Budget Dollars
7	Coffee	Capuccino	Northeast	44,785	44,432	\$542,095	\$561,491
8	Coffee	Capuccino	Southeast	73,264	75,353	\$944,000	\$956,661
9	Coffee	Capuccino	West	71,168	70,585	\$895,495	\$877,304
10	Coffee	Espresso	Midwest	101,154	101,869	\$1,294,947	\$1,258,232
11	Coffee	Espresso	Northeast	68,127	69,776	\$850,107	\$872,902
12	Coffee	Espresso	Southeast	68,030	66,785	\$853,572	\$849,465
13	Coffee	Espresso	West	71,675	72,927	\$907,617	\$923,941
14	Coffee	Latte	Midwest	231,623	233,657	\$2,883,566	\$2,827,800
15	Coffee	Latte	Northeast	222,866	221,712	\$2,771,815	\$2,818,069
16	Coffee	Latte	Southeast	209,654	213,555	\$2,617,836	\$2,625,303
17	Coffee	Latte	West	213,920	215,272	\$2,670,405	\$2,722,718
18	Food	Biscotti	Midwest	86,105	85,839	\$1,091,727	\$1,067,629
19	Food	Biscotti	Northeast	145,242	145,152	\$1,802,005	\$1,848,682
20	Food	Biscotti	Southeast	119,594	120,549	\$1,505,717	\$1,512,019
21	Food	Biscotti	West	70,436	67,780	\$863,868	\$861,804
22	Food	Croissant	Midwest	139,182	139,648	\$1,751,124	\$1,708,733
23	Food	Croissant	Northeast	137,394	137,864	\$1,670,818	\$1,739,522
24	Food	Croissant	Southeast	156,456	157,148	\$1,902,359	\$1,969,906
25	Food	Croissant	West	197,022	195,329	\$2,425,601	\$2,406,554
26	Food	Scone	Midwest	116,127	113,776	\$1,495,420	\$1,444,359
27	Food	Scone	Northeast	70,732	68,415	\$907,171	\$865,703
28	Food	Scone	Southeast	73,779	73,812	\$900,655	\$927,363
	Region-Northeast	Region-Southeast	Region-West	Sales Summary	Sales Detail		

**Reference:** Guidelines for Using the Legacy OPEN, CLOSE, and NOBREAK Keywords and SET COMPOUND

The keywords OPEN, CLOSE, and NOBREAK are used to control Excel compound reports. They can be specified with the HOLD or PCHOLD command or with a separate SET COMPOUND command.

- ❑ OPEN is used on the first report of a sequence of component reports to specify that a compound report should be started.
- ❑ CLOSE is used to designate the last report in a compound report.
- ❑ NOBREAK specifies that the next report be placed on the same worksheet as the current report. If it is not present, the default behavior is to place the next report on a separate worksheet.

- ❑ When used with the HOLD or PCHOLD syntax, the compound report keywords OPEN, CLOSE, and NOBREAK must appear immediately after FORMAT XLSX. For example, you can specify:

- ❑ `ON TABLE PCHOLD FORMAT XLSX OPEN`

- ❑ `ON TABLE HOLD AS MYHOLD FORMAT XLSX OPEN NOBREAK`

- ❑ As with PDF compound reports, compound report keywords can be alternatively specified using SET COMPOUND:

- ❑ `SET COMPOUND = OPEN`

- ❑ `SET COMPOUND = 'OPEN NOBREAK'`

- ❑ `SET COMPOUND = NOBREAK`

- ❑ `SET COMPOUND = CLOSE`

### **Reference:** Guidelines for Producing Excel Compound Reports Using XLSX

- ❑ **Naming of Worksheets.** The default worksheet tab names will be Sheet1, Sheet2, and so on. You have the option to specify a different worksheet tab name by using the TITLETEXT keyword in the StyleSheet. For example:

```
TYPE=REPORT, TITLETEXT='Summary Report', $
```

Excel limits the length of worksheet titles to 31 characters. The following special characters cannot be used: ':', '?', '\*', and '/'.

- ❑ **File Names and Formats.** The output file name (AS name, or HOLD by default) is obtained from the first report of the compound report (the report with the OPEN keyword). Output file names on subsequent reports are ignored.

The HOLD FORMAT syntax used in the first component report in a compound report applies to all subsequent reports in the compound report, regardless of their format.

- ❑ **NOBREAK Behavior.** When NOBREAK is specified, the following report appears on the row immediately after the last row of the report with the NOBREAK. If additional spacing is required between the reports, a FOOTING or an ON TABLE SUBFOOT can be placed on the report with the NOBREAK, or a HEADING or an ON TABLE SUBHEAD can be placed on the following report. This allows the most flexibility, since if blank rows were added by default, there would be no way to remove them.

**Example:**     **Creating a Simple Compound Report Using XLSX**

```
SET PAGE-NUM=OFF
TABLE FILE GGSALES
HEADING
"Report 1: Coffee - Budget"
" "
SUM BUDDOLLARS BUDUNITS COLUMN-TOTAL AS 'Total'
BY REGION
ON TABLE SET STYLE *
TYPE=REPORT, TITLETEXT=Coffee Budget,$
TYPE=HEADING, SIZE=14,$
ENDSTYLE
ON TABLE PCHOLD AS EX1 FORMAT XLSX OPEN
END

TABLE FILE GGSALES
HEADING
"Report 2: Coffee - Actual "
SUM DOLLARS UNITS COLUMN-TOTAL AS 'Total'
BY REGION
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET STYLE *
TYPE=REPORT, TITLETEXT=Coffee Actual,$
TYPE=HEADING, SIZE=14,$
ENDSTYLE
END

TABLE FILE GGSALES
HEADING
"Report 3: Food - Budget"
SUM BUDDOLLARS BUDUNITS COLUMN-TOTAL AS 'Total'
BY REGION
ON TABLE SET STYLE *
TYPE=REPORT, TITLETEXT=Food Budget,$
TYPE=HEADING, SIZE=14,$
ENDSTYLE
ON TABLE PCHOLD FORMAT XLSX CLOSE
END
```



The output is:

Report 1: Coffee - Budget		
Region	Budget Dollars	Budget Units
Midwest	11194373	907107
Northeast	11576921	914359
Southeast	11807971	942247
West	11641513	930781
Total	46220778	3694494

Report 2: Coffee - Actual		
Region	Dollar Sales	Unit Sales
Midwest	11400665	905045
Northeast	11392300	916675
Southeast	11710379	935232
West	11652946	932039
Total	46156290	3688991

Report 3: Food - Budget		
Region	Budget Dollars	Budget Units
Midwest	11194373	907107
Northeast	11576921	914359
Southeast	11807971	942247
West	11641513	930781
Total	46220778	3694494

### Example: Creating a Compound Report Using NOBREAK

In this example, the first two reports are on the first worksheet, and the last two reports are on the second worksheet, since NOBREAK appears on both the first and third reports.

```
TABLE FILE GGSales
HEADING
"Report 1: Coffee - Budget"
SUM BUDDOLLARS BUDUNITS COLUMN-TOTAL AS 'Total'
BY REGION
IF CATEGORY EQ Coffee
ON TABLE PCHOLD FORMAT XLSX OPEN NOBREAK
ON TABLE SET STYLE *
TYPE=REPORT, TITLETEXT=Coffee, FONT=ARIAL, SIZE=10, STYLE=NORMAL,$
TYPE=TITLE, STYLE=BOLD,$
TYPE=HEADING, SIZE=12, STYLE=BOLD, COLOR=BLUE,$
TYPE=GRANDTOTAL, STYLE=BOLD,$
END
```

```
TABLE FILE GGSales
HEADING
" "
"Report 2: Coffee - Actual "
SUM DOLLARS UNITS COLUMN-TOTAL AS 'Total'
BY REGION
IF CATEGORY EQ Coffee
ON TABLE PCHOLD FORMAT XLSX
ON TABLE SET STYLE *
TYPE=REPORT, FONT=ARIAL, SIZE=10, STYLE=NORMAL,$
TYPE=GRANDTOTAL, STYLE=BOLD,$
TYPE=HEADING, SIZE=12, STYLE=BOLD, COLOR=BLUE,$
END
```

```
TABLE FILE GGSALES
HEADING
"Report 3: Food - Budget"
SUM BUDDOLLARS BUDUNITS COLUMN-TOTAL AS 'Total'
BY REGION
IF CATEGORY EQ Food
ON TABLE PCHOLD FORMAT XLSX NOBREAK
ON TABLE SET STYLE *
TYPE=REPORT, TITLETEXT=Food, FONT=ARIAL, SIZE=10, STYLE=NORMAL,$
TYPE=HEADING, STYLE=BOLD, SIZE=12, COLOR=BLUE,$
TYPE=TITLE, STYLE=BOLD,$
TYPE=GRANDTOTAL, STYLE=BOLD,$
END
```

```
TABLE FILE GGSALES
HEADING
" "
"Report 4: Food - Actual"
SUM DOLLARS UNITS COLUMN-TOTAL AS 'Total'
BY REGION
IF CATEGORY EQ Food
ON TABLE PCHOLD FORMAT XLSX CLOSE
ON TABLE SET STYLE *
TYPE=REPORT, FONT=ARIAL, SIZE=10, $
TYPE=TITLE, STYLE=BOLD,$
TYPE=HEADING, SIZE=12, STYLE=BOLD, COLOR=BLUE,$
TYPE=GRANDTOTAL, STYLE=BOLD,$
END
```

Report output is displayed in two separate tabs.

Region	Budget Dollars	Budget Units
Midwest	4086032	335526
Northeast	4252462	335920
Southeast	4431429	355693
West	4523963	358784
<b>Total</b>	<b>17293886</b>	<b>1385923</b>

Region	Dollar Sales	Unit Sales
Midwest	4178513	332777
Northeast	4164017	335778
Southeast	4415408	350948
West	4473517	356763
<b>Total</b>	<b>17231455</b>	<b>1376266</b>

Region	Budget Dollars	Budget Units
Midwest	4220721	339263
Northeast	4453907	351431
Southeast	4409288	351509
West	4183244	335361
<b>Total</b>	<b>17267160</b>	<b>1377564</b>

Region	Dollar Sales	Unit Sales
Midwest	4338271	341414
Northeast	4379994	353368
Southeast	4308731	349829
West	4202337	340234
<b>Total</b>	<b>17229333</b>	<b>1384845</b>

## Using XLSX FORMULA With Compound Reports

In new compound syntax, the implementation of compound workbooks with XLSX FORMULA can be activated in either of the following ways. Each of these approaches will generate a workbook with all of the component reports in FORMULA mode.

1. Add FORMAT XLSX FORMULA to the compound syntax header, as shown in the following syntax:

```
COMPOUND LAYOUT PCHOLD FORMAT XLSX FORMULA
UNITS=IN, $
SECTION=section1, LAYOUT=ON, METADATA='prop_with_names, Margins_Left=0.5,
Margins_Top=0.5, Margins_Right=0.5, Margins_Bottom=0.5, thumbnailscale=4,
MERGE=OFF, ORIENTATION=PORTRAIT, PAGESIZE=Letter, SHOW_GLOBALFILTER=OFF,
$
PAGELAYOUT=1, NAME='Page layout 1', text='Page layout 1', TOC-LEVEL=1,
BOTTOMMARGIN=0.5, TOPMARGIN=0.5, METADATA='BOTTOMMARGIN=0.5,
TOPMARGIN=0.5, LEFTMARGIN=0, RIGHTMARGIN=0,', $
COMPONENT='report1', TEXT='report1', TOC-LEVEL=2, POSITION=(0.567 0.667),
DIMENSION=(6.883 2.314), BYTOC=0, ARREPORTSIZE=DIMENSION,
METADATA='left: 0.567in; top: 0.667in; width: 6.883in; height: 2.314in;
position: absolute; z-index: 1;', $
COMPONENT='report2', TEXT='report2', TOC-LEVEL=2, POSITION=(0.567 3.250),
DIMENSION=(7.000 2.833), BYTOC=0, ARREPORTSIZE=DIMENSION,
METADATA='left: 0.567in; top: 3.25in; width: 7in; height: 2.833in;
position: absolute; z-index: 2;', $
END
```

2. Define XLSX FORMULA as the output setting for the first component, as shown in the following syntax:

```

SET COMPONENT='report1'
-*component_type report
-*File: IBFS:/localhost/EDA/9999/APPPATH/xlsx2015/Report1.fex
-*Created by WebFOCUS AppStudio
DEFINE FILE GGSales
D_UOVBUD/D12C=GGSales.SALES01.UNITS - GGSales.SALES01.BUDUNITS;
END
TABLE FILE GGSales
SUM
    GGSales.SALES01.UNITS
    GGSales.SALES01.BUDUNITS
    GGSales.SALES01.DOLLARS
    GGSales.SALES01.BUDDOLLARS
    GGSales.SALES01.D_UOVBUD
    COMPUTE C_DVBUD/D12.2CM = GGSales.SALES01.DOLLARS -
        GGSales.SALES01.BUDDOLLARS;
BY GGSales.SALES01.REGION
BY GGSales.SALES01.CATEGORY
HEADING
"XLSX FORMULA - the difference between DEFINE & COMPUTES"
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE SET ASNAMES ON
ON TABLE COLUMN-TOTAL AS 'TOTAL'
ON TABLE PCHOLD FORMAT XLSX FORMULA
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
    INCLUDE = endeflt,
$
ENDSTYLE

```

**Note:**

- ❑ Although both of these approaches will work, from a tool perspective, it is advantageous to use the first approach (add FORMAT XLSX FORMULA to the compound syntax header) since the user may not have control of the first table when system tables are generated with page masters and in other page layout scenarios. This approach ensures that the XLSX FORMULA designation is always moved to the first table.
- ❑ If you change the format for the compound workbook so that the compound header requests XLSX, not XLSX FORMULA, this will *not* automatically turn off FORMULA. When FORMULA is set in the first component, this will always override the compound heading setting. This may cause confusion when you think you have turned FORMULA off at the document level, but it is actually retained at the first component level.

## FORMAT XLSX Limitations

Format XLSX does not support the following features, currently supported for EXL2K:

- ❑ Cell locking
- ❑ XLSX reports are available on a z/OS USS server but are not currently supported on a z/OS PDS server or on a z/OS USS server with the setting DYNAM TEMP ALLOC MVS.

For additional support on the implementation of features supported by the XLSX format, see *WebFOCUS XLSX Format Supported Features Roadmap*, located at the following link:

[https://techsupport.informationbuilders.com/tech/wbf/wbf\\_rln\\_formatXLSX\\_support.html](https://techsupport.informationbuilders.com/tech/wbf/wbf_rln_formatXLSX_support.html)

## Using PowerPoint PPT Display Format

Specifying PowerPoint (PPT) as the output format creates a PowerPoint document with a single slide that includes the report.

You can add multiple graphs and images to a PowerPoint presentation. The PowerPoint output format can contain a variety of graphs positioned anywhere on a slide to create a visual layout.

You can also place report output on a specific slide in a PowerPoint template.

## Using PowerPoint PPT Templates

A WebFOCUS report can be placed inside of an existing PowerPoint presentation. This enables you to populate existing presentations with preset Slide Masters, styling, and other business content. PowerPoint PPT templates are stored on the server with a .MHT extension and can be distributed automatically with ReportCaster.

### **Syntax:** How to Create PowerPoint PPT Report Output

```
ON TABLE {PCHOLD|HOLD|SAVE} [AS name] FORMAT PPT
      [TEMPLATE 'template' SLIDENUMBER n]
```

where:

*name*

Is the name of the PowerPoint output file.

*template*

Is the name of the PowerPoint template file. The template file must have at least one blank slide and must be saved as a Web Archive (.MHT extension) on your WebFOCUS Reporting Server application directory.

**Tip:** Since the Reporting Server cannot differentiate between Excel and PowerPoint template files (.MHT), it is important to apply a naming convention to your template files. This will help you organize and distinguish the different Excel and PowerPoint template files when developing your reports. For example, Excel\_template.MHT or PPT\_template.MHT.

*n*

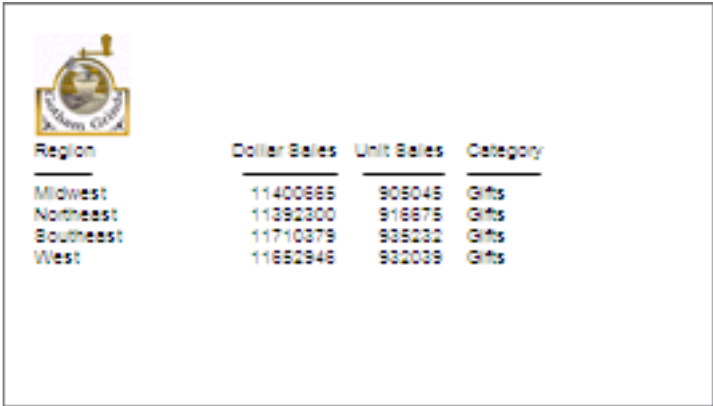
Is the number of the slide on which to place the report output. This number is optional if the template has only one slide.

**Example:**    **Using a PowerPoint PPT Template**

The following request against the GGSALES data source inserts a WebFOCUS report into a PowerPoint template named mytemplate.mht, which is stored in the application directory:

```
TABLE FILE GGSALES
HEADING
" "
" "
" "
" "
" "
SUM DOLLARS UNITS CATEGORY
BY REGION
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PPT TEMPLATE 'mytemplate' SLIDENUMBER 1
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE=REPORT, FONT=ARIAL, SIZE=10,$
TYPE=HEADING, image=gglogo.gif, POSITION=(0.000000 0.000000),$
ENDSTYLE
END
```

The output is:



Region	Dollar Sales	Unit Sales	Category
Midwest	11400665	905045	Gifts
Northeast	11392300	916875	Gifts
Southeast	11710379	935232	Gifts
West	11652946	932039	Gifts

## Saving Report Output in PPTX Format

WebFOCUS Release 8.0 Version 08 introduced the capability to retrieve data from any WebFOCUS supported data source and to generate a PPTX formatted (PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013) presentation.

The PPTX format in WebFOCUS supports the following Microsoft Office software products:

- ☐ Microsoft Office 2013/2010/2007.
- ☐ Microsoft Office 2000/2003 with the Microsoft Office Compatibility Pack.

**Note:** As per the Microsoft [Support has ended for Office 2003](#) statement, support and updates for Office 2003 are no longer available. Although you will still be able to start and work in the Microsoft Office 2003 application, Microsoft recommends that you upgrade to a newer version of Office to get continuing support and updates. If your organization uses Office 2003, go to the [end of support page for Windows XP SP3 and Office 2003](#) for more information.

- ☐ Open Office Support (FORMAT PPTX). Core PowerPoint functionality generated by the PPTX format is supported for Open Office as of WebFOCUS Release 8.0 Version 08. For details on Open Office, see <http://www.openoffice.org/>.
- ☐ MAC Office 2008 and 2011. FORMAT PPTX is supported as of WebFOCUS Release 8.0 Version 08.

WebFOCUS generates PPTX presentations based on the Microsoft PPTX standard. These presentations are accessible through all browsers and the Microsoft PowerPoint mobile application.

**Note:** This section, which describes important PPTX features, applies to PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013, unless otherwise indicated.

## Building the .pptx Presentation File

Microsoft changed the format and structure of the PowerPoint presentation file in PowerPoint 2007. The new .pptx file is a binary compilation of a group of .xml files. Generating this new file format using WebFOCUS is a two-step process that consists of generating the .xml files containing the report output and zipping the .xml documents into the binary .pptx format. The Reporting Server performs the xml generation process. The zipping process can be completed either by the Client (WebFOCUS Servlet) or the Server (JSCOM3):

- ☐ **WebFOCUS Servlet.** The WebFOCUS Client within the application server performs the zipping process. This can be done within the local client or through a remotely accessed client. The servlet method is the default approach defined for each WebFOCUS Client, with the client pointing to itself, by default.

- ❑ **JSCOM3.** The Java layer of the Reporting Server performs the zipping operation. This option should be used when the WebFOCUS Servlet is configured on a secured web or application server. This is because JSCOM3 does not require URL access to a remote WebFOCUS Client.

### **Syntax:**      **How to Select the Method for Zipping the .pptx File**

You designate the method and location where the zipping will occur by setting EXCELSERVURL to a URL (for the WebFOCUS Servlet) or to a blank (for JSCOM3). Even though this setting is prefaced with EXCEL, when applied, the same results are achieved for PowerPoint. You can set this value for a specific procedure or for the entire environment:

- ❑ **For a procedure.** Issue the SET EXCELSERVURL command within the procedure.
- ❑ **For the entire environment.** Edit the IBIF\_excelservurl variable in the WebFOCUS Administration Console by selecting *Configuration/Client Settings/General/IBIF\_excelservurl*.

For more information on accessing the WebFOCUS Administration Console and setting the IBIF\_excelservurl variable, see the *WebFOCUS Security and Administration* manual.

The value you assign to EXCELSERVURL determines whether the WebFOCUS Servlet or JSCOM3 performs the zipping operation:

- ❑ **Specifying the Servlet.** To specify that the WebFOCUS Servlet should be used, set the EXCELSERVURL parameter or the IBIF\_excelservurl variable to the URL of a WebFOCUS Release 8.0 Version 09 or higher client configuration.

In a procedure:

```
SET EXCELSERVURL = http://servername:8080/ibi_apps
```

In the WebFOCUS Administration Console:

```
IBIF_excelservurl = http://servername:8080/ibi_apps
```

- ❑ **Specifying JSCOM3.** To specify that JSCOM3 should be used within the current Reporting Server, set EXCELSERVURL to a blank or an empty string.

In a procedure:

```
SET EXCELSERVURL = ' '
```

In the WebFOCUS Administration Console:

```
IBIF_excelservurl = ' '
```



By default, each WebFOCUS Client contains the following URL definition that points to itself:

```
&URL_PROTOCOL: //&servername:&server_port&IBIF_webapp
```

### **Syntax:** How to Generate a PPTX Presentation

You can specify that a report should be saved to a PPTX presentation, displayed in the browser, or displayed in the PowerPoint application.

```
ON TABLE {PCHOLD|HOLD} AS name FORMAT PPTX
```

where:

**PCHOLD**

Displays the generated presentation in either the browser or the PowerPoint application, based on your desktop settings. For information, see [Viewing PowerPoint Presentations in the Browser vs. the PowerPoint Application](#) on page 687.

**HOLD**

Saves a presentation with a .pptx extension to the designated location.

*name*

Specifies a file name for the generated presentation.

**Note:** To assign a file name to the generated presentation, set the Save Report option to YES for the .pptx file extension in the WebFOCUS Client Redirection Settings. When opened in the PowerPoint application, the generated presentation will retain the designated AS name. For more information on the Redirection Settings, see the *WebFOCUS Security and Administration* manual.

### **Opening PPTX Report Output**

To open PPTX presentations, the user must have an account for Microsoft Office365 or Microsoft PowerPoint 2013, 2010, or 2007 must be installed on the desktop.

Upon execution of a report with FORMAT PPTX, the user is prompted to Open or Save the PPTX file. The file name displayed before the .pptx extension is an internally generated name.

The WebFOCUS procedure generates a presentation containing as many slides as required to display output. A report may contain defined elements, such as headings, subtotals, and titles, as well as StyleSheet syntax, such as conditional styling and drill downs.

## Opening PPTX Report Output in Microsoft PowerPoint 2000/2003

PowerPoint 2000 and PowerPoint 2003 can be updated to read PowerPoint PPTX presentations using the Microsoft Office Compatibility Pack available from the Microsoft download site (<http://www.microsoft.com/downloads/en/default.aspx>). When the file extension of the file being opened is .pptx (PPTX presentation), the Microsoft Office Compatibility Pack performs the necessary conversion to allow PowerPoint 2000/2003 to read and open it.

In addition to the Microsoft Office Compatibility Pack, it is important to enable the WebFOCUS Client Redirection Settings Save As option so that PowerPoint 2000/2003 will be able to open the PPTX report output without users first having to save it to their machine with the .pptx file extension. The WebFOCUS Client processing Redirection Settings Save As option configures how the WebFOCUS Client sends each report output file type to the user machine.

This option can be set as follows:

- ☐ **Save As Option disabled (NO).** The WebFOCUS Client Redirection Setting Save As is disabled, by default. When the Save As option is disabled, the WebFOCUS Client sends report output to the user machine in memory with the application association specified for the report format in the WebFOCUS Client Redirection Settings configuration file (mime.wfs).

A user machine that does not have PowerPoint 2007 or higher installed will not recognize the application association for PowerPoint and PowerPoint will display a message. The PowerPoint 2000/2003 user can select Save and provide a file name with the .pptx extension to save the report output to their machine. The user can then open the .pptx file directly from PowerPoint 2000/2003.

- ☐ **Save As Option enabled (YES).** When the WebFOCUS Redirection Save As option is enabled, the WebFOCUS Client sends the report output to the user as a file with the extension specified in the WebFOCUS Client Redirection Settings configuration file (mime.wfs). Upon receiving the file, Windows will display the File Download prompt asking the user to Open or Save the file with the identified application type. The File Download prompt displays the Name with the .pptx file extension for the report output that is recognized as a PowerPoint PPTX file type.

**Note:** The download prompt will display for all users, including users who have PowerPoint 2007 or higher installed on their machines.

If a PowerPoint 2000/2003 user chooses to open the file, the Microsoft Office Compatibility Pack will recognize the .pptx file extension and perform the necessary conversion to allow PowerPoint 2000/2003 to read the PowerPoint PPTX presentation.

If a PowerPoint 2007 or higher user chooses to open the file, PowerPoint will recognize the .pptx file extension and read the PowerPoint PPTX presentation.

For additional information on WebFOCUS Client Redirection Settings, see the *WebFOCUS Security and Administration* manual.

### Viewing PowerPoint Presentations in the Browser vs. the PowerPoint Application

Your Operating System and desktop settings determine whether PowerPoint output sent to the client is displayed in an Internet Browser window or within the PowerPoint application. When PowerPoint output has been defined within the Windows environment to Browse in same window, the workbook generated by a WebFOCUS request is opened within an Internet Explorer® browser window. When the Browse in same window option is unchecked for the .ppt file type, the browser window created by WebFOCUS is blank because the report output is displayed in the stand-alone PowerPoint application window.

- ❑ In Windows XP and earlier, file type specific settings are managed on the desktop within Windows Explorer by selecting *Tools/Folder Options*, clicking the *File Types* tab, selecting the extension (.ppt or .pptx), clicking the *Advanced* button, and checking the *Browse in same window* check box.

As per the Microsoft [Support has ended for Office 2003](#) statement, support and updates for Office 2003 are no longer available. Although you will still be able to start and work in the Microsoft Office 2003 application, Microsoft recommends that you upgrade to a newer version of Office to get continuing support and updates. If your organization uses Office 2003, go to the [end of support page for Windows XP SP3 and Office 2003](#) for more information.

- ❑ In Windows 7, Microsoft removed the desktop settings that support opening worksheets in the browser. This means that to change this behavior, you can no longer simply navigate to the Folder Options dialog box, but you must change a registry setting. This change is documented in the Microsoft Knowledge Base Article ID 927009 at the following website:

<http://support.microsoft.com/kb/927009>

**Note:** This works the same for both PPT and PPTX formats. The only difference is the selection of file type based on the version of PowerPoint output you will be generating.

### Grouping Tables and Components in a PowerPoint Slide

When table elements are placed on a PowerPoint slide, the elements are placed in individual text boxes to allow for explicit positioning to match the other positioned drivers, such as PDF and DHTML.

The PPTXGROUP parameter enables you to group elements together in a PPTX report. You can rotate, flip, move, or resize objects within a group at the same time as though they were a single object. You can also change the attributes of all of the objects in a group at one time, including font, color, or size, and you can ungroup a group of objects at any time, and then regroup them later.

In WebFOCUS, grouping is done within each report component. Objects within the report component (or stand-alone report), including data, all headings and footings, and images, are grouped together. In compound reports, each component report is grouped individually and non-component elements, such as drawing objects, lines, and images, are not included in any group.

### ***Syntax:*** How to Group Tables and Components in a PowerPoint Slide

```
SET PPTXGROUP = {ON|OFF}
```

The command can also be issued from within a report using:

```
ON TABLE SET PPTXGROUP = {ON|OFF}
```

In a StyleSheet:

```
TYPE=REPORT, PPTXGROUP = {ON|OFF}
```

where:

ON

Enables you to group elements together in a PPTX report.

OFF

Indicates no grouping of elements, which is the legacy behavior. OFF is the default value.

### ***Example:*** Displaying Group Tables and Components in a Standard Report

In the following standard report, the grouping is defined for the core report elements, excluding images and drawing objects, defined at the TYPE=REPORT level.

```

TABLE FILE GGSales
SUM DOLLARS/D12CM UNITS/D12C BUDDOLLARS/D12CM BUDUNITS/D12C
COMPUTE SHOWCAT/A100=CATEGORY||'.GIF';
BY REGION
BY CATEGORY
ON TABLE SUBHEAD
"Report Heading Here"
" "
" "
" "
HEADING
"Page Heading here"
FOOTING
"Page Footing here"
ON TABLE SUBFOOT
"Report Footing Here"
ON TABLE PCHOLD AS GROUPTTEST FORMAT PPTX
ON TABLE SET STYLE *
TYPE=REPORT, PPTXGROUP = ON, SQUEEZE=ON, FONT=TAHOMA,SIZE=12,
ORIENTATION=LANDSCAPE, $
TYPE=REPORT, OBJECT=LINE, POSITION=(9 1), DIMENSION=(1 5), COLOR=BLUE, $
TYPE=REPORT, IMAGE=IBILOGO, POSITION=(0 0), $
TYPE=TABHEADING, IMAGE=GGLOGO.GIF, POSITION=(6.5 .10), $
TYPE=DATA,COLUMN=SHOWCAT, IMAGE=(SHOWCAT), PRESERVERATIO=ON,
SIZE=(.5 .5), $
END












```

The output is:


Information Builders  
The Standard for Enterprise Business Intelligence

Report Heading Here

Page Heading here

Region	Category	Dollar Sales	Unit Sales	Budget Dollars	Budget Units	SHOWCAT
Midwest	Coffee	\$4,178,513	332,777	\$4,086,032	335,526	
	Food	\$4,338,271	341,414	\$4,220,721	339,263	
	Gifts	\$2,883,881	230,854	\$2,887,620	232,318	
Northeast	Coffee	\$4,164,017	335,778	\$4,252,462	335,920	
	Food	\$4,379,994	353,368	\$4,453,907	351,431	
	Gifts	\$2,848,289	227,529	\$2,870,552	227,008	
Southeast	Coffee	\$4,415,408	350,948	\$4,431,429	355,693	
	Food	\$4,308,731	349,829	\$4,409,288	351,509	
	Gifts	\$2,986,240	234,455	\$2,967,254	235,045	
West	Coffee	\$4,473,517	356,763	\$4,523,963	358,784	
	Food	\$4,202,337	340,234	\$4,183,244	335,361	

Page Footing here



**Example:**    **Displaying Group Tables and Components in a Compound Report**

In the following compound report, the grouping is defined for the component report. Additional objects on the page, including the chart image, logo image, lines, and text box are not included in the grouping.

```

SET HTMLARCHIVE=ON
SET PPTXGRAPHTYPE=PNG
SET PPTXGROUP=ON
COMPOUND LAYOUT PCHOLD FORMAT PPTX
UNITS=IN, $
SECTION=section1, LAYOUT=ON, MERGE=OFF, ORIENTATION=LANDSCAPE, PAGESIZE=PPT
Slide, $
PAGELAYOUT=1, NAME='Page layout 1', TEXT='Page layout 1', BOTTO MMARGIN=0.5,
TOPMARGIN=0.5, $
OBJECT=STRING, NAME='text1', TEXT='<left>Grouping is supported within the
component reports of a compound report:
<ul type=disc>
<LI>Each table / report is grouped together. </LI>
<LI>Drawing objects such as images and lines are not included in any group.
</LI>
<LI>Charts are inserted as images.</LI></ul><div><br></div><br></left>',
POSITION=(2.764 4.958), MARKUP=ON, WRAP=ON, DIMENSION=(4.000 1.992),
FONT='TREBUCHET MS', COLOR=RGB(0 0 0), SIZE=10, $
COMPONENT='report1', TEXT='report1', POSITION=(1.028 2.083),
DIMENSION=(4.431 1.667), $
COMPONENT='chart2', TEXT='chart2', POSITION=(4.972 1.319), DIMENSION=(4.456
3.085), COMPONENT-TYPE=GRAPH, $
OBJECT=BOX, NAME='line1', POSITION=(0.498 6.500),
DIMENSION=(9.167 0.022), BACKCOLOR=BLACK, BORDER-COLOR=BLACK, $
OBJECT=BOX, NAME='line2', POSITION=(0.502 1.097),
DIMENSION=(9.167 0.022), BACKCOLOR=BLACK, BORDER-COLOR=BLACK, $
OBJECT=IMAGE, NAME='image4', IMAGE=ibolologo.gif, ALT='',
POSITION=(0.502 0.499), DIMENSION=(1.861 0.506), $
END

SET COMPONENT='report1'
TABLE FILE GGSales
SUM
    GGSales.SALES01.DOLLARS
BY GGSales.SALES01.REGION
ACROSS LOWEST GGSales.SALES01.CATEGORY
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
    INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
$
ENDSTYLE
END

```

```

SET COMPONENT='chart2'
SET PAGE-NUM=NOLEAD
SET ARGRAPHENGINE=JSCHART
SET EMBEDHEADING=ON
SET GRAPHDEFAULT=OFF
GRAPH FILE GGSales
SUM GGSales.SALES01.DOLLARS
BY GGSales.SALES01.CATEGORY
ACROSS GGSales.SALES01.REGION
ON GRAPH PCHOLD FORMAT HTML
ON GRAPH SET VZERO OFF
ON GRAPH SET GRWIDTH 1
ON GRAPH SET UNITS 'PIXELS'
ON GRAPH SET HAXIS 770.0
ON GRAPH SET VAXIS 405.0
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 0
ON GRAPH SET GRLEGEND 1
ON GRAPH SET GRXAXIS 1
ON GRAPH SET LOOKGRAPH HBAR
ON GRAPH SET STYLE *

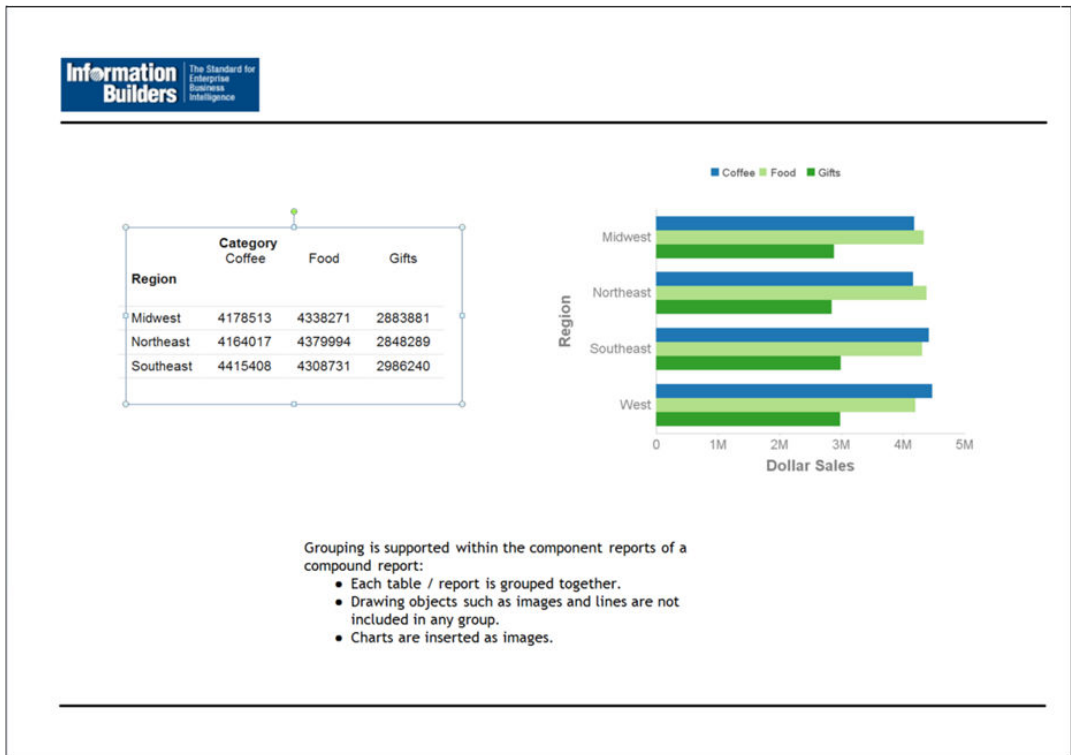
*GRAPH_SCRIPT
setPieDepth(0);
setPieTilt(0);
setDepthRadius(0);
setCurveFitEquationDisplay(false);
setPlace(true);
*END
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, TITLETEXT='WebFOCUS Report', $
*GRAPH_SCRIPT
setLegendPosition(4);
*GRAPH_JS_FINAL
"blaProperties": {
    "orientation": "horizontal"
},
"agnosticSettings": {
    "chartTypeFullName": "Bar_Clustered_Horizontal"
}
*END
ENDSTYLE
END
-RUN

COMPOUND END

```



The output is:



## Date and Page/Slide Number

You can add the date and page numbers to both single and Compound Reports.

- ☐ For single reports, use the TABPAGENO feature and the associated attributes.
- ☐ For Compound Reports, add your data and page number to text objects. For more information, see [Text Formatting Markup Tags for a Text Object](#) on page 693.

## Text Formatting Markup Tags for a Text Object

**Note:** If your text contains any open caret characters (<), you must put a blank space after each open caret that is part of the text, for example, "< 250". If you do not, everything following the open caret will be interpreted as the start of a markup tag and will not display as text.

### Font Properties

The font tag supports three attributes: face, size, and color (where the color must be specified as the hexadecimal number code for the color):

```
<font face="font" size=[+|-]n color=color_code>text</font>
```

For example:

```
<font face="New Century Schoolbook">Test1</font>  
<font face="Times" size=12>test2</font>  
<font face="Times New Roman" color=#0000FF size=+4>Test3</font>  
<font size=-2 face="Times New Roman" color=#0000FF >Test4</font>
```

### Text Styles

The supported text styles are bold, italic, underline, and superscript:

Bold: `<b>text</b>`

Italic: `<i>text</i>`

Underline: `<u>text</u>`

Superscript: `<sup>text</sup>`

### Line Breaks

The line break tag after a portion of text begins the next portion of text on a new line. Note that there is no closing tag for a line break:

```
<br>
```

### Text Alignment

The alignment options pertain to wrapped text, as well as specified line breaks. Both horizontal justification and vertical alignment are supported.

#### Horizontal Justification

Left Justification:

```
<left>text</left>
```

Right Justification:

```
<right>text</right>
```

Center Justification:

```
<center>text</center>
```

Full Justification:

```
<full>text</full>
```

#### ❏ Vertical Alignment

Top Alignment:

```
<top>text</top>
```

Middle Alignment:

```
<mid>text</mid>
```

Bottom Alignment:

```
<bottom>text</bottom>
```

#### Unordered (Bullet) List

The unordered (ul) list tag encloses a bullet list. Each item is enclosed in a list item tag (li). The start tag and end tag for the list must each be on its own line. Each list item must start on a new line:

```
<ul>
<li>list item1</li>
<li>list item2</li>
.
.
.
</ul>
```

By default, the bullet type is disc. You can also specify circle or square:

```
<ul type=disc>
```

```
<ul type=circle>
```

```
<ul type=square>
```

**Ordered (Number or Letter) List**

The ordered (ol) list tag encloses a list in which each item has a consecutive number or letter. Each item is enclosed in a list item tag (li). The start tag and end tag for the list must each be on its own line. Each list item must start on a new line:

```
<ol>
<li>list item1</li>
<li>list item2</li>
.
.
.
</ol>
```

By default, Arabic numerals (type=1) are used for the ordering of the list. You can specify the following types of order:

Arabic numerals (the default): `<ol type=1>`

Lowercase letters: `<ol type=a>`

Uppercase letters: `<ol type=A>`

Lowercase Roman numerals: `<ol type=I>`

Uppercase Roman numerals: `<ol type=I>`

**Hyperlinks**

Hyperlinks can be included within text markup in a PPTX output file.

The syntax for the anchor markup tag is a subset of the HTML anchor syntax:

```
<a href="hyperlink">Text to display</a>
```

where:

*hyperlink*

Is the hyperlink to jump to when the text is clicked.

*Text to display*

Is the text to display for the hyperlink.

For example:

```
<a href="http://www.example.com/help.htm">Click here for help</a>
```

No other attributes are supported in the anchor markup tag.

## Page Numbering

There are two pseudo-HTML tags for embedding page numbers in text on a Page Master for a Coordinated Compound Layout report:

Current page number: `<ibi-page-number/>`

Total number of pages: `<ibi-total-pages/>`

Note that when MARKUP=ON, space is allocated for the largest number of pages, so there may be a wide gap between the page number and the text that follows. To remove the extra space in the text object that has the page numbering tags:

- ❑ If specific styling of the text object is not required, do not insert markup tags, and turn MARKUP=OFF.

```
MARKUP=OFF, TEXT='Page <ibi-page-number/> of <ibi-total-pages/> of Sales
Report', $
```

This displays the following output:

`Page 1 of 100 of Sales Report`

- ❑ If specific styling of the text object is required, you must set MARKUP=ON. With MARKUP=ON, set WRAP=OFF and do not place any styling tags between the page number variables within the string. Tags can be used around the complete *Page n of m* string. The following code produces a page number string without the extra spaces:

```
MARKUP=ON, WRAP=OFF, TEXT='<font face="ARIAL" size=10><i>Page <ibi-page-
number/> of <ibi-total-pages/> of Sales Report </i>', $
```

This displays the following output:

*Page 1 of 100 of Sales Report*

## Dates

To display a date in the report output, insert a WebFOCUS date variable in a text object on a Page Master (such as &DATEtrMDYY) in the text object.

**Example: Formatting a Compound Layout Text Object With Markup Tags**

The following request displays a text object with markup tags in a PPTX output file.

**Important:** Text markup syntax cannot contain hidden carriage return or line feed characters. For purposes of presenting the example in this documentation, line feed characters have been added so that the sample code wraps to fit within the printed page. To run this example in your environment, copy the code into a text editor and delete any line feed characters within the text markup object by going to the end of each line and pressing the Delete key. In some instances, you may need to add a space to maintain the structure of the string. For additional information on displaying carriage returns within the text object see [Text Formatting Markup Tags for a Text Object](#) on page 693.

```
SET PAGE-NUM=OFF
SET LAYOUTGRID=ON
TABLE FILE GGSALES
BY REGION NOPRINT
ON TABLE PCHOLD AS LINESP1 FORMAT PPTX
ON TABLE SET STYLE *
type=report, size=8, $
object=string, position=(1 1), dimension=(7 3), wrap=on, markup=on,
  linespacing=multiple(3),
  text='<b><font face="Arial" size=12>This paragraph is triple-spaced
(LINESPACING=MULTIPLE(3)):</font></b>
<full>Our <i>primary</I> goal for fiscal 2006 was to accelerate our
transformation to customer centricity. In this letter, I'd like to
give you an update on this work, which contributed to the 22-percent
increase in earnings from continuing operations we garnered for fiscal
2006. Since the past is often prologue to the future, I'd like to
describe how customer centricity is influencing not only our goals for
fiscal 2007, but also our long-term plans. At Gotham Grinds, customer
centricity means treating each customer as a unique individual, meeting
their needs with end-to-end solutions, and engaging and energizing our
employees to serve them.</full>', $
ENDSTYLE
END
```

In this request:

- ☐ No fields from the data source are displayed. Only the text object displays on the output.
- ☐ The SET LAYOUTGRID command displays a grid to indicate the coordinates and dimensions of the text object.
- ☐ The OBJECT=STRING declaration specifies triple spacing: LINESPACING=MULTIPLE(3).
- ☐ The following text is displayed in boldface, in the Arial font face, and with a font size of 12 (the default is 8 from the TYPE=REPORT declaration):

'This paragraph is triple-spaced (LINESPACING=MULTIPLE(3)):'

The markup for this formatting is:

```
<b><font face="Arial" size=12>This paragraph is triple-spaced  
(LINESPACING=MULTIPLE(3)):</font></b>
```

Note, however, that the image has been resized to fit the page so the font may appear smaller:

- ❑ The remainder of the text is displayed with full justification (left and right sides align):

<full>Our ... </full>

- ❑ The following markup displays the text ‘primary’ in italics:

*primary*

The output is:

**This paragraph is triple-spaced (LINESPACING=MULTIPLE(3)):** Our primary goal for fiscal 2006 was to accelerate our transformation to customer centricity. In this letter, I'd like to give you an update on this work, which contributed to the 22-percent increase in earnings from continuing operations we garnered for fiscal 2006. Since the past is often prologue to the future, I'd like to describe how customer centricity is influencing not only our goals for fiscal 2007, but also our long-term plans. At Gotham Grinds, customer centricity means treating each customer as a unique individual, meeting their needs with end-to-end solutions, and engaging and energizing our employees to serve them.

**Example:**     **Drawing Text and Line objects on a Page Master**

The following request places a line on the Page Master between the header report and the component reports and places a line and a text string on the bottom of each page:

```
SET PAGE-NUM=OFF
SET SQUEEZE=ON
COMPOUND LAYOUT PCHOLD FORMAT PPTX
SECTION=S1, LAYOUT=ON, MERGE=ON, ORIENTATION=LANDSCAPE, $
PAGELAYOUT=ALL, $
COMPONENT=HEADER, TYPE=REPORT, POSITION=(1 1), DIMENSION=(4 4), $
OBJECT=STRING, POSITION=(1 6.6), MARKUP=ON,
TEXT='<font face="Arial" color=#0000FF size=12> Slide <ibi-page-number/>
      </font> ', WRAP=ON, DIMENSION=(4 4),$
OBJECT=LINE, POSITION=(1 2.5), ENDPOINT=(9.5 2.5),
      BORDER-COLOR=BLUE,$
OBJECT=LINE, POSITION=(1 6.5), ENDPOINT=(9.5 6.5),
      BORDER-COLOR=BLUE,$
PAGELAYOUT=1, $
COMPONENT=R1, TYPE=REPORT, POSITION=(1 3), DIMENSION=(4 4), $
COMPONENT=R2, TYPE=REPORT, POSITION=(6 3), DIMENSION=(4 4), $
PAGELAYOUT=2, $
COMPONENT=R3, TYPE=REPORT, POSITION=(4 3), DIMENSION=(4 4), $
END
```



```

SET COMPONENT=HEADER
TABLE FILE GGSales
" "
"Report package for <REGION"
BY REGION NOPRINT
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, SIZE=20, $
TYPE=REPORT, IMAGE=gglogo.gif, POSITION=(+.25 +.25), $
TYPE=HEADING, LINE=2, ITEM=1, POSITION=1.5, $
END
SET COMPONENT=R1
TABLE FILE GGSales
"Sales report for <REGION"
" "
SUM DOLLARS/F8M
BY REGION NOPRINT
BY ST
BY CITY
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, COLOR=RED, SQUEEZE=ON, $
END
SET COMPONENT=R2
TABLE FILE GGSales
"Number of unit sales per product for <REGION"
" "
SUM CNT.UNITS AS 'Number of units sold'
BY REGION NOPRINT
BY PRODUCT
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, COLOR=BLUE, SQUEEZE=ON, $
END
SET COMPONENT=R3
TABLE FILE GGSales
"Report R3 for <REGION"
BY REGION NOPRINT
SUM DOLLARS BY ST
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, COLOR=GREEN, $
END
COMPOUND END

```

The first page of output is:

The screenshot shows a PowerPoint slide titled "Report package for Midwest". The slide contains two tables. The first table, "Sales report for Midwest", lists sales data for three states. The second table, "Number of unit sales per product for Midwest", lists the number of units sold for various products.

State	City	Dollar Sales
IL	Chicago	\$3,924,401
MO	St. Louis	\$3,761,296
TX	Houston	\$3,714,979

Product	Number of units sold
Bacotti	99
Coffee Grinder	71
Coffee Pot	72
Crossant	144
Espresso	117
Latte	243
Mug	144
Scout	127
Thermos	72

Slide 1

The second page of output has the same drawing objects:

The screenshot shows a PowerPoint slide titled "Report package for Midwest". The slide contains a table titled "Report R3 for Midwest" which lists dollar sales for three states. The slide is labeled "Slide 2" at the bottom.

State	Dollar Sales
IL	3924401
MO	3761296
TX	3714979

Slide 2

**Example: Vertically Aligning Text Markup in PPTX Report Output**

The following request creates three boxes and places a text string object within each of them:

- ☐ In the left box, the text is aligned vertically at the top.
- ☐ In the middle box, the text is aligned vertically at the middle.
- ☐ In the right box, the text is aligned vertically at the bottom.

**Important:** Text markup syntax cannot contain hidden carriage return or line feed characters. For purposes of presenting the example in this documentation, line feed characters have been added so that the sample code wraps to fit within the printed page. To run this example in your environment, copy the code into a text editor and delete any line feed characters within the text markup object by going to the end of each line and pressing the Delete key. In some instances, you may need to add a space to maintain the structure of the string. For additional information on displaying carriage returns within the text object see [Text Formatting Markup Tags for a Text Object](#) on page 693.

```
SET PAGE-NUM=OFF
TABLE FILE GGSales
BY REGION NOPRINT
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
type=report, font=arial, size=10, $
object=box, position=(1 1), dimension=(6 1), $
object=line, position=(3 1), endpoint=(3 2), $
object=line, position=(5 1), endpoint=(5 2), $
object=string, text='<top>Vertically aligned text within a text object
using top alignment.</top>', position=(1.05 1), dimension=(2 1),
linespacing=exact(.15), markup=on, wrap=on, $
object=string, text='<mid>Vertically aligned text within a text object
using middle alignment.</mid>', position=(3.05 1), dimension=(2 1),
linespacing=exact(.15), markup=on, wrap=on, $
object=string, text='<bottom>Vertically aligned text within a text object
using bottom alignment.</bottom>', position=(5.05 .9), dimension=(2
1),linespacing=exact(.15), markup=on, wrap=on, $
END
```

The output is:

Vertically aligned text within a text object using top alignment.	Vertically aligned text within a text object using middle alignment.	Vertically aligned text within a text object using bottom alignment.
---	--	--

## Display Unordered Lists With Bullets, Discs, Squares, and Circles

The unordered (ul) list tag encloses a bulleted list in which each item is marked by a bullet of a particular shape or design. Each item or point is enclosed in a list item tag (li). The start and end tags for each point may occupy one continuous line or be placed on different lines. When each list is placed on a new line, insert a backslash (\) after each closing tag.

### *Example:* Displaying Unordered Lists With Bullets, Squares, and Circles

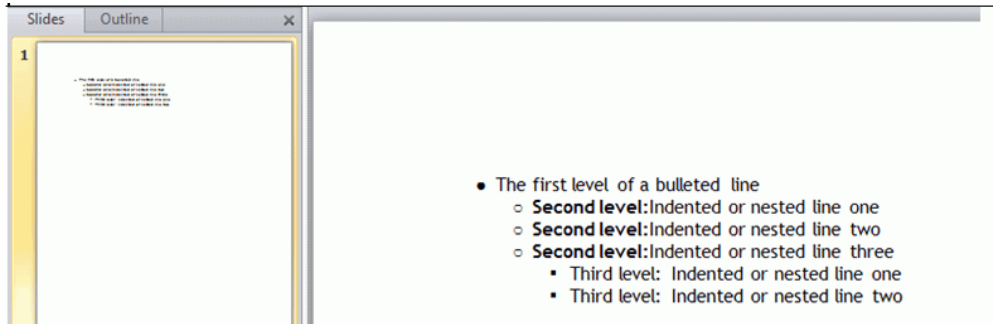
The following request displays a bulleted list with bullets of particular shapes and designs.

```
<ul>
<li>list item1</li>\
<li>list item2</li>\
</ul>

SET HTMLARCHIVE=ON
COMPOUND LAYOUT PCHOLD FORMAT PPTX
UNITS=IN, $
SECTION=section1, LAYOUT=ON, MERGE=OFF, ORIENTATION=PORTRAIT,
PAGE SIZE=Letter, SHOW_GLOBALFILTER=OFF, $
PAGELAYOUT=1, NAME='Page layout 1', text='Page layout 1',
BOTTOMMARGIN=0.5, TOPMARGIN=0.5, $
OBJECT=STRING, NAME='text1', TEXT='<font face="TREBUCHET MS" size=10>\
<UL>
<LI>The first level of a bulleted line </LI>\
<UL>
<LI><b>Second level:</b> Indented or nested line one</LI>\
<LI><b>Second level:</b> Indented or nested line two</LI>\
<LI><b>Second level:</b> Indented or nested line three</LI>\
<UL>\
<LI>Third level: Indented or nested line one</LI>\
<LI>Third level: Indented or nested line two</LI>\
</UL>\

<BR><BR><BR><U><DIV><BR></DIV></U></font>', POSITION=(0.938 0.938),
MARKUP=ON, WRAP=ON, DIMENSION=(6.563 4.167), $
COMPONENT='DfltCmpt1', POSITION=(0 0), DIMENSION=(0 0), $
END
SET COMPONENT='DfltCmpt1'
TABLE FILE SYSCOLUM
" "
SUM TBNAME NOPRINT
IF READLIMIT EQ 1
ON TABLE SET PREVIEW ON
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
END
COMPOUND END
```

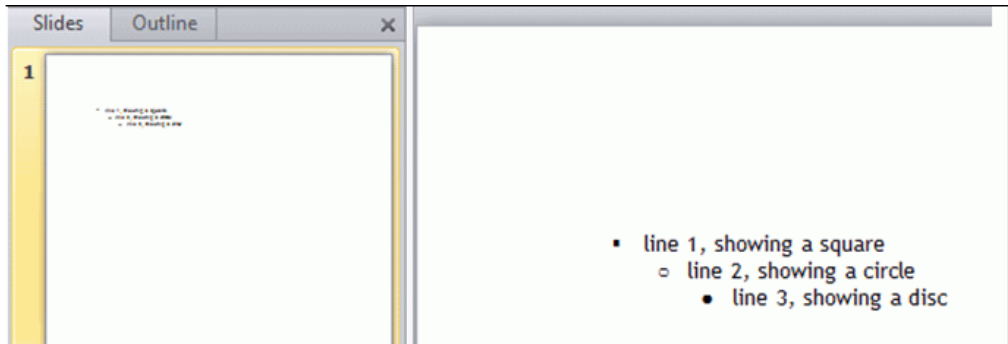
The output is:



The default bullet type is disc. You may also specify circle or square, as shown in the following request.

```
SET HTMLARCHIVE=ON
COMPOUND LAYOUT PCHOLD FORMAT PPTX
UNITS=IN, $
SECTION=section1, LAYOUT=ON, MERGE=OFF, ORIENTATION=PORTRAIT,
PAGESIZE=Letter, SHOW_GLOBALFILTER=OFF, $
PAGELAYOUT=1, NAME='Page layout 1', text='Page layout 1',
BOTTOMMARGIN=0.5, TOPMARGIN=0.5, $
OBJECT=STRING, NAME='text1', TEXT='<font face="TREBUCHET MS" size=10>\
<UL type=square>\
<LI> line 1, showing a square </LI>\
<UL type=circle>\
<LI> line 2, showing a circle </LI>\
<UL type=disc>\
<LI> line 3, showing a disc </LI>\
</UL>\
<BR><BR><BR><U><DIV><BR></DIV></U></font>', POSITION=(0.938 0.938),
MARKUP=ON, WRAP=ON, DIMENSION=(6.563 4.167), $
COMPONENT='DfltCmpt1', POSITION=(0 0), DIMENSION=(0 0), $
END
SET COMPONENT='DfltCmpt1'
TABLE FILE SYSCOLUM
" "
SUM TBNOME NOPRINT
IF READLIMIT EQ 1
ON TABLE SET PREVIEW ON
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
END
COMPOUND END
```

The output is:



### Inserting Images In Various Elements of PowerPoint PPTX Reports

WebFOCUS supports the placement of images within each element or node of the report. An image, such as a logo, gives corporate identity to a report, or provides visual appeal. Data specific images can be placed in headers and footers to provide additional clarity and style. The image must reside on the WebFOCUS Reporting Server in a directory named on EDAPATH or APPPATH. If the file is not on the search path, supply the full path name.

All images will be placed in the defined area, based on the explicit positioning defined by the POSITION attribute within the style sheet.

Images can be placed in any available WebFOCUS reporting node or element. Supported image formats include .gif, .jpg, and .png. Images may be positioned and resized by using the POSITION and SIZE attributes to set the x, y coordinates and height, width settings, respectively. Justification of images is not supported.

**Note:** The highest quality image format for charts is PNG, which allows for transparency, as well as better integration with the styling within slide backgrounds.

**Syntax:**      **How to Insert Images Into WebFOCUS PPTX Reports**

```
TYPE={REPORT|HEADING|data}, IMAGE={file|(column)}  
[,BY=byfield] [,SIZE=(w h)] , $
```

where:

**REPORT**

Embeds an image in the body of a report. The image appears in the background of the report. REPORT is the default value.

**HEADING**

Embeds an image in a heading or footing. Valid values are TABHEADING, TABFOOTING, FOOTING, HEADING, SUBHEAD, and SUBFOOT. Provide sufficient blank space in the heading or footing so that the image does not overlap the heading or footing text. You may also want to place heading or footing text to the right of the image using spot markers.

*data*

Defines a data column in which to place the image. Must be used with COLUMNS=*column title* to identify the specific report column where the image should be anchored.

*file*

Is the name of the image file. It must reside on the WebFOCUS Reporting Server in a directory named on EDAPATH or APPPATH. If the file is not on the search path, supply the full path name. When specifying a GIF file, you can omit the file extension.

**Example:**      **Inserting Images in the Headers and Footers of a Report**

The following request inserts images in the headers and footers of a report.

```
TABLE FILE EMPDATA
SUM
EMPDATA.EMPDATA.SALARY
BY LOWEST EMPDATA.EMPDATA.DEPT
BY EMPDATA.EMPDATA.LASTNAME
ON EMPDATA.EMPDATA.DEPT SUBFOOT "Subfoot"
" "
ON EMPDATA.EMPDATA.DEPT PAGE-BREAK
ON TABLE SUBHEAD
"Report Heading"
" "
HEADING
"Page Heading"
" "
FOOTING
"Page Footing"
" "
ON TABLE SUBFOOT "Report Footing"
" "
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
PAGESIZE='PPT Slide',
    ORIENTATION=LANDSCAPE,
$
TYPE=REPORT,
    OBJECT=STATUS-AREA,
    JUSTIFY=LEFT,
    PAGE-LOCATION=BOTTOM,
$
```



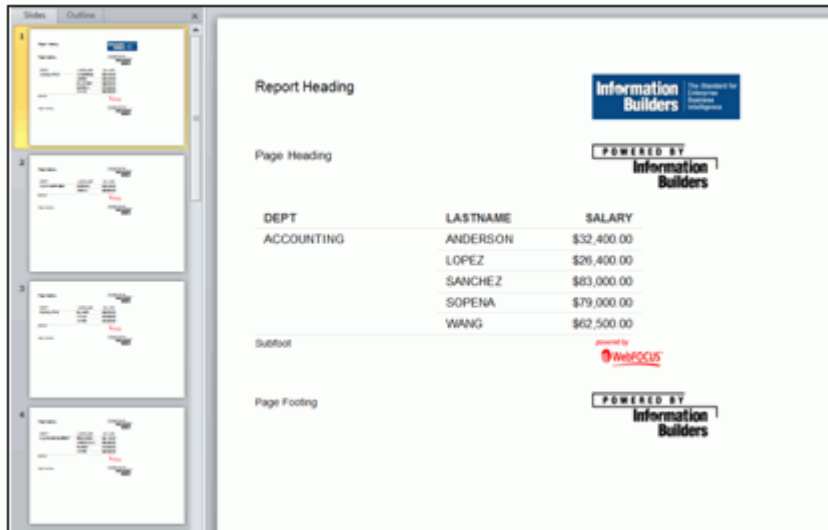
```

TYPE=REPORT,
  GRID=OFF,
  FONT='ARIAL',
  SIZE=12,
  STYLE=NORMAL,
  SQUEEZE=ON,
  TOPGAP=0.05,
  BOTTOMGAP=0.05,
  BORDER-COLOR=RGB(219 219 219),
  TITLELINE=SKIP,
  TOPMARGIN=.75,
  LEFTMARGIN=.5,
$
TYPE=TITLE,
  COLOR=RGB(51 51 51),
  STYLE=-UNDERLINE +BOLD,
$
TYPE=DATA,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
TYPE=SUBTOTAL,
  STYLE=BOLD,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
TYPE=TABHEADING,
  SIZE=14,
  JUSTIFY=LEFT,
$
TYPE=TABHEADING,
  IMAGE=smplogol.gif,
  POSITION=(+4.500000 +0.000000),
$
TYPE=TABFOOTING,
  SIZE=10,
$

```

```
TYPE=SUBHEAD,
  BACKCOLOR=RGB(246 246 246),
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
TYPE=SUBHEAD,
  BY=1,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(102 102 102),
$
TYPE=SUBHEAD,
  OBJECT=FIELD,
  STYLE=BOLD,
$
TYPE=SUBFOOT,
  SIZE=9,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
TYPE=TABFOOTING,
  IMAGE=smplogol.gif,
  POSITION=(+4.500000 +0.000000),
$
TYPE=HEADING,
  SIZE=12,
$
TYPE=HEADING,
  IMAGE=poweredbyibi.gif,
  POSITION=(+4.500000 +0.000000),
$
TYPE=FOOTING,
  SIZE=10,
$
TYPE=FOOTING,
  STYLE=BOLD,
  JUSTIFY=LEFT,
  IMAGE=poweredbyibi.gif,
  POSITION=(+4.500000 +0.000000),
$
TYPE=SUBFOOT,
  SIZE=10,
$
TYPE=SUBFOOT,
  IMAGE=webfocus1.gif,
  POSITION=(+4.500000 +0.000000),
  SIZE=(1.000000 0.500000),
$
ENDSTYLE
END
```

The output is:



Report Heading

Page Heading

DEPT	LASTNAME	SALARY
ACCOUNTING	ANDERSON	\$32,400.00
	LOPEZ	\$26,400.00
	SANCHEZ	\$83,000.00
	SOPENA	\$79,000.00
	WANG	\$62,500.00

Subtotal

Page Footer

**Example:** Inserting Images in the Data Cells of a Report

The following request inserts images in the data cells of a report.

```

APP PATH IBISAMP IBIDEMO
TABLE FILE GGSales
SUM DOLLARS/D17M AS 'Revenue'
COMPUTE Surplus/A15 = IF DOLLARS GE 4000000 THEN 'g1.gif' ELSE 'r1.gif';
BY REGION
BY ST
ON TABLE SUBHEAD
"Current Year Revenue"
FOOTING
"Revenue in excess of 4 million"
"Revenue less than 4 million"
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET SQUEEZE ON
ON TABLE SET STYLE *
TYPE=REPORT,
    GRID=OFF,
    FONT='ARIAL',
    SIZE=14,
    STYLE=NORMAL,
    SQUEEZE=ON,
    TOPGAP=0.05,
    BOTTOMGAP=0.05,
    BORDER-COLOR=RGB(219 219 219),
    TITLELINE=SKIP,
    ORIENTATION = LANDSCAPE,TOPMARGIN=1,LEFTMARGIN=1,
$
TYPE=DATA, COLUMN=Surplus, IMAGE=(Surplus), SIZE=(.2 .2),$
TYPE=FOOTING,IMAGE=g1.gif, position=(.122 .055), SIZE=(.2 .2 ),$
TYPE=FOOTING,line=1,item=1,position=.5,$
TYPE=FOOTING,IMAGE='r1.gif', position=(.122 .33), SIZE=(.2 .2),$
TYPE=FOOTING,line=2,item=1, position=.5,$
TYPE=DATA,
    BORDER-TOP=LIGHT,
    BORDER-TOP-COLOR=RGB(219 219 219),
$

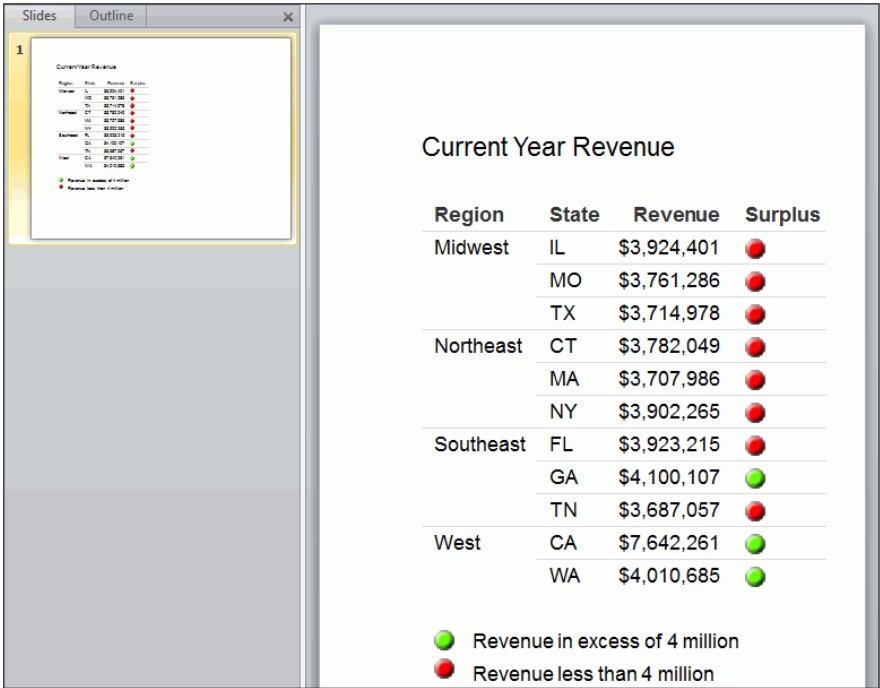
```

```

TYPE=TITLE,
    COLOR=RGB(51 51 51),
    STYLE=-UNDERLINE +BOLD,
$
TYPE=TABHEADING,
    SIZE=18,
    JUSTIFY=LEFT,
$
TYPE=TABFOOTING,
    SIZE=10,
$
TYPE=HEADING,
    JUSTIFY=LEFT,
    SIZE=12,
$
TYPE=SUBHEAD,
    BACKCOLOR=RGB(246 246 246),
    BORDER-TOP=LIGHT,
    BORDER-TOP-COLOR=RGB(219 219 219),
$
TYPE=SUBHEAD,
    BY=1,
    BORDER-TOP=LIGHT,
    BORDER-TOP-COLOR=RGB(102 102 102),
$
TYPE=SUBHEAD,
    OBJECT=FIELD,
    STYLE=BOLD,
$
TYPE=SUBFOOT,
    SIZE=9,
    BORDER-TOP=LIGHT,
    BORDER-TOP-COLOR=RGB(219 219 219),
$
END

```

The report output is as follows:



Displaying PPTX Charts in PNG Image Format

The PPTXGRAPHTYPE attribute enhances the quality of charts embedded into PowerPoint (PPTX) slides. As of Release 8.2.01M, you can use the PNG output format to enhance the image and text quality and support transparency.

This is useful for a number of important scenarios, including use of templates with background color and for overlapping a chart with other components and drawing objects.

*Syntax:*      How to Display PPTX Charts in PNG Image Format

```
SET PPTXGRAPHTYPE={ PNG | PNG_NOSCALE | JPEG }
```

where:

#### [PNG](#)

Scales the PNG image to twice its dimensions to get significantly improved quality. This may cause problems if you have non-scalable items in the chart, such as text with absolute point sizes (including embedded scales headings). The output file is also larger due to the larger bitmap. Text within the chart is noticeable sharper than the legacy JPEG format.

PNG preserves font sizes in the chart when it is internally rescaled for increased resolution. It converts absolute font sizes set in the stylesheet (\*GRAPH\_SCRIPT) to sizes expressed in virtual coordinates (which are relative to the dimensions of the chart) and generates font sizes for embedded headings and footings in virtual coordinates.

#### [PNG\\_NOSCALE](#)

Renders in PNG, but does not scale. This produces slightly better quality than JPEG. Going from JPEG to PNG\_NOSCALE makes the chart sharper, but has only a slight effect on the text.

#### [JPEG](#)

Indicates legacy format. This is the default value.

### **Example:** Displaying a PNG Chart With Transparency

Transparency enables greater control over how components and drawing objects can be placed together on a slide. The following report contains a text box, report, and chart that can be intertwined on the page because the background of the chart does not cover the contents of the other objects.

```
SET PPTXGRAPHPTYPE=PNG
```

```
COMPOUND LAYOUT PCHOLD FORMAT PPTX
UNITS=IN, $
SECTION=section1, LAYOUT=ON, MERGE=OFF, ORIENTATION=LANDSCAPE,
PAGESIZE=PPT Slide, $
PAGELAYOUT=1, NAME='Page layout 1', text='Page layout 1',
BOTTOMMARGIN=0.5, TOPMARGIN=0.5, $
COMPONENT='report1', TEXT='report1', POSITION=(5.088 1.375),
DIMENSION=(2.260 2.500), $
COMPONENT='chart2', TEXT='chart2', POSITION=(0.815 1.351),
DIMENSION=(5.104 2.917), COMPONENT-TYPE=GRAPH, $
OBJECT=STRING, NAME='text1', TEXT='<left>PNG charts can be defined with
transparency to allow the background to show through and allow for
overlapping components to optimize the use of space on the slide.
</left>',
POSITION=(0.500 0.979), MARKUP=ON, WRAP=ON, DIMENSION=(4.635 0.729),
font='TREBUCHET MS', color=RGB(0 0 0), size=10, $
END
```

```
SET COMPONENT='report1'
TABLE FILE GGSales
SUM
    GGSales.SALES01.DOLLARS
BY GGSales.SALES01.REGION
ACROSS LOWEST GGSales.SALES01.CATEGORY
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
$
ENDSTYLE
END
```

```
SET COMPONENT='chart2'
GRAPH FILE ggsales
SUM GGSales.SALES01.DOLLARS
BY GGSales.SALES01.CATEGORY
ACROSS GGSales.SALES01.REGION
ON GRAPH PCHOLD FORMAT HTML
ON GRAPH SET VZERO OFF
ON GRAPH SET HTMLENCODE ON
ON GRAPH SET GRAPHDEFAULT OFF
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET GRWIDTH 1
ON GRAPH SET UNITS 'PIXELS'
ON GRAPH SET HAXIS 770.0
ON GRAPH SET VAXIS 405.0
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 0
ON GRAPH SET GRLEGEND 1
ON GRAPH SET GRXAXIS 1
ON GRAPH SET LOOKGRAPH HBAR
ON GRAPH SET STYLE *
```



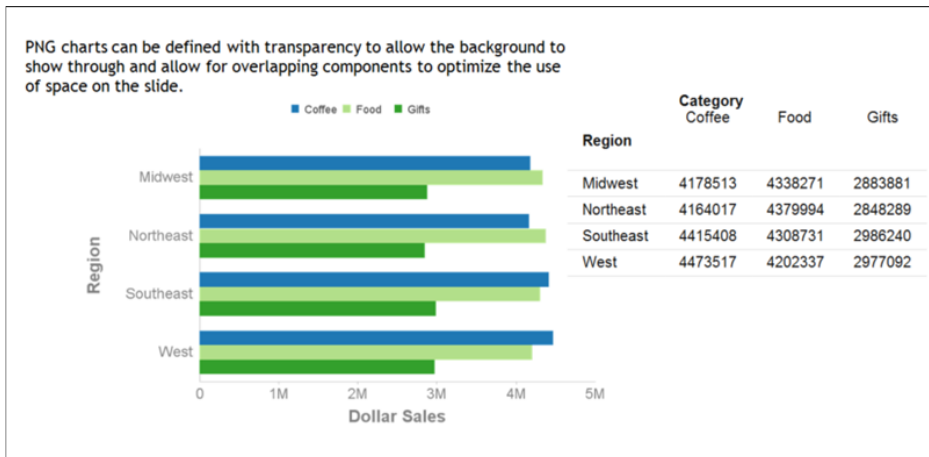
```

*GRAPH_SCRIPT
setPieDepth(0);
setPieTilt(0);
setDepthRadius(0);
setPlace(true);
setCurveFitEquationDisplay(false);
*END
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, TITLETEXT='WebFOCUS Report', $
*GRAPH_SCRIPT
setFillColor(getChartBackground(),new Color(255,255,255,0));
setLegendPosition(4);
*GRAPH_JS_FINAL
"blaProperties": {
  "orientation": "horizontal"
},
"agnosticSettings": {
  "chartTypeFullName": "Bar_Clustered_Horizontal"
}
*END
ENDSTYLE
END
-RUN

COMPOUND END

```

The output is:



**Example:** Displaying a PNG Image With Transparency in a Designated Template

The following compound report places a chart defined with transparency on a slide from the designated template that contains background colors and patterns.

```
SET PPTXGRAPHTYPE=PNG

COMPOUND LAYOUT PCHOLD FORMAT PPTX
UNITS=IN, $
SECTION=section1, LAYOUT=ON, MERGE=OFF, ORIENTATION=LANDSCAPE,
PAGESIZE=Letter, $
PAGELAYOUT=1, NAME='Page layout 1', text='Page layout 1', $
COMPONENT='ppt_template', $
COMPONENT='chart1', TEXT='chart1', POSITION=(0.500 2.10),
DIMENSION=(9.336 3.437),
COMPONENT-TYPE=GRAPH, ARREPORTSIZE=DIMENSION, $
END

SET COMPONENT='ppt_template'
TABLE FILE SYSCOLUM
SUM TBNAME NOPRINT
IF READLIMIT EQ 1
ON TABLE PCHOLD FORMAT PPTX TEMPLATE 'golden.potx' SLIDENUMBER 1
END

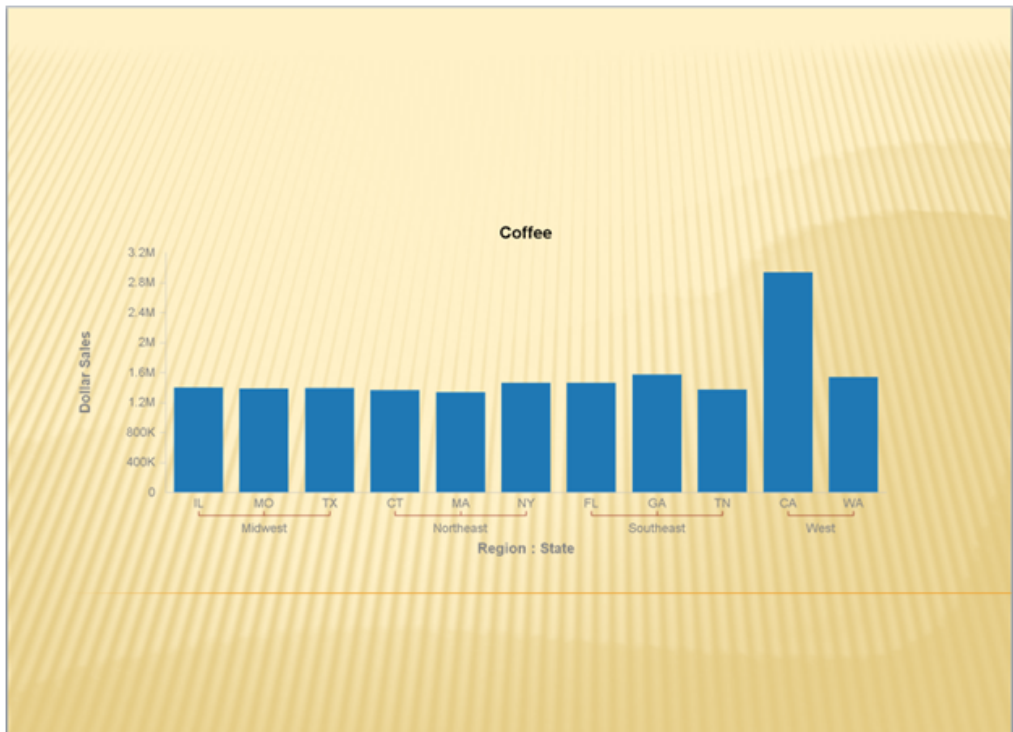
SET COMPONENT='chart1'
GRAPH FILE ggsales
SUM GGSales.SALES01.DOLLARS
BY GGSales.SALES01.CATEGORY
BY GGSales.SALES01.REGION
BY GGSales.SALES01.ST
ON GRAPH PCHOLD FORMAT HTML
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET GRWIDTH 1
ON GRAPH SET UNITS PIXELS
ON GRAPH SET HAXIS 770.0
ON GRAPH SET VAXIS 405.0
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 0
ON GRAPH SET GRXAXIS 2
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
```

```

*GRAPH_SCRIPT
setPieDepth(0);
setPieTilt(0);
setDepthRadius(0);
setCurveFitEquationDisplay(false);
setPlace(true);
*END
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, TITLETEXT='WebFOCUS Report', $
*GRAPH_SCRIPT
setFillType(getChartBackground(),2);
setGradientNumPins(getChartBackground(),2);
setFillColor(getChartBackground(),new Color(255,255,255,0));
setFillType(getChartBackground(),1);
*END
ENDSTYLE
END
-RUN
COMPOUND END

```

The output is:



## Drill Down From Microsoft PowerPoint

Two types of drill downs are supported:

- ☐ WebFOCUS content
- ☐ External URL

When working in the WebFOCUS Repository or Content environment, drill-down hyperlinks in PPTX reports will not work when Microsoft PowerPoint opens in a PowerPoint application window instead of in a browser. The current security context and any previously established session-related cookies are not retained and this changes user authorization. The recommendation is to configure one of the three security models described below, to allow successful drill down from reports displayed in a Microsoft PowerPoint application.

### The Remember Me Security Model

The Remember Me Security model is a method of user authentication that enables WebFOCUS to store a trusted sign-in cookie locally, on the workstation, for a default period of 14 days. WebFOCUS does not however, store the user password in the sign-in cookie. Enable the Remember Me feature on the Sign-in page. If the end-user uses the Remember Me feature, a persistent cookie is used.

### Public Access

Public Access is useful for procedures that are available to everyone within an organization or the general public and do not require authentication. Set up WebFOCUS security such that the PUBLIC user has the necessary permissions to drill down to reports.

### Integrated Windows Authentication

Integrated Windows Authentication (IWA) is enabled by configuring the browser. Use SSO with IIS/Tomcat Integrated Windows Authentication. Renegotiation occurs automatically and the PowerPoint formatted reports display correctly.

Refer to the *WebFOCUS Security and Administration* manual for details on these authentication models.

### **Example:** Drilling Down to an External URL

The following request places a reference to an external URL on the grand total line tag.

```
TABLE FILE GGSALES
SUM
    GGSALES.SALES01.BUDDOLLARS/D12CM
    GGSALES.SALES01.DOLLARS/D12CM
BY GGSALES.SALES01.REGION
BY GGSALES.SALES01.CATEGORY
BY GGSALES.SALES01.PRODUCT
HEADING
"REVENUE BY REGION "
ON GGSALES.SALES01.REGION SUBTOTAL AS '*TOTAL'
WHERE GGSALES.SALES01.CATEGORY NE 'Gifts';
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE COLUMN-TOTAL AS 'TOTAL'
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
PAGESIZE='PPT Slide',
    ORIENTATION=LANDSCAPE,
$
```

```
TYPE=REPORT,
  OBJECT=STATUS-AREA,
  JUSTIFY=LEFT,
  PAGE-LOCATION=BOTTOM,
$
TYPE=REPORT,
  GRID=OFF,
  FONT='ARIAL',
  SIZE=12,
  STYLE=NORMAL,
  SQUEEZE=ON,
  TOPGAP=0.05,
  BOTTOMGAP=0.05,
  BORDER-COLOR=RGB(219 219 219),
  TITLELINE=SKIP,
  TOPMARGIN=1,
  LEFTMARGIN=1,
$
TYPE=TITLE,
  COLOR=RGB(51 51 51),
  STYLE=-UNDERLINE +BOLD,
$
TYPE=DATA,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
TYPE=HEADING,
  JUSTIFY=LEFT,
  SIZE=14,
$
TYPE=SUBTOTAL,
  STYLE=BOLD,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
TYPE=GRANDTOTAL,
  OBJECT=TAG,
  URL=http://www.ibi.com,
$

TYPE=REPORT,
  OBJECT=STATUS-AREA,
  JUSTIFY=LEFT,
  PAGE-LOCATION=BOTTOM,
$
TYPE=GRANDTOTAL,
  COLOR=RGB(51 51 51),
  STYLE=BOLD,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(102 102 102),
$
ENDSTYLE
END
```

The output is:

Slides

Outline

1

REVENUE BY REGION

Region	Category	Product	Budget Dollars	Dollar Sales	
Southeast	Coffee	Capuccino	\$956,661	\$944,000	
		Espresso	\$849,465	\$853,572	
		Latte	\$2,625,303	\$2,617,836	
	Food	Biscotti	\$1,512,019	\$1,505,717	
		Croissant	\$1,969,906	\$1,902,359	
		Scone	\$927,363	\$900,655	
		<b>*TOTAL Southeast</b>			<b>\$8,840,717</b>
	West	Coffee	Capuccino	\$877,304	\$895,495
			Espresso	\$923,941	\$907,617
Latte			\$2,722,718	\$2,670,405	
Food		Biscotti	\$861,804	\$863,868	
		Croissant	\$2,406,554	\$2,425,601	
		Scone	\$914,886	\$912,868	
<b>*TOTAL West</b>			<b>\$8,707,207</b>	<b>\$8,675,854</b>	
<b><u>TOTAL REVENUE</u></b>			<b>\$34,561,046</b>	<b>\$34,460,788</b>	

2

REVENUE BY REGION

Region	Category	Product	Budget Dollars	Dollar Sales	
Southeast	Coffee	Capuccino	\$956,661	\$944,000	
		Espresso	\$849,465	\$853,572	
		Latte	\$2,625,303	\$2,617,836	
	Food	Biscotti	\$1,512,019	\$1,505,717	
		Croissant	\$1,969,906	\$1,902,359	
		Scone	\$927,363	\$900,655	
		<b>*TOTAL Southeast</b>			<b>\$8,840,717</b>
	West	Coffee	Capuccino	\$877,304	\$895,495
			Espresso	\$923,941	\$907,617
Latte			\$2,722,718	\$2,670,405	
Food		Biscotti	\$861,804	\$863,868	
		Croissant	\$2,406,554	\$2,425,601	
		Scone	\$914,886	\$912,868	
<b>*TOTAL West</b>			<b>\$8,707,207</b>	<b>\$8,675,854</b>	
<b><u>TOTAL REVENUE</u></b>			<b>\$34,561,046</b>	<b>\$34,460,788</b>	

## PowerPoint PPTX Presentations Using Templates

PPTX report output can be generated based on PowerPoint templates. This feature allows for the integration of WebFOCUS reports into presentations containing multiple slides. Any native PowerPoint template can be used to generate a new presentation containing a WebFOCUS report.

The following PowerPoint file types can be used as template files to generate PPTX presentations.

Template File Type	Presentation Output Generated
Template (.potx)	Presentation (.pptx)
Macro-Enabled Template (.potm)	Macro-Enabled presentation (.pptm)
Presentation (.pptx)	Presentation (.pptx)
Macro-Enabled presentation (.pptm)	Macro-Enabled presentation (.pptm)

**Note:** For more information on working with active content in macro-enabled templates, see the Microsoft webpage: <https://support.office.com/en-nz/article/Enable-or-disable-macros-in-Office-documents-7b4fdd2e-174f-47e2-9611-9efe4f860b12>

**Example:**    **Using Standard PowerPoint Templates (POTX)**

In the following request, the report occupies multiple slides. The designated slide is replaced by as many slides as are needed to display the report output.

```
TABLE FILE TRAINING
SUM
TRAINING.TRAINING.EXPENSES/D12CM
BY TRAINING.TRAINING.LOCATION
BY TRAINING.TRAINING.PIN
BY LOWEST TRAINING.TRAINING.COURSECODE
BY TRAINING.TRAINING.COURSESTART
BY TRAINING.TRAINING.GRADE
ON TRAINING.TRAINING.LOCATION SUBTOTAL AS 'TOTAL EXPENSES FOR'
ON TRAINING.TRAINING.LOCATION PAGE-BREAK
HEADING
"MONTHLY EXPENSES BY STATE"
" "
ON TABLE SET ASNAMES ON
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PPTX TEMPLATE 'ibi_template.potx' SLIDENUMBER 2
ON TABLE SET STYLE *
PAGESIZE='PPT Slide',
    ORIENTATION=LANDSCAPE,
$
TYPE=REPORT,
    OBJECT=STATUS-AREA,
    JUSTIFY=LEFT,
    PAGE-LOCATION=BOTTOM,
$
TYPE=REPORT,
    GRID=OFF,
    FONT='ARIAL',
    SIZE=14,
    STYLE=NORMAL,
    SQUEEZE=ON,
    TOPGAP=0.05,
    BOTTOMGAP=0.05,
    BORDER-COLOR=RGB(219 219 219),
    TITLELINE=SKIP,
    TOPMARGIN=.1,
    LEFTMARGIN=1.5,
$
```

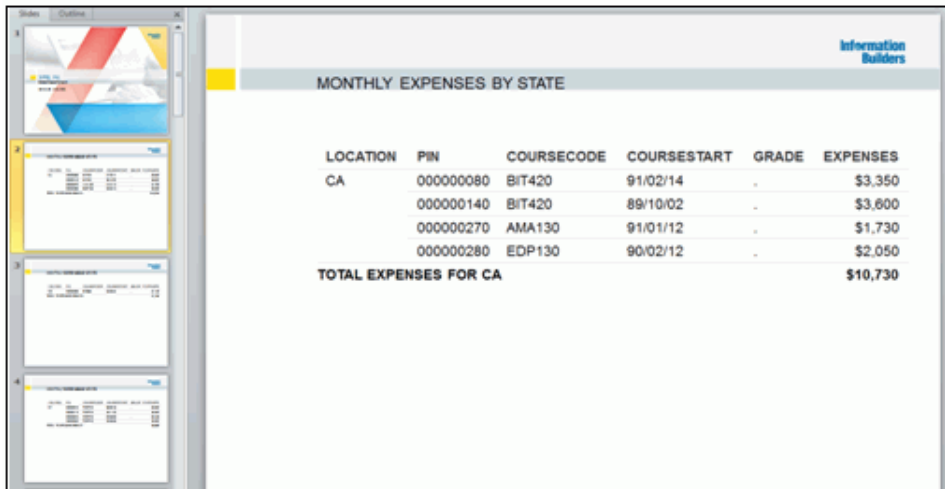


```

TYPE=TITLE,
  COLOR=RGB(51 51 51),
  STYLE=-UNDERLINE +BOLD,
$
TYPE=DATA,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
TYPE=HEADING,
  JUSTIFY=LEFT,
  SIZE=16,
$
TYPE=SUBTOTAL,
  STYLE=BOLD,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
ENDSTYLE
END

```

The output is:



LOCATION	PIN	COURSECODE	COURSESTART	GRADE	EXPENSES
CA	000000080	BIT420	91/02/14	-	\$3,350
	000000140	BIT420	89/10/02	-	\$3,600
	000000270	AMA130	91/01/12	-	\$1,730
	000000280	EDP130	90/02/12	-	\$2,050
<b>TOTAL EXPENSES FOR CA</b>					<b>\$10,730</b>

**Example:** Using a Multi-Report Request to Populate Designated Slides in a Template

The following request is a technique for inserting different components across multiple slides.

```
-* Replace Slide #2
TABLE FILE GGSales
HEADING
"FIRST SLIDE"
SUM
DOLLARS/D12CM UNITS
BY REGION AS 'My Field'
BY CATEGORY
ON TABLE COLUMN-TOTAL
ON TABLE HOLD AS SLIDE_A FORMAT PPTX TEMPLATE 'ibi_template.potx'
SLIDENUMBER 2
ON TABLE SET STYLE *
TYPE=REPORT,
    OBJECT=STATUS-AREA,
    JUSTIFY=LEFT,
    PAGE-LOCATION=BOTTOM,
$
TYPE=REPORT,
    GRID=OFF,
    FONT='ARIAL',
    SIZE=14,
    STYLE=NORMAL,
    SQUEEZE=ON,
    TOPGAP=0.05,
    BOTTOMGAP=0.05,
    BORDER-COLOR=RGB(219 219 219),
    TITLELINE=SKIP,
    ORIENTATION = LANDSCAPE,
    TOPMARGIN=.1,
    LEFTMARGIN=1.5,
$
TYPE=TITLE,
    COLOR=RGB(51 51 51),
    STYLE=-UNDERLINE +BOLD,
$
```

```

TYPE=DATA,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
TYPE=HEADING,
  JUSTIFY=LEFT,
  SIZE=16,
$
TYPE=DATA,
  COLUMN=DOLLARS,
  COLOR=BLUE,
$
TYPE=REPORT,
  COLUMN=REGION,
  COLOR=RED,
$
TYPE=REPORT,
  COLUMN=CATEGORY,
  COLOR=GREEN,
$
TYPE=GRANDTOTAL,
  COLOR=RGB(51 51 51),
  STYLE=BOLD,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(102 102 102),
$
END
-* Replace Slide #3
TABLE FILE GGSales HEADING
"SECOND SLIDE"
SUM DOLLARS/D12CM UNITS
BY REGION AS 'My Field'
BY CATEGORY
ON TABLE COLUMN-TOTAL
ON TABLE HOLD AS SLIDE_1 FORMAT PPTX TEMPLATE 'slide_a.pptx' SLIDENUMBER 3
ON TABLE COLUMN-TOTAL
ON TABLE SET STYLE *
$
TYPE=REPORT,
  OBJECT=STATUS-AREA,
  JUSTIFY=LEFT,
  PAGE-LOCATION=BOTTOM,
$

```

```
TYPE=REPORT,
  GRID=OFF,
  FONT='ARIAL',
  SIZE=14,
  STYLE=NORMAL,
  SQUEEZE=ON,
  TOPGAP=0.05,
  BOTTOMGAP=0.05,
  BORDER-COLOR=RGB(219 219 219),
  TITLELINE=SKIP,
  ORIENTATION = LANDSCAPE,
  TOPMARGIN=.1,
  LEFTMARGIN=1.5,
$
TYPE=TITLE,
  COLOR=RGB(51 51 51),
  STYLE=-UNDERLINE +BOLD,
$
TYPE=DATA,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
TYPE=HEADING,
  JUSTIFY=LEFT,
  SIZE=16,
$
TYPE=DATA,
  COLUMN=DOLLARS,
  COLOR=BLUE,
$
TYPE=REPORT,
  COLUMN=REGION,
  COLOR=RED,
$
TYPE=REPORT,
  COLUMN=CATEGORY,
  COLOR=GREEN,
$
TYPE=GRANDTOTAL,
  COLOR=RGB(51 51 51),
  STYLE=BOLD,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(102 102 102),
$
END
```

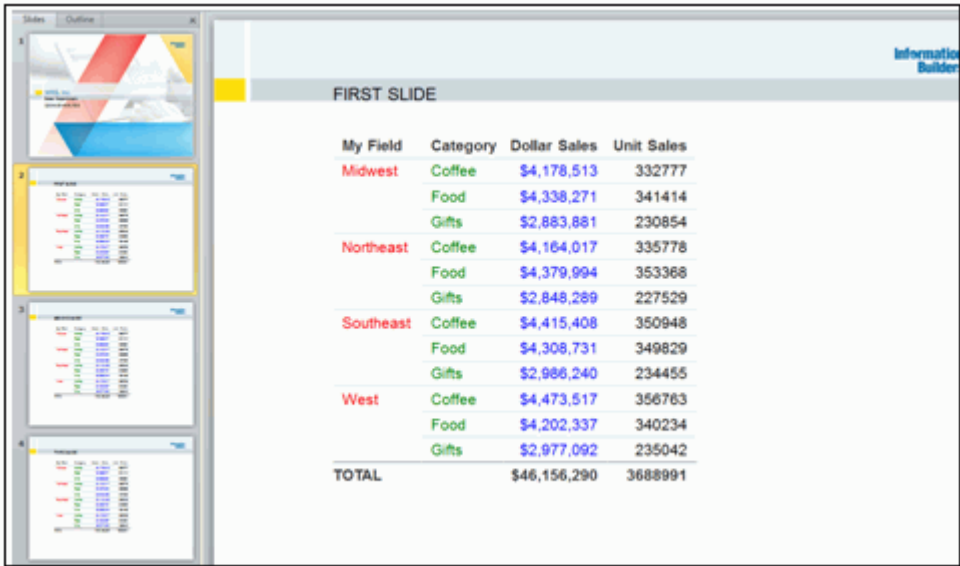
```

-* Replace Slide #4
TABLE FILE GGSALES
HEADING
"THIRD SLIDE"
SUM
DOLLARS/D12CM
UNITS
BY REGION AS 'My Field'
BY CATEGORY
ON TABLE COLUMN-TOTAL
ON TABLE PCHOLD AS THRID_SLIDE FORMAT PPTX TEMPLATE 'SLIDE_1.pptx'
SLIDENUMBER 4
ON TABLE SET STYLE *
TYPE=REPORT,
    OBJECT=STATUS-AREA,
    JUSTIFY=LEFT,
    PAGE-LOCATION=BOTTOM,
$
TYPE=REPORT,
    GRID=OFF,
    FONT='ARIAL',
    SIZE=14,
    STYLE=NORMAL,
    SQUEEZE=ON,
    TOPGAP=0.05,
    BOTTOMGAP=0.05,
    BORDER-COLOR=RGB(219 219 219),
    TITLELINE=SKIP,
    ORIENTATION = LANDSCAPE,
    TOPMARGIN=.1,
    LEFTMARGIN=1.5,
$
TYPE=TITLE,
    COLOR=RGB(51 51 51),
    STYLE=-UNDERLINE +BOLD,
$
TYPE=DATA,
    BORDER-TOP=LIGHT,
    BORDER-TOP-COLOR=RGB(219 219 219),
$
TYPE=HEADING,
    JUSTIFY=LEFT,
    SIZE=16,

```

```
$
TYPE=DATA,
  COLUMN=DOLLARS,
  COLOR=BLUE,
$
TYPE=REPORT,
  COLUMN=REGION,
  COLOR=RED,
$
TYPE=REPORT,
  COLUMN=CATEGORY,
  COLOR=GREEN,
$
TYPE=GRANDTOTAL,
  COLOR=RGB(51 51 51),
  STYLE=BOLD,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(102 102 102),
$
END
```

The output is:



PowerPoint PPTX Compound Syntax

PowerPoint Compound Documents generate presentations that may contain multiple slides. The components of a PowerPoint Compound Document can include standard tables and charts.

**Example: Generating a Compound Document**

The following request creates a slide deck, which presents the selected information in standard tables and charts.

```

SET HTMLARCHIVE=ON
COMPOUND LAYOUT PCHOLD FORMAT PPTX
UNITS=IN, $
SECTION=section1, LAYOUT=ON, MERGE=OFF,
ORIENTATION=LANDSCAPE, PAGESIZE=PPT Slide, SHOW_GLOBALFILTER=OFF, $
PAGELAYOUT=1, NAME='Page layout 1', text='Page layout 1',
BOTTOMMARGIN=0.2, TOPMARGIN=0.5, LEFTMARGIN=2.0, $
COMPONENT='chart1', TEXT='chart1', POSITION=(0.707 0.520),
DIMENSION=(8.750 2.917), COMPONENT-TYPE=GRAPH, $
COMPONENT='report1', TEXT='report1', POSITION=(0.500 3.542),
DIMENSION=(9.271 3.646), $
END
SET COMPONENT='chart1'
ENGINE INT CACHE SET ON
-DEFAULTH &WF_STYLE_UNITS='PIXELS';
-DEFAULTH &WF_STYLE_HEIGHT='405.0';
-DEFAULTH &WF_STYLE_WIDTH='770.0';
-DEFAULTH &WF_TITLE='WebFOCUS Report';
GRAPH FILE ibisamp/ggsales
SUM GGSales.SALES01.DOLLARS
BY GGSales.SALES01.REGION
BY TOTAL HIGHEST GGSales.SALES01.DOLLARS NOPRINT
BY GGSales.SALES01.ST
ON GRAPH PCHOLD FORMAT PPTX
ON GRAPH SET HTMLENCODE ON
ON GRAPH SET GRAPHDEFAULT OFF
ON GRAPH SET ARGRAPHENGINE JSCHART
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET VZERO OFF
ON GRAPH SET GRWIDTH 1

```

```

ON GRAPH SET UNITS &WF_STYLE_UNITS
ON GRAPH SET HAXIS &WF_STYLE_WIDTH
ON GRAPH SET VAXIS &WF_STYLE_HEIGHT
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 0
ON GRAPH SET GRXAXIS 2
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
setPieDepth(0);
setPieTilt(0);
setDepthRadius(0);
setCurveFitEquationDisplay(false);
setPlace(true);
setPieFeelerTextDisplay(1);
setUseSeriesShapes(true);
setMarkerSizeDefault(50);
*END
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/
warm.sty,$
TYPE=REPORT, TITLETEXT=&WF_TITLE.QUOTEDSTRING, $
*GRAPH_SCRIPT
setReportParsingErrors(false);
setSelectionEnableMove(false);
setFillType(getSeries(0),2);
setGradientPinLeftColor0(getSeries(0),new Color(0,127,192));
setGradientPinRightColor0(getSeries(0),new Color(0,127,192));
setGradientPinLeftColor2(getSeries(0),new Color(0,127,192));
setGradientPinRightColor2(getSeries(0),new Color(0,127,192));
setGradientPinLeftColor1(getSeries(0),new Color(0,64,128));
setGradientPinRightColor1(getSeries(0),new Color(0,64,128));
setGradientPinPosition0(getSeries(0),0.0);
setGradientPinPosition1(getSeries(0),1.0);
setPieTilt(0);
setPieDepth(0);
setDepthRadius(0);
setDepthAngle(0);
setFillType(getSeries(7),2);
setGradientPinPosition0(getSeries(7),0.0);
setGradientPinPosition1(getSeries(7),1.0);
setFillType(getSeries(9),2);
setGradientDirection(getSeries(9),16);
setGradientPinPosition0(getSeries(9),0.0);
setGradientPinPosition1(getSeries(9),1.0);
*END
ENDSTYLE
END
-RUN

```



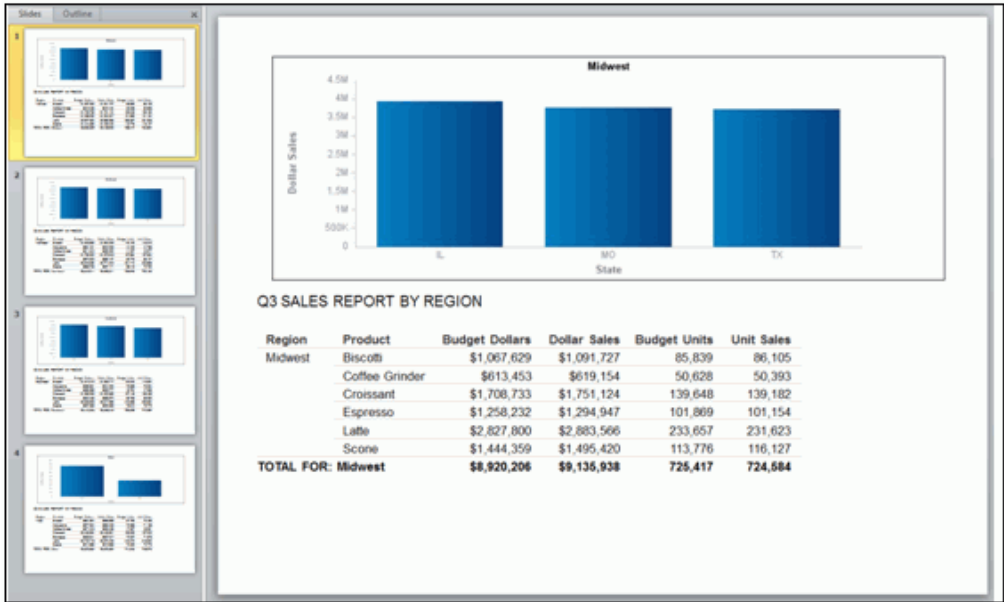
```

SET COMPONENT='report1'
TABLE FILE IBISAMP/GGSALES
SUM
    GGSALES.SALES01.BUDDOLLARS/D12CM
    GGSALES.SALES01.DOLLARS/D12CM
    GGSALES.SALES01.BUDUNITS/D12C
    GGSALES.SALES01.UNITS/D12C
BY GGSALES.SALES01.REGION
BY GGSALES.SALES01.PRODUCT
ON GGSALES.SALES01.REGION SUBTOTAL AS 'TOTAL FOR:'
ON GGSALES.SALES01.REGION PAGE-BREAK
HEADING
"Q3 SALES REPORT BY REGION"
WHERE GGSALES.SALES01.PRODUCT NE 'Coffee Pot' OR 'Mug' OR 'Thermos';
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE SET ASNAMES ON
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
    PAGESIZE='PPT Slide',
    LEFTMARGIN=1.000000,
    TOPMARGIN=0.500000,
    BOTTOMMARGIN=0.000000,
    SQUEEZE=ON,
    ORIENTATION=LANDSCAPE,
$
TYPE=REPORT,
    BORDER-TOP-COLOR=RGB(219 219 219),
    BORDER-BOTTOM-COLOR=RGB(219 219 219),
    BORDER-LEFT-COLOR=RGB(219 219 219),
    BORDER-RIGHT-COLOR=RGB(219 219 219),
    FONT='ARIAL',
    SIZE=12,
    TITLELINE=SKIP,
    STYLE=NORMAL,
    TOPGAP=0.041667,
$
TYPE=DATA,
    BORDER-TOP=LIGHT,
    BORDER-TOP-COLOR=RGB(219 219 219),
$

```

```
TYPE=TITLE,
  COLOR=RGB(51 51 51),
  STYLE=-UNDERLINE+BOLD,
$
TYPE=HEADING,
  SIZE=14,
  JUSTIFY=LEFT,
$
TYPE=SUBTOTAL,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
  STYLE=BOLD,
$
TYPE=REPORT,
  OBJECT=STATUS-AREA,
  JUSTIFY=LEFT,
  PAGE-LOCATION=BOTTOM,
$
ENDSTYLE
END
COMPOUND END
```

The resulting Compound Document output is:



## Coordinated Compound Layout Reports

A Coordinated Compound Layout report is coordinated so that all reports and graphs that contain a common sort field are burst into separate page layouts. Pages are generated for each value of the common sort field, with every component displaying the data it retrieved for that value on that page. You create a Coordinated Compound Layout report by specifying `MERGE=ON` in the `SECTION` declaration for the Compound Layout report.

In a Coordinated Compound Layout report, if at least one component contains data for a specific sort field value, a page is generated for that value even though some of the components may be missing.

While the length of the report will always include all of the rows of data generated by the query, the width of the report is limited by the size of the defined component container. This means that paneling is not supported for Compound Reports, although it is for non-Compound PPTX Reports.

If the width of the report data is wider than the defined page size, a panel (or horizontal overflow page) is automatically generated.

In legacy compound syntax, if one of the component reports is too large to fit within the defined page width, execution is halted and the user is presented with an error message stating that paneling is not supported.

In Compound Layout syntax, if a component is too wide to fit within the defined container, the report wraps the contents within the container. The container size is defined through a combination of the `POSITION` and `DIMENSIONS` parameters for the component within the compound syntax.

### ***Example:*** Generating a Coordinated Compound Layout Report

The following request generates a Coordinated Compound Layout report. This Compound Report is coordinated by Region, so that individual slides are generated for each value of the primary Key, Region.

```
SET HTMLARCHIVE=ON
*-HOLD_SOURCE
COMPOUND LAYOUT PCHOLD FORMAT PPTX
UNITS=IN, $
SECTION=section1, LAYOUT=ON, MERGE=ON, ORIENTATION=LANDSCAPE,
PAGESIZE=PPT Slide, SHOW_GLOBALFILTER=OFF,$
PAGELAYOUT=1, NAME='Page layout 1', text='Page layout 1', BOTTOMMARGIN=0.5,
TOPMARGIN=0.5, $
COMPONENT='report1', TEXT='report1', POSITION=(0.500 0.625),
DIMENSION=( * *), $
COMPONENT='report2', TEXT='report2', POSITION=(0.712 0.771),
DIMENSION=( * *), $
COMPONENT='report3', TEXT='report3', POSITION=(5.702 0.759),
DIMENSION=( * *), $
END
SET COMPONENT='report1'
TABLE FILE WF_RETAIL
BY WF_RETAIL.WF_RETAIL_GEOGRAPHY_CUSTOMER.BUSINESS_REGION NOPRINT
HEADING
"PROFIT REPORTS FOR <WF_RETAIL.WF_RETAIL_GEOGRAPHY_CUSTOMER.BUSINESS_REGION
<+15>
DISCOUNTS APPLIED: <WF_RETAIL.WF_RETAIL_GEOGRAPHY_CUSTOMER.BUSINESS_REGION"
" "
ON TABLE SET ASNAMES ON
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
```

```

TYPE=REPORT,
  GRID=OFF,
  FONT='ARIAL',
  SIZE=9,
  STYLE=NORMAL,
  SQUEEZE=ON,
  TOPGAP=0.05,
  BOTTOMGAP=0.05,
  PAGECOLOR='WHITE',
  BORDER-COLOR=RGB(219 219 219),
  TITLELINE=SKIP,
  TOPMARGIN=.5,
  LEFTMARGIN=.15,
$
TYPE=HEADING,
  JUSTIFY=LEFT,
  SIZE=14,
$
ENDSTYLE
END
SET COMPONENT='report2'
TABLE FILE WF_RETAIL
SUM
WF_RETAIL.WF_RETAIL_SALES.COGS_US
WF_RETAIL.WF_RETAIL_SALES.GROSS_PROFIT_US
BY WF_RETAIL.WF_RETAIL_GEOGRAPHY_CUSTOMER.BUSINESS_REGION NOPRINT
BY WF_RETAIL.WF_RETAIL_PRODUCT.PRODUCT_CATEGORY
ON WF_RETAIL.WF_RETAIL_GEOGRAPHY_CUSTOMER.BUSINESS_REGION SUBTOTAL AS
'TOTAL FOR'
ON TABLE SET ASNAMES ON
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
PAGESIZE='PPT Slide',
  ORIENTATION=LANDSCAPE,
$
TYPE=REPORT,
  OBJECT=STATUS-AREA,
  JUSTIFY=LEFT,
  PAGE-LOCATION=BOTTOM,
$

```

```

TYPE=REPORT,
  GRID=OFF,
  FONT='ARIAL',
  SIZE=14,
  STYLE=NORMAL,
  SQUEEZE=ON,
  TOPGAP=0.05,
  BOTTOMGAP=0.05,
  PAGECOLOR='WHITE',
  BORDER-COLOR=RGB(219 219 219),
  TITLELINE=SKIP,
    TOPMARGIN=.5,
    LEFTMARGIN=.15,
$
TYPE=DATA,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
TYPE=TITLE,
  COLOR=RGB(51 51 51),
  STYLE=-UNDERLINE +BOLD,
$
TYPE=DATA,
  COLUMN=ROWTOTAL(*),
  STYLE=BOLD,
$
TYPE=TITLE,
  COLUMN=ROWTOTAL(*),
  COLOR=RGB(51 51 51),
  STYLE=-UNDERLINE +BOLD,
$
TYPE=SUBTOTAL,
  STYLE=BOLD,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
ENDSTYLE
END
SET COMPONENT='report3'
TABLE FILE WF_RETAIL
SUM
WF_RETAIL.WF_RETAIL_SALES.DISCOUNT_US
BY WF_RETAIL.WF_RETAIL_GEOGRAPHY_CUSTOMER.BUSINESS_REGION NOPRINT
BY WF_RETAIL.WF_RETAIL_PRODUCT.PRODUCT_CATEGORY
ON WF_RETAIL.WF_RETAIL_GEOGRAPHY_CUSTOMER.BUSINESS_REGION SUBTOTAL
AS 'TOTAL FOR'

```

```

ON TABLE SET ASNAMES ON
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
PAGESIZE='PPT Slide',
  ORIENTATION=LANDSCAPE,
$
TYPE=REPORT,
  OBJECT=STATUS-AREA,
  JUSTIFY=LEFT,
  PAGE-LOCATION=BOTTOM,
$
TYPE=REPORT,
  GRID=OFF,
  FONT='ARIAL',
  SIZE=14,
  STYLE=NORMAL,
  SQUEEZE=ON,
  TOPGAP=0.05,
  BOTTOMGAP=0.05,
  PAGECOLOR='WHITE',
  BORDER-COLOR=RGB(219 219 219),
  TITLELINE=SKIP,
$
TYPE=DATA,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
TYPE=TITLE,
  COLOR=RGB(51 51 51),
  STYLE=-UNDERLINE +BOLD,
$
TYPE=DATA,
  COLUMN=ROWTOTAL(*),
  STYLE=BOLD,
$
TYPE=TITLE,
  COLUMN=ROWTOTAL(*),
  COLOR=RGB(51 51 51),
  STYLE=-UNDERLINE +BOLD,
$
TYPE=SUBTOTAL,
  STYLE=BOLD,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
ENDSTYLE
END
COMPOUND END

```

The output is:

Slides

Outline

1

REPORT: REPORT FOR EMEA  
Product Category  
Accessories  
Camcorder  
Computers  
Media Player  
Stereo Systems  
Televisions  
Video Production  
TOTAL FOR EMEA  
Cost of Goods  
\$143,987.00  
\$187,000.00  
\$47,616.00  
\$328,401.00  
\$361,756.00  
\$100,443.00  
\$78,722.00  
\$1,247,925.00  
Gross Profit  
\$67,490.99  
\$93,235.28  
\$30,624.81  
\$96,506.47  
\$152,933.93  
\$23,478.37  
\$33,419.72  
\$497,689.57

2

REPORT: REPORT FOR EMEA  
Product Category  
Accessories  
Camcorder  
Computers  
Media Player  
Stereo Systems  
Televisions  
Video Production  
TOTAL FOR EMEA  
Cost of Goods  
\$143,987.00  
\$187,000.00  
\$47,616.00  
\$328,401.00  
\$361,756.00  
\$100,443.00  
\$78,722.00  
\$1,247,925.00  
Gross Profit  
\$67,490.99  
\$93,235.28  
\$30,624.81  
\$96,506.47  
\$152,933.93  
\$23,478.37  
\$33,419.72  
\$497,689.57

3

REPORT: REPORT FOR EMEA  
Product Category  
Accessories  
Camcorder  
Computers  
Media Player  
Stereo Systems  
Televisions  
Video Production  
TOTAL FOR EMEA  
Cost of Goods  
\$143,987.00  
\$187,000.00  
\$47,616.00  
\$328,401.00  
\$361,756.00  
\$100,443.00  
\$78,722.00  
\$1,247,925.00  
Gross Profit  
\$67,490.99  
\$93,235.28  
\$30,624.81  
\$96,506.47  
\$152,933.93  
\$23,478.37  
\$33,419.72  
\$497,689.57

4

REPORT: REPORT FOR EMEA  
Product Category  
Accessories  
Camcorder  
Computers  
Media Player  
Stereo Systems  
Televisions  
Video Production  
TOTAL FOR EMEA  
Cost of Goods  
\$143,987.00  
\$187,000.00  
\$47,616.00  
\$328,401.00  
\$361,756.00  
\$100,443.00  
\$78,722.00  
\$1,247,925.00  
Gross Profit  
\$67,490.99  
\$93,235.28  
\$30,624.81  
\$96,506.47  
\$152,933.93  
\$23,478.37  
\$33,419.72  
\$497,689.57

## PROFIT REPORTS FOR EMEA

Product Category	Cost of Goods	Gross Profit
Accessories	\$143,987.00	\$67,490.99
Camcorder	\$187,000.00	\$93,235.28
Computers	\$47,616.00	\$30,624.81
Media Player	\$328,401.00	\$96,506.47
Stereo Systems	\$361,756.00	\$152,933.93
Televisions	\$100,443.00	\$23,478.37
Video Production	\$78,722.00	\$33,419.72
<b>TOTAL FOR EMEA</b>	<b>\$1,247,925.00</b>	<b>\$497,689.57</b>

## DISCOUNTS APPLIED: EMEA

Product Category	Discount
Accessories	\$8,533.01
Camcorder	\$8,898.26
Computers	\$3,549.30
Media Player	\$20,719.34
Stereo Systems	\$22,125.93
Televisions	\$8,307.24
Video Production	\$6,176.48
<b>TOTAL FOR EMEA</b>	<b>\$78,309.56</b>

Templates for Compound Reports

For Compound Reports, the template is defined in the first component report. For compound reports with Page Masters, this table will be the default table used to substantiate the page layout. For uncoordinated standard compound reports, this default component can be created using the system table, as shown in the following request.

```
SET COMPONENT='ppt_template'  
TABLE FILE SYSCOLUM  
SUM TBNAME NOPRINT  
IF READLIMIT EQ 1  
ON TABLE PCHOLD FORMAT PPTX TEMPLATE 'template_plus.potx' SLIDENUMBER 1  
END
```

For Coordinated Compound Reports, this table must contain the same primary key as the other components in the report. This includes any preprocessing of the data to define the universe of available primary key values including JOINS and DEFINES.

```
SET COMPONENT='ppt_template'  
TABLE FILE GGSales  
BY REGION NOPRINT  
IF READLIMIT EQ 1  
ON TABLE PCHOLD FORMAT PPTX TEMPLATE 'template_plus.potx' SLIDENUMBER 1  
END
```



## Adding Images to a Compound Request

Images may be inserted on the Page Master, Page Layout, and PowerPoint template (POTX) to enhance the Compound Document. Images inserted on the Page Master will be visible on every Page Layout within the Compound Document. Images on a Page Layout will be displayed only on that page. If the Document is to be displayed on a PowerPoint Template, images may be saved on the Template so that they will be displayed as positioned on the individual slides.

**Important:** Compound Layout syntax cannot contain hidden carriage return or line feed characters. For purposes of presenting this example, line feed characters have been added so that the sample code wraps to fit within the printed page. To run this example in your environment, copy the code into a text editor and delete any line feed characters within the Compound Layout syntax by going to the end of each line and pressing the Delete key.

### *Example:* Adding Images to a Compound Request

The following compound syntax creates a Document with images in the Page Master, the Page Layout, and that is displayed on a PowerPoint template on which an image has been inserted.

```
SET HTMLARCHIVE=ON
*-HOLD_SOURCE
COMPOUND LAYOUT PCHOLD FORMAT PPTX
UNITS=IN, $
SECTION=section1, LAYOUT=ON, MERGE=OFF,
ORIENTATION=LANDSCAPE, PAGESIZE=PPT Slide, SHOW_GLOBALFILTER=OFF, $
PAGELAYOUT=ALL, NAME='Page Master', $
COMPONENT='ppt_template', $
OBJECT=BOX, NAME='line1', POSITION=(0.052 0.500), DIMENSION=(10.000 0.031),
BACKCOLOR=RGB(176 196 222), BORDER-COLOR=RGB(176 196 222), $
OBJECT=BOX, NAME='line2', POSITION=(-12.000 -10.000), DIMENSION=(0.000
0.000), BACKCOLOR=BLACK, BORDER-COLOR=BLACK, $
OBJECT=BOX, NAME='line3', POSITION=(0.479 0.000), DIMENSION=(0.025 7.600),
BACKCOLOR=RGB(176 196 222), BORDER-COLOR=RGB(176 196 222), $
OBJECT=IMAGE, NAME='image1', IMAGE=webfocus1.gif, ALT='',
POSITION=(0.601 6.700), DIMENSION=(1.248 0.436), $
PAGELAYOUT=1, NAME='Page layout 1', text='Page layout 1',
BOTTOMMARGIN=0.15, TOPMARGIN=0.5, $
COMPONENT='report1', TEXT='report1', POSITION=(1.853 1.289),
DIMENSION=(6.458 5.417), $
OBJECT=IMAGE, NAME='image2', IMAGE=analyst_logo.gif, ALT='',
POSITION=(0.499 0.509), DIMENSION=(2.081 0.477), $
END
```

```

SET COMPONENT='ppt_template'
TABLE FILE SYSCOLUM
SUM TBNAME NOPRINT
IF READLIMIT EQ 1
ON TABLE PCHOLD FORMAT PPTX TEMPLATE '_ibi_template.potx' SLIDENUMBER 2
END
SET COMPONENT='report1'
TABLE FILE IBISAMP/EMPDATA
SUM
    EMPDATA.EMPDATA.SALARY
BY LOWEST EMPDATA.EMPDATA.DEPT
BY EMPDATA.EMPDATA.LASTNAME
ON EMPDATA.EMPDATA.DEPT SUBFOOT
"Subfoot"
" "
ON EMPDATA.EMPDATA.DEPT PAGE-BREAK
ON TABLE SUBHEAD
"Report Heading "
" "
" "
HEADING
"Page Heading "
" "
FOOTING
"Page Footing"
" "
ON TABLE SUBFOOT
"Report Footing"
WHERE EMPDATA.EMPDATA.SALARY GE 40900;
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE SET ASNAMES ON
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
    PAGESIZE='PPT Slide',
    ORIENTATION=LANDSCAPE,
$
TYPE=REPORT,
    OBJECT=STATUS-AREA,
    JUSTIFY=LEFT,
    PAGE-LOCATION=BOTTOM,
$

```

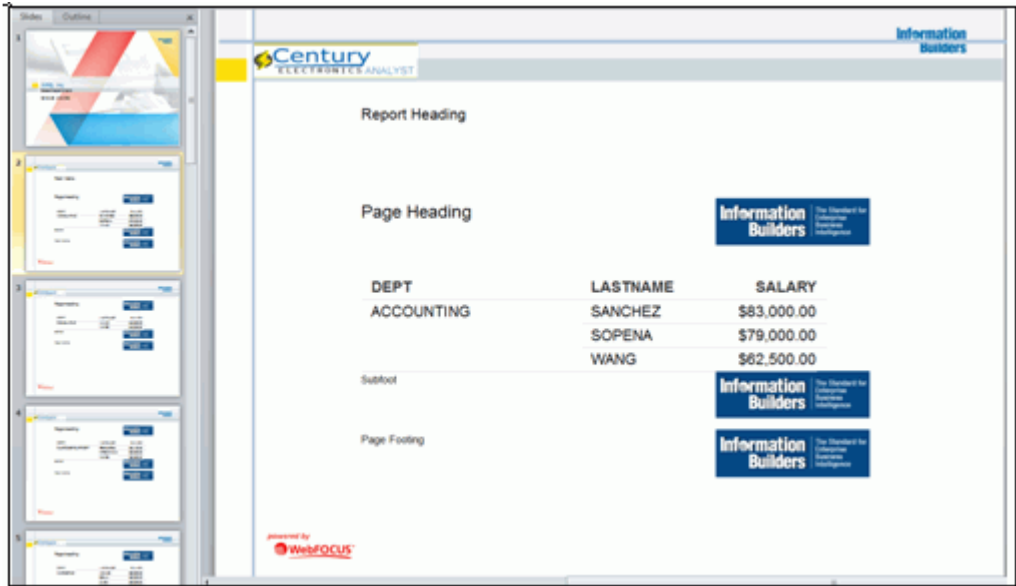
```

TYPE=REPORT,
  GRID=OFF,
  FONT='ARIAL',
  SIZE=14,
  STYLE=NORMAL,
  SQUEEZE=ON,
  TOPGAP=0.05,
  BOTTOMGAP=0.05,
  BORDER-COLOR=RGB(219 219 219),
  TITLELINE=SKIP,
  TOPMARGIN=.25,
  LEFTMARGIN=2,
  BOTTOMMARGIN=0.05,
$
TYPE=TITLE,
  COLOR=RGB(51 51 51),
  STYLE=-UNDERLINE +BOLD,
$
TYPE=DATA,
  BORDER-TOP=LIGHT,
  BORDER-TOP-COLOR=RGB(219 219 219),
$
TYPE=HEADING,
  JUSTIFY=LEFT,
  SIZE=16,
$
TYPE=HEADING,
  IMAGE=smplogol.gif,
  POSITION=(+4.500000 +0.000000),
$
TYPE=TABFOOTING,
  IMAGE=smplogol.gif,
  POSITION=(+4.500000 +0.000000),
$
TYPE=TABFOOTING,
  SIZE=10,
$
TYPE=FOOTING,
  SIZE=10,
$
TYPE=FOOTING,
  IMAGE=smplogol.gif,
  POSITION=(+4.500000 +0.000000),
$

```

```
TYPE=SUBFOOT,  
  SIZE=9,  
  BORDER-TOP=LIGHT,  
  BORDER-TOP-COLOR=RGB(219 219 219),  
$  
TYPE=SUBFOOT,  
  BY=1,  
  IMAGE=smplogo1.gif,  
  POSITION=(+4.500000 +0.000000),  
$  
ENDSTYLE  
END  
COMPOUND END
```

The output is:



## Template Masters and Slide Layouts

A Microsoft PPTX 2007 and higher template can contain one or more Slide Masters, defining a variety of different Slide Layouts.

A Slide Master is the top slide in a hierarchy of slides that stores information about the theme and Slide Layouts of a presentation, including the background, color, fonts, effects, placeholder sizes, and positioning.

You can incorporate two or more different styles or themes, such as backgrounds, color schemes, fonts, and effects, by inserting an individual Slide Master into the template for each different theme.

**Note:** Additional information on Microsoft PowerPoint Slide Layouts is available in an article titled [What is a slide layout?](#) on the Microsoft support site.

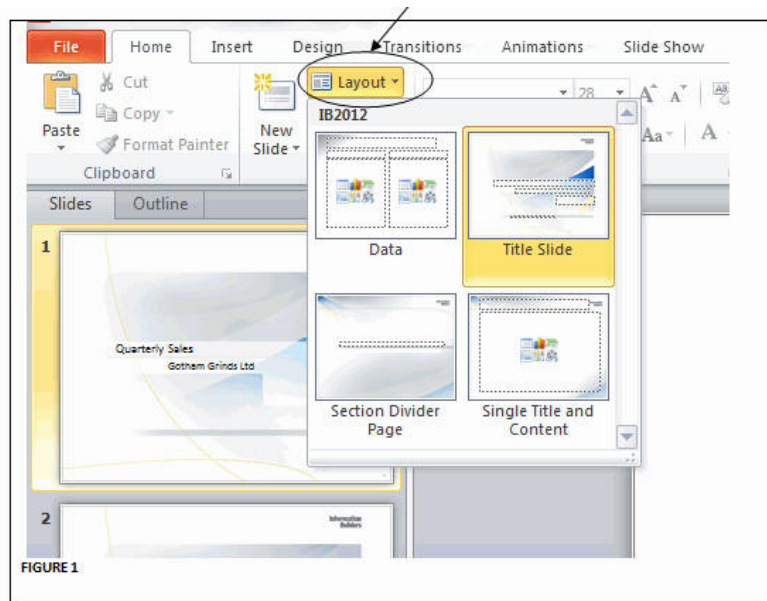
By default, the first Slide Layout in the first Slide Master is applied to slides on which WebFOCUS data is displayed.

With this new feature, WebFOCUS enables the developer to select any Slide Layout in any Slide Master in a PowerPoint template (POTX/POTM) or Presentation file (PPTX/PPTM). One Slide Layout may be applied to a slide or slides, displaying the output of a standard report, while one or different Slide Layouts may be applied to each Page Layout in a PPTX formatted Compound Document. The WebFOCUS generated output is placed on top of the styling on the selected Slide Layout.

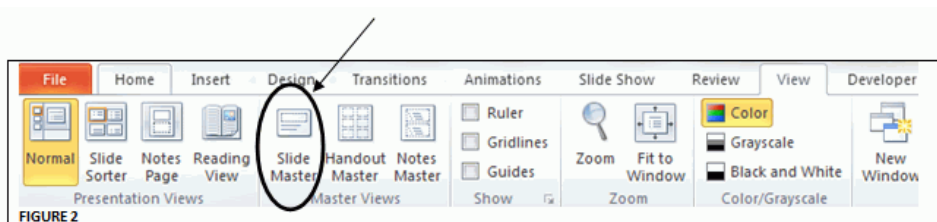
### Identifying Slide Master Attributes in PowerPoint

To identify Slide Masters and Slide Layouts in a template, open the file in the PowerPoint application, select the *Home* tab, and click the *Layout* button on the ribbon.

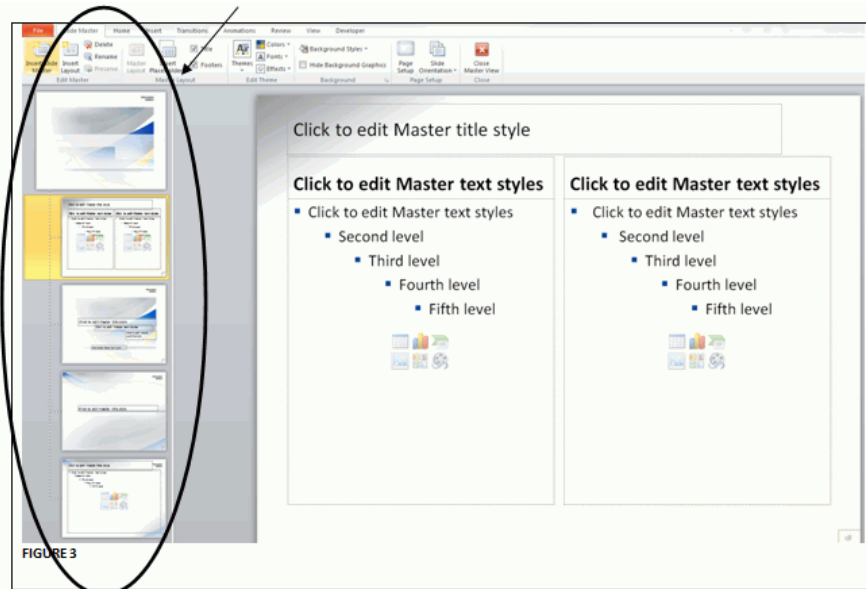
A context menu displays all Slide Layouts with labels, as shown in the following image.



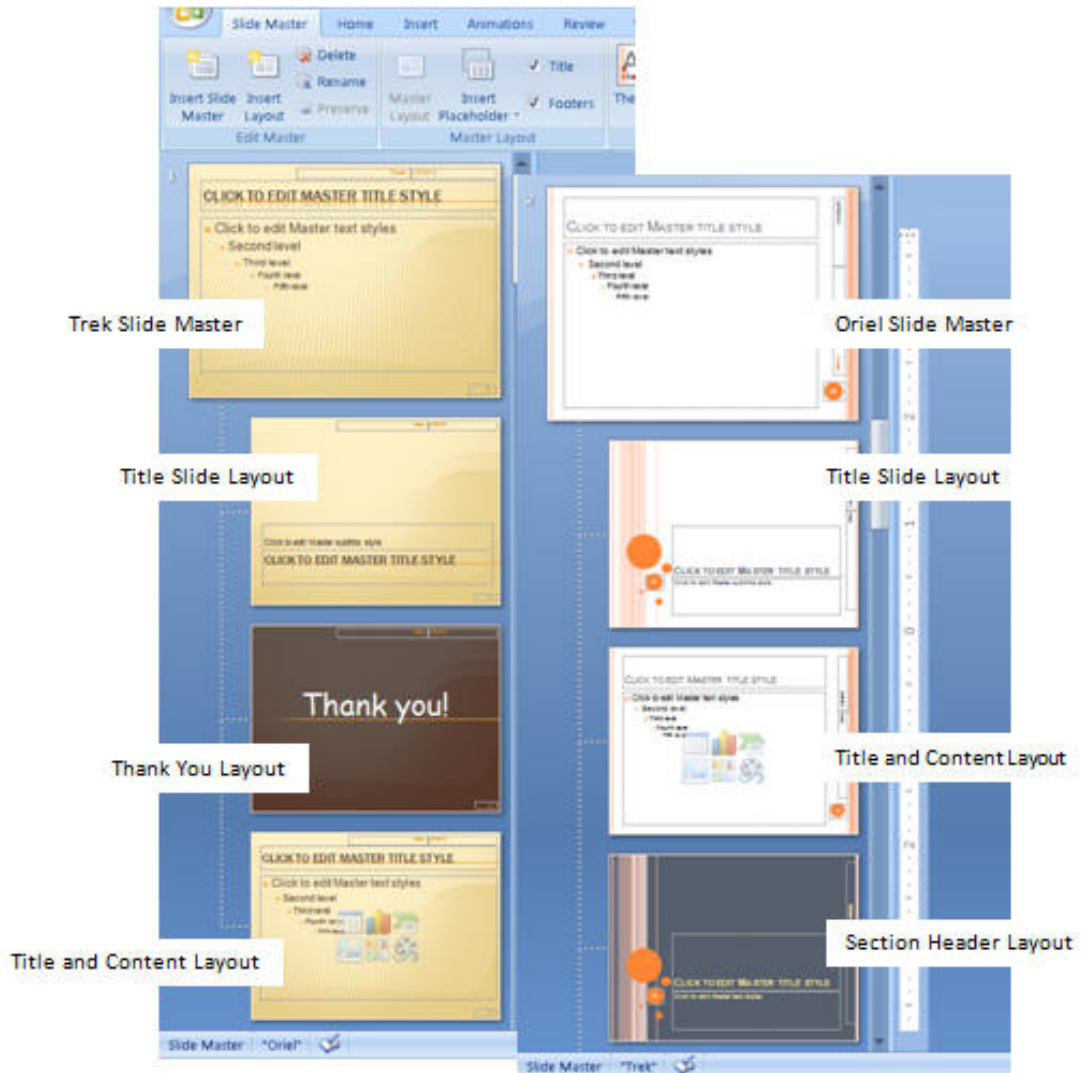
To view the Slide Master, select the *View* tab, and in the Master Views group on the ribbon, click the *Slide Master* button, as shown in the following image.



The Master View opens to show the Slide Master and its associated Slide Layouts, as shown in the following image.



The following image shows the Slide Master view of the template used in this example. It contains two Slide Masters: (1) Trek and (2) Oriel. Within each Master, the image displays the layouts selected for the report generation. Notice that the Slide Master and Layout names can be identified by hovering over the slide image. Use the name without the *Slide Master* or *Slide Layout* suffixes.





**Syntax:**      **How to Identify Slide Master Attributes in PowerPoint**

For single reports:

```
TYPE=REPORT,SLIDE-MASTER='slidemaster_name',SLIDE-LAYOUT='layout_name', $
```

For Compound syntax:

```
PAGELAYOUT=n, NAME='Page layout (n)', SLIDE-MASTER='slidemaster_name',  
SLIDE-LAYOUT='layout_name', $
```

**Note:** Slide Masters and Slide Layouts can be defined on the Page Master within the Section syntax or on any Page Layout.

**Example:**      **Compound Report Accessing Multiple Masters**

The following syntax creates a full PPTX presentation based on a template with two Slide Masters and four individual Slide Layouts. Each of the five individual procedures (.fex) shown in this example need to be copied to a separate file.

```
COMPOUND LAYOUT PCHOLD FORMAT PPTX  
UNITS=IN, $  
SECTION=section1, LAYOUT=ON, SLIDE-MASTER='TREK',  
SLIDE-LAYOUT='Title and Content',  
MERGE=OFF, ORIENTATION=LANDSCAPE,  
PAGESIZE=PPT Slide, SHOW_GLOBALFILTER=OFF, $  
PAGELAYOUT=1, NAME='Page layout 1', SLIDE-MASTER='TREK',  
SLIDE-LAYOUT='Title Slide', text='Page layout 1',  
BOTTOMMARGIN=0.5, TOPMARGIN=0.5, $  
COMPONENT='ppt_template', $  
OBJECT=STRING, NAME='text1', TEXT='Gotham Grinds Sales Summary',  
POSITION=(1.083 3.117), MARKUP=OFF, WRAP=ON, DIMENSION=(6.799 0.620),  
font='TREBUCHET MS', color=RGB(0 0 0), size=36, $  
OBJECT=STRING, NAME='text2', TEXT='Profit By Category',  
POSITION=(1.100 3.733), MARKUP=OFF, WRAP=ON, DIMENSION=(6.833 0.500),  
font='TREBUCHET MS', color=RGB(0 0 0), size=18, $  
OBJECT=STRING, NAME='text3', TEXT='Prepared By:<br><br>Anne T Jones,  
EVP of Sales<br>Joe F Smith, VP of Sales<br><br>&DATEtrMDYY',  
POSITION=(5.683 5.433), MARKUP=ON, WRAP=ON, DIMENSION=(4.049 1.721),  
font='TREBUCHET MS', color=RGB(0 0 0), size=18, $  
PAGELAYOUT=2, NAME='Page layout 2', text='Page layout 2',  
BOTTOMMARGIN=0.025, TOPMARGIN=4.8, $
```

```

OBJECT=STRING, NAME='pl2_text2', TEXT='Sales By Region',
POSITION=(0.5 0.325), MARKUP=ON, WRAP=ON, DIMENSION=(4.146 0.609),
font='TREBUCHET MS', color=RGB(0 0 0), style=bold, size=24, $
OBJECT=STRING, NAME='pl2_text3', TEXT=' ', POSITION=(0.5 .725), MARKUP=ON,
WRAP=ON, DIMENSION=(4.146 0.609), font='TREBUCHET MS', color=RGB(0 0 0),
style=bold, size=20, $
COMPONENT='report1', TEXT='report1', POSITION=(2.2 4.8),
DIMENSION=(* *), $
COMPONENT='chart1', TEXT='chart1', POSITION=(0.733 1.25),
DIMENSION=(8.680 3.10), COMPONENT-TYPE=GRAPH, $
PAGELAYOUT=5, NAME='Page layout 5', SLIDE-MASTER='ORIEL',
SLIDE-LAYOUT='Section Header', text='Page layout 3',
BOTTOMMARGIN=0.5, TOPMARGIN=0.5, $
OBJECT=STRING, NAME='pl3text1', TEXT='Sales Performance Regional
Breakdowns', POSITION=(1.0 3.35), MARKUP=OFF, WRAP=ON,
DIMENSION=(6.799 0.620), font='TREBUCHET MS', color=RGB(0 0 0), size=24,$
COMPONENT='DfltCmpt2_3', POSITION=(0 0), DIMENSION=(0 0), $
PAGELAYOUT=6, NAME='Page layout 2', text='Page layout 4',
SLIDE-MASTER='ORIEL', SLIDE-LAYOUT='Title and Content',
BOTTOMMARGIN=0.025, TOPMARGIN=4.8, $
OBJECT=STRING, NAME='pl4_text2', TEXT='Sales By Category',
POSITION=(0.5 0.325), MARKUP=ON, WRAP=ON, DIMENSION=(4.146 0.609),
font='TREBUCHET MS', color=RGB(0 0 0), style=bold, size=24, $
OBJECT=STRING, NAME='pl4_text3', TEXT=' ', POSITION=(0.5 .725), MARKUP=ON,
WRAP=ON, DIMENSION=(4.146 0.609), font='TREBUCHET MS', color=RGB(0 0 0),
style=bold, size=20, $
COMPONENT='report2_2', TEXT='report2', POSITION=(2.2 4.8),
DIMENSION=(* *), $
COMPONENT='chart2_2', TEXT='chart2', POSITION=(0.733 1.25),
DIMENSION=(8.680 3.10), COMPONENT-TYPE=GRAPH, $
COMPONENT='report2_3', TEXT='report3', POSITION=(0.5 .725),
DIMENSION=(3 3), $
PAGELAYOUT=9, NAME='Page layout 9', SLIDE-MASTER='TREK',
SLIDE-LAYOUT='Thank You', text='Page layout 3',
BOTTOMMARGIN=0.5, TOPMARGIN=0.5, $
COMPONENT='DfltCmpt9', POSITION=(0 0), DIMENSION=(0 0), $
END

SET COMPONENT='ppt_template'
TABLE FILE SYSCOLUM
SUM TBNAME NOPRINT
IF READLIMIT EQ 1
ON TABLE PCHOLD FORMAT PPTX TEMPLATE 'template_plus.potx' SLIDENUMBER 1
END

```

```

SET COMPONENT='report1'
-INCLUDE GG_RPT1

SET COMPONENT='chart1'
-INCLUDE GG_CHART1

SET COMPONENT='DfltCmpt2_3'
TABLE FILE SYSCOLUM
" "
SUM TBNAME NOPRINT
IF READLIMIT EQ 1
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
END

SET COMPONENT='report2_2'
-INCLUDE GG_RPT2
SET COMPONENT='chart2_2'
-INCLUDE GG_CHART2
SET COMPONENT='report2_3'
-INCLUDE GG_RPT3

SET COMPONENT='DfltCmpt9'
TABLE FILE SYSCOLUM
" "
SUM TBNAME NOPRINT
IF READLIMIT EQ 1
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
END

COMPOUND END

-*gg_rpt1.fex
TABLE FILE GGSales
SUM DOLLARS/D12CM BUDDOLLARS/D12CM UNITS/D12 BUDUNITS/D12
BY REGION
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
        INCLUDE = warm,
$
ENDSTYLE
END

```

**-\*gg\_rpt2.fex**

```
TABLE FILE GGSales
SUM DOLLARS/D12CM BUDDOLLARS/D12CM UNITS/D12 BUDUNITS/D12
BY REGION NOPRINT
BY CATEGORY
ON REGION PAGE-BREAK
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
        INCLUDE = warm,
$
ENDSTYLE
END
```

**-\*gg\_chart1.fex**

```
ENGINE INT CACHE SET ON
-DEFAULTH &WF_STYLE_UNITS='INCHES';
-DEFAULTH &WF_STYLE_HEIGHT='4.21875';
-DEFAULTH &WF_STYLE_WIDTH='8.020833';
-DEFAULTH &WF_TITLE='WebFOCUS Report';
GRAPH FILE GGSales
SUM DOLLARS UNITS
BY REGION NOPRINT
ON GRAPH PCHOLD FORMAT PPTX
ON GRAPH SET HTMLENCODE ON
ON GRAPH SET GRAPHDEFAULT OFF
ON GRAPH SET ARGPHENGIN JSCHART
ON GRAPH SET VZERO OFF
ON GRAPH SET GRWIDTH 1
ON GRAPH SET UNITS &WF_STYLE_UNITS
ON GRAPH SET HAXIS &WF_STYLE_WIDTH
ON GRAPH SET VAXIS &WF_STYLE_HEIGHT
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 0
ON GRAPH SET GRLEGEND 1
ON GRAPH SET GRXAXIS 0
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
setPieDepth(0);
setPieTilt(0);
setDepthRadius(0);
setCurveFitEquationDisplay(false);
setPlace(true);
*END
```

```

INCLUDE = warm,$
TYPE=REPORT, TITLETEXT=&WF_TITLE.QUOTEDSTRING, $
*GRAPH_SCRIPT
setReportParsingErrors(false);
setSelectionEnableMove(false);
setTransparentBorderColor(getChartBackground(),true);
setTransparentFillColor(getFrameSide(),true);
setTransparentBorderColor(getFrameSide(),true);
setTransparentFillColor(getFrameBottom(),true);
setTransparentBorderColor(getFrameBottom(),true);
*GRAPH_SCRIPT
-* Make the chart background, borders, etc. transparent:
setTransparentFillColor(getChartBackground(),true);
setFillColor(getChartBackground(),new Color(255,255,255,0));
setTransparentBorderColor(getChartBackground(),true);
setTransparentFillColor(getFrameSide(),true);
setTransparentBorderColor(getFrameSide(),true);
setTransparentFillColor(getFrameBottom(),true);
setTransparentBorderColor(getFrameBottom(),true);
setPlace(true);
*END
ENDSTYLE
END
-RUN

-*gg_chart2.fex
-DEFAULTH &WF_STYLE_UNITS='INCHES';
-DEFAULTH &WF_STYLE_HEIGHT='4.21875';
-DEFAULTH &WF_STYLE_WIDTH='8.020833';
-DEFAULTH &WF_TITLE='WebFOCUS Report';
GRAPH FILE GGSales
SUM DOLLARS UNITS
BY REGION NOPRINT
BY CATEGORY
ON GRAPH PCHOLD FORMAT PPTX
ON GRAPH SET HTMLENCODE ON
ON GRAPH SET GRAPHDEFAULT OFF
ON GRAPH SET ARGRAPHENGIN JSCHART
ON GRAPH SET VZERO OFF
ON GRAPH SET GRWIDTH 1
ON GRAPH SET UNITS &WF_STYLE_UNITS
ON GRAPH SET HAXIS &WF_STYLE_WIDTH
ON GRAPH SET VAXIS &WF_STYLE_HEIGHT
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 1
ON GRAPH SET GRXAXIS 0
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *

```

```

*GRAPH_SCRIPT
setPieDepth(0);
setPieTilt(0);
setDepthRadius(0);
setCurveFitEquationDisplay(false);
setPlace(true);
*END
INCLUDE = warm,$
TYPE=REPORT, TITLETEXT=&WF_TITLE.QUOTEDSTRING, $
*GRAPH_SCRIPT
setReportParsingErrors(false);
setSelectionEnableMove(false);
setTransparentBorderColor(getChartBackground(),true);
setTransparentFillColor(getFrameSide(),true);
setTransparentBorderColor(getFrameSide(),true);
setTransparentFillColor(getFrameBottom(),true);
setTransparentBorderColor(getFrameBottom(),true);
*GRAPH_SCRIPT
-* Make the chart background, borders, etc. transparent:
setTransparentFillColor(getChartBackground(),true);
setFillColor(getChartBackground(),new Color(255,255,255,0));
setTransparentBorderColor(getChartBackground(),true);
setTransparentFillColor(getFrameSide(),true);
setTransparentBorderColor(getFrameSide(),true);
setTransparentFillColor(getFrameBottom(),true);
setTransparentBorderColor(getFrameBottom(),true);
setPlace(true);
*END
ENDSTYLE
END
-RUN

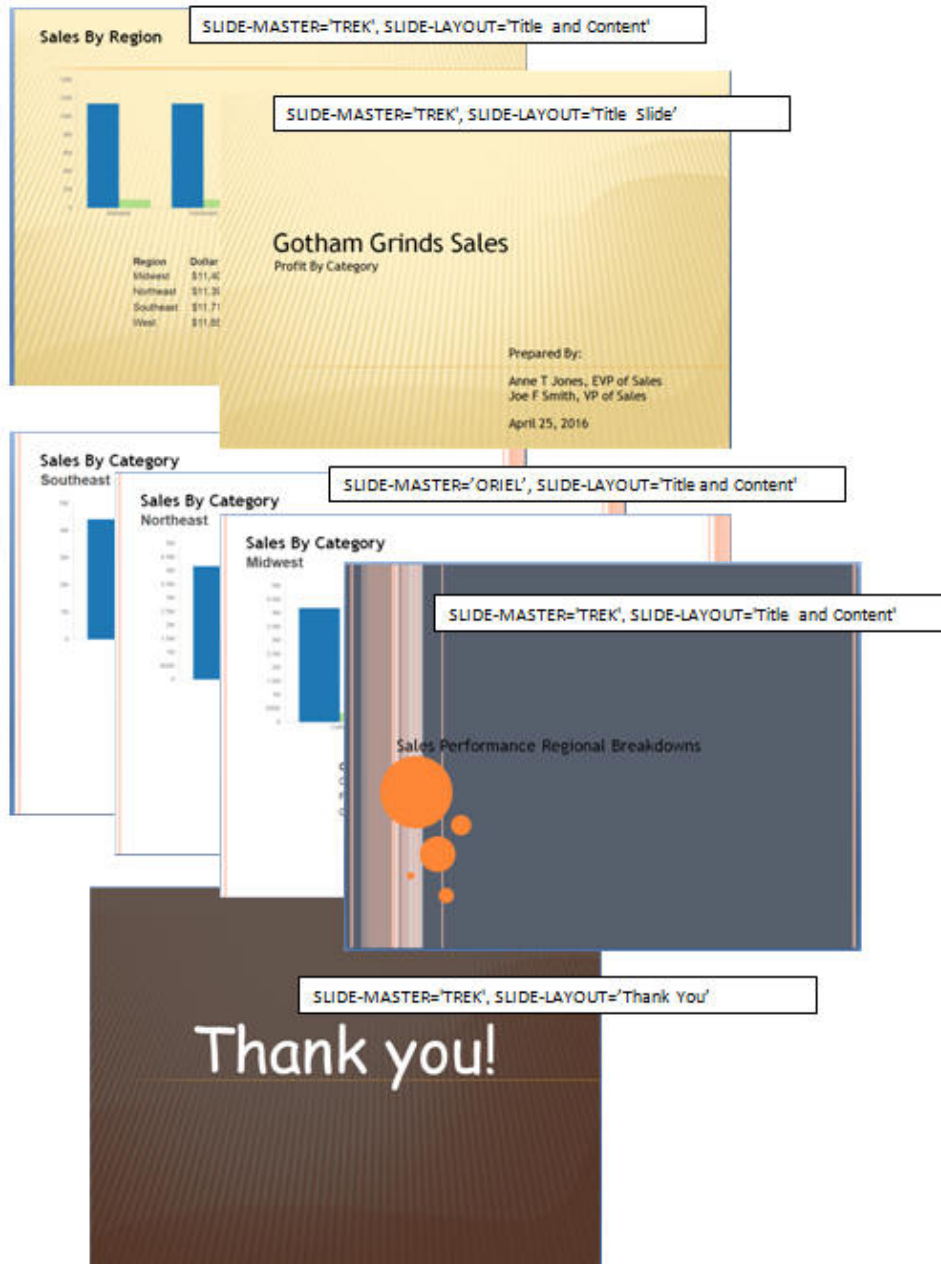
```

```

-*gg_rpt3.fex
TABLE FILE GGSales
BY REGION PAGE-BREAK NOPRINT
HEADING
"<REGION"
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
        INCLUDE = warm,
$
TYPE=HEADING, SIZE=20,
$
ENDSTYLE
END

```

The output is:



## WebFOCUS Pivot Support for XLSX

The WebFOCUS XLSX format can generate a workbook based on a template that contains predefined pivot tables. These pivot tables can be built based on data fed from a report and/or exist independently of the WebFOCUS data on other worksheets.

### **Example:** Feeding Data From a WebFOCUS Report Into a Pivot Table and Pivot Chart

The following sample procedure shows how to feed data from a WebFOCUS report into a pivot table and pivot chart within an existing Excel template called *wf2pivot.xltx*.

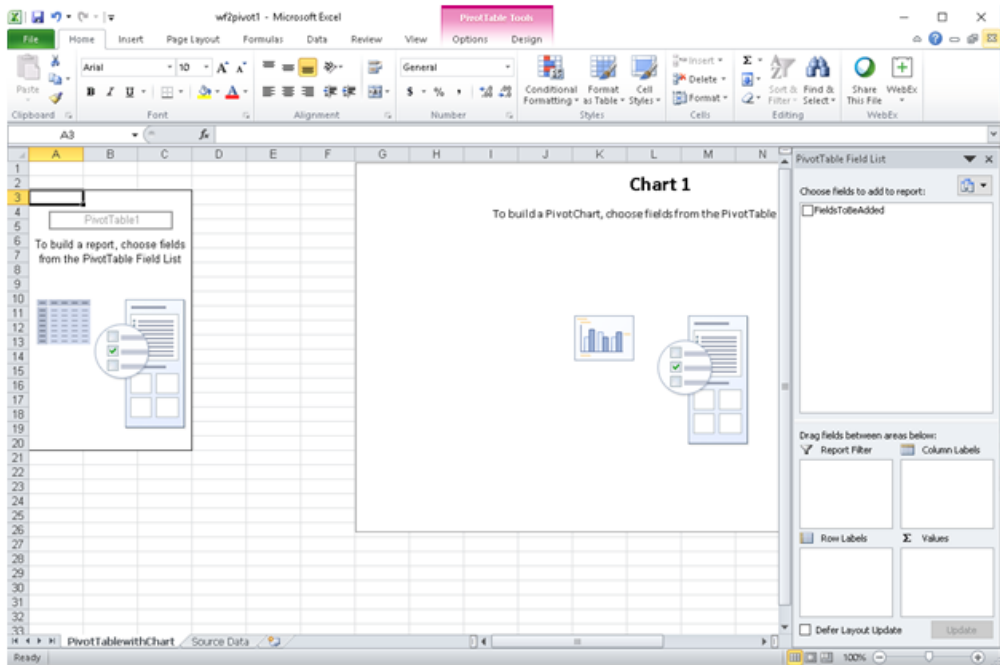
**Note:** Template names containing embedded blanks must be enclosed in single quotation marks.

```
TABLE FILE GGSALES
PRINT
UNITS/D12C DOLLARS/D12CM
BUDUNITS/D12C BUDDOLLARS/D12CM
BY LOWEST REGION
BY LOWEST ST
BY HIGHEST CATEGORY
BY LOWEST PRODUCT
ON TABLE SET BYDISPLAY ON
ON TABLE PCHOLD AS PIVOTWITHCHART FORMAT XLSX TEMPLATE wf2pivot.xltx
SHEETNUMBER 2
ON TABLE SET STYLE *
TYPE=DATA,IN-RANGES='DATAwithHEADERS',$,
TYPE=TITLE,IN-RANGES='DATAwithHEADERS',$,
ENDSTYLE
END
```

The wf2pivot.xltx template file must be in the Reporting Server path. The following images show the default of the first and second worksheets in the wf2pivot.xltx template, before executing the sample procedure.

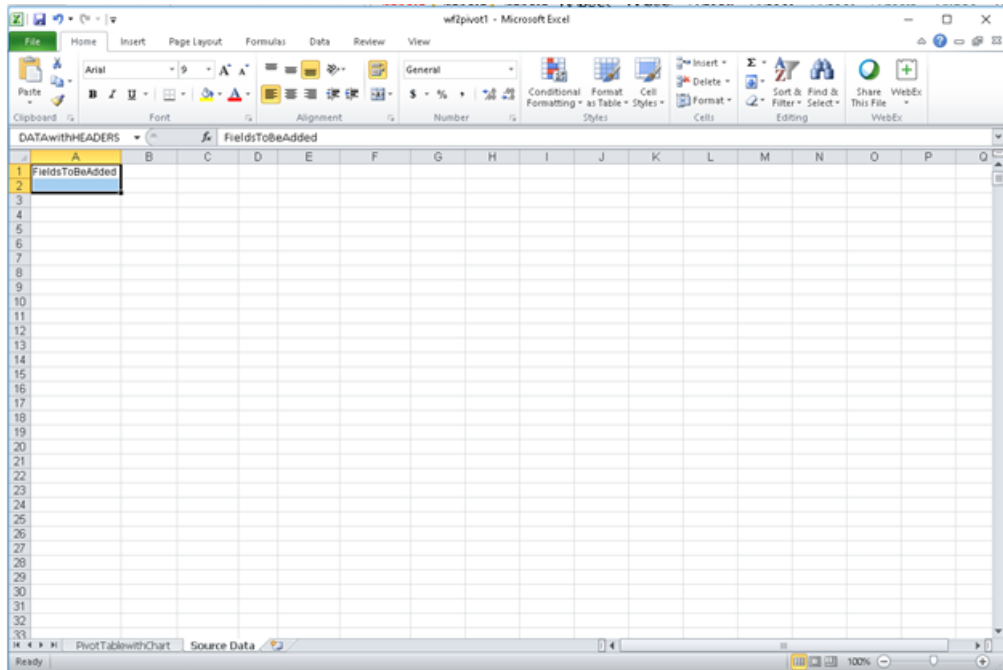


The first worksheet, PivotTablewithChart, contains an empty pivot table and pivot chart. It also contains an empty PivotTableFieldList. The first worksheet is shown in the following image.



## Saving Report Output in PPTX Format

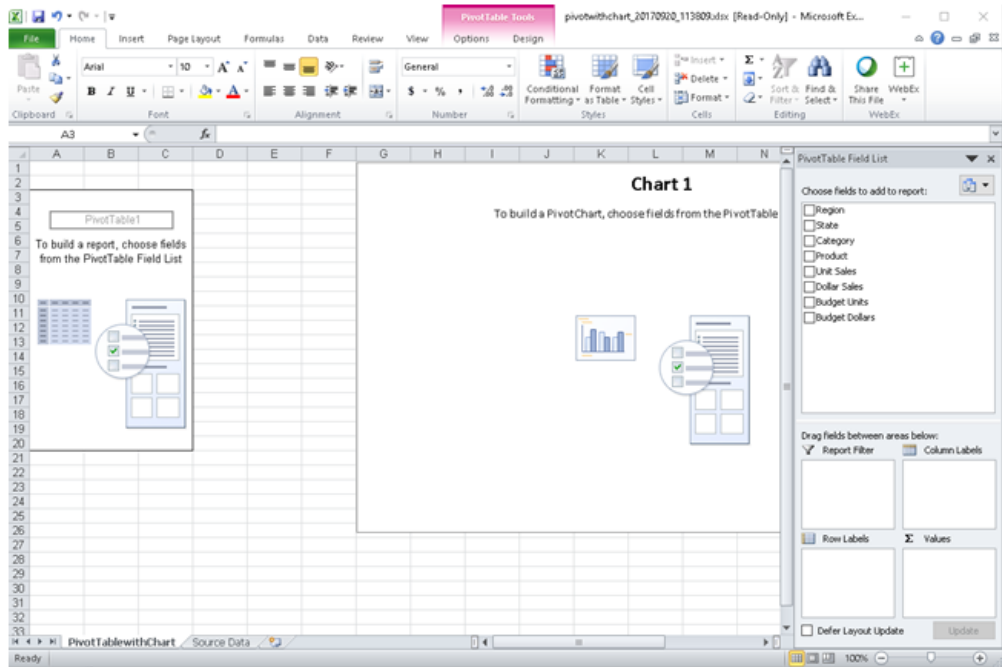
The second worksheet, Source Data, contains one column called FieldsToBeAdded, for which there is initially no data. The second worksheet is shown in the following image.



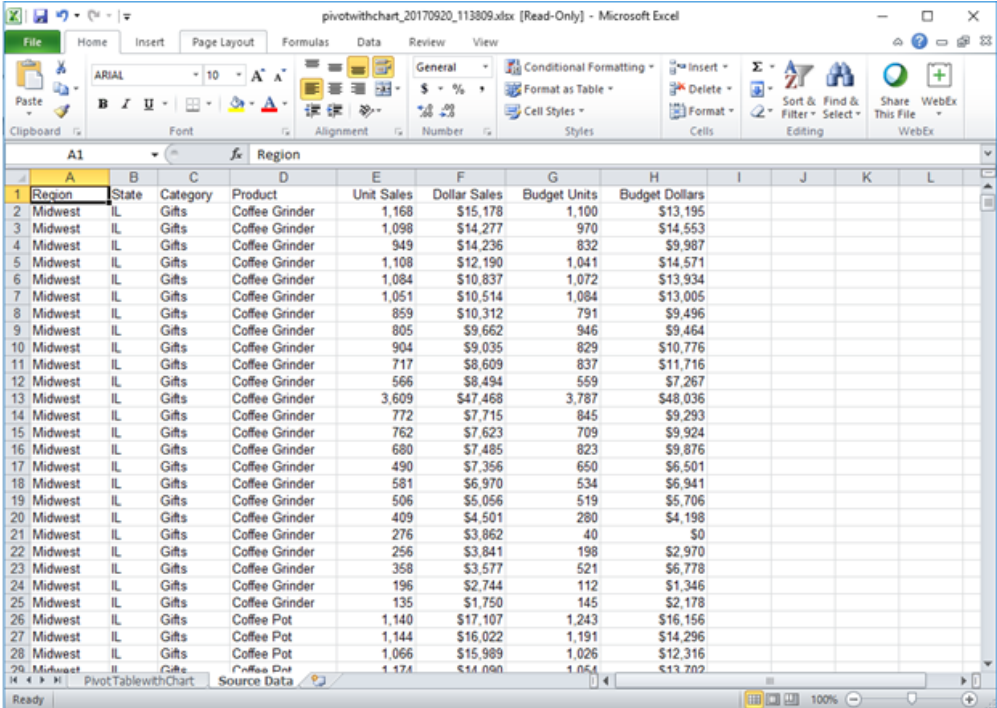
When you run the sample procedure, a pivotwithchart.xlsx workbook is generated, with the WebFOCUS report data stored in the second worksheet.

The following images show the first and second worksheets in the pivotwithchart.xlsx workbook after you run the sample procedure.

## First Worksheet



## Second Worksheet

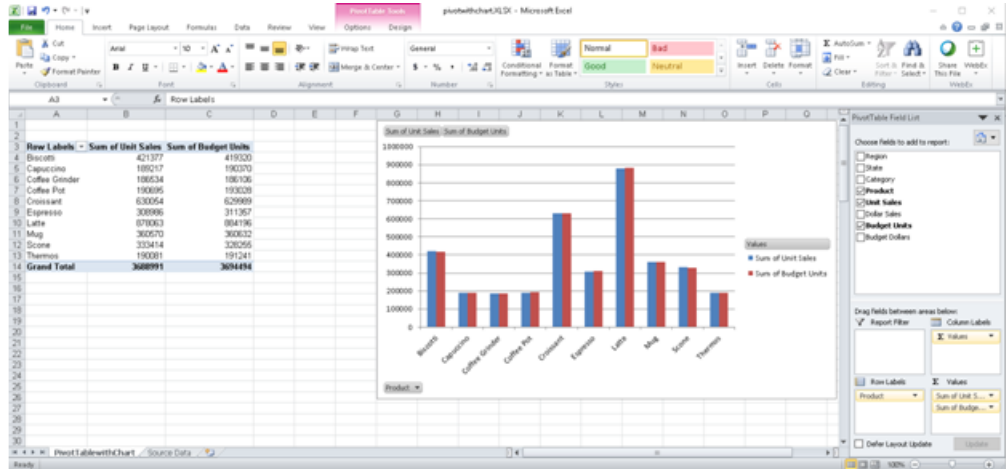


Region	State	Category	Product	Unit Sales	Dollar Sales	Budget Units	Budget Dollars
Midwest	IL	Gifts	Coffee Grinder	1,168	\$15,178	1,100	\$13,195
Midwest	IL	Gifts	Coffee Grinder	1,098	\$14,277	970	\$14,553
Midwest	IL	Gifts	Coffee Grinder	949	\$14,236	832	\$9,987
Midwest	IL	Gifts	Coffee Grinder	1,108	\$12,190	1,041	\$14,571
Midwest	IL	Gifts	Coffee Grinder	1,084	\$10,837	1,072	\$13,934
Midwest	IL	Gifts	Coffee Grinder	1,051	\$10,514	1,084	\$13,005
Midwest	IL	Gifts	Coffee Grinder	859	\$10,312	791	\$9,496
Midwest	IL	Gifts	Coffee Grinder	805	\$9,662	946	\$9,464
Midwest	IL	Gifts	Coffee Grinder	904	\$9,035	829	\$10,776
Midwest	IL	Gifts	Coffee Grinder	717	\$8,609	837	\$11,716
Midwest	IL	Gifts	Coffee Grinder	566	\$8,494	559	\$7,267
Midwest	IL	Gifts	Coffee Grinder	3,609	\$47,468	3,787	\$48,036
Midwest	IL	Gifts	Coffee Grinder	772	\$7,715	845	\$9,293
Midwest	IL	Gifts	Coffee Grinder	762	\$7,623	709	\$9,924
Midwest	IL	Gifts	Coffee Grinder	680	\$7,485	823	\$9,876
Midwest	IL	Gifts	Coffee Grinder	490	\$7,356	650	\$6,501
Midwest	IL	Gifts	Coffee Grinder	581	\$6,970	534	\$6,941
Midwest	IL	Gifts	Coffee Grinder	506	\$5,056	519	\$5,706
Midwest	IL	Gifts	Coffee Grinder	409	\$4,501	280	\$4,198
Midwest	IL	Gifts	Coffee Grinder	276	\$3,862	40	\$0
Midwest	IL	Gifts	Coffee Grinder	256	\$3,841	198	\$2,970
Midwest	IL	Gifts	Coffee Grinder	358	\$3,577	521	\$6,778
Midwest	IL	Gifts	Coffee Grinder	196	\$2,744	112	\$1,346
Midwest	IL	Gifts	Coffee Grinder	135	\$1,750	145	\$2,178
Midwest	IL	Gifts	Coffee Pot	1,140	\$17,107	1,243	\$16,156
Midwest	IL	Gifts	Coffee Pot	1,144	\$16,022	1,191	\$14,296
Midwest	IL	Gifts	Coffee Pot	1,066	\$15,989	1,026	\$12,316
Midwest	IL	Gifts	Coffee Pot	1,174	\$14,090	1,051	\$13,702

In the PivotTableWithChart worksheet, note that the PivotTableFieldList is populated with the fields from the WebFOCUS report. There is one check box for each field in the report procedure. All the check boxes are not selected, by default.

The data used to populate the check boxes is obtained from the Source Data worksheet, where the data from the WebFOCUS report was saved upon executing the sample procedure.

To start building a pivot report and pivot chart, you can select the check boxes for the desired fields in the PivotTableFieldList. For example, selecting the check boxes for Product, Unit Sales, and Budget Units will automatically feed the data from the Source Data worksheet into the pivot table and pivot chart in the PivotTablewithChart worksheet. The resulting pivot table and pivot chart are shown in the following image.



The wf2pivot.xlsx template is provided, by default, with the WebFOCUS Reporting Server installation as part of the Legacy Samples. For information on how to download the Reporting Server Legacy Samples, see the *Server Administration* manual.

You can use the WebFOCUS XLSX template as is, or you can customize it to meet the your business requirements.

**Note:** The wf2pivot.xlsx template includes a pivot table and pivot chart on the same worksheet, but you can use a macro-enabled template to generate a pivot table and a pivot chart on separate worksheets, which are linked to a common data source.

## ReportCaster Distribution and ReportCaster Bursting

ReportCaster Distribution is supported for simple reports, reports with images, PPTX Templates, Coordinated Compound Reports, and graphs. Bursting is supported for simple reports, reports with images, PPTX Templates, and Coordinated Compound Reports, but not for graphs. For more information, see the *ReportCaster* manual.

## PPTX Limitations

The following are limitations when using PPTX output format:

- ☐ ReportCaster bursting of graphs.

- ❑ Justification of images in report elements.

In Compound syntax, the following are limitations when using PPTX output format:

- ❑ Paneled reports.
- ❑ Nested syntax.
- ❑ Multi-pane reports or reports with multiple columns.
- ❑ OPEN, CLOSE, or NOBREAK command.

### Related Information

For related information on these topics, see the following WebFOCUS manuals:

- ❑ *Describing Data With WebFOCUS Language*
- ❑ *Developing Reporting Applications*
- ❑ *Using Functions*
- ❑ *ReportCaster*

## Linking a Report to Other Resources

---

You can use StyleSheet declarations to define links from any report component. You can use links to:

- ☐ Create a series of drill-down reports by linking the procedures that generate these reports.
- ☐ Link to URLs. These can be other webpages, websites, Servlet programs, or non-World Wide Web resources, such as an email application.
- ☐ Execute JavaScript functions to perform additional analysis of the report data.

You can create links from report data as well as graphical images within a report. You can also create links from a graph. For details on linking from a graph, see [Creating a Graph](#) on page 1657.

### In this chapter:

- |  |  |
|--|--|
| <input type="checkbox"/> <a href="#">Linking Using StyleSheets</a>   | <input type="checkbox"/> <a href="#">Creating Parameters</a>                                     |
| <input type="checkbox"/> <a href="#">Linking to Another Report</a>   | <input type="checkbox"/> <a href="#">Linking With Conditions</a>                                 |
| <input type="checkbox"/> <a href="#">Linking to a URL</a>  | <input type="checkbox"/> <a href="#">Linking From a Graphic Image</a>                            |
| <input type="checkbox"/> <a href="#">Linking to a JavaScript Function</a>                                  | <input type="checkbox"/> <a href="#">Specifying a Base URL</a>                                   |
| <input type="checkbox"/> <a href="#">Linking to a Maintain Data Procedure</a>                              | <input type="checkbox"/> <a href="#">Specifying a Target Frame</a>                               |
| <input type="checkbox"/> <a href="#">Multi-Drill Feature With Cascading Menus and User-Defined Styling</a> | <input type="checkbox"/> <a href="#">Creating a Compound Report</a>                              |
|  | <input type="checkbox"/> <a href="#">Creating a PDF Compound Report With Drill Through Links</a> |
- 

### Linking Using StyleSheets

You can use StyleSheets to define a link from any report component. You can create links from report data (including headings and footings) as well as graphic images (such as a company logo or product image), to other reports, procedures, URLs, or JavaScript functions.

The links you create can be dynamic. With a dynamic link, your selection passes the value of the selected report component to the linked report (procedure, URL, or JavaScript function). The resource uses the passed value to dynamically determine the results that are returned. You can pass one or more parameters. For details, see [Creating Parameters](#) on page 797.

### **Procedure:** How to Create Links Using StyleSheets

This procedure is a basic overview of how to create links using StyleSheets.

1. Identify the report component that the user selects in the web browser to execute the link.
2. Specify the name of the embedded procedure, URL, or JavaScript function to execute.
3. Identify the parameters that define the specifics of your link, if necessary.

## Linking to Another Report

A link allows you to drill down to a report for more details or execute a procedure by selecting a designated hot spot (the link) in the report. By linking reports you provide easy access to more detailed data that supplements the information in your base report. The drill-down report can contain information that is either independent of the data in the base report or depends and expands on a specific data value in the base report.

To create a link, you must have a report to link from (the base report) and a report to link to (the drill-down report). If the drill-down report depends on a specific data value in the base report, you also need to pass that value to the drill-down report by creating parameters. For details, see [Creating Parameters](#) on page 797.

### **Syntax:** How to Link to Reports and Procedures

```
TYPE=type, [subtype], FOCEXEC=fex[(parameters ...)], [TARGET=frame,]  
[ALT = 'description',] $
```

where:

*type*

Identifies the report component that you select in the web browser to execute the link. The TYPE attribute and its value must appear at the beginning of the declaration.

*subtype*

Are any additional attributes, such as COLUMN, LINE, or ITEM, that are needed to identify the report component that you are formatting. For information on identifying report components, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.



*fex*

Identifies the file name of the linked procedure to run when you select the report component.

**Note:** The procedure cannot be named NONE (all uppercase). Using NONE as the procedure name will result in a syntax error. Mixed or lowercase is allowed.

To determine the file name in WebFOCUS, see [How to Determine a WebFOCUS File Name in Managed Reporting](#) on page 766.

The maximum length of a FOCEXEC=*fex* argument, including any associated parameters, is 2400 characters. The FOCEXEC argument can span more than one line, as described in [Creating and Managing a WebFOCUS StyleSheet](#) on page 1117.

*parameters*

Values that are passed to the report, URL, or JavaScript function. For details, see [Creating Parameters](#) on page 797.

*frame*

Identifies the target frame in the webpage in which the output from the drill-down link is displayed. For details, see [Specifying a Target Frame](#) on page 814.

*description*

Is a textual description of the link supported in an HTML report for compliance with Section 508 accessibility. Enclose the description in single quotation marks.

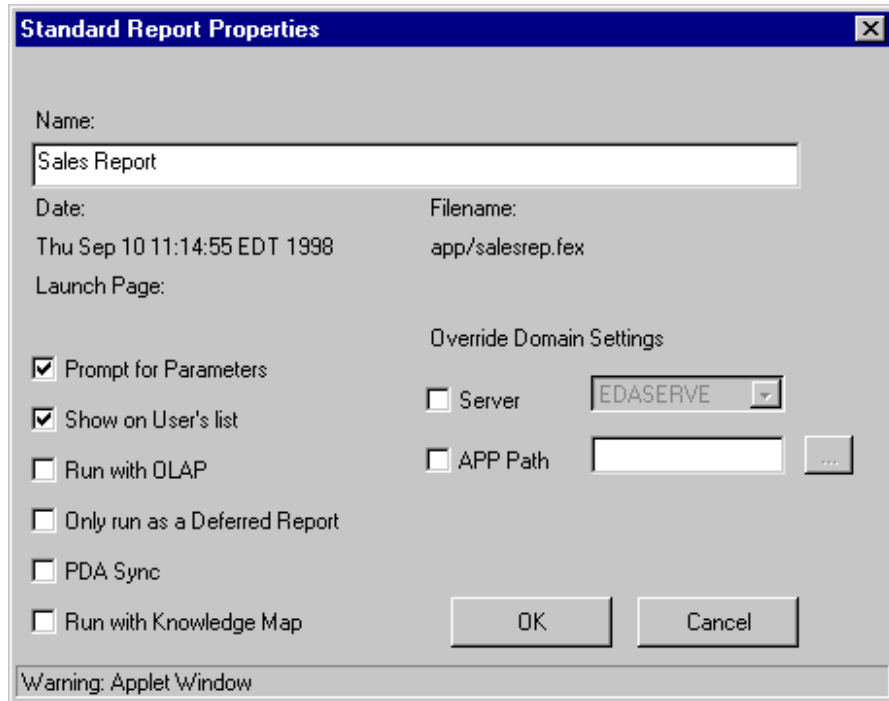
The description also displays as a pop-up description when your mouse or cursor hovers over the link in the report output.

### **Reference: Usage Notes for Drilldown Reports in PDF Format**

When going back to the original report from a drilldown report in PDF format, you must click the Back button twice quickly. The alternative is to use the drop-down list presented to the right of the Back button to view the browser history and select the link two steps back. The first history item will point to the redirection page and be titled based on the method used to access the WFServlet. The previous item will be titled *WebFOCUS Report* and will point back to the original PDF report.

**Procedure: How to Determine a WebFOCUS File Name in Managed Reporting**

1. Right-click the report name and select *Properties*. The Report Properties dialog box opens.
2. The file name appears under *Filename*. In the example below, the name of the file is "salesrep". Do not include the file extension (.fex) or the directory location and slash (/).



**Example: Linking to a Report From a Footing**

The following report request summarizes product sales and sorts the data by region, state, and store code. The store code also displays in the subfootings where links to detailed reports about the store's sales (by product or by date) display. Each line of the subfoot contains two text objects and one embedded field. The relevant StyleSheet declarations are highlighted in the request.

The main report is:

```

TABLE FILE GGSALES
HEADING
"Sales Report"
SUM DOLLARS/I08M
BY REGION BY ST BY STCD
ON STCD SUBFOOT
"View Store <STCD Sales By Product"
" "
"View Store <STCD Sales By Date"
ON REGION PAGE-BREAK
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=HEADING, SIZE=12, STYLE=BOLD, $
TYPE=SUBFOOT, LINE=1, OBJECT=TEXT, ITEM=2, COLOR=GREEN,
FOCEXEC=PRDSALES (STOREID=STCD), $
TYPE=SUBFOOT, LINE=3, OBJECT=TEXT, ITEM=2, COLOR=BLUE,
FOCEXEC=HSTSALES (STOREID=STCD), $
ENDSTYLE
END

```

Using StyleSheet declarations, the subfoot phrase *Sales By Product* links to a second procedure named PRDSALES and passes it the value of STCD displayed in the subfoot. The subfoot phrase *Sales By Date* links to a procedure named HSTSALES and passes it the value of STCD displayed in the subfoot.

The request for the linked report HSTSALES is:

```

TABLE FILE GGSALES
SUM UNITS
BY STCD
BY DATE
WHERE STCD = '&STOREID'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END

```

The request for the linked report PRDSALES is:

```

TABLE FILE GGSALES
SUM UNITS
BY STCD
BY PRODUCT
WHERE STCD = '&STOREID'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END

```

The first page of output for the main report follows. If you select *Sales By Product* for Store R1020, the value R1020 is passed to the PRDSALES procedure. If you select *Sales By Date* for Store R1019, the value R1019 is passed to the HSTSALES procedure.

The output is:

Sales Report			
Region	State	Store ID	Dollar Sales
Midwest	IL	R1020	\$3,924,401
View Store R1020 <a href="#">Sales By Product</a>			
View Store R1020 <a href="#">Sales By Date</a>			
	MO	R1250	\$3,761,286
View Store R1250 <a href="#">Sales By Product</a>			
View Store R1250 <a href="#">Sales By Date</a>			
	TX	R1019	\$3,714,978
View Store R1019 <a href="#">Sales By Product</a>			
View Store R1019 <a href="#">Sales By Date</a>			

If you click the *Sales By Product* link for store R1020, the output is:

<u>Store ID</u>	<u>Product</u>	<u>Unit Sales</u>
R1020	Biscotti	29413
	Coffee Grinder	19339
	Coffee Pot	15785
	Croissant	43300
	Espresso	32237
	Latte	77344
	Mug	30157
	Scone	45355
	Thermos	14651

## Linking to a URL

You can define a link from any report component to any URL including webpages, websites, Servlet programs, or non-World Wide Web resources, such as an email application. After you have defined a link, you can select the report component to access the URL.

The links you create can be dynamic. With a dynamic link, your selection passes the value of the selected report component to the URL. The resource uses the passed value to dynamically determine the results that are returned. You can pass one or more parameters. For details, see [Creating Parameters](#) on page 797.

### **Syntax:** How to Link to a URL

```
TYPE=type, [subtype], URL=url[(parameters ...)], [TARGET=frame,] [ALT =  
'description',] $
```

where:

#### *type*

Identifies the report component that you select in the web browser to execute the link. The TYPE attribute and its value must appear at the beginning of the declaration.

#### *subtype*

Are any additional attributes, such as COLUMN, LINE, or ITEM, that are needed to identify the report component that you are formatting. For information on identifying report components, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

#### *url*

Identifies any valid URL, including a URL that specifies a WebFOCUS Servlet program, or the name of a report column enclosed in parentheses whose value is a valid URL to which the link will jump.

#### **Note:**

- ❑ The maximum length of a URL=*url* argument, including any associated variable=object parameters, is limited by the maximum number of characters allowed by the browser. For information about this limit for your browser, search on your browser vendor's support site. The URL argument can span more than one line, as described in [Creating and Managing a WebFOCUS StyleSheet](#) on page 1117.

Note that the length of the URL is limited by the maximum number of characters allowed by the browser. For information about this limit for your browser, search on your browser vendor's support site.

- ❑ If the URL refers to a WebFOCUS Servlet program that takes parameters, the URL must end with a question mark (?).

### *parameters*

Values that are passed to the URL. For details, see [Creating Parameters](#) on page 797.

### *frame*

Identifies the target frame in the webpage in which the output from the drill-down link is displayed. For details, see [Specifying a Target Frame](#) on page 814.

### *description*

Is a textual description of the link supported in an HTML report for compliance with Section 508 accessibility. Enclose the description in single quotation marks.

The description also displays as a pop-up description when your mouse or cursor hovers over the link in the report output.

## **Example:**    **Linking to a URL**

The following example illustrates how to link to a URL from a report. The heading *Click here to access the IB homepage* is linked to the URL [www.ibi.com](http://www.ibi.com). The relevant StyleSheet declarations are highlighted in the request.

Note that *webserver* indicates the name of the webserver that runs WebFOCUS.

```
TABLE FILE GGSales
ON TABLE SET PAGE-NUM OFF
SUM UNITS AND DOLLARS
BY CATEGORY BY REGION
HEADING
"Regional Sales Report"
"Click here to access the IB homepage."
" "
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=HEADING, LINE=2, OBJECT=TEXT, ITEM=1,
URL=http://www.ibi.com, $
ENDSTYLE
END
```

The output is:

Regional Sales Report

[Click here to access the IB homepage.](#)

<u>Category</u>	<u>Region</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	Midwest	332777	4178513
	Northeast	335778	4164017
	Southeast	350948	4415408
	West	356763	4473517
Food	Midwest	341414	4338271
	Northeast	353368	4379994
	Southeast	349829	4308731
	West	340234	4202337
Gifts	Midwest	230854	2883881
	Northeast	227529	2848289
	Southeast	234455	2986240
	West	235042	2977092

When you click the link the site displays in your browser.

**Example:** **Linking to a URL to Run a Drilldown WebFOCUS Server Procedure**

The following request is initiated from a browser session and runs a drill down report stored on the WebFOCUS Reporting Server.

This procedure is run from a browser, so the drilldown in the example is specified as a relative URL (it does not have protocol, host, or port) because it will be submitted using the protocol, host, and port of the current browser session.

**Note:** This technique is useful in a Managed Reporting procedure for creating a drill down to a WebFOCUS Server procedure. The FOCEXEC= technique for running a drill down procedure does not work because Managed Reporting always looks for the procedure in the Managed Reporting repository.

The main procedure is:

```
TABLE FILE GGSALES
ON TABLE SET PAGE-NUM OFF
SUM UNITS AND DOLLARS
BY CATEGORY BY REGION
HEADING
"Regional Sales Report"
" "
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=DATA, COLUMN=REGION,
URL=/ibi_apps/WFServlet?(IBIF_ex='ggdrill' AREA=REGION
IBIC_server='EDASERVE' IBI_APPS='IBISAMP'),$
ENDSTYLE
END
```

The drilldown report, which must be in application ibisamp, is:

```
-DEFAULTS &REGION='$*';
TABLE FILE GGSALES
ON TABLE SET PAGE-NUM OFF
SUM UNITS AND DOLLARS
BY PRODUCT
WHERE REGION = '&AREA'
HEADING
"Sales Report for Region &AREA"
" "
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```



The output of the main report is:

### Regional Sales Report

Click a region to see a report for that region.

<u>Category</u>	<u>Region</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	<a href="#"><u>Midwest</u></a>	332777	4178513
	<a href="#"><u>Northeast</u></a>	335778	4164017
	<a href="#"><u>Southeast</u></a>	350948	4415408
	<a href="#"><u>West</u></a>	356763	4473517
Food	<a href="#"><u>Midwest</u></a>	341414	4338271
	<a href="#"><u>Northeast</u></a>	353368	4379994
	<a href="#"><u>Southeast</u></a>	349829	4308731
	<a href="#"><u>West</u></a>	340234	4202337
Gifts	<a href="#"><u>Midwest</u></a>	230854	2883881
	<a href="#"><u>Northeast</u></a>	227529	2848289
	<a href="#"><u>Southeast</u></a>	234455	2986240
	<a href="#"><u>West</u></a>	235042	2977092

If you click the region Northeast, the output is:

### Sales Report for Region Northeast

<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Biscotti	145242	1802005
Capuccino	44785	542095
Coffee Grinder	40977	509200
Coffee Pot	46185	590780
Croissant	137394	1670818
Espresso	68127	850107
Latte	222866	2771815
Mug	91497	1144211
Scone	70732	907171
Thermos	48870	604098

## Defining a Hyperlink Color

You can use the HYPERLINK-COLOR attribute to designate a color for a hyperlink within a report. This applies to all hyperlinks generated in the report. You can define a single color for the entire report or different colors for each individual element.

### **Syntax:** How to Define a Hyperlink Color

`TYPE = type, HYPERLINK-COLOR = color`

where:

*type*

Is the report component you wish to affect. You can apply this keyword to the entire report using TYPE=REPORT. The attribute can also individually be set for any other element of the report. For details, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

*color*

Can use any style sheet supported color value designation. For available color values that can be utilized with the syntax, see [Color Values in a Report](#) on page 1615.

**Example: Defining a Hyperlink Color**

The following PDF request illustrates how to define hyperlink colors for the entire report, as well as individual elements.

- ❑ The default font color for the entire report is grey and the default hyperlink color for the entire report is slate blue.
- ❑ For the Dollar Sales column (DOLLARS), the font color is green and the hyperlink color is purple.
- ❑ For both the Dollar Sales column (DOLLARS) and the Unit Sales column (UNITS), conditional styling has been applied using the same condition (REGION GE 'O').
- ❑ For the Unit Sales column (UNITS), when the conditional styling is met, the hyperlink color is inherited from the default hyperlink color for the report (slate blue).
- ❑ For the Dollar Sales column (DOLLARS), when the conditional styling is met, the hyperlink color is purple.

```
TABLE FILE GGSales
SUM DOLLARS/D12CM UNITS/D12C
BY REGION
BY CATEGORY
HEADING
"Hyperlinks of Many Colors"
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, SQUEEZE=ON, FONT=ARIAL, GRID=OFF, COLOR=GREY,
    HYPERLINK-COLOR='SLATE BLUE', $
TYPE=DATA, COLUMN=UNITS, WHEN=REGION GE 'O', URL='http://www.ibi.com', $
TYPE=DATA, COLUMN=DOLLARS, COLOR=GREEN, HYPERLINK-COLOR='PURPLE', $
TYPE=DATA, COLUMN=DOLLARS, WHEN=REGION GE 'O', URL='http://www.ibi.com', $
ENDSTYLE
END
```

The output is:

Hyperlinks of Many Colors

Region	Category	Dollar Sales	Unit Sales
Midwest	Coffee	<a href="#">\$4,178,513</a>	332,777
	Food	<a href="#">\$4,338,271</a>	341,414
	Gifts	<a href="#">\$2,883,881</a>	230,854
Northeast	Coffee	<a href="#">\$4,164,017</a>	335,778
	Food	<a href="#">\$4,379,994</a>	353,368
	Gifts	<a href="#">\$2,848,289</a>	227,529
Southeast	Coffee	<a href="#">\$4,415,408</a>	<a href="#">350,948</a>
	Food	<a href="#">\$4,308,731</a>	<a href="#">349,829</a>
	Gifts	<a href="#">\$2,986,240</a>	<a href="#">234,455</a>
West	Coffee	<a href="#">\$4,473,517</a>	<a href="#">356,763</a>
	Food	<a href="#">\$4,202,337</a>	<a href="#">340,234</a>
	Gifts	<a href="#">\$2,977,092</a>	<a href="#">235,042</a>

Reference: Usage Notes for HYPERLINK-COLOR

- ❑ By default, drill-down links are presented in hyperlink blue and underlined.
- ❑ In HTML and DHTML reports, any designated report font color overrides the drill-down default font color.
- ❑ For standard reports, set the HYPERLINK-COLOR attribute using the TYPE=REPORT declaration of the style sheet.
- ❑ For compound reports, set the HYPERLINK-COLOR attribute using the TYPE=REPORT declaration of the style sheet of the first component report (excluding anything on the Page Master).
- ❑ For PPTX, the hyperlink color is stored as part of the PPTX Slide Master theme. Only one HYPERLINK-COLOR attribute can be defined for each request (report/compound report).

Linking to a JavaScript Function

You can use a StyleSheet to define a link to a JavaScript function from any report component. After you have defined the link, you can select the report component to execute the JavaScript function.

Just as with drill-down links to procedures and URLs, you can specify optional parameters that allow values of a report component to be passed to the JavaScript function. The function will use the passed value to dynamically determine the results that are returned to the browser. For details, see [Creating Parameters](#) on page 797.

**Note:**

- ❑ JavaScript functions can, in turn, call other JavaScript functions.
- ❑ You cannot specify a target frame if you are executing a JavaScript function. However, the JavaScript function itself can specify a target frame for its results.

**Syntax:****How to Link to a JavaScript Function**

`TYPE=type, [subtype], JAVASCRIPT=function[(parameters ...)], $`

where:

*type*

Identifies the report component that you select in the web browser to execute the link. The TYPE attribute and its value must appear at the beginning of the declaration.

*subtype*

Are any additional attributes, such as COLUMN, LINE, or ITEM, that are needed to identify the report component that you are formatting. See [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167 for details.

*function*

Identifies the JavaScript function to run when you select the report component.

The maximum length of a JAVASCRIPT=function argument, including any associated parameters, is 2400 characters and can span more than one line. If you split a single argument across a line, you need to use the \ character at the end of the first line, as continuation syntax. If you split an argument at a point where a space is required as a delimiter, the space must be before the \ character or be the first character on the next line. The \ character does not act as the delimiter.

In this example,

```
JAVASCRIPT=myfunc(COUNTRY \
CAR MODEL 'ABC' ),$
```

the argument correctly spans two lines.

**Note:**

- ❑ You can use the Dialogue Manager -HTMLFORM command to embed the report into an HTML document in which the function is defined.
- ❑ When you have an HTML document called by -HTMLFORM, ensure that the file extension is .HTM (not .HTML).

For more information about the -HTMLFORM command, see the *Developing Reporting Applications* manual.

### *parameters*

Values that are passed to the JavaScript function. For details, see [Creating Parameters](#) on page 797.

### **Example:** Linking to a JavaScript Function

The following displays the report and StyleSheet syntax used to link to a JavaScript function. It also shows the JavaScript function that is executed, and the result that is displayed in the browser.

The report request (which contains the inline StyleSheet) is:

```
TABLE FILE GGORDER
SUM PRODUCT_ID
BY STORE_CODE
BY PRODUCT_DESCRIPTION NOPRINT
IF STORE_CODE EQ 'R1250'
ON TABLE HOLD AS JAVATEMP FORMAT HTMTABLE
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=DATA, COLUMN=PRODUCT_ID, JAVASCRIPT=showitem(PRODUCT),$
ENDSTYLE
END
-RUN
-HTMLFORM JAVAFORM
```

The JAVAFORM.HTM file that contains the JavaScript function is:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
// This function will display the value in the text box
function showitem(string) {
document.form1.text1.value = string;
}
// End the hiding here
</SCRIPT>
</HEAD>
<BODY>
!IBI.FIL.JAVATEMP;
<HR>
<B>Product Description:</B>
<FORM NAME="form1">
<INPUT TYPE="text" NAME="text1" SIZE="16"> </FORM>
</BODY>
</HTML>
```

When you execute the report procedure, the following report displays in the web browser. If you select a Product Code link, the JavaScript function *ShowItem* executes, and displays the value of the PRODUCT\_DESCRIPTION field (a NOPRINT field) in the text box in the form below the report. For example, if you select the Product Code *G104*, "Thermos" displays in the Product Description field.

<u>Store</u> <u>Code</u>	<u>Product</u> <u>Code</u>
R1250	<a href="#">F102</a>
	<a href="#">G110</a>
	<a href="#">G121</a>
	<a href="#">F103</a>
	<a href="#">B142</a>
	<a href="#">B141</a>
	<a href="#">G100</a>
	<a href="#">F101</a>
	<a href="#">G104</a>

---

**Product Description:**

Thermos

## Linking to a Maintain Data Procedure

You can provide update capabilities directly from your report by linking it to a Maintain Data procedure.

The link can be either a URL for the WebFOCUS Servlet or a JavaScript drilldown to the Maintain Data procedure.

If it is a URL for the WebFOCUS Servlet, it must include the IBIF\_cmd command with the MNTCON RUN or MNTCON EX syntax to invoke an existing Maintain Data form procedure. The link can pass control to a Maintain form, or run a batch mode Maintain procedure that does not display a user interface.

If it is a JavaScript drilldown, it uses the parent.IbComposer\_drillMntdata function.

### **Syntax:** How to Link to a Maintain Data Procedure Using a URL

```
TYPE=type, [subtype, ] URL=/ibi_apps/WFServlet? IBIF_cmd='MNTCON
{RUN|EX} procname' IBIS_passthru='on' IBIS_connect='on'
[(parameters...)], $
```

where:

#### *type*

Identifies the report component that you select in the web browser to execute the link. The TYPE attribute and its value must appear at the beginning of the declaration.

#### *subtype*

Are any additional attributes, such as COLUMN, LINE, or ITEM, that are needed to identify the report component that you are formatting. See [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167 for information on identifying report components.

#### *procname*

Is the name of the Maintain Data procedure.

#### *parameters*

Values that are passed to the Maintain Data procedure. For details, see [Creating Parameters](#) on page 797.

### **Example:** Linking to a Maintain Data Procedure

The following report allows you to update the unit price for a product directly from the report output by linking the report to the appropriate Maintain procedure.

The report request is:

```
TABLE FILE GGPRODS
PRINT PRODUCT_DESCRIPTION VENDOR_CODE VENDOR_NAME UNIT_PRICE
BY PRODUCT_ID
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=DATA, COLUMN=N1,
    URL=/ibi_apps/WFServlet?(PRODUCT_ID=N1 IBIF_cmd='MNTCON RUN GGUPD1'
    IBIS_passthru=\
        'on' IBIS_connect='on'), $
ENDSTYLE
END
```

The Maintain Data procedure (ggupd1) is:



```

MAINTAIN FILE ggprods
module import(mntuws FOCCOMP)
$$Declarations
Declare pcode/a4;
Case Top
compute timechk/a0=HHMMSS();
document.referer='/ibi_apps/WFServlet?IBIF_ex=ggprod&IBIS_connect
=on'||
'&timechk='|timechk;
compute pcode = IWC.getAppCGIValue("PRODUCT_ID");
Infer ggprods.prods01.product_id into ggstk1;
Reposition PRODUCT_ID
Stack clear ggstk1 ;
For all next ggprods.prods01.product_id into ggstk1
  where product_id eq pcode;
Winform Show Form1;
EndCase
Case Updtel
for all Update ggprods.prods01.unit_price from ggstk1(1) ;
EndCase
END

```

**Note:** This is an interactive form to display data and is created in App Studio.


The report is:

Product				Unit
<u>Code</u>	<u>Product</u>	<u>Vendor ID</u>	<u>Vendor Name</u>	<u>Price</u>
<a href="#">B141</a>	Hazelnut	V082	Coffee Connection	59.00
<a href="#">B142</a>	French Roast	V083	European Specialities,	81.00
<a href="#">B144</a>	Kona	V081	Evelina Imports, Ltd	76.00
<a href="#">F101</a>	Scone	V102	Ridgewood Bakeries	13.00
<a href="#">F102</a>	Biscotti	V100	Delancey Bakeries	17.00
<a href="#">F103</a>	Croissant	V103	West Side Bakers	30.00
<a href="#">G100</a>	Mug	V305	NY Ceramic Supply	26.00
<a href="#">G104</a>	Thermos	V305	ThermoTech, Inc	97.00
<a href="#">G110</a>	Coffee Grinder	V303	Appliance Craft	125.00
<a href="#">G121</a>	Coffee Pot	V304	Appliance Craft	140.00

When you click a Product Code, the Maintain procedure ggupd1 is invoked, which uses the IWC.getAppCGIValue function to retrieve the correct value.

Form 1 in the Maintain Data procedure ggupd1 opens and you can update the unit price for that product:

powered by



Unit Price Update

	Product Id	Product Description	Vendor Code	Vendor Name	Unit Price
1	F103	Croissant	V103	West Side Bakers	28.00

Update Unit Price

Finish

**Syntax:**      **How to Link to a Maintain Data Procedure Using a JavaScript Drilldown**

```
TYPE=DATA,
  DRILLMENUITEM='DrillDown 1',
    JAVASCRIPT=parent.IbComposer_drillMntdata( \
      'Request2' \
      'mntcase' \
      'stack_field' \
      rptcol \
    ),
    TARGET='_parent',
$
```

where:

`'DrillDown 1'`

Is the text that displays for the drilldown link.

`'Request2'`

Is the name of the Maintain Data procedure.

`'mntcase'`

Is the name of the Maintain case.

`'stack_field'`

Is the stack and stack field associated with the report column.

*rptcol*

Is the report column specification.

**Note:** Multiple stack fields and report columns can be specified to pass additional values from the report to Maintain.

In the Maintain procedure, make sure the stack retrieving the values is created and the case being performed exists.

The following is a sample drilldown from a report against the MOVIES data source to a Maintain procedure. It passes the value from the first report column (N1) to the Moviecode field in the Movstk stack in the the LoadData case of the Maintain procedure named Request2.

```
TYPE=DATA,
  DRILLMENUITEM='DrillDown 1',
    JAVASCRIPT=parent.IbComposer_drillMntdata( \
      'Request2' \
      'LoadData' \
      'Movstk.Moviecode' \
      N1 \
    ),
    TARGET='_parent',
$
```

The following is the Maintain code needed in order to pass these values from the report. The Maintain procedure is named Request2.

```
MAINTAIN FILE movies

$$Declarations

Case Top
infer moviecode into movstk
perform loaddata
Winform Show Form1;
EndCase

Case LoadData
reposition moviecode
stack clear detailstk
next moviecode into detailstk
  where moviecode eq movstk.moviecode;
endcase

END
```

When the report and the Maintain form are placed on the same HTML page, clicking one of the links in the report passes the values to the Maintain form, as shown in the following image.

MOVIECODE	TITLE	CATEGORY
<a href="#">001MCA</a>	<a href="#">JAWS II</a>	<a href="#">ACTION</a>
<a href="#">005WAR</a>	<a href="#">EAST OF EDEN</a>	<a href="#">CLASSIC</a>
<a href="#">020TUR</a>	<a href="#">CITIZEN KANE</a>	<a href="#">CLASSIC</a>
<a href="#">024WAR</a>	<a href="#">DOG DAY AFTERNOON</a>	<a href="#">DRAMA</a>
<a href="#">031KKV</a>	<a href="#">SMURFS, THE</a>	<a href="#">CHILDREN</a>
<a href="#">035CBS</a>	<a href="#">CABARET</a>	<a href="#">MUSICALS</a>
<a href="#">040ORI</a>	<a href="#">BABETTE'S FEAST</a>	<a href="#">FOREIGN</a>
<a href="#">043DIS</a>	<a href="#">SHAGGY DOG, THE</a>	<a href="#">CHILDREN</a>
<a href="#">053WAR</a>	<a href="#">TIN DRUM, THE</a>	<a href="#">FOREIGN</a>
<a href="#">076WAR</a>	<a href="#">ALTERED STATES</a>	<a href="#">SCI/FI</a>
<a href="#">081MCA</a>	<a href="#">REAR WINDOW</a>	<a href="#">MYSTERY</a>
<a href="#">082MCA</a>	<a href="#">VERTIGO</a>	<a href="#">MYSTERY</a>
<a href="#">090CBS</a>	<a href="#">ALIEN</a>	<a href="#">SCI/FI</a>
<a href="#">093WOR</a>	<a href="#">SCOOPY-DOO-A DOG IN THE RUFF</a>	<a href="#">CHILDREN</a>
<a href="#">095CBS</a>	<a href="#">ALL THAT JAZZ</a>	<a href="#">MUSICALS</a>
<a href="#">103LOR</a>	<a href="#">LEARN TO SKI BETTER</a>	<a href="#">TRAIN/EX</a>
104MED	BOY AND HIS DOG, A	SCI/FI

Moviecode	<input type="text" value="076WAR"/>
Title	<input type="text" value="ALTERED STATES"/>
Category	<input type="text" value="SCI/FI"/>
Director	<input type="text" value="RUSSELL K."/>
Rating	<input type="text" value="R"/>
Reldate	<input type="text" value="81/07/22"/>
Wholesalepr	<input type="text" value="14.99"/>
Listpr	<input type="text" value="19.98"/>
Copies	<input type="text" value="1"/>

## Multi-Drill Feature With Cascading Menus and User-Defined Styling

The multi-drill feature supports multiple menu items, as well as multiple cascading levels, that you can incorporate into any WebFOCUS report that works with JavaScript (for example, HTML and DHTML). Styling of the menu can be customized using WebFOCUS StyleSheet syntax.

The multi-drill feature for HTML and DHTML provides:

- ☐ Flexible cascading menus.
- ☐ Fully customizable styling at each element and level within the menu.
- ☐ Menus intelligently positioned in relationship to the data elements selected.

For PDF, PS, PPT, PPTX, EXL2K, and XLSX formats, the first active link is used to create a hyperlink on the designated location.

## Accessibility Support

The multi-drill feature provides 508 accessibility support to the cascading multi-drill menus available in HTML format.

In reports, for the multi-drill cascading menu options Auto Drill and Auto Link, accessibility users should:

- ☐ Turn the Virtual PC cursor setting mode off.
- ☐ Use keystrokes to navigate to the link in the report.

- ❑ Open the cascading menu item and then press Enter to view the item.
- ❑ Turn the Virtual PC cursor setting on to navigate through the report.

For more information, see *TM4505: WebFOCUS HTML Report Accessibility Support*.

### Creating Multiple Drill-Down Links

You can customize the drill-down menu at two levels:

- ❑ Global styling of all menus within the current procedure (fex).
- ❑ Item level styling for each entry in the menu.

### Global Menu Styling

To define styling attributes for all menus within the current procedure (fex):

```
TYPE=REPORT, OBJECT=MENU, [FONT=font], [SIZE=size], [COLOR=color],  
[HOVER-COLOR=hover_color], [BACKCOLOR=backcolor],  
[HOVER-BACKCOLOR=hover_backcolor], [BORDER={ON|OFF|n}],  
[BORDER-COLOR=border_color], [BORDER-STYLE=border_style]  
$
```

where:

*font*

Defines the font typeface for the menu item. The default is inherited from the report.

*size*

Defines the font size for the menu item. The default font size is 9.

*color*

Defines the text color for the menu item (named colors or RGB/HEX values). The default text color for the menu item is RGB(#6B6B6B). Also used to define the color of the SEPARATOR line and the control caret.

*hover\_color*

Defines the text color for the hover over or select menu item (named colors or RGB/HEX values). The default text color for hover over or select menu item is RGB(#495263).

*backcolor*

Defines the background color for the menu item (named colors or RGB/HEX values). The default background color is RGB(#F8F8F8).

*hover\_backcolor*

Defines the background color for hover over or select menu item (named colors or RGB/HEX values). The default background color for hover over or select menu item is RGB(#DFDFDF).

`BORDER={ON|OFF|n}`

where:

- ☐ ON displays borders around the menu objects and as the separator lines, based on user defined styling or system defaults.
- ☐ OFF displays no border around the menu objects.
- ☐ *n* is border weight in pixels (valid values 1, 2, 3, light, medium, heavy).

The default border weight is light.

*border\_color*

Defines border coloring to be used for borders around the menu. This will also be used for the color of the separator lines within the menu. Is one of the preset color values. The default border color is RGB(#D6D6D6).

*border\_style*

Defines line styles to be used for borders around the menu, as well as separator lines. Possible values are listed in the following table. This will also be used for separator style within the menu. The default border style is solid. Seeing the distinction in border style may require using a heavier weight (for example, border=heavy, or border=3).

Style	Description
NONE	No border/divider
SOLID	Solid line
DOTTED	Dotted line
DASHED	Dashed line
DOUBLE	Double line
GROOVE	3D groove
RIDGE	3D ridge

Style	Description
INSET	3D inset
OUTSET	3D outset

**Note:**

- ❑ If a multi-drill menu is tagged with the DRILL-SOURCE attribute, it indicates that the menu was generated by WebFOCUS and should be merged into the existing drilldowns added by the user for the specified report element, if any. The value of the attribute indicates which WebFOCUS feature generated the menu. This attribute is reserved for Information Builders internal use only.
- ❑ When you create multiple drill-down links, you cannot specify a single drill-down action (for example, FOCEXEC or URL) before the first DRILLMENUITEM.

**Menu Items Styling**

The syntax for cascading menus is an extension of the existing multi-drill (DRILLMENUITEM) syntax. Any syntax that is currently valid should behave the same after the extended syntax is implemented.

To define individual menus and items attached to a report node or data element:

```
TYPE=type, [subtype], [DRILLMENUITEM='description', action|'keyword' ],  
[NAME=name], [PARENT=parentname],
```

where:

*type*

Identifies the report component that you select in the web browser to execute the link. The TYPE attribute and its value must appear at the beginning of the declaration.

*subtype*

Are any additional attributes, such as COLUMN, LINE, or ITEM, that are needed to identify the report component that you are formatting.

Each DRILLMENUITEM item must have a description or a keyword pair. Descriptions without actions will automatically be inactive by default.



The exception to this rule will be parent items containing children entries linked with the NAME/PARENT pairing. In this instance, the action will be to present the children in the cascading menu.

#### *description*

Is the text that appears on the pop-up menu of drill-down options on the report output. The default value is DrillDown *n*, where *n* is a consecutive integer, such as DrillDown 1, DrillDown 2, and so on.

#### **Note:**

- ☐ If DRILLMENUITEM is set to the special value 'SEPARATOR':
  - ☐ A horizontal separator line will be drawn using the styling and color attributes defined for the menu borders at the location within the menu.
  - ☐ A separator cannot be associated with an action.
- ☐ The DRILLMENUITEM value cannot be empty or blank.

#### *action*

Is the type of link, as described in [Drill-Down Action Options](#) on page 789. For example, a link to a detail report or URL.

The following attributes are optional. They are only required for cascading menus where a hierarchy must be defined.

#### *name*

An optional unique identifier for the current item to use as a link between parent and children items. Only required if this node serves as a parent to children menu items where a link must be identified.

#### *parentname*

An optional unique identifier/name of the parent menu item for the current child item. Only required if this node serves as a parent to another item in the hierarchy.

### **Drill-Down Action Options**

Each drill menu item can be linked to a single instance of the actions below:

`FOCEXEC=report.fex`

Another report. The StyleSheet attribute is FOCEXEC.

```
TYPE=type, [subtype], FOCEXEC=fex(parameters...), [TARGET=frame,]
[ALT='description',] $
```

*URL=url string*

A URL. The StyleSheet attribute is URL. You pass a valid URL. Note that the length of the URL is limited by the maximum number of characters allowed by the browser. For information about this limit for your browser, see the browser support site.

```
TYPE=type, [subtype], URL=url[(parameters...)], [TARGET=frame,]  
[ALT='description'], $
```

*URL=(field)*

A URL from a field. The StyleSheet attribute is URL. You pass the name of a report column whose value is a valid URL to which the link will jump.

```
TYPE=type, [subtype], URL=url[(parameters ...)], [TARGET=frame,]  
[ALT='description'], $
```

*JAVASCRIPT = function*

A JavaScript function. The StyleSheet attribute is JAVASCRIPT.

```
TYPE=type, [subtype], JAVASCRIPT=function[(parameters ...)], $
```

### Summary of Drill-Down Links

Within a multi-drill menu, you can link to:

- ❑ Another report. The StyleSheet attribute is FOCEXEC. For details on the syntax, see [Linking to Another Report](#) on page 764.
- ❑ A URL. The StyleSheet attribute is URL. You pass a valid URL. For details on the syntax, see [Linking to a URL](#) on page 769.  
  
Note that the length of the URL is limited by the maximum number of characters allowed by the browser. For information about this limit for your browser, search on your browser vendor support site.
- ❑ A URL from a field. The StyleSheet attribute is URL. You pass the name of a report column whose value is a valid URL to which the link will jump. For details on the syntax, see [Linking to a URL](#) on page 769.
- ❑ A JavaScript function. The StyleSheet attribute is JAVASCRIPT. For details on the syntax, see [Linking to a JavaScript Function](#) on page 776.
- ❑ A Maintain Data procedure. The StyleSheet attribute is URL with the keyword MNTCON EX. For details on the syntax, see [Linking to a Maintain Data Procedure](#) on page 779.

- ❑ A WebFOCUS compiled Maintain Data procedure. The StyleSheet attribute is URL with the keyword MNTCON RUN. For details on the syntax, see [Linking to a Maintain Data Procedure](#) on page 779.

### Sample Drill Menu Stylesheet Code

```
TABLE FILE GGSALES
SUM
    GGSALES.SALES01.UNITS
    GGSALES.SALES01.DOLLARS
BY GGSALES.SALES01.REGION
BY GGSALES.SALES01.CATEGORY
BY GGSALES.SALES01.PRODUCT
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE SET ASNAMES ON
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLRENDERING ON
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
FONT=TAHOMA, GRID=OFF, $
```

```

TYPE=DATA,COLUMN=B2,
  DRILLMENUITEM='Sales Details', NAME=menu2,
  DRILLMENUITEM='By Month',
    PARENT=menu2, NAME=menu21,
    FOCEXEC=detailreport.fex(PARAMETER=CATEGORY),TARGET=_blank,
  DRILLMENUITEM='By Quarter',
    PARENT=menu2, NAME=menu23,
    FOCEXEC=detailreport.fex(PARAMETER=CATEGORY),TARGET=_blank,
  DRILLMENUITEM=SEPARATOR, PARENT=menu2,
  DRILLMENUITEM='By Product',
    PARENT=menu2, NAME=menu24,
    FOCEXEC=detailreport.fex(PARAMETER=CATEGORY),TARGET=_blank,
  DRILLMENUITEM='By Customer',
    PARENT=menu2,NAME=menu25,
    FOCEXEC=detailreport.fex(PARAMETER=CATEGORY),TARGET=_blank,
  DRILLMENUITEM=SEPARATOR, PARENT=menu2,
  DRILLMENUITEM='Profitablity Analysis',
    PARENT=menu2,NAME=menu3,
  DRILLMENUITEM='By Month',
    PARENT=menu3, NAME=menu31,
    FOCEXEC=detailreport.fex(PARAMETER=CATEGORY),TARGET=_blank,
  DRILLMENUITEM='By Region',
    PARENT=menu3, NAME=menu32,
    FOCEXEC=detailreport.fex(PARAMETER=CATEGORY),TARGET=_blank,
  DRILLMENUITEM='Forecasts',
    FOCEXEC=detailreport.fex(PARAMETER=CATEGORY),TARGET=_blank,
$
TYPE=DATA,COLUMN=B3,
  DRILLMENUITEM='IBI Links',NAME=menu4a,
  DRILLMENUITEM='Information Builders',
    PARENT=menu4a, NAME=menu41,
    URL=http://www.ibi.com,TARGET=_blank,
  DRILLMENUITEM='Summit 2015',
    PARENT=menu4a, NAME=menu42,
    URL=http://www.ibi.com,TARGET=_blank,
  DRILLMENUITEM='Competative Analysis',
    PARENT=menu4a, NAME=menu43,
    URL=http://www.ibi.com,TARGET=_blank,
  DRILLMENUITEM='External Links', NAME=menu4b,
  DRILLMENUITEM='Google',
    PARENT=menu4b, NAME=menu45, URL=http://www.google.com,TARGET=_blank,
  DRILLMENUITEM='Weather',
    PARENT=menu4b, NAME=menu46, URL=http://www.weather.com,TARGET=_blank,
  DRILLMENUITEM='CNN',
    PARENT=menu4b,NAME=menu47, URL=http://www.cnn.com,TARGET=_blank,
$
ENDSTYLE
END

```

This code generates a menu structure that looks like the following images.

Region	Category	Product	Unit Sales	Dollar Sales
Midwest	<a href="#">Coffee</a>	<a href="#">Espresso</a>	101154	1294947
		<a href="#">Sales Details &gt;</a>	231623	2883566
	<a href="#">Food</a>	<a href="#">Biscotti</a>	86105	1091727
		<a href="#">Croissant</a>	139182	1751124
		<a href="#">Scone</a>	116127	1495420
	<a href="#">Gifts</a>	<a href="#">Coffee Grinder</a>	50393	619154
		<a href="#">Coffee Pot</a>	47156	599878
		<a href="#">Mug</a>	86718	1086943
		<a href="#">Thermos</a>	46587	577906
Northeast	<a href="#">Coffee</a>	<a href="#">Capuccino</a>	44785	542095

Region	Category	Product	Unit Sales	Dollar Sales
Midwest	<a href="#">Coffee</a>	<a href="#">Espresso</a>	101154	1294947
		<a href="#">Sales Details &gt;</a>	231623	2883566
	<a href="#">Food</a>	<a href="#">Forecasts &gt;</a>	86105	1091727
		<a href="#">By Month</a>	139182	1751124
		<a href="#">By Quarter</a>	116127	1495420
	<a href="#">Gifts</a>	<a href="#">By Product</a>	50393	619154
		<a href="#">By Customer</a>	47156	599878
		<a href="#">Profitability Analysis &gt;</a>	86718	1086943
		<a href="#">Thermos</a>	46587	577906
Northeast	<a href="#">Coffee</a>	<a href="#">Capuccino</a>	44785	542095

Region	Category	Product	Unit Sales	Dollar Sales
Midwest	<a href="#">Coffee</a>	<a href="#">Espresso</a>	101154	1294947
		<a href="#">Sales Details &gt;</a>	231623	2883566
	<a href="#">Food</a>	<a href="#">Forecasts &gt;</a>	86105	1091727
		<a href="#">By Month</a>	139182	1751124
		<a href="#">By Quarter</a>	116127	1495420
	<a href="#">Gifts</a>	<a href="#">By Product</a>	50393	619154
		<a href="#">By Customer</a>	47156	599878
		<a href="#">Profitability Analysis &gt;</a>	86718	1086943
		<a href="#">By Region</a>	46587	577906
Northeast	<a href="#">Coffee</a>	<a href="#">Capuccino</a>	44785	542095

Region	Category	Product	Unit Sales	Dollar Sales
Midwest	<a href="#">Coffee</a>	<a href="#">Espresso</a>	101154	1294947
		<a href="#">Latte</a>	231623	2883566
	<a href="#">Food</a>	<a href="#">Biscotti</a>	86105	1091727
		<a href="#">Croissant</a>	139182	1751124
	<a href="#">Gifts</a>	<a href="#">Scone</a>	116127	1495420
		<a href="#">Coffee Grinder</a>	50393	619154
		<a href="#">Coffee Pot</a>	47156	599878
		<a href="#">Mug</a>	86718	1086943
		<a href="#">Thermos</a>	46587	577906
		<a href="#">Capuccino</a>	44785	542095
Northeast	<a href="#">Coffee</a>	<a href="#">Capuccino</a>	44785	542095

Region	Category	Product	Unit Sales	Dollar Sales
Midwest	<a href="#">Coffee</a>	<a href="#">Espresso</a>	101154	1294947
		<a href="#">Latte</a>	231623	2883566
	<a href="#">Food</a>	<a href="#">Biscotti</a>	86105	1091727
		<a href="#">Croissant</a>	139182	1751124
	<a href="#">Gifts</a>	<a href="#">Scone</a>	116127	1495420
		<a href="#">Coffee Grinder</a>	50393	619154
		<a href="#">Coffee Pot</a>	47156	599878
		<a href="#">Mug</a>	86718	1086943
		<a href="#">Thermos</a>	46587	577906
		<a href="#">Capuccino</a>	44785	542095
Northeast	<a href="#">Coffee</a>	<a href="#">Capuccino</a>	44785	542095

To apply custom styling to the menus, add the following syntax to the StyleSheet:

```
TYPE=REPORT, OBJECT=MENU, FONT="COMIC SANS MS", COLOR=NAVY,
BACKCOLOR=GREY, HOVER-COLOR=GREY, HOVER-BACKCOLOR=NAVY, $
```

The menu structure will now look like the following images.

<u>Region</u>	<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Midwest	<a href="#">Coffee</a>	<a href="#">Espresso</a>	101154	1294947
		<a href="#">Latte</a>	231623	2883566
	<a href="#">Food</a>	<a href="#">Biscotti</a>	86105	1091727
		<a href="#">Sales By Month</a>	39182	1751124
		<a href="#">Forecast By Quarter</a>	16127	1495420
	<a href="#">Gifts</a>	<a href="#">By Product</a>	50393	619154
		<a href="#">By Customer</a>	47156	599878
		<a href="#">Profitability By Month</a>	6718	1086943
		<a href="#">Thermos By Region</a>	6587	577906
	<a href="#">Capuccino</a>		44785	542095
Northeast				

<u>Region</u>	<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Midwest	<a href="#">Coffee</a>	<a href="#">Espresso</a>	101154	1294947
		<a href="#">Latte</a>	231623	2883566
	<a href="#">Food</a>	<a href="#">Biscotti</a>	86105	1091727
		<a href="#">Croissant</a>	139182	1751124
		<a href="#">Scones</a>	116127	1495420
	<a href="#">Gifts</a>	<a href="#">Coffee Pot</a>	50393	619154
		<a href="#">Mug</a>	47156	599878
		<a href="#">Thermos</a>	6718	1086943
		<a href="#">Capuccino</a>	46587	577906
	<a href="#">Capuccino</a>		44785	542095
Northeast				

### **Reference:** Usage Notes for Multi-Drill Menus

The following interactive reports are supported with the multi-drill cascading menus with user defined styling:

- ☐ HFREEZE
- ☐ Accordion by Row (EXPANDBYROW, EXPANDBYROWTREE)
- ☐ Accordion by Column (EXPANDABLE)
- ☐ OLAP
- ☐ HTML TOC

**Note:** As of Release 8.2 Version 01, when the Multi-Drill and On Demand Paging features are enabled in a report, the cascading Multi-Drill menus do not display. Instead, the legacy Multi-Drill menus generated prior to Release 8.2.01 will be generated. Cascading menus are not available and all hyperlinks display on the same level on the menu.

### **Applying Conditional Styling**

You can apply conditional styling to a report component, using a phrase such as WHEN, and use it to select one of a number of different actions, depending on the value of fields in the report.

The WHEN condition must precede the DRILLMENUITEM syntax.

For details on creating conditions, see [Linking With Conditions](#) on page 806.

### **Example:** Applying Conditional Styling to a Multiple Drill-Down Report

Add the following boldface code to the sample summary report in [Creating Multiple Drill-Down Links](#) on page 786. Notice that the WHEN condition precedes the code for DRILLMENUITEM, as required.

When you run the summary report, the State field is in red instead of blue whenever budget dollars is greater than dollar sales, and the pop-up menu of drill-down options shows Detail Budget Report instead of DrillDown 1 and DrillDown 2.



```

.
.
.
TYPE=DATA,
    COLUMN=N1,
    COLOR='BLUE',
    STYLE=UNDERLINE,
    DRILLMENUITEM='DrillDown 1',
        URL=http://www.informationbuilders.com?,
    DRILLMENUITEM='DrillDown 2',
        FOCEXEC=DETAILREPORT(PARAMETER=N1),$.

TYPE=DATA,
    COLUMN=N1,
    COLOR='RED',
    STYLE=UNDERLINE,
    WHEN=BUDDOLLARS GT DOLLARS,
    DRILLMENUITEM='Detail Budget Report',
        FOCEXEC=DETAILREPORT(PARAMETER=N1),$.
.
.

```

Sample output is:

<u>State</u>	<u>Dollar Sales</u>	<u>Budget Dollars</u>
<u>CA</u>	\$7,642,261.00	\$7,586,347.00
<u>CT</u>	\$3,782,049.00	\$3,832,202.00
<u>FL</u>	\$3,923,215.00	\$3,870,405.00
<u>GA</u>	\$4,100,107.00	\$4,247,587.00
<b>Detail Budget Report</b>		866,856.00
<u>MA</u>	\$3,707,986.00	\$3,818,397.00
<u>MO</u>	\$3,761,286.00	\$3,646,838.00
<u>NY</u>	\$3,902,265.00	\$3,926,322.00
<u>TN</u>	\$3,687,057.00	\$3,689,979.00
<u>TX</u>	\$3,714,978.00	\$3,680,679.00
<u>WA</u>	\$4,010,685.00	\$4,055,166.00

## Creating Parameters

If your drill-down report depends on a specific data value in the base report, you must create a parameter (or parameters) that can pass one or more values to the report you are drilling down to.

Parameters are useful when you want to create a dynamic link. For example, your first report is a summary report that lists the total number of products ordered by a company on a specific date. You can drill down from a specific product in that report to a more detailed report that shows the name of the product's vendor and the individual number of units ordered by order number. With a dynamic link, you create only one drill-down report that uses the value passed from the first report to determine what information to display, instead of several static reports.

You can create multiple parameters. The entire string of parameters must be enclosed in parentheses, separated from each other by a blank space, and cannot exceed 2400 characters.

You can use any combination of the following methods to create parameters in your StyleSheet declaration. You can specify:

- ☐ A constant value.
- ☐ The name or the position of a field.
- ☐ The name of an amper variable to pass its value. Amper variables can only be used with inline StyleSheets. For details on inline StyleSheets, see [Creating and Managing a WebFOCUS StyleSheet](#) on page 1117.

### **Syntax:**      **How to Create Parameters**

*parameter=value*

where:

*parameter*

Is the name of the variable in the linked procedure.

**Note:** To avoid conflicts, do not name variables beginning with Date, IBI, or WF. Variable names beginning with these values are reserved for Information Builders use.

*value*

Identifies the value to be passed. Values can be any of the following:

*'constant\_value'* identifies an actual value to be passed. The value must be enclosed in single quotation marks.

*field* identifies the field in the report whose value is to be passed to the procedure. You can identify the field using either the field name or the field position. For details on field position, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

*'&variable'* identifies an amper variable whose value is to be passed to the procedure. The name of the amper variable must be enclosed in single quotation marks. You can use amper variables only in inline StyleSheets.

**Note:** The usual use of an amper variable is to pass a constant value. If the amper variable corresponds to an alphanumeric field, the amper variable would have to be embedded in single quotation marks, for example:

```
'&ABC' .
```

The entire string of parameter names and values must be enclosed in parentheses. Each *parameter=value* pair must be separated by a blank space. You can include multiple parameters in your request but the entire string cannot exceed 2400 characters.

**Note:** If the drill-down report contains a -DEFAULTS statement that sets a default value to the same amper variable passed from the main report, the amper variable value passed down overwrites the -DEFAULTS statement in the target procedure.

### **Example:** Creating Parameters by Specifying a Constant Value

The following example illustrates how to create parameters by specifying a constant value. The relevant StyleSheet declarations are highlighted in the request.

Main report:

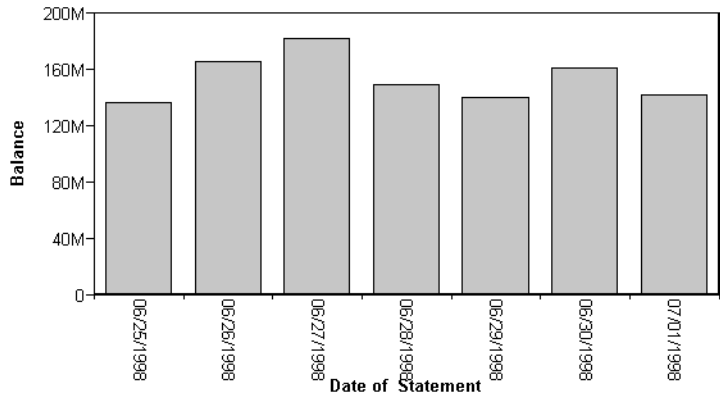
```
SET LOOKGRAPH BAR
SET 3D=OFF
GRAPH FILE SHORT
HEADING
"Sum of Balance Across Short Date"
"Click Any Bar For a Report on Projected Returns Since June 29, 1998 "
SUM BALANCE
ACROSS SHORT_DATE
ON GRAPH SET STYLE *
TYPE=DATA, ACROSSCOLUMN=N1,FOCEXEC=PROJRET(Short_Date='06291998'),$
ENDSTYLE
END
```

Drill-down report (PROJRET):

```
TABLE FILE SHORT
HEADING
"Projected Returns Since June 29, 1998 "
SUM PROJECTED_RETURN
BY SHORT_DATE
BY REGION
WHERE SHORT_DATE GE '&Short_Date';
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output for the main report is:

**Sum of Balance Across Short Date**  
Click Any Bar For a Report on Projected Returns Since June 29, 1998



When you click a bar the output is:

Projected Returns Since June 29, 1998

Date of Statement	Region	Projected Annualized Return
06/29/1998	CENTRAL AMERICA	1.360
	EASTERN EUROPE	2.300
	FAR EAST	1.300
	MIDDLE EAST	1.140
	NORTH AMERICA	1.780
	SOUTH AMERICA	1.200
	WESTERN EUROPE	1.140
06/30/1998	CENTRAL AMERICA	1.360
	EASTERN EUROPE	2.350
	FAR EAST	1.300
	MIDDLE EAST	1.140

	NORTH AMERICA	1.780
	SOUTH AMERICA	1.200
	WESTERN EUROPE	1.140
07/01/1998	CENTRAL AMERICA	1.360
	EASTERN EUROPE	2.300
	FAR EAST	1.300
	MIDDLE EAST	1.140
	NORTH AMERICA	1.780
	SOUTH AMERICA	1.200
	WESTERN EUROPE	1.140

### **Example:** Creating Parameters By Specifying a Field

The following example illustrates how to create a parameter by specifying a field, in this case CATEGORY. The SALES drill-down report (the report that is linked to the main report) sets the CATEGORY field equal to &TYPE. In the base report, TYPE is set to equal the field CATEGORY.

When you run the report, the values for the field CATEGORY (Coffee, Food, Gifts) are linked to a report that contains the product and regional breakdowns for the respective value.

Main report:

```
TABLE FILE GGSales
SUM UNITS DOLLARS
BY CATEGORY
HEADING
"* Click category to see product and regional breakdowns."
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=DATA, COLUMN=CATEGORY, FOCExec=SALES (TYPE=CATEGORY), $
ENDSTYLE
FOOTING
"This report was created on &DATE ."
END
```

Drill-down report (SALES):

```
TABLE FILE GGSales
ON TABLE SET PAGE-NUM OFF
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT
ACROSS REGION
WHERE CATEGORY = '&TYPE';
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output for the main report is:

### Sales Report

\* Click category to see product and regional breakdowns.

Category	Unit Sales	Dollar Sales
Coffee	1376266	17231455
Food	1384845	17229333
Gifts	927880	11695502

This report was created on 05/11/01 .

Click *Coffee* and the product and regional breakdown for Coffee displays:

		Region							
		Midwest		Northeast		Southeast		West	
Category	Product	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales
Coffee	Capuccino	.	.	44785	542095	73264	944000	71168	895495
	Espresso	101154	1294947	68127	850107	68030	853572	71675	907617
	Latte	231623	2883566	222866	2771815	209654	2617836	213920	2670405

**Example:**    **Creating Parameters by Specifying an Amper Variable**

The following request illustrates how to create a parameter by specifying an amper variable. The relevant StyleSheet declarations are highlighted in the request.

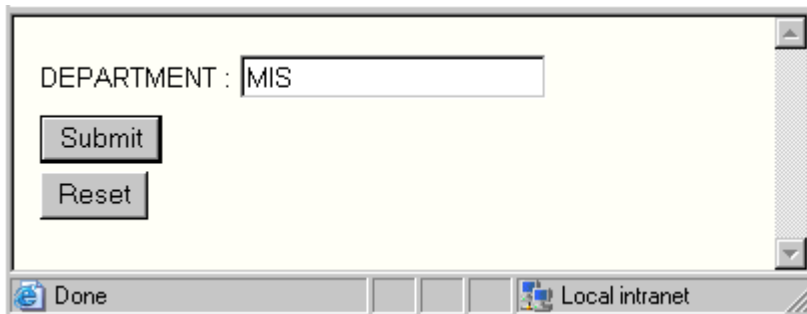
Main report:

```
SET3D=OFF
GRAPH FILE EMPLOYEE
HEADING
"Salary Report Per Employee ID"
"Click A Bar For The List of Employees in the '&DEPARTMENT' Department"
SUM SALARY
ACROSS EMP_ID AS 'EMPLOYEE ID'
ON GRAPH SET STYLE *
TYPE=DATA, ACROSSCOLUMN=SALARY,
FOCEXEC=EMPBYDEP (DEPARTMENT='&DEPARTMENT'), $
ENDSTYLE
END
```

Linked report (EMPBYDEP):

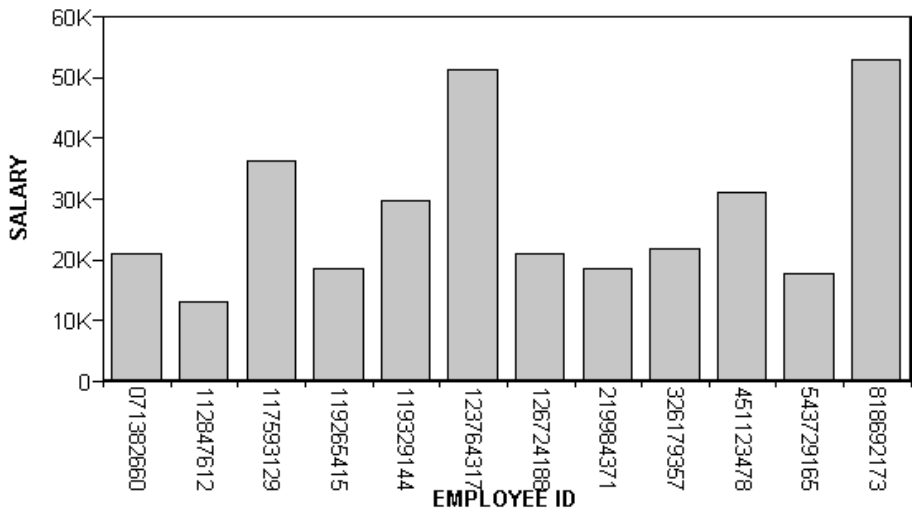
```
TABLE FILE EMPLOYEE
HEADING
"List Of Employees in the '&DEPARTMENT' Department "
PRINT FIRST_NAME LAST_NAME
BY DEPARTMENT
WHERE DEPARTMENT EQ '&DEPARTMENT';
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

When the main report request is run, the following prompt opens:



Enter *MIS* and click *Submit*. The output is:

**Salary Report Per Employee ID**  
Click A Bar For The List of Employees in the 'MIS' Department



When you click a bar on the graph, the output is:

List Of Employees in the 'MIS' Department

<u>DEPARTMENT</u>	<u>FIRST NAME</u>	<u>LAST NAME</u>
MIS	MARY	SMITH
	DIANE	JONES
	JOHN	MCCOY
	ROSEMARIE	BLACKWOOD
	MARY	GREENSPAN
	BARBARA	CROSS

**Example: Using Multiple Parameters**

When using multiple parameters, the entire string must be enclosed in parentheses and separated from each other by a blank space. The relevant StyleSheet declarations are highlighted in the request.



Main report:

```
SET 3D=OFF
GRAPH FILE EMPLOYEE
SUM CURR_SAL
ACROSS DEPARTMENT
ON GRAPH SET STYLE *
TYPE=DATA, ACROSSCOLUMN=CURR_SAL,
FOCEXEC=REPORT2 (DEPARTMENT='&DEPARTMENT' LAST_NAME='SMITH'), $
ENDSTYLE
END
```

Drill-down report (REPORT2):

```
TABLE FILE EMPLOYEE
PRINT SALARY
BY DEPARTMENT
BY FIRST_NAME
BY LAST_NAME
WHERE DEPARTMENT EQ '&DEPARTMENT'
WHERE LAST_NAME EQ '&LAST_NAME'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

When the main report request is run, the following prompt opens:

WebFOCUS DESCRIBE Generated Form - Microsoft Int...

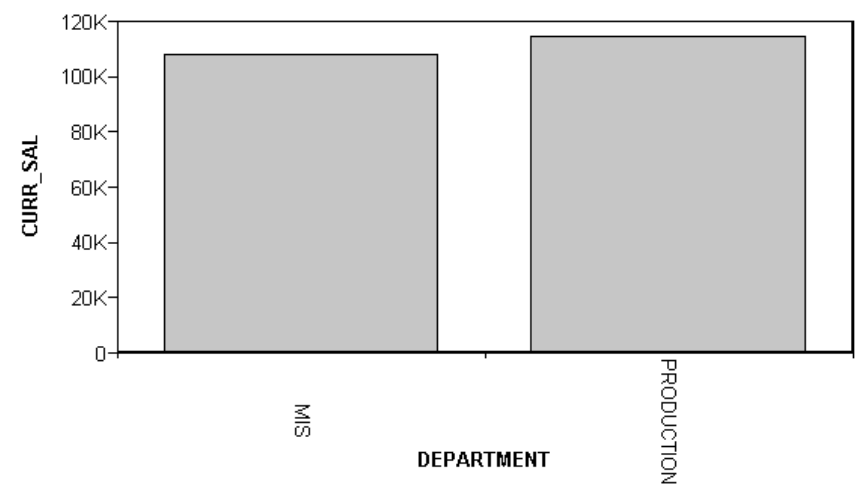
File Edit View Favorites Tools Help

Address [http://localhost/cgi-bin/ibi\\_cgi/webapi.dll](http://localhost/cgi-bin/ibi_cgi/webapi.dll) Go

DEPARTMENT:

Done Local intranet

Enter *MIS* and click *Submit*. The output is:



When you click the MIS bar, the output is:

<u>DEPARTMENT</u>	<u>FIRST NAME</u>	<u>LAST NAME</u>	<u>SALARY</u>
MIS	MARY	SMITH	\$13,200.00

Linking With Conditions

You can create conditions when linking to a report, URL, or JavaScript function from a report or graph. For example, you may only be interested in displaying current salaries for a particular department. You can accomplish this by creating a WHEN condition.

For complete details on WHEN, see [Controlling Report Formatting](#) on page 1139.

**Note:** Linking with conditions is not supported in GRAPH requests.

*Syntax:*      **How to Link With Conditions**

To specify a conditional link to a report use:

```
TYPE=type, [subtype], FOCEXEC=fex[(parameters...)],  
  WHEN=expression,[TARGET=frame,] $
```

To specify a conditional link to a URL use:

```
TYPE=type, [subtype], URL=url[(parameters...)],  
  WHEN=expression,[TARGET=frame,] $
```

To specify a conditional link to a JavaScript function use:

```
TYPE=type, [subtype], JAVASCRIPT=function(parameters...),  
    WHEN=expression, [TARGET=frame,] $
```

where:

#### *type*

Identifies the report component that you select in the web browser to execute the link. The TYPE attribute and its value must appear at the beginning of the declaration.

#### *subtype*

Are any additional attributes, such as COLUMN, LINE, or ITEM, that are needed to identify the report component that you are formatting. For information on identifying report components, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

#### *fex*

Identifies the file name of the linked procedure to run when you select the report component. For details about linking to another procedure, see [Linking to Another Report](#) on page 764.

#### *url*

Identifies any valid URL, or the name of a report column enclosed in parentheses whose value is a valid URL. For details about linking to a URL, see [Linking to a URL](#) on page 769.

#### *function*

Identifies the JavaScript function to run when you select the report component. For details about calling a JavaScript function, see [Linking to a JavaScript Function](#) on page 776.

#### *parameters*

Values that are passed to the report, URL, or JavaScript function. For details, see [Creating Parameters](#) on page 797.

#### *expression*

Is any Boolean expression that would be valid on the right side of a COMPUTE expression.

**Note:** IF... THEN... ELSE logic is not necessary in a WHEN clause and is not supported. All non-numeric literals in a WHEN expression must be specified within single quotation marks.

#### *frame*

Identifies the target frame in the webpage in which the output from the drill-down link is displayed. For details, see [Specifying a Target Frame](#) on page 814.

**Example:    Linking With Conditions**

In this example, we only want to link the MIS value of the DEPARTMENT field to REPORT3. To do this we include the phrase WHEN=DEPARTMENT EQ 'MIS' in the StyleSheet declaration. The relevant declarations are highlighted in the requests.

Main report:

```
TABLE FILE EMPLOYEE
SUM CURR_SAL AS 'Total,Current,Salaries'
BY DEPARTMENT AS 'Department'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=DATA, COLUMN=N1, FOCEXEC=REPORT3 (DEPARTMENT=N1),
WHEN=DEPARTMENT EQ 'MIS', $
ENDSTYLE
END
```

Drill-down report (REPORT3):

```
TABLE FILE EMPLOYEE
PRINT SALARY
BY DEPARTMENT
BY LAST_NAME
WHERE DEPARTMENT EQ '&DEPARTMENT'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

In the following output, note that only the MIS department is linked:

<u>Department</u>	<u>Total Current Salaries</u>
<u>MIS</u>	\$108,002.00
PRODUCTION	\$114,282.00

When you click MIS, the following output displays:

<u>DEPARTMENT</u>	<u>LAST NAME</u>	<u>SALARY</u>
MIS	BLACKWOOD	\$21,780.00
	CROSS	\$27,062.00
		\$25,755.00

GREENSPAN	\$9,000.00
	\$8,650.00
JONES	\$18,480.00
	\$17,750.00
MCCOY	\$18,480.00
SMITH	\$13,200.00

## Linking From a Graphic Image

You can link to a report or procedure from an image in an HTML report. The image can be attached to the entire report or to the report heading or footing (this includes table headings/table footings, and sub-headings/sub-footings).

The syntax for linking from a graphic image is the same as for linking from a report component. The only difference is adding `IMAGE=image` to the StyleSheet declaration.

**Note:** You can only link to a report or procedure from an image when you are using HTML format.

### *Syntax:* How to Specify Links From a Graphic Image

To specify a link from an image in a report or procedure use:

```
TYPE=type, [subtype], IMAGE=image, FOCEXEC=fex
[(parameters ...)],[TARGET=frame,] $
```

To specify a link from an image in an URL use:

```
TYPE=type, [subtype], IMAGE=image, URL=url
[(parameters ...)],[TARGET=frame,] $
```

To specify a link from an image in a JavaScript function use:

```
TYPE=type, [subtype], IMAGE=image, JAVASCRIPT=function
[(parameters ...)],$
```

where:

*type*

Identifies the report component that the user selects to execute the link. The TYPE attribute and its value must appear at the beginning of the declaration. You can specify the following types of components:

**REPORT** enables you to drill down from a graphical image that is attached to the entire report.

[TABHEADING](#) or [TABFOOTING](#) enables you to drill down from a graphical image that is attached to a report heading or footing.

[HEADING](#) or [FOOTING](#) enables you to drill down from a graphical image that is attached to a page heading or footing.

[SUBHEAD](#) or [SUBFOOT](#) enables you to drill down from a graphical image that is attached to a sub heading or sub footing.

Report components are described in [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

### *subtype*

Are any additional attributes, such as COLUMN, LINE, or ITEM, that are needed to identify the report component that you are formatting. See [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167 for information on identifying report components.

### *image*

Specifies the file name of a graphical image file. The image must exist as a separate graphic file in a format that your browser supports. Most browsers support GIF and JPEG file types.

You can specify a local image file, or identify an image elsewhere on the network using a URL. URLs can be absolute, such as, <http://www.ibi.com/graphic.gif>, or relative alias that can be identified to the application server or web server, such as, [/ibi\\_apps/ibi\\_html/ibi\\_logo.gif](#).

Alternatively, you can specify an alphanumeric field in the report (either a BY sort field or a display field) whose value corresponds to the name of the image file. For information about using StyleSheets to incorporate and position graphical images in a report, see [Laying Out the Report Page](#) on page 1249.

### *fex*

Identifies the file name of the linked procedure to run when the user selects the report component. For details about linking to another procedure, see [Linking to Another Report](#) on page 764.

### *url*

Identifies any valid URL, or the name of a report column enclosed in parentheses whose value is a valid URL. For details about linking to an URL, see [Linking to a URL](#) on page 769.

### *function*

Identifies the JavaScript function to run when the user selects the report component. For details about calling a JavaScript function, see [Linking to a JavaScript Function](#) on page 776.

*parameters*

Are values that are passed to the report, URL, or JavaScript function. You can pass one or more parameters. The entire string of parameters must be enclosed in parentheses, and separated from each other by a blank space. For details, see [Creating Parameters](#) on page 797.

*frame*

Identifies the target frame in the webpage in which the output from the drill-down link is displayed. For details, see [Specifying a Target Frame](#) on page 814.

**Note:** You cannot specify a target frame if you are executing a JavaScript function. However, the JavaScript function itself can specify a target frame for its results.

**Example: Specifying a Link From an Image**

The following example illustrates how to link a report from an image. The relevant StyleSheet declarations are highlighted in the request.

Main report:

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME BY EMP_ID
HEADING
"List Of Employees By Employee ID"
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=HEADING, STYLE=BOLD, $
TYPE=REPORT, GRID=OFF, $
TYPE=REPORT,
IMAGE=E:\IBI\WEBFOCUS81\APPS\IBINCCEN\IMAGES\LEFTLOGO.GIF,
FOCEXEC=IMAGE-D, $
ENDSTYLE
END
```

**Note:** The IBINCCEN directory contains the English version of the samples.

Drill-down report (IMAGE-D):

```
TABLE FILE EMPDATA
PRINT SALARY
BY DIV
WHERE DIV LE 'CORP';
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output for the main report is:



**List Of Employees By Employee ID**

<u>EMP ID</u>	<u>LAST NAME</u>
071382660	STEVENS
112847612	SMITH
117593129	JONES
119265415	SMITH
119329144	BANNING
123764317	IRVING
126724188	ROMANS
219984371	MCCOY
326179357	BLACKWOOD
451123478	MCKNIGHT
543729165	GREENSPAN
818692173	CROSS

When you click the graphic, the output is:

<u>DIV</u>	<u>SALARY</u>
CE	\$62,500.00
	\$54,100.00
	\$25,400.00
	\$115,000.00
	\$33,300.00
	\$25,000.00
	\$49,000.00
	\$40,900.00
	\$43,000.00
	\$45,000.00
CORP	\$55,500.00
	\$83,000.00



\$32,000.00  
 \$62,500.00  
 \$79,000.00  
 \$35,200.00  
 \$62,500.00  
 \$26,400.00

## Specifying a Base URL

If you want to link to files, images, and Java files, but do not know their full, physical URLs, you can specify a default location where the browser searches for relative URLs.

To specify a default URL location, use the SET BASEURL command. Using SET BASEURL puts `<BASE HREF="url">` into the HTML file that WebFOCUS generates. When a report is run, the specified directory is searched for the HTML files, graphics files, and Java applet CLASS files that are called by the generated webpage.

For more details on specifying URLs, see [Navigating Within an HTML Report](#) on page 905.

### **Syntax:** How to Specify a Base URL

```
SET BASEURL=url
```

where:

*url*

Is the default location where the browser searches for relative URLs specified in the HTML documents created by your application.

The URL must begin with `http://` and end with a closing delimiter (`/`).

### **Example:** Specifying a Base URL

The following illustrates how to specify a base URL:

```
SET BASEURL=http://host[:port]/ibi_apps/ibi_html/
```

where:

*host*

Is the host name where the WebFOCUS Web application is deployed.

*port*

Is the port number (specified only if you are not using the default port number) where the WebFOCUS Web application is deployed.

If you are including a graphic image in your report that is stored in the specified base URL, you can add the following declaration to your StyleSheet instead of typing the entire URL:

```
TYPE=HEADING, IMAGE=ib_logo.gif, ..., $
```

**Note:** If the URL is at a remote website, it may take longer to retrieve. Whenever possible, store graphic image files on your WebFOCUS system.

## Specifying a Target Frame

You can use frames to subdivide application HTML pages into separate scrollable sections. Frames enable users to explore various information items on a page by scrolling through a section, instead of linking to a separate page. When defining a link from a report component to a report procedure or URL, you can specify that the results of the drill-down link be displayed in a target frame on a webpage.

There are two ways to specify a target frame. You can specify:

- ❑ A target frame in a StyleSheet declaration using the TARGET attribute. You can use StyleSheets to specify that drill-down links from a report or graph are displayed in a target frame on the webpage displaying the report or graph. However, using StyleSheets to specify target frames adds extra HTML syntax to every HREF that is generated.

**Note:** When specifying a target frame from the Report canvas, manually added commands in the StyleSheet are not recognized. The Report canvas removes commands that it does not generate itself.

- ❑ A default target frame with a SET command. SET TARGETFRAME puts the HTML code `<BASE TARGET="framename">` into the header of the HTML file that WebFOCUS displays. All drill-down links from the base report or graph are directed to the specified frame, unless overridden by the TARGET attribute in the StyleSheet.

To use the TARGET attribute or the SET TARGETFRAME command, you must create multiple frames on the webpage.

**Note:** You cannot specify a target frame if you are executing a JavaScript function. However, the JavaScript function itself can specify a target frame for its results.

**Syntax:**      **How to Specify a Target Frame**

To specify a target frame in a report or procedure use:

```
TYPE=type, [subtype], FOCEXEC=fex[(parameters ...)], [TARGET=frame,] $
```

To specify a target frame for an URL use:

```
TYPE=type, [subtype], URL=url[(parameters ...)], [TARGET=frame,] $
```

where:

*type*

Identifies the report component that the user selects in the web browser to execute the link. The TYPE attribute and its value must appear at the beginning of the declaration.

*subtype*

Are any additional attributes, such as COLUMN, LINE, or ITEM, that are needed to identify the report component that you are formatting. See [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167 for information on identifying report components.

*fex*

Identifies the file name of the linked procedure to run when the user selects the report component. For details about linking to another procedure, see [Linking to Another Report](#) on page 764.

*url*

Identifies any valid URL, or the name of a report column enclosed in parentheses whose value is a valid URL to which the link will jump. For details about linking to an URL, see [Linking to a URL](#) on page 769.

*parameters*

Are values being passed to the procedure or URL. You can pass one or more parameters. The entire string of values must be enclosed in parentheses, and separated from each other by a blank space. For details, see [Creating Parameters](#) on page 797.

*frame*

Identifies the target frame in the web page in which the output from the drill-down link (either a FOCEXEC or URL) is displayed.

If the name of the target frame contains embedded spaces, the name will be correctly interpreted without enclosing the name in quotation marks. For example:

```
TYPE=DATA, COLUMN=N1,
FOCEXEC=MYREPORT, TARGET=MY FRAME, $
```

The name of the target frame is correctly interpreted to be MY FRAME.

You can also use the following standard HTML frame names: `_BLANK`, `_SELF`, `_PARENT`, `_TOP`.

### **Syntax:** How to Specify a Default Target Frame

```
SET TARGETFRAME=frame
```

where:

*frame*

Identifies the target frame in the webpage in which the output from the drill-down link (either a FOCEXEC or URL) is displayed.

### **Example:** Specifying a Target Frame

The following illustrates how to specify a default target frame:

```
SET TARGETFRAME=_SELF
```

The following illustrates how to specify a target frame in a request. The relevant StyleSheet declaration is highlighted in the request.

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL
BY DEPARTMENT
ON TABLE SET STYLE *
TYPE=DATA, COLUMN=N1, URL=http:\\www.informationbuilders.com,
TARGET=_SELF, $
ENDSTYLE
END
```

## Creating a Compound Report

Compound reports combine multiple reports into a single file. This enables you to concatenate reports with styled formats (such as PDF, DHTML, PS, EXL2K, or XLSX). You can also embed image files, including graphs saved as images, in a compound report.

Three types of compound reports exist:

- ❑ **Legacy Compound Reports (or, simply, Compound Reports).** These reports string the individual reports or graphs together sequentially into a single output file. The functionality for this type of compound report has been stabilized. The syntax is included in this document for upward compatibility, and applications using this technique will continue to work. For more information, see [Creating a Compound PDF or PS Report](#) on page 870 and [Creating a Compound Excel Report Using EXL2K](#) on page 879.

- ❑ **Compound Layout Reports.** A Compound Layout report is comprised of individual component reports or graphs, either embedded or external. Reports and graphs can be positioned anywhere on the page. You can assign specific pages to combinations of reports and specify how to handle overflow onto additional pages. For more information, see [Creating a Compound Layout Report With Document Syntax](#) on page 818.
- ❑ **Coordinated Compound Layout Reports.** A Coordinated Compound Layout report is *coordinated* so that all reports and graphs that contain a common sort field are burst into separate page layouts. Pages are generated for each value of the common sort field, with every component displaying the data it retrieved for that value on that page. You create a Coordinated Compound Layout report by specifying MERGE=ON in the SECTION declaration for the Compound Layout report.

In a Coordinated Compound Layout report, if at least one component contains data for a specific sort field value, a page is generated for that value even though some of the components may be missing. For more information about Coordinated Compound Layout reports with missing data, see [Coordinated Compound Layout Reports With Missing Data](#) on page 851.

While the length of the report will always include all of the rows of data generated by the query, the width of the report is limited by the size of the defined component container. This means that paneling is not supported for compound reports, although it is for non-compound PDF reports. For non-compound PDF documents, if the width of the report data is wider than the defined page size, a panel (or horizontal overflow page) is automatically generated. This paneling feature is not supported for compound PDF documents, so each compound component must fit within the width of the defined container in order for the report to be successfully generated. The container size is defined within each type of report.

- ❑ In legacy compound syntax, if one of the component reports is too large to fit within the defined page width, execution is halted and the user is presented with an error message stating that paneling is not supported.
- ❑ In Compound Layout Syntax, if a component is too wide to fit within the defined container, the report wraps the contents within the container. The container size is defined through a combination of the POSITION and DIMENSIONS parameters for the component within the compound syntax. For more information about compound layout syntax, see [Creating a Compound Layout Report With Document Syntax](#) on page 818.

For information about creating PDF Compound Reports with Drill Through links, see [How to Create a Drill Through in a PDF Compound Report](#) on page 892.

## Creating a Compound Layout Report With Document Syntax

Typically, you create a compound layout report by using the options in the Document canvas. Alternatively, you may create a compound layout report by modifying the syntax in any text editor.

Syntax for a compound layout report is structured by a compound layout block, which places all of the layout information in a single block that precedes the report. This block begins with a COMPOUND LAYOUT declaration and is terminated with END. The language it contains is based on StyleSheet syntax and is parsed by the StyleSheet parser.

This is supported with styled formats, such as PDF, PS, DHTML, EXL2K, or XLSX.

**Tip:** For details about StyleSheet syntax, see [Creating and Managing a WebFOCUS StyleSheet](#) on page 1117.

The compound layout block consists of SECTION, PAGELAYOUT, and COMPONENT declarations. The general structure of the compound layout block of syntax is:

```
COMPOUND LAYOUT PCHOLD AS filename FORMAT format

SECTION
    PAGELAYOUT
        COMPONENT
        COMPONENT
    ...
    PAGELAYOUT
        COMPONENT
        COMPONENT
    ...
END
...
COMPOUND END
```

**Note on Compound Layout Declaration:** The available compound layout output formats are PDF, DHTML, PowerPoint, AHTML, Excel, FLEX, and APDF. The selected compound layout format will override any report output format from the individual components. The output file name can be defined using an AS *filename* phrase in the COMPOUND block. If none is defined, the file name is taken from the ON TABLE HOLD phrase in the first component report.

END signifies the end of the COMPOUND LAYOUT block, whereas COMPOUND END signifies the end of the compound report.

Additionally, the syntax SET COMPONENT=report(n) is added after each component, followed by the actual WebFOCUS code to generate the report.

**Reference: SECTION Declaration and Syntax**

A compound report section, or SECTION declaration, is a grouping of component reports within a compound report. While the current functionality only supports reports with a single section, this structure is used to support more complex reports. The SECTION declaration is mandatory when creating a compound layout report.

The SECTION syntax appears as:

```
SECTION=section-name, LAYOUT=ON, [MERGE=ON|OFF],  
[UNITS=IN|CM|PTS,] [PAGESIZE=size,] [ORIENTATION=PORTRAIT|LANDSCAPE,]  
[LEFTMARGIN=m,] [RIGHTMARGIN=m,] [TOPMARGIN=m,] [BOTTOMMARGIN=m,] $
```

where:

*section-name*

Is the unique identifier of the section, up to 16 characters.

LAYOUT=ON

Specifies that the section uses a complex layout.

**Note:** LAYOUT=ON is the only applicable option at this time.

MERGE={ON|OFF}

Specifies whether the section is coordinated (merged) based on the value of the initial BY field.

**Note:** The default value is OFF.

*m*

Specifies the margins (LEFT, RIGHT, TOP, BOTTOM) in inches, centimeters, or points.

If the optional items, UNITS, PAGESIZE, ORIENTATION, or MARGIN are present in the SECTION declaration, they override any settings of these parameters within the component reports, global SET commands, ON TABLE SET commands, and StyleSheet keywords.

**Reference: PAGELAYOUT Declaration**

A SECTION consists of one or more PAGELAYOUT declarations, each of which group together a number of COMPONENT declarations that are laid out on that particular page of the section.

The PAGELAYOUT keyword brackets a group of COMPONENT declarations that follow, up to the next PAGELAYOUT keyword, or the end of the section.

The PAGELAYOUT syntax appears as:

```
PAGELAYOUT={n|ALL},  
[TOPMARGIN=m,] [BOTTOMMARGIN=m,] $
```

where:

`{n|ALL}`

Specifies on what page of a multi-page layout the components appear. The value (*n*) is either a page number within the current section, or ALL to indicate that the component appears on every page of the section.

- ❑ The PAGELAYOUT values are numbered starting with 1. For example, if a compound report is printed on two sides of a page, the component reports on the front side would be specified as PAGELAYOUT [1](#), and the reverse side would be PAGELAYOUT [2](#).

**Note:** Syntax is required even if the report only contains a single page.

`PAGELAYOUT=1, $`

- ❑ The PAGELAYOUT=ALL syntax specifies a component that appears on every page. This is useful for components that generate page headers or footers.

`PAGELAYOUT=ALL, $`

*m*

Defines the boundaries (TOP, BOTTOM) for flowing reports in current units. For a description of flowing reports, see [COMPONENT Declaration](#) on page 822.

## Reference: Page Masters

Components included in a declaration for PAGELAYOUT=ALL appear on every page of the report output. This is useful for creating a design theme for the compound report output:

- ❑ Page masters must have a default report component (since COMPONENT is a required syntax element), but that report component should not display any data, except in the heading. If the component displays data or graphics in the heading, you must leave space for the heading on every page, being careful to place the other components in areas on the page that do not overlap with the heading.

For example, the following report does not display any data:

```
SET COMPONENT='DfltCmpt1'
TABLE FILE SYSCOLUM
SUM TBNAME NOPRINT
IF READLIMIT EQ 1
ON TABLE SET PREVIEW ON
ON TABLE SET PAGE-NUM NOLEAD
END
```



- ❑ Page masters for coordinated compound reports must have the same primary BY field as other components in the compound layout report. The best way to create the default component for the page master in a coordinated compound report is to use one of the data sources for one of the compound layout reports. For example, if a component report for PAGELAYOUT1 is against the GGSALES data source, and the primary BY field is PCD, the default component for the page master could be:

```
SET COMPONENT='DfltCmpt1'
TABLE FILE GGSALES
SUM UNITS NOPRINT
BY PCD NOPRINT
IF READLIMIT EQ 1
ON TABLE SET PREVIEW ON
ON TABLE SET PAGE-NUM NOLEAD
END
```

Note that the recommended way to create a design theme with repeating text and images on a page master is to place drawing objects on the page master. For information, see [How to Draw Objects With Document Syntax](#) on page 834.

- ❑ Page master styling for size and orientation only applies to the default document level. Page master elements do not automatically resize and position for various page orientations throughout the document. For documents requiring mixed page orientations, each page layout can be defined with its own orientation. In this scenario, styling elements should be applied to the individual page layouts so that the styling can be appropriately applied for each change in page orientation.

**Reference: COMPONENT Declaration**

The order of COMPONENT declarations in the COMPOUND LAYOUT block must match the order in which the component reports are executed, and there must be a COMPONENT declaration for each component report.

There are two types of components: fixed and flowing. A fixed component fills the container defined by the dimension parameters on the page and, if additional data exists, it overflows onto the next page in the same fixed size in the same location. The size and location of the fixed overflow component can be customized on the overflow page (using the OVERFLOW-POSITION and OVERFLOW-DIMENSION parameters). In a flowing component, the data flows from the top of the defined component to the bottom page margin and then begins to flow again on the top page margin of the overflow page, until the data is complete. The starting position of each type of component is defined by the POSITION parameter. The fixed or flowing aspect of the component is determined by the DIMENSION parameter. For a fixed component, the DIMENSION parameter specifies sizes for the dimensions of the bounding box. However, for a flowing component the DIMENSION parameter specifies asterisks (\* \*) for the dimensions.

The COMPONENT syntax appears as:

```
COMPONENT=component-name, TYPE=component-type,
        POSITION=(x y), DIMENSION=(xsize ysize),
        [OVERFLOW-POSITION=(x y),] [OVERFLOW-DIMENSION=(xsize ysize),]
[DRILLMAP=((L1 targetreport)),]      $
```

where:

*component-name*

The name of the component must be a unique identifier, up to 16 characters. It designates a component report that appears later in the request (in the same procedure (FOCEXEC) or in a called procedure), and is identified by SET COMPONENT=*component-name* syntax, using the same name.

**Note:** The SET syntax only tags styled reports that can participate in a compound report, so it can be placed before unstyled reports that precede the report to be named. For example, reports that generate extract files.

*component-type*

Specifies the type of component being declared. Currently, only REPORT is supported.

POSITION=(*x y*)

Specifies the (x y) coordinate on the page where the upper-left corner of the component is to be placed. All coordinates are in current UNITS (default inches), and (0 0) is the upper-left corner of the physical page.

**Note:** By default, coordinates are absolute locations on the physical page. If x or y is preceded by a plus sign (+) or a minus sign (-), for example, (+.25 +0), the coordinate is relative to the left or top page margin.

`DIMENSION=(xsize ysize)`

Specifies the size of the bounding box of the component (in current UNITS).

For a fixed component, *xsize* and *ysize* must be numeric dimension sizes.

For a flowing component, *xsize* and *ysize* must both be asterisks `DIMENSION = (* *)`.

`OVERFLOW-POSITION=(x y)` and `OVERFLOW-DIMENSION=(xsize ysize)`

These optional items specify the position and dimension of a fixed component on subsequent pages, if it overflows its initial bounding box.

`DRILLMAP=((L1 targetreport))`

Identifies the link identifier and the target report for a Drill Through hyperlink from this report. L1 is a sufficient link identifier at this time. For more information on Drill Through reports, see [How to Create a Drill Through in a PDF Compound Report](#) on page 892.

**Note:** The double parentheses are required.

### **Example:** Creating a Compound Layout Report With Document Syntax

In this simple example using the GGSales Master File, the MERGE keyword specifies that a Coordinated Compound Layout Report is to be generated. Since the first BY field of each component is REGION, a page will be generated for each value of REGION, with the first report (Sales) positioned at (1 1) and the second report (Units) at (6.25 1).

Enter the following syntax in the Text Editor.

```
SET PAGE-NUM=OFF
COMPOUND LAYOUT PCHOLD FORMAT PDF
SECTION=S1, LAYOUT=ON, MERGE=ON, ORIENTATION=LANDSCAPE, $
PAGELAYOUT=1, $
COMPONENT=Sales, TYPE=REPORT, POSITION=(1 1), DIMENSION=(4 4), $
COMPONENT=Units, TYPE=REPORT, POSITION=(6.25 1), DIMENSION=(4 4), $
END
SET COMPONENT=Sales
TABLE FILE GGSales
"Sales report for <REGION>"
" "
SUM DOLLARS/F8M
BY REGION NOPRINT
BY ST
BY CITY
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, COLOR=RED, SQUEEZE=ON, $
END
SET COMPONENT=Units
TABLE FILE GGSales
"Number of unit sales per product for <REGION>"
" "
SUM CNT.UNITS AS 'Number of units sold'
BY REGION NOPRINT
BY PRODUCT
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, COLOR=BLUE, SQUEEZE=ON, $
ENDSTYLE
END
COMPOUND END
```

The following syntax is an example of what the same report might look like if the component reports were in pre-existing procedures (FOCEXECs), as indicated by R1 and R2.

```
SET PAGE-NUM=OFF
SET SQUEEZE=ON
COMPOUND LAYOUT PCHOLD FORMAT PDF
SECTION=S1, LAYOUT=ON, MERGE=ON, ORIENTATION=LANDSCAPE, $
PAGELAYOUT=1, $
COMPONENT=R1, TYPE=REPORT, POSITION=(1 1), DIMENSION=(4 4), $
COMPONENT=R2, TYPE=REPORT, POSITION=(6 1), DIMENSION=(4 4), $
END
SET COMPONENT=R1
EX REPORT1
SET COMPONENT=R2
EX REPORT2
COMPOUND END
```

The first page of output is:

PAGE 1			PAGE 5	
Sales report for Midwest			Number of unit sales per product for Midwest	
State	City	Dollar Sales	Product	Number of units sold
IL	Chicago	\$3,924,401	Biscotti	89
MO	St. Louis	\$3,761,288	Coffee Grinder	71
TX	Houston	\$3,714,978	Coffee Pot	72
			Croissant	144
			Espresso	117
			Latte	243
			Mug	144
			Scone	127
			Thermos	72

### **Example:** Creating a Coordinated Graph With Document Syntax

This example, using the GGSales Master File, generates a Coordinated Compound Layout report that contains a graph and a report (by replacing the first report in the previous example with a graph). Note that a graph request with two BY fields will generate a graph for each value of the first BY field (REGION), and that these files are named by appending sequence numbers to the HOLD file name. For example, HOLD0.SVG, HOLD1.SVG, and so on.

To place these graphs into a report as a component of a Coordinated Compound Layout report, several COMPUTE commands are required to construct the name of each graph file (HOLD0.SVG, HOLD1.SVG, and so on). Additionally, a COMPUTE command will add the image files into the HEADING of the TABLE request so that they are associated with the same value of REGION, from which they were originally produced.

Enter the following syntax in the Text Editor.

```
SET PAGE-NUM=OFF
COMPOUND LAYOUT PCHOLD FORMAT PDF
SECTION=S1, LAYOUT=ON, MERGE=ON, ORIENTATION=LANDSCAPE, $
PAGELAYOUT=1, $
COMPONENT=Sales, TYPE=REPORT, POSITION=(0.25 1), DIMENSION=(4 4), $
COMPONENT=Fuel, TYPE=REPORT, POSITION=(7.25 1), DIMENSION=(4 4), $
END
```

```

SET COMPONENT=Sales
GRAPH FILE GGSales
SUM PCT.DOLLARS
BY REGION NOPRINT
BY PRODUCT
ON GRAPH SET LOOKGRAPH HBAR
ON GRAPH HOLD AS HOLD FORMAT SVG
ON GRAPH SET GRAPHSTYLE *
setPlace(true);
setColorMode(1);
setDepthRadius(0);
setDepthAngle(0);
setDisplay(getOlMajorGrid(),false);
setTransparentColor(getFrame(),true);
setDisplay(getDataText(),true);
setTextFormatPreset(getDataText(),28);
setFontSizeAbsolute(getDataText(),true);
setFontSizeInPoints(getDataText(),9);
setPlaceResize(getDataText(),0);
setFontStyle(getDataText(),0);
setTransparentColor(getSeries(0),true);
setDisplay(getOlAxisLine(),false);
setFontSizeAbsolute(getOlLabel(),true);
setFontSizeInPoints(getOlLabel(),9);
setPlaceResize(getOlLabel(),0);
setFontSizeAbsolute(getYlLabel(),true);
setFontSizeInPoints(getYlLabel(),9);
setPlaceResize(getYlLabel(),0);
setTextFormatPreset(getYlLabel(),28);
setGridStyle(getYlMajorGrid(),3);
setDisplay(getYlAxisLine(),false);
setDisplay(getYlMajorGrid(),false);
setDisplay(getYlLabel(),false);
setDataTextPosition(3);
setTextString(getOlTitle(),"");
setTextString(getYlTitle(),"");
setFontStyle(getTitle(),0);
ENDSTYLE
END

```

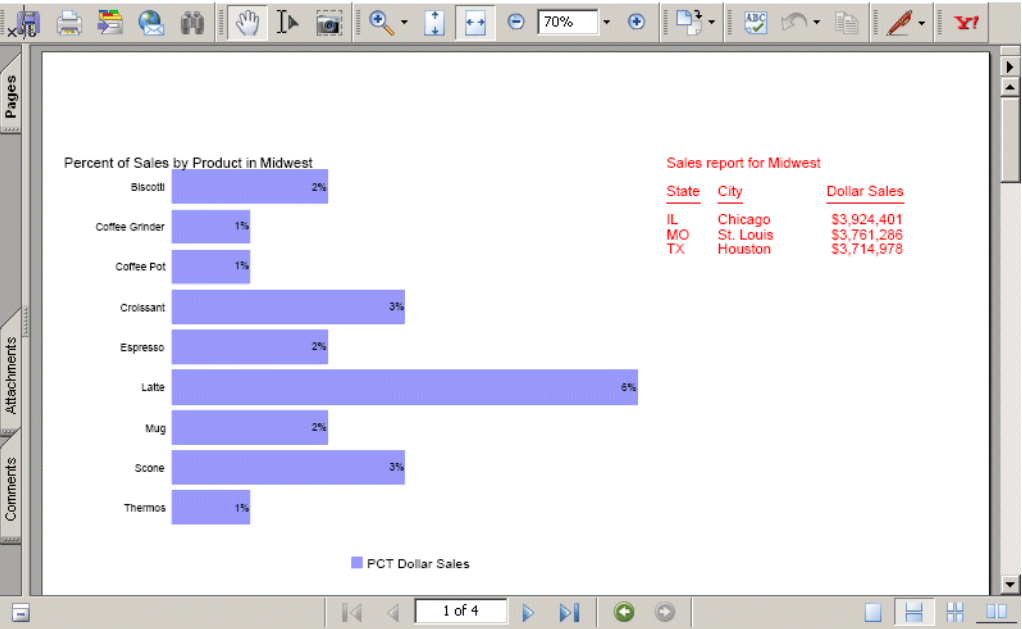
```

TABLE FILE GGSales
"Percent of Sales by Product in <REGION"
" "
SUM
COMPUTE CNTR/I4 = CNTR + 1; NOPRINT
COMPUTE CNTR2/A4 = IF &FOCGRAPHCNT EQ 1 THEN ' ' ELSE
FTOA(CNTR-1, '(F4)', 'A4'); NOPRINT
COMPUTE IMG/A16 = 'HOLD' || LJUST(4, CNTR2, 'A4') || '.svg'; NOPRINT
BY REGION NOPRINT
ON REGION PAGE-BREAK
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
type=HEADING, IMAGE=(IMG), position=(0 0), $
TYPE=REPORT, PAGE-LOCATION=OFF, $
TYPE=REPORT, FONT=HELVETICA, COLOR=BLACK, SQUEEZE=ON, $
ENDSTYLE
END

SET COMPONENT=Fuel
TABLE FILE GGSales
"Sales report for <REGION"
" "
SUM DOLLARS/F8M
BY REGION NOPRINT
BY ST
BY CITY
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, COLOR=RED, SQUEEZE=ON, $
ENDSTYLE
END
COMPOUND END

```

The first page of output is:



**Example:**    **Creating Multi-Page Layouts With Document Syntax**

In this example using the GGSALES Master File, multi-page layouts allow components to be placed in fixed locations on multiple pages. For example, a Coordinated Compound Layout report can contain component reports R1 and R2 for each value of the first sort field on the odd-numbered pages (front side), and R3 on the even-numbered pages (reverse side). Additionally, you can place the same heading that contains a logo and some text with the embedded value of the first sort field at the top of each side.

For the heading report, create a procedure (named HEADER.FEX), and enter the following syntax:

```
TABLE FILE GGSALES
" "
"Report package for <REGION>"
BY REGION NOPRINT
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, SIZE=20, $
TYPE=REPORT, IMAGE=poweredbyibi.gif, POSITION=(+.25 +.25), $
TYPE=HEADING, LINE=2, ITEM=1, POSITION=4, $
ENDSTYLE
END
```



We will use components R1 and R2 from the previous example. If you did not already do so, save them as REPORT1.FEX and REPORT2.FEX. Enter the following syntax as the R3 report component, by creating a procedure named REPORT3.FEX.

```
TABLE FILE GGSales
"Report R3 for <REGION"
BY REGION NOPRINT
SUM DOLLARS BY ST
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, COLOR=GREEN, $
ENDSTYLE
END
```

From the Text Editor, enter the following syntax specifying that the R1 and R2 report components appear on page 1. The R3 report component appears on page 2, and the heading report will appear on all pages of the document.

```
SET PAGE-NUM=OFF
SET SQUEEZE=ON
COMPOUND LAYOUT PCHOLD FORMAT PDF
SECTION=S1, LAYOUT=ON, MERGE=ON, ORIENTATION=LANDSCAPE, $
PAGELAYOUT=ALL, $
COMPONENT=HEADER, TYPE=REPORT, POSITION=(1 1), DIMENSION=(4 4), $
PAGELAYOUT=1, $
COMPONENT=R1, TYPE=REPORT, POSITION=(1 3), DIMENSION=(4 4), $
COMPONENT=R2, TYPE=REPORT, POSITION=(6 3), DIMENSION=(4 4), $
PAGELAYOUT=2, $
COMPONENT=R3, TYPE=REPORT, POSITION=(4 3), DIMENSION=(4 4), $
END

SET COMPONENT=HEADER
EX HEADER
SET COMPONENT=R1
EX REPORT1
SET COMPONENT=R2
EX REPORT2
SET COMPONENT=R3
EX REPORT3
COMPOUND END
```

Page 1 of the output is:

POWERED BY Information Builders				
Sales report for Midwest			Number of unit sales per product for Midwest	
State	City	Dollar Sales	Product	Number of units sold
IL	Chicago	\$3,924,401	Biscotti	89
MO	St. Louis	\$3,761,286	Coffee Grinder	71
TX	Houston	\$3,714,978	Coffee Pot	72
			Croissant	144
			Espresso	117
			Latte	243
			Mug	144
			Scone	127
			Thermos	72

Page 2 of the output is:

POWERED BY Information Builders				
		Report R3 for Midwest		
State		Dollar Sales		
IL		3924401		
MO		3761286		
TX		3714978		

**Example: Creating Page Overflow With Document Syntax**

A common type of report contains a fixed layout at the top of the page, followed by a report containing detail records of unfixed length. For example, a brokerage statement may contain the customer name and address, an asset-allocation graph, and a comparison of the portfolio with market indexes at the top, followed by a list of securities held in the account. If the list of securities overflows the first page, we would like it to continue on the second page, underneath the common heading which appears on all pages (a logo, account number, page number for instance). The OVERFLOW-POSITION and OVERFLOW-DIMENSION syntax enables us to specify where on the overflow page the report continues and what its maximum length on each overflow page should be. (Note that its width should not vary from one page to the next.)

The following example, using the GGSales Master File, demonstrates how you can use OVERFLOW-POSITION and OVERFLOW-DIMENSION to reposition the second report component (R2) so that it begins below the first component on the initial page, and two inches below the top of the page on subsequent pages. Note that this leaves enough space for the header report component (HEADER) at the top of each page.

Additionally, PAGELAYOUT=ALL forces the HEADER component to appear at the top of each overflow page.

Enter the following syntax in the Text Editor.

```
SET PAGE-NUM=OFF
COMPOUND LAYOUT PCHOLD FORMAT PDF
SECTION=Example, LAYOUT=ON, MERGE=OFF, $
  PAGELAYOUT=1, $
    COMPONENT=R1, TYPE=REPORT, POSITION=(1.5 2), DIMENSION=(8 3), $
    COMPONENT=R2, TYPE=REPORT, POSITION=(.5 5), DIMENSION=(8 5),
      OVERFLOW-POSITION=(.5 2), OVERFLOW-DIMENSION=(8 8.5), $
    PAGELAYOUT=ALL, $
    COMPONENT=HEADER, TYPE=REPORT, POSITION=(1.25 1), DIMENSION=(6 1), $
  END

SET COMPONENT=R1
TABLE FILE GGSales
HEADING CENTER
"Report 1"
"Sales Summary by Category"
" "
SUM UNITS BUDDOLLARS BY CATEGORY
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, SQUEEZE=ON, $
ENDSTYLE
END
```

```
SET COMPONENT=R2
TABLE FILE GGSales
HEADING CENTER
"Report 2"
"Sales Detail Report"
" "
SUM UNITS BUDUNITS DOLLARS BUDDOLLARS
BY CATEGORY BY PRODUCT BY REGION
ON CATEGORY UNDER-LINE
ON PRODUCT SUB-TOTAL
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, SQUEEZE=ON, $
ENDSTYLE
END
```

```
SET COMPONENT=HEADER
TABLE FILE GGSales
HEADING
"Gotham Grinds sales to Information Builders, October 1997"
BY CATEGORY NOPRINT
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, IMAGE=gotham.gif, POSITION=(3.25 .25), DIMENSION=(2 .75), $
ENDSTYLE
END
```

The first page of output is:

Gotham Grinds sales to Information Builders, October 1997

Report 1  
Sales Summary by Category

Category	Unit Sales	Budget Units	Dollar Sales	Budget Dollars
Coffee	1376266	1385923	17231455	17293886
Food	1384845	1377564	17229333	17267160
Gifts	927880	931007	11695502	11659732

Report 2  
Sales Detail Report

Category	Product	Region	Unit Sales	Budget Units	Dollar Sales	Budget Dollars
Coffee	Capuccino	Northeast	44785	44432	542095	561491
		Southeast	73264	75353	944000	956661
		West	71168	70585	895495	877304
		*TOTAL PRODUCT Capuccino		189217	190370	2381590
	Espresso	Midwest	101154	101869	1294947	1258232
		Northeast	68127	69776	850107	872902
		Southeast	68030	66785	853572	849465
		West	71675	72927	907617	923941
		*TOTAL PRODUCT Espresso		308986	311357	3906243
	Latte	Midwest	231623	233657	2883566	2827800
		Northeast	222866	221712	2771815	2818069
		Southeast	209654	213555	2617836	2625303
		West	213920	215272	2670405	2722718
		*TOTAL PRODUCT Latte		878063	884196	10943622
*TOTAL CATEGORY Coffee			1376266	1385923	17231455	17293886

The second page of output is:

Gotham Grinds sales to Information Builders, October 1997						
Report 2 Sales Detail Report						
Category	Product	Region	Unit Sales	Budget Units	Dollar Sales	Budget Dollars
Food	Biscotti	Midwest	86105	85839	1091727	1067629
		Northeast	145242	145152	1802006	1848682
		Southeast	119594	120549	1505717	1512019
		West	70436	67780	863868	861804
	*TOTAL PRODUCT Biscotti		421377	419320	5263317	5290134
	Croissant	Midwest	139182	139648	1751124	1708733
		Northeast	137394	137864	1670818	1739522
		Southeast	156456	157148	1902359	1969906
		West	197022	195329	2425601	2406554
	*TOTAL PRODUCT Croissant		630054	629689	7749902	7824715
	Scone	Midwest	116127	113776	1495420	1444359
		Northeast	70732	68415	907171	865703
		Southeast	73779	73812	900655	927363
		West	72776	72252	912868	914886
*TOTAL PRODUCT Scone		333414	328255	4216114	4152311	
*TOTAL CATEGORY Food		1384845	1377564	17229333	17267160	
<hr/>						
Gifts	Coffee Grinder	Midwest	50393	50628	619154	613453
		Northeast	40977	41297	509200	511642
		Southeast	47083	46784	605777	595585
		West	48081	47397	603436	571316
	*TOTAL PRODUCT Coffee Grinder		186534	186106	2337567	2265996
	Coffee Pot	Midwest	47156	47779	599878	614007
		Northeast	46185	45404	590780	572349
		Southeast	49922	50637	645303	654579
		West	47432	49208	613624	630196
	*TOTAL PRODUCT Coffee Pot		190695	193028	2449585	2472131
	Mug	Midwest	86718	87082	1086943	1096150
		Northeast	91497	90540	1144211	1170314
		Southeast	88474	88371	1102703	1124345
		West	93881	93629	1188664	1156976
*TOTAL PRODUCT Mug		360570	360632	4522521	4547785	

## **Syntax:** How to Draw Objects With Document Syntax

A variety of objects can be drawn on the page to enhance a report. The currently supported objects include Lines, Boxes, Static text strings, and Images.

The syntax for drawing these objects may appear in the StyleSheet of a report, but they may also be included within a PAGELAYOUT grouping in the COMPOUND LAYOUT declarations. The syntax for each drawing object is described below.

☐ **Lines.** To draw a line from point (x1 y1) to point (x2 y2), enter the following syntax:

```
OBJECT=LINE, POSITION=(x1 y1), ENDPOINT=(x2 y2),
[BORDER=b,] [BORDER-COLOR=c,] [BORDER-STYLE=s,]$
```

Optionally, the border attributes BORDER, BORDER-COLOR, and BORDER-STYLE follow the existing BORDER syntax, as shown below:

```
OBJECT=LINE, POSITION=(1 1), ENDPOINT=(8 1),
      BORDER=HEAVY, BORDER-COLOR=RED, BORDER-STYLE=DASHED, $
```

- ❑ **Boxes.** To draw a box, whose upper left corner is at (x y), and whose dimensions are xdim by ydim, enter the following syntax:

```
OBJECT=BOX, POSITION=(x y), DIMENSION=(xdim ydim),
      BACKCOLOR=c,
      [BORDER=b,] [BORDER-COLOR=bc,] [BORDER-STYLE=bs,] $
```

**Tip:** The background color, which is c, specifies the color with which the box is filled and can be any valid color. For example, yellow or RGB(200 200 200).

As in the BORDER syntax, the individual sides of the box can be styled separately. For example:

```
OBJECT=BOX, POSITION=(1 1), DIMENSION=(2 3),
      BACKCOLOR=YELLOW,
      BORDER=HEAVY, BORDER-TOP-COLOR=RED, BORDER-BOTTOM-COLOR=BLUE, $
```

Note that, as in the BORDER syntax, attributes of lines or boxes that are not explicitly specified have the following defaults:

- ❑ Color: black
- ❑ Style: solid
- ❑ Width border attribute: medium
- ❑ **Static text strings.** Static text strings display text that you enter as part of the object description.

You can format the text displayed in the text object by including markup tags within the text portion of the text object. A report with markup tags in a text object is called a *markup report*. A markup report can be generated as a PDF, DHTML, PPT, or PPTX output file. WebFOCUS supports a subset of HTML tags and its own page numbering tags. To activate these markup tags (so that they are treated as formatting elements instead of displaying as text), add the attribute MARKUP=ON to the string object. For additional information, see [Text Formatting Markup Tags for a Text Object](#) on page 840.

To draw a static text string at position (x y), enter the following syntax:

```
OBJECT=STRING, POSITION=(X Y), TEXT='any text you like', [MARKUP={ON|
OFF},] [FONT=f,] [SIZE=sz,] [STYLE=st,] [COLOR=c,]
      [WRAP=ON, DIMENSION=(xdim ydim),] [LINESPACING=linesoption,]
$
```

where:

**POSITION**=( *xy*)

Specifies the (x y) coordinate on the page where the upper-left corner of the component is to be placed. All coordinates are in current UNITS (default is inches), and (0 0) is the upper-left corner of the physical page.

**Note:** By default, coordinates are absolute locations on the physical page. If x or y is preceded by a plus sign (+) or minus sign (-), for example, (+.25 +0), the coordinate is relative to the left or top page margin.

**TEXT**='*any text you like*'

Is the text to be placed in the text object.

**Note:** If your text contains any open caret characters (<), you must put a blank space after each open caret that is part of the text. If you do not, everything following the open caret will be interpreted as the start of a markup tag and will not display as text.

**MARKUP**={ **ON** | **OFF** }

ON causes the markup tags to be interpreted as formatting options. OFF displays the tags as text. OFF is the default value.

**FONT**=*f*

Is the default font to be used for the text.

**SIZE**=*sz*

Is the default font size to be used for the text.

**STYLE**=*st*

Is the default font style to be used for the text.

**COLOR**=*c*

Is the default font color to be used for the text.

**WRAP**=**ON**

Specifies that the text should wrap when it reaches the end of the text object bounding box.

**DIMENSION**=( *xdimydim*)

Specifies the size of the text object bounding box (in current UNITS).

**LINESPACING**=*linesoption*

Determines the amount of vertical space between lines of text in a paragraph. Two types of LINESPACING attributes are supported:

**LINESPACING**={ **SINGLE** | 1.5**LINES** | **DOUBLE** }

or



`LINESPACING=type(value)`

where:

**SINGLE**

Accommodates the largest font in that line, plus a small amount of extra space. The amount of extra space varies depending on the font used. SINGLE is the default option.

**1.5LINES**

Is one-and-one-half times that of single line spacing.

**DOUBLE**

Is twice that of single line spacing.

**type(value)**

Can be one of the following where *value* is a positive number:

Type	Value	Example
<b>MULTIPLE</b>	The percentage by which to increase or decrease the line space.	<code>LINESPACING=MULTIPLE(1.2)</code> increases line space by 20 percent.
<b>MIN</b>	The minimum line space (in the unit specified by UNITS parameter) needed to fit the largest font on the line.	<code>LINESPACING=MIN(0.5)</code> provides a minimum line space of 0.5 inch when UNITS=IN.
<b>EXACT</b>	The fixed line space (in the unit specified by UNITS parameter) that WebFOCUS does not adjust.	<code>LINESPACING=EXACT(.3)</code> provides a fixed line space of 0.3 inch when UNITS=IN.

Optionally, you may specify the FONT, SIZE, STYLE, and COLOR attributes as you would for any textual object in a report. For example:

```
OBJECT=STRING, POSITION=(1 1), TEXT='Hello world!',  
      FONT=TIMES, SIZE=12, STYLE=BOLD, COLOR=RED, $
```

**Note:** A position of a string is measured from its bottom left to allow strings with different heights to be aligned to a common base-line. However, if WRAP=ON is present, it indicates that the string should be wrapped to a bounding box whose top-left corner is at (x y) and whose dimensions are (xdim ydim). In this case, the top left of the text string is positioned at point (x y).

❑ **Images.** An image can be drawn as a drawing object by entering the following syntax:

```
OBJECT=IMAGE, IMAGE=file, POSITION=(x y), DIMENSION=(xdim ydim), $
```

**Note:** The image file name=*file* can be any image file valid in a PDF report. POSITION is used as with a conventional image, and DIMENSION is used in place of the SIZE attribute of a conventional image.

**Example: Drawing Objects With Document syntax**

The following example shows how drawing objects can be placed inside the COMPOUND LAYOUT syntax using the GGSALES Master File. Note that a drawing object, like a COMPONENT, appears on the page whose PAGELAYOUT declaration it follows.

```

SET PAGE-NUM=OFF
SET SQUEEZE=ON
COMPOUND LAYOUT PCHOLD FORMAT PDF
SECTION=S1, LAYOUT=ON, MERGE=ON, ORIENTATION=LANDSCAPE, $
PAGELAYOUT=1, $
COMPONENT=Sales, TYPE=REPORT, POSITION=(1 1), DIMENSION=(4 4), $
COMPONENT=Budget, TYPE=REPORT, POSITION=(6.25 1), DIMENSION=(4 4), $
OBJECT=IMAGE, IMAGE=gglogo.gif, POSITION=(1 4.5), DIMENSION=(1 1), $
OBJECT=BOX, POSITION=(1 1), DIMENSION=(5 3), BACKCOLOR=GOLDENROD, $
END
SET COMPONENT=Sales
TABLE FILE GGSALES
"Sales report for <REGION>"
" "
SUM DOLLARS
BY REGION NOPRINT
BY CATEGORY
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, STYLE=BOLD, $
ENDSTYLE
END
SET COMPONENT=Budget
TABLE FILE GGSALES
"Budget report for <REGION>"
" "
SUM BUDDOLLARS
BY REGION NOPRINT
BY CATEGORY
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, COLOR=BLUE, $
ENDSTYLE
END
COMPOUND END

```

The first page of output is:

Sales report for Midwest	
Category	Dollar Sales
Coffee	4178513
Food	4338271
Gifts	2883881

Budget report for Midwest	
Category	Budget Dollars
Coffee	4086032
Food	4220721
Gifts	2887620



**Note:** A drawing object will not be drawn unless there is at least one COMPONENT in its PAGELAYOUT.

**Syntax:**      **How to Display Grids With Document Syntax**

The SET LAYOUTGRID command can be used as an aid to manually develop report layouts. Displaying grids superimposes a light one-inch by one-inch grid on the page so that the locations of the various report components can be verified. Currently, the grid only works in inches.

Enter the following syntax to display grids:

```
SET LAYOUTGRID=ON
```

**Reference:**      **Text Formatting Markup Tags for a Text Object**

**Note:** If your text contains any open caret characters (<), you must put a blank space after each open caret that is part of the text. If you do not, everything following the open caret will be interpreted as the start of a markup tag and will not display as text.

### Font Properties

The font tag supports three attributes: face, size, and color (where the color must be specified as the hexadecimal number code for the color):

```
<font face="font" size=[+|-]n color=color_code>text</font>
```

For example:

```
<font face="New Century Schoolbook">Test1</font>
<font face="Times" size=12>test2</font>
<font face="Times New Roman" color=#0000FF size=+4>Test3</font>
<font size=-2 face="Times New Roman" color=#0000FF >Test4</font>
```

### Text Styles

The supported text styles are bold, italic, underline, and superscript:

Bold: `<b>text</b>`

Italic: `<i>text</i>`

Underline: `<u>text</u>`

Superscript: `<sup>text</sup>`

### Line Breaks

The line break tag after a portion of text begins the next portion of text on a new line. Note that there is no closing tag for a line break:

```
<br>
```

### Text Alignment

The alignment options pertain to wrapped text, as well as specified line breaks. Both horizontal justification and vertical alignment are supported.

#### ❑ Horizontal Justification

Left Justification:

```
<left>text</left>
```

Right Justification:

```
<right>text</right>
```

Center Justification:

```
<center>text</center>
```

Full Justification:

```
<full>text</full>
```

### ❑ Vertical Alignment

Top Alignment:

```
<top>text</top>
```

Middle Alignment:

```
<mid>text</mid>
```

Bottom Alignment:

```
<bottom>text</bottom>
```

### Unordered (Bullet) List

The unordered (ul) list tag encloses a bullet list. Each item is enclosed in a list item tag (li). The start tag and end tag for the list must each be on its own line. Each list item must start on a new line:

```
<ul>
<li>list item1</li>
<li>list item2</li>
.
.
.
</ul>
```

By default, the bullet type is disc. You can also specify circle or square:

```
<ul type=disc>
```

```
<ul type=circle>
```

```
<ul type=square>
```

### Ordered (Number or Letter) List

The ordered (ol) list tag encloses a list in which each item has a consecutive number or letter. Each item is enclosed in a list item tag (li). The start tag and end tag for the list must each be on its own line. Each list item must start on a new line:

```
<ol>
<li>list item1</li>
<li>list item2</li>
.
.
.
</ol>
```

By default, Arabic numerals (type=1) are used for the ordering of the list. You can specify the following types of order:

Arabic numerals (the default): `<ol type=1>`

Lowercase letters: `<ol type=a>`

Uppercase letters: `<ol type=A>`

Lowercase Roman numerals: `<ol type=I>`

Uppercase Roman numerals: `<ol type=I>`

### Hyperlinks

Hyperlinks can be included within text markup in PDF documents.

The syntax for the anchor markup tag is a subset of the HTML anchor syntax:

```
<a href="hyperlink">Text to display</a>
```

where:

*hyperlink*

Is the hyperlink to jump to when the text is clicked.

*Text to display*

Is the text to display for the hyperlink.

For example:

```
<a href="http://www.example.com/help.htm">Click here for help</a>
```

No other attributes are supported in the anchor markup tag.

### Page Numbering and Dates

There are two pseudo-HTML tags for embedding page numbers in text on a Page Master for a Coordinated Compound Layout report:

Current page number: `<ibi-page-number/>`

Total number of pages: `<ibi-total-pages/>`

Note that when MARKUP=ON, space is allocated for the largest number of pages, so there may be a wide gap between the page number and the text that follows. To remove the extra space in the text object that has the page numbering tags:

- ❑ If specific styling of the text object is not required, do not insert markup tags, and turn MARKUP=OFF.

```
MARKUP=OFF, TEXT='Page <ibi-page-number/> of <ibi-total-pages/> of Sales Report', $
```

This displays the following

`Page 1 of 100 of Sales Report`

- ❑ If specific styling of the text object is required, you must set MARKUP=ON. With MARKUP=ON, set WRAP=OFF and do not place any styling tags between the page number variables within the string. Tags can be used around the complete *Page n of m* string. The following code produces a page number string without the extra spaces:

```
MARKUP=ON, WRAP=OFF, TEXT='<font face="ARIAL" size=10><i>Page <ibi-page-number/> of <ibi-total-pages/> of Sales Report </i>', $
```

This displays the following

*Page 1 of 100 of Sales Report*

To display a date in the report output, insert a WebFOCUS date variable in a text object on a Page Master (such as &DATEtrMDYY) in the text object.



**Example: Formatting a Compound Layout Text Object With Markup Tags**

The following request displays a text object with markup tags in a PDF output file.

**Note:** Text markup syntax cannot contain hidden carriage return or line feed characters. For purposes of presenting the example in this documentation, line feed characters have been added so that the sample code wraps to fit within the printed page. To run this example in your environment, copy the code into a text editor and delete any line feed characters within the text markup object by going to the end of each line and pressing *Delete*. In some instances, you may need to add a space to maintain the structure of the string. For additional information on displaying carriage returns within the text object see [Text Formatting Markup Tags for a Text Object](#) on page 840.

```
SET PAGE-NUM=OFF
SET LAYOUTGRID=ON
TABLE FILE GGSALES
BY REGION NOPRINT
ON TABLE PCHOLD AS LINESP1 FORMAT PDF
ON TABLE SET STYLE *
type=report, size=8, $
object=string, position=(1 1), dimension=(7 3), wrap=on, markup=on,
  linespacing=multiple(3),
  text='<b><font face="Arial" size=12>This paragraph is triple-spaced
(LINESPACING=MULTIPLE(3)):</font></b>
<full>Our <i>primary</I> goal for fiscal 2006 was to accelerate our
transformation to customer centricity. In this letter, I'd like to
give you an update on this work, which contributed to the 22-percent
increase in earnings from continuing operations we garnered for fiscal
2006. Since the past is often prologue to the future, I'd like to
describe how customer centricity is influencing not only our goals for
fiscal 2007, but also our long-term plans. At Gotham Grinds, customer
centricity means treating each customer as a unique individual, meeting
their needs with end-to-end solutions, and engaging and energizing our
employees to serve them.</full>', $
ENDSTYLE
END
```

In this request:

- ☐ No fields from the data source are displayed. Only the text object displays on the output.
- ☐ The SET LAYOUTGRID command displays a grid to indicate the coordinates and dimensions of the text object.
- ☐ The OBJECT=STRING declaration specifies triple spacing: LINESPACING=MULTIPLE(3).
- ☐ The following text is displayed in boldface, in the Arial font face, and with a font size of 12 (the default is 8 from the TYPE=REPORT declaration):

‘This paragraph is triple-spaced (LINESPACING=MULTIPLE(3)):

The markup for this formatting is:




**Example: Drawing Text and Line objects on a Page Master**

The following request places a line on the page master between the header report and the component reports and places a line and a text string on the bottom of each page:

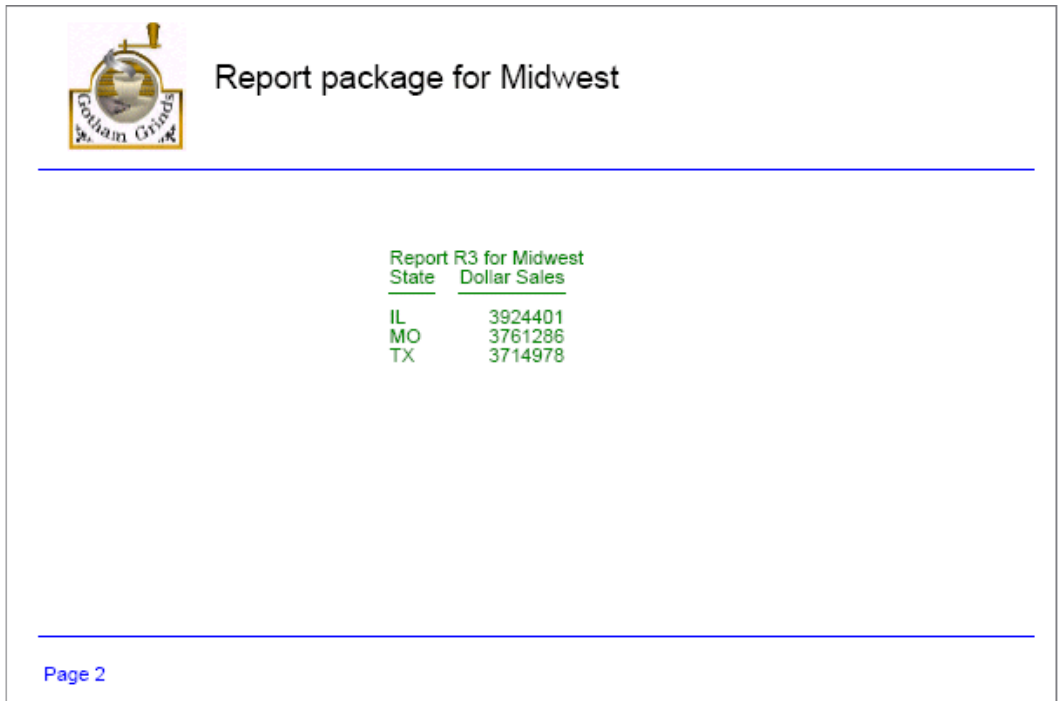
```
SET PAGE-NUM=OFF
SET SQUEEZE=ON
COMPOUND LAYOUT PCHOLD FORMAT PDF
SECTION=S1, LAYOUT=ON, MERGE=ON, ORIENTATION=LANDSCAPE, $
PAGELAYOUT=ALL, $
COMPONENT=HEADER, TYPE=REPORT, POSITION=(1 1), DIMENSION=(4 4), $
OBJECT=STRING, POSITION=(1 6.6), MARKUP=ON,
TEXT='<font face="Arial" color=#0000FF size=12> Page <ibi-page-number/>
      </font> ', WRAP=ON, DIMENSION=(4 4),$
OBJECT=LINE, POSITION=(1 2.5), ENDPOINT=(9.5 2.5),
      BORDER-COLOR=BLUE,$
OBJECT=LINE, POSITION=(1 6.5), ENDPOINT=(9.5 6.5),
      BORDER-COLOR=BLUE,$
PAGELAYOUT=1, $
COMPONENT=R1, TYPE=REPORT, POSITION=(1 3), DIMENSION=(4 4), $
COMPONENT=R2, TYPE=REPORT, POSITION=(6 3), DIMENSION=(4 4), $
PAGELAYOUT=2, $
COMPONENT=R3, TYPE=REPORT, POSITION=(4 3), DIMENSION=(4 4), $
END
```

```
SET COMPONENT=HEADER
TABLE FILE GGSales
" "
"Report package for <REGION"
BY REGION NOPRINT
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, SIZE=20, $
TYPE=REPORT, IMAGE=gglogo.gif, POSITION=(+.25 +.25), $
TYPE=HEADING, LINE=2, ITEM=1, POSITION=1.5, $
ENDSTYLE
END
SET COMPONENT=R1
TABLE FILE GGSales
"Sales report for <REGION"
" "
SUM DOLLARS/F8M
BY REGION NOPRINT
BY ST
BY CITY
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, COLOR=RED, SQUEEZE=ON, $
ENDSTYLE
END
SET COMPONENT=R2
TABLE FILE GGSales
"Number of unit sales per product for <REGION"
" "
SUM CNT.UNITS AS 'Number of units sold'
BY REGION NOPRINT
BY PRODUCT
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, COLOR=BLUE, SQUEEZE=ON, $
ENDSTYLE
END
SET COMPONENT=R3
TABLE FILE GGSales
"Report R3 for <REGION"
BY REGION NOPRINT
SUM DOLLARS BY ST
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, COLOR=GREEN, $
ENDSTYLE
END
COMPOUND END
```

The first page of output is:

			Report package for Midwest	
Sales report for Midwest			Number of unit sales per product for Midwest	
<u>State</u>	<u>City</u>	<u>Dollar Sales</u>	<u>Product</u>	<u>Number of units sold</u>
IL	Chicago	\$3,924,401	Biscotti	89
MO	St. Louis	\$3,761,286	Coffee Grinder	71
TX	Houston	\$3,714,978	Coffee Pot	72
			Croissant	144
			Espresso	117
			Latte	243
			Mug	144
			Scone	127
			Thermos	72
Page 1				

The second page of output has the same drawing objects:



**Example: Vertically Aligning Text Markup in PDF Report Output**

The following request creates three boxes and places a text string object within each of them:

- ☐ In the left box, the text is aligned vertically at the top.
- ☐ In the middle box, the text is aligned vertically at the middle.

☐ In the right box, the text is aligned vertically at the bottom.

**Note:** Text markup syntax cannot contain hidden carriage return or line feed characters. For purposes of presenting the example in this documentation, line feed characters have been added so that the sample code wraps to fit within the printed page. To run this example in your environment, copy the code into a text editor and delete any line feed characters within the text markup object by going to the end of each line and pressing *Delete*. In some instances, you may need to add a space to maintain the structure of the string. For additional information on displaying carriage returns within the text object see [Text Formatting Markup Tags for a Text Object](#) on page 840.

```
SET PAGE-NUM=OFF
TABLE FILE GGSales
BY REGION NOPRINT
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
type=report, font=arial, size=10, $
object=box, position=(1 1), dimension=(6 1), $
object=line, position=(3 1), endpoint=(3 2), $
object=line, position=(5 1), endpoint=(5 2), $
object=string, text='<top>Vertically aligned text within a text object
using top alignment.</top>', position=(1.05 1), dimension=(2 1),
linespacing=exact(.15), markup=on, wrap=on, $
object=string, text='<mid>Vertically aligned text within a text object
using middle alignment.</mid>', position=(3.05 1), dimension=(2 1),
linespacing=exact(.15), markup=on, wrap=on, $
object=string, text='<bottom>Vertically aligned text within a text object
using bottom alignment.</bottom>', position=(5.05 .9), dimension=(2
1),linespacing=exact(.15), markup=on, wrap=on, $
ENDSTYLE
END
```

The output is:

Vertically aligned text within a text object using top alignment.	Vertically aligned text within a text object using middle alignment.	Vertically aligned text within a text object using bottom alignment.
---	--	--

**Reference:** Coordinated Compound Layout Reports With Missing Data

A Coordinated Compound Layout report is comprised of individual component reports or graphs with a common first sort field. The compound procedure generates an output document with a separate page (or set of pages) for each individual value of the sort field, with the embedded components segmented to display the data that corresponds to that sort field value.

A Coordinated Compound Layout report page is generated in the designated page layout for every sort field value found in at least one of the component reports, presenting the appropriate data for those components where data exists for that value, and presenting an empty component report where data does not exist.

The way an empty component is represented on the report page is dependent on how the component positioning is defined within the Coordinated Compound Layout report. Components can be defined with absolute positioning or relative to other components in the layout. If the empty component and subsequent components on the page are defined with absolute positioning, the empty component report will display as blank space in the designated location. If relative positioning is defined between the empty component and subsequent components, the subsequent components will float up on the page, and no empty space will be displayed in the area defined for the empty component.

In compound reports with relative positioning defined between reports, when an empty report is encountered, the report following the empty report is positioned vertically relative to the bottom of the last non-empty report and horizontally relative to the page margin. This means that when a report contains no data, the subsequent report will float up (vertically) and begin relative to the previous report but it will not move horizontally on the page relative to either of the previous reports

Using POSITION = (X Y), the placement of a report is designated by the left coordinate (X = horizontal) and the top coordinate (Y = vertical). Each of these coordinates can be defined independently as relative to the previous report or in a fixed position on the page.

To define positioning so that the report floats up on the page to replace the empty report but is anchored in a fixed left position on the page, you can define the Y coordinate (top) as relative and anchor the X coordinate (left). To anchor one of the position coordinates, change the reference from a relative position (+/-) to an absolute position. For example:

Both coordinates relative: POSITION(+0.003 +0.621)

Anchor horizontal / flow vertical: POSITION( 0.520 +0.621)

You do not need to change or add any WebFOCUS syntax to your request in order to take advantage of this feature. You will, however, want to pay attention to how you select and relate the data within your coordinated components to ensure you are generating the desired output.

### ***Example:*** Setup: Creating a Coordinated Compound Report With Missing Data

In this example, we will create a set of statements reporting the outstanding inventory orders for a select group of stores. Each store may have unfilled orders in any of three inventory categories: *Food*, *Coffee*, and *Gifts*.



To demonstrate how this works we will first build a set of data files: a header file containing contact information for the selected set of stores, and transaction files for each inventory category. We are selecting specific data to demonstrate how this will work when different component reports are empty.

After creating the data files, we will build four component reports, one to display the header information and one for each inventory category.

Finally, we will bring them together in a Coordinated Compound Layout report that merges all of this information into a single statement page for each store.

### **Example:** Step 1: Creating the Data Files

The following four data files will be created from a join of the GGORDER and GGSales data sources:

<b>Data File Created</b>	<b>Type of Information Included</b>	<b>Stores included</b>
GGHDR	Store information	R1019, R1020, R1040, R1041
GG1	Order transactions for Coffee	R1019, R1040, R1088
GG2	Order transactions for Food	R1019, R1020, R1041, R1088
GG3	Order transactions for Gifts	R1019, R1020, R1040, R1088

The APP HOLD, JOIN, and DEFINE commands are:

```
APP HOLD baseapp
JOIN
  GGORDER.ORDER01.STORE_CODE IN GGORDER TO
    UNIQUE GGSTORES.STORES01.STORE_CODE
    IN GGSTORES AS J1
END
DEFINE FILE GGORDER
PRODUCT_CATEGORY/A15=IF (PRODUCT_DESCRIPTION IN
    ('Biscotti','Croissant','Scone')) THEN 'Food'
    ELSE IF (PRODUCT_DESCRIPTION IN
    ('French Roast','Hazelnut','Kona')) THEN 'Coffee' ELSE
'Gifts';
END
```

The following procedure creates the data source GGHDR:

```
TABLE FILE GGORDER
SUM
    FST.STORE_NAME
    FST.ADDRESS1
    FST.ADDRESS2
    FST.CITY
    FST.STATE
    FST.ZIP
BY STORE_CODE
WHERE STORE_CODE IN ('R1019','R1020','R1040','R1041');
ON TABLE NOTOTAL
ON TABLE HOLD AS GGHDR FORMAT FOCUS INDEX 'STORE_CODE'
END
```

The following procedure creates the data source GG1:

```
TABLE FILE GGORDER
PRINT
    QUANTITY
    UNIT_PRICE
    PACKAGE_TYPE
    SIZE
    VENDOR_NAME
    PRODUCT_CATEGORY
BY STORE_CODE
BY ORDER_DATE
BY PRODUCT_DESCRIPTION
WHERE ( PRODUCT_CATEGORY EQ 'Coffee' ) AND ( ORDER_DATE GE '09/01/97' );
WHERE STORE_CODE IN ('R1019','R1040','R1088');
ON TABLE HOLD AS GG1 FORMAT FOCUS INDEX 'STORE_CODE'
END
```

The following procedure creates the data source GG2:

```
TABLE FILE GGORDER
PRINT
    QUANTITY
    UNIT_PRICE
    PACKAGE_TYPE
    SIZE
    VENDOR_NAME
    PRODUCT_CATEGORY
BY STORE_CODE
BY ORDER_DATE
BY PRODUCT_DESCRIPTION
WHERE ( PRODUCT_CATEGORY EQ 'Food' ) AND ( ORDER_DATE GE '09/01/97' );
WHERE STORE_CODE IN ('R1019','R1020','R1041','R1088');
ON TABLE HOLD AS GG2 FORMAT FOCUS INDEX 'STORE_CODE'
END
```

The following procedure creates the data source GG3:

```
TABLE FILE GGORDER
PRINT
    QUANTITY
    UNIT_PRICE
    PACKAGE_TYPE
    SIZE
    VENDOR_NAME
    PRODUCT_CATEGORY
BY STORE_CODE
BY ORDER_DATE
BY PRODUCT_DESCRIPTION
WHERE ( PRODUCT_CATEGORY EQ 'Gifts' ) AND ( ORDER_DATE GE '09/01/97' );
WHERE STORE_CODE IN ( 'R1019','R1020','R1040','R1088' );
ON TABLE HOLD AS GG3 FORMAT FOCUS INDEX 'STORE_CODE'
END
```

### **Example:** Step 2: Creating the Component Reports

The following procedure, GGHDR.FEX, creates the first report component for the Coordinated Compound Layout report from the GGHDR data source. This will function as a header for each page of the Compound Layout report. The shared sort field for all of the report components is STORE\_CODE. On each page of the PDF output file, this procedure lists the name and address of one store:

```
TABLE FILE GGHDR
BY STORE_CODE NOPRINT
HEADING
    "<STORE_NAME "
    "<ADDRESS1 "
    "<ADDRESS2 "
    "<CITY , <STATE <ZIP "
    " "
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
    UNITS=IN,
    SQUEEZE=ON,
    ORIENTATION=PORTRAIT,$
TYPE=REPORT,
    GRID=OFF,
    FONT='ARIAL' ,
    SIZE=9,$
TYPE=HEADING,
    SIZE=10,
    STYLE=BOLD,
    COLOR=BLUE,$
ENDSTYLE
END
```

The following procedure, GGRPT1.FEX, creates the second report component for the Coordinated Compound Layout report. For the same store code value in the header report, it displays data from the GG1 data source about the product category Coffee:

```
TABLE FILE GG1
SUM
    QUANTITY
    UNIT_PRICE
    FST.PACKAGE_TYPE AS ',Package'
    FST.SIZE AS ',Size'
    VENDOR_NAME
BY STORE_CODE AS 'Store'
BY PRODUCT_DESCRIPTION
HEADING
"<PRODUCT_CATEGORY"
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
    UNITS=IN,
    SQUEEZE=ON,
    ORIENTATION=PORTRAIT,$
TYPE=REPORT,
    GRID=OFF,
    FONT='ARIAL',
    SIZE=9,$
TYPE=DATA,
    SIZE=10,$
TYPE=TITLE,
    STYLE=BOLD,
    SIZE=10,$
TYPE=HEADING,
    SIZE=10,
    STYLE=BOLD,
    COLOR=RED,$
ENDSTYLE
END
```

The following procedure, GGRPT2.FEX, creates the third report component for the Coordinated Compound Layout report. For the same store code value in the header report, it displays data from the GG2 data source about the product category Food:

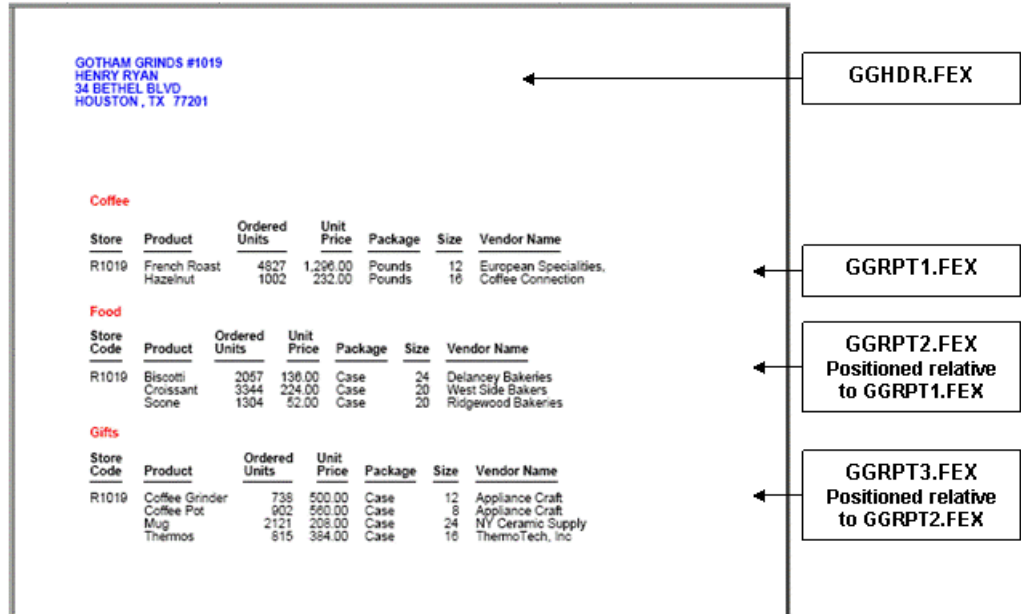
```
TABLE FILE GG2
SUM
    QUANTITY
    UNIT_PRICE
    FST.PACKAGE_TYPE AS ',Package'
    FST.SIZE AS ',Size'
    VENDOR_NAME
BY STORE_CODE
BY PRODUCT_DESCRIPTION
HEADING
"<PRODUCT_CATEGORY"
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
    UNITS=IN,
    SQUEEZE=ON,
    ORIENTATION=PORTRAIT,$
TYPE=REPORT,
    GRID=OFF,
    FONT='ARIAL',
    SIZE=9,$
TYPE=DATA,
    SIZE=10,$
TYPE=TITLE,
    STYLE=BOLD,
    SIZE=10,$
TYPE=HEADING,
    SIZE=10,
    STYLE=BOLD,
    COLOR=RED,$
ENDSTYLE
END
```

The following procedure, GGRPT3.FEX, creates the final report component for the Coordinated Compound Layout report. For the same store code value in the header report, it displays data from the GG3 data source about the product category Gifts:

```
TABLE FILE GG3
SUM
    QUANTITY
    UNIT_PRICE
    FST.PACKAGE_TYPE AS ',Package'
    FST.SIZE AS ',Size'
    VENDOR_NAME
BY STORE_CODE
BY PRODUCT_DESCRIPTION
HEADING
"<PRODUCT_CATEGORY"
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
    UNITS=IN,
    SQUEEZE=ON,
    ORIENTATION=PORTRAIT,$
TYPE=REPORT,
    GRID=OFF,
    FONT='ARIAL',
    SIZE=9,$
TYPE=DATA,
    SIZE=10,$
TYPE=TITLE,
    STYLE=BOLD,
    SIZE=10,$
TYPE=HEADING,
    SIZE=10,
    STYLE=BOLD,
    COLOR=RED,$
ENDSTYLE
END
```

**Example: Step 3: Building the Coordinated Compound Layout Report**

The following procedure, GGCMPD.FEX, combines the four components into a Coordinated Compound Layout report. The reports and relative positioning for each component is presented in the following diagram:



The Coordinated Compound Layout syntax is:

```
SET HTMLARCHIVE=ON
COMPOUND LAYOUT PCHOLD FORMAT PDF
UNITS=IN, $
SECTION=section1, LAYOUT=ON, MERGE=ON, ORIENTATION=PORTRAIT,
  PAGESIZE=Letter, $
PAGELAYOUT=1, NAME='Page layout 1', text='Page layout 1',
  TOC-LEVEL=1, BOTTOMMARGIN=0.5, TOPMARGIN=0.5, $
COMPONENT='report1', TEXT='report1', TOC-LEVEL=2,
  POSITION=(0.667 1.083), DIMENSION=(3.417 1.412), $
COMPONENT='report2', TEXT='report2', TOC-LEVEL=2,
  POSITION=(0.837 2.584), DIMENSION=(* *), $
COMPONENT='report3', TEXT='report3', TOC-LEVEL=2,
  POSITION=(-0.006 +0.084), DIMENSION=(* *),
  RELATIVE-TO='report2', RELATIVE-POINT=BOTTOM-LEFT,
  POSITION-POINT=TOP-LEFT, $
COMPONENT='report4', TEXT='report4', TOC-LEVEL=2,
  POSITION=(+0.010 +0.080), DIMENSION=(* *),
  RELATIVE-TO='report3', RELATIVE-POINT=BOTTOM-LEFT,
  POSITION-POINT=TOP-LEFT, $
END
SET COMPONENT='report1'
-INCLUDE GGHDR.FEX
SET COMPONENT='report2'
-INCLUDE GGRPT1.FEX
SET COMPONENT='report3'
-INCLUDE GGRPT2.FEX
SET COMPONENT='report4'
-INCLUDE GGRPT3.FEX
COMPOUND END
```



On the first page of the PDF output file, all components have data and appear on the report output:

<b>GOTHAM GRINDS #1019</b> <b>HENRY RYAN</b> <b>34 BETHEL BLVD</b> <b>HOUSTON, TX 77201</b>						
<b>Coffee</b>						
<u>Store</u>	<u>Product</u>	<u>Ordered Units</u>	<u>Unit Price</u>	<u>Package</u>	<u>Size</u>	<u>Vendor Name</u>
R1019	French Roast	4827	1,298.00	Pounds	12	European Specialties,
	Hazelnut	1002	232.00	Pounds	16	Coffee Connection
<b>Food</b>						
<u>Store Code</u>	<u>Product</u>	<u>Ordered Units</u>	<u>Unit Price</u>	<u>Package</u>	<u>Size</u>	<u>Vendor Name</u>
R1019	Biscotti	2057	136.00	Case	24	Delancey Bakeries
	Croissant	3344	224.00	Case	20	West Side Bakers
	Scone	1304	52.00	Case	20	Ridgewood Bakeries
<b>Gifts</b>						
<u>Store Code</u>	<u>Product</u>	<u>Ordered Units</u>	<u>Unit Price</u>	<u>Package</u>	<u>Size</u>	<u>Vendor Name</u>
R1019	Coffee Grinder	738	500.00	Case	12	Appliance Craft
	Coffee Pot	902	580.00	Case	8	Appliance Craft
	Mug	2121	208.00	Case	24	NY Ceramic Supply
	Thermos	815	384.00	Case	16	ThermoTech, Inc

On the second page, report GGRPT2.FEX did not retrieve any data for the store in the header. Therefore, the Coffee component is missing. Note that because Component 3 and 4 are positioned RELATIVE-TO the components defined above them in the Compound Layout syntax, the Food and Gifts components move up instead of leaving blank space where the Coffee component would have been:

<b>GOTHAM GRINDS #1020</b> <b>MARY MULLER</b> <b>79 RICHMOND LANE</b> <b>CHICAGO, IL 60601</b>						
<b>Food</b>						
<b>Store Code</b>	<b>Product</b>	<b>Ordered Units</b>	<b>Unit Price</b>	<b>Package</b>	<b>Size</b>	<b>Vendor Name</b>
R1020	Biscotti	2475	138.00	Case	24	Delancey Bakeries
	Croissant	2417	224.00	Case	20	West Side Bakers
	Soone	1589	52.00	Case	20	Ridgewood Bakeries
<b>Gifts</b>						
<b>Store Code</b>	<b>Product</b>	<b>Ordered Units</b>	<b>Unit Price</b>	<b>Package</b>	<b>Size</b>	<b>Vendor Name</b>
R1020	Coffee Grinder	796	500.00	Case	12	Appliance Craft
	Coffee Pot	863	560.00	Case	8	Appliance Craft
	Mug	1906	208.00	Case	24	NY Ceramic Supply
	Thermos	1043	384.00	Case	16	ThermoTech, Inc

On page 5, the header report did not retrieve any data for a store code value present in the other three components. A page is still generated for this store code. Since the second component (Coffee report) was positioned absolutely in the Compound Layout syntax, not RELATIVE-TO the first component (header report), the space where the header report would have been is left blank:

<b>Coffee</b>						
Store	Product	Ordered Units	Unit Price	Package	Size	Vendor Name
R1088	French Roast	4535	972.00	Pounds	12	European Specialities,
	Hazelnut	1147	232.00	Pounds	16	Coffee Connection
	Kona	1316	304.00	Pounds	12	Evelina Imports, Ltd
<b>Food</b>						
Store Code	Product	Ordered Units	Unit Price	Package	Size	Vendor Name
R1088	Biscotti	815	68.00	Case	24	Delancey Bakeries
	Croissant	4017	336.00	Case	20	West Side Bakers
	Scone	1209	52.00	Case	20	Ridgewood Bakeries
<b>Gifts</b>						
Store Code	Product	Ordered Units	Unit Price	Package	Size	Vendor Name
R1088	Coffee Grinder	648	500.00	Case	12	Appliance Craft
	Coffee Pot	1354	560.00	Case	8	Appliance Craft
	Mug	1281	208.00	Case	24	NY Ceramic Supply
	Thermos	915	384.00	Case	16	ThermoTech, Inc

## Generating a Table of Contents With BY Field Entries for PDF Compound Layout Reports

Using compound layout syntax, you can generate a Table of Contents for a PDF compound report.

The Table of Contents can be presented as either PDF bookmarks displayed by Adobe Reader®, or as a Table of Contents page placed at the beginning of the document, or both. Both the Table of Contents entries and the bookmarks provide links to each of the components and included sort fields. These links position the reader on the page where the Table of Contents link is located.

Include any report or graph component in the Table of Contents at a specific level by defining a TOC Description and TOC Level in the compound layout syntax for the component. Additionally, the BY field values of any of the report components can be presented within the TOC tree indented one level within the component report entry. BYTOC entries are supported for report components only (not graph components).

PDF Compound reports defining the Table of Contents page or Bookmarks based on BY field entries are supported for both non-coordinated (MERGE=OFF) and coordinated reports (MERGE=ON). For coordinated reports, the primary sort key is presented as the top-level entry for individual instances of the report, and subsequent keys are presented within the appropriate components within the tree. These coordinated reports can also be burst into separate documents by the primary sort key and distributed using ReportCaster.

### Table of Contents Features

The Table of Contents (TOC) page shows a summary of the contents of the document, along with page numbers, and can be printed with the document. The entries in the Table of Contents enable you to easily navigate to a particular section while viewing the document online. The entries can link to any component of the compound output (page, report, or graph), any object (image, text box) within the compound report, and vertical sort field values (BY field values) within each component report.

The actual content of the Table of Contents is represented as a text element in the compound layout syntax. When using a Table of Contents page, you can:

- ☐ Customize the title of the Table of Contents, as well as format all of the text.
- ☐ Specify a TOC level for each component to be included in the Table of Contents.
- ☐ Enable TOC page numbering so that each element in the Table of Contents is numbered. You can also add tab leaders (dots) from the entry to the page number for easier selection of the contents.
- ☐ Control which reports and graphs show up in the Table of Contents by customizing the object properties.
- ☐ Use hypertext links in the Table of Contents page which enable you to click on an entry and jump to the specified page in the document.

**Note:** If the Table of Contents overflows to more than one page at run time, the remaining content is executed with the same size and dimensions as the first page until the entire TOC has been output.

**Syntax: How to Generate a Table of Contents in a PDF Compound Layout Report**

You can generate bookmarks, a Table of Contents page, or both by including the BOOKMARKS and/or TOC object in the compound layout syntax. Then in the component definitions, specify the Table of Contents level and description and, optionally, the BYTOC levels.

**TOC Attributes**

```
COMPOUND LAYOUT PCHOLD FORMAT PDF
[OBJECT=BOOKMARKS, $]
[OBJECT=TOC, NAME='text1',
  TEXT='<font face="font1" size=sz1>Table of Contents</font>',
  MARKUP=ON, TOC-NUMBERING={OFF|ON}, POSITION=(xy), [TOC-FILL=DOTS,]
  DIMENSION=(mn),
  font='font2', color={color|RGB(rgb)}, size=sz2,
  METADATA=' TOCTITLE: Table of Contents', $]
```

where:

**OBJECT=BOOKMARKS**

Generates PDF bookmarks for the Table of Contents entries specified in the COMPONENT declarations.

**OBJECT=TOC**

Generates a Table of Contents page for the Table of Contents entries specified in the COMPONENT declarations.

**NAME='text1'**

Specifies a name for the Table of Contents page item.

**TEXT='<font face="font1" size=sz1>Table of Contents</font>'**

Specifies the title and text characteristics for the Table of Contents title. In order for this information to be interpreted correctly, the MARKUP=ON attribute must be specified. The text will not wrap and must fit within the width of the overall text element.

**MARKUP=ON**

Causes the markup tags used with the Table of Contents title to be interpreted as formatting options, not as text.

**TOC-NUMBERING={OFF|ON}**

Specifies whether the entries on the Table of Contents page are numbered. ON is the default value.

**POSITION=( xy)**

Defines the x and y coordinates for the object on the page.

**DIMENSION=( ab)**

Defines the size of the bounding box for the object.

`font='font2', color={color|RGB(rgb)},size=sz2,`

Specifies text characteristics for the Table of Contents body. If omitted, the attributes are taken from the TEXT attribute.

`TOC-FILL=DOTS`

Places tab leader dots from the entry to the page number. If this attribute is not included, the entry and the page number are separated by blank space.

### Component Entries

`COMPONENT=component1, TITLE='title1', TOC-LEVEL=n, [BYTOC=m,]  
POSITION=(xy),DIMENSION=(ab), $`

where:

`component1`

Is a component to be included in the Table of Contents.

`title1`

Is the title for the component to be used as the TOC entry.

`TOC-LEVEL=n`

Defines *n* as the Table of Contents level for the report, graph, or page layout object. This option defines the hierarchical order of objects within the Table of Contents.

0 = the object is not shown in the Table of Contents.

1 = the object is shown as a first level item in the Table of Contents.

2 = the object is shown as a second level item in the Table of Contents and so on.

`BYTOC=m`

Specifies the number of BY fields to be included within the current component entry (*m*).

`POSITION=(xy)`

Defines the x and y coordinates for the object on the page.

`DIMENSION=(ab)`

Defines the size of the bounding box for the object.

### Reference: Usage Notes for Table of Contents

#### General Notes

- ❑ The Table of Contents entry is a link to the overall page, not a direct link to the location of the selected data element on the page.

- ❑ Reports used for headings or footings on overflow pages (DisplayOn=OVERFLOW-ONLY) should not be included in any TOC or BYTOC entries. These components will generate duplicate entries in the Table of Contents because they are repeated on the second page and all subsequent pages of a given page layout based on page count rather than sort field values.
- ❑ For an uncoordinated document, the TOC presents all of the page layouts and components designated in the TOC parameters.
- ❑ For a coordinated document, the TOC presents a reference to each of the coordinated sort fields, with a link to the first page of the associated pages for that common sort field.

### BYTOC Notes

- ❑ If the BYTOC value designated is greater than the count of BY fields in the component report, the value defaults to the total count of available BY fields.
- ❑ Any valid field that can be used as a BY field can be used in a BYTOC entry, including those taken directly from a data source or created in a DEFINE.
- ❑ BY fields designated not to be displayed within the body of the component report (NOPRINT fields) will still be displayed within the Table of Contents.
- ❑ The TOC-level values defined cannot skip levels. Any skip in level will cause the bookmarks below that level not to display.

### *Example:* Creating Bookmarks and a Table of Contents Page

The following example has two component reports. Both PDF bookmarks and a Table of Contents page are generated. Each component is at TOC level 1 and each has two levels of BY fields under the Level 1 entries:

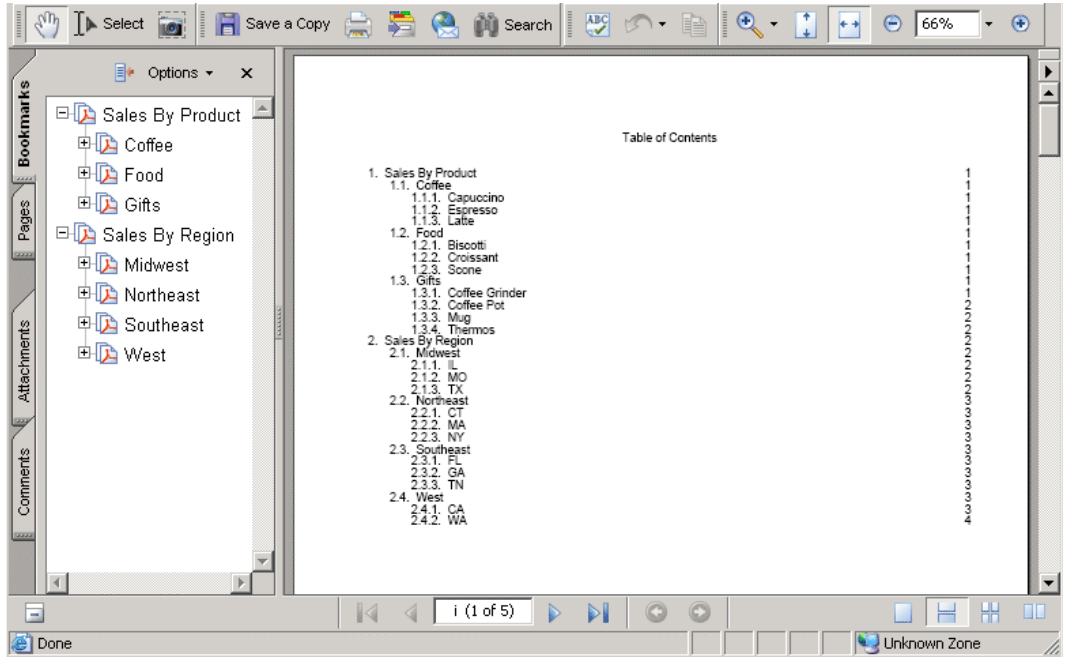
```
COMPOUND LAYOUT PCHOLD FORMAT PDF
OBJECT=BOOKMARKS, $
OBJECT=TOC, NAME='text1',
  TEXT='<font face="ARIAL" size=10>Table of Contents</font>',
  MARKUP=ON, TOC-NUMBERING=ON, POSITION=(0.854 0.854),
  DIMENSION=(7.000 9.500), font='ARIAL', color=RGB(0 0 0),
  size=10, METADATA=' TOCTITLE: Table of Contents', $
SECTION=S1, LAYOUT=ON, MERGE=OFF, ORIENTATION=PORTRAIT, $
PAGELAYOUT=1, $
COMPONENT=report1, TEXT='Sales By Product', TOC-LEVEL=1, BYTOC=2,
  POSITION=(1 1), DIMENSION=(* *), $
COMPONENT=report2, TEXT='Sales By Region', TOC-LEVEL=1, BYTOC=2,
  POSITION=(+0.00 +0.519), DIMENSION=(* *), RELATIVE-TO='report1',
  RELATIVE-POINT=BOTTOM-LEFT, POSITION-POINT=TOP-LEFT, $
END
```

```
SET COMPONENT=report1
TABLE FILE GGSALES
SUM DOLLARS/F8M
BY CATEGORY
BY PRODUCT
BY REGION
BY ST
HEADING
"Sales by Category"
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, SQUEEZE=ON, $
ENDSTYLE
END
```

```
SET COMPONENT=report2
TABLE FILE GGSALES
SUM DOLLARS/F8M
BY REGION
BY ST
BY CATEGORY
BY PRODUCT
HEADING
"Sales by Region"
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, SQUEEZE=ON, $
ENDSTYLE
END
COMPOUND END
```



The output shows that each component reports is at Table of Contents level 1 and has two levels of sort fields under it. For the Sales by Product report, the BY fields are Category and Product. For the Sales by Region report, the BY fields are Region and State. Each entry in the Table of Contents is a link to the page containing that value:



Clicking any entry on the Table of Contents page or in the bookmarks pane opens the page containing that entry. For example, clicking the Sales by Region/Southeast entry opens the following page:

Region	State	Category	Product	Dollar Sales
Southeast	FL	Coffee	Capuccino	242,000
		Coffee	Espresso	221,000
		Coffee	Latte	221,000
	FL	Food	Capuccino	221,000
		Food	Espresso	221,000
		Food	Latte	221,000
	FL	Gifts	Capuccino	221,000
		Gifts	Espresso	221,000
		Gifts	Latte	221,000
Southeast	GA	Coffee	Capuccino	221,000
		Coffee	Espresso	221,000
		Coffee	Latte	221,000

Creating a Compound PDF or PS Report

Compound reports combine multiple reports into a single PDF or PS file. The first PDF or PS report defines the format for the concatenated report, enabling you to intersperse intermediate reports of other formats into one encompassing report. Using compound reports, you can gather data from different data sources and combine reports into one governing report that runs each request and concatenates the output into a single PDF or PS file.

You can then run or distribute the report with ReportCaster, which displays the compound PDF report in Adobe Reader or sends the compound PS report directly to a printer. See the ReportCaster documentation for details about this product.

This is supported with styled formats, such as PDF, PS, EXL2K, or XLSX.

For information about creating Drill Through PDF Compound Reports, see [How to Create a Drill Through in a PDF Compound Report](#) on page 892. For information about creating Excel Compound Reports, see [Creating a Compound Excel Report Using EXL2K](#) on page 879.

**Syntax:**      **How to Display Compound Reports**

For a compound report that may contain different report types, use the syntax

```
SET COMPOUND= {OPEN|CLOSE} [NOBREAK]
```

or

```
ON TABLE SET COMPOUND {OPEN|CLOSE}
```

Note that when you are using this syntax, you must also include the following code to identify the display format of each of the reports to be concatenated:

```
ON TABLE {PCHOLD|HOLD|SAVE} [AS name] FORMAT formatname
```

If all of the reports in the compound set are of the same type, either PDF or PS, you can use the following, more compact, syntax

```
ON TABLE {PCHOLD|HOLD|SAVE} [AS name] FORMAT {PDF|PS} {OPEN|CLOSE}
[NOBREAK]
```

where:

*name*

Is the name of the generated file. The name is taken from the first request in the compound report. If no name is specified in the first report, the name HOLD is used.

OPEN

Is specified with the first report, and begins the concatenation process. A report that contains the OPEN attribute must be PDF or PS format.

CLOSE

Is specified with the last report, and ends the concatenation process.

NOBREAK

Is an optional phrase that suppresses page breaks. By default, each report is displayed on a separate page.

You can use NOBREAK selectively in a request to control which reports are displayed on the same page.

**Note:**

- ❑ Compound reports cannot be nested.
- ❑ You can save or hold the output from a compound report. For details, see [Saving and Reusing Your Report Output](#) on page 471.
- ❑ Multi-Pane reports cannot be used in a Compound Report.

**Example: Creating a Compound PDF Report**

The following illustrates how to combine three separate PDF reports into one by creating a compound report. Notice that:

- ❑ Report 1 specifies ON TABLE PCHOLD FORMAT PDF OPEN. This defines the report as the first report and sets the format for the entire compound report as PDF.
- ❑ Report 2 species only the format, ON TABLE PCHOLD FORMAT PDF.
- ❑ Report 3 specifies ON TABLE PCHOLD FORMAT PDF CLOSE. This defines the report as the last report.

Note that in this example, all reports are set to PDF format. However, when you create a compound report, only the first report must be in either PDF or PS format. Subsequent reports can be in any styled format. For an illustration, see [How to Embed Graphics in a Compound Report](#) on page 874.

**Report 1:**

```
SET PAGE-NUM=OFF
TABLE FILE CENTORD
HEADING
"Sales Report"
" "
SUM LINEPRICE
BY PRODCAT
ON TABLE SET STYLE *
TYPE=HEADING, SIZE=18, $
ENDSTYLE
ON TABLE PCHOLD FORMAT PDF OPEN NOBREAK
END
```

**Report 2:**

```
TABLE FILE CENTORD
HEADING
"Inventory Report"
" "
SUM QUANTITY
BY PRODCAT
ON TABLE SET STYLE *
TYPE=HEADING, SIZE=18, $
ENDSTYLE
ON TABLE PCHOLD FORMAT PDF NOBREAK
END
```

**Report 3:**

```
TABLE FILE CENTORD
HEADING
"Cost of Goods Sold Report"
" "
SUM LINE_COGS
BY PRODCAT
ON TABLE SET STYLE *
TYPE=HEADING, SIZE=18, $
ENDSTYLE
ON TABLE PCHOLD FORMAT PDF CLOSE
END
```

The output displays as a PDF report. Because the syntax for reports 1 and 2 contain the NOBREAK command, the three reports appear on a single page. (Without NOBREAK, each report displays on a separate page.)

## Sales Report

<u>PRODCAT</u>	<u>Line Total</u>
CD Players	\$40,694,863.53
Camcorders	\$333,884,927.52
Cameras	\$9,718,752.55
DVD	\$53,182,674.89
Digital Tape Recorders	\$38,773,229.69
PDA Devices	\$233,967,566.49
VCRs	\$23,801,009.34

## Inventory Report

<u>PRODCAT</u>	<u>Quantity:</u>
CD Players	308,274
Camcorders	950,700
Cameras	98,141
DVD	241,287
Digital Tape Recorders	524,027
PDA Devices	610,844
VCRs	170,977

## Cost of Goods Sold Report

<u>PRODCAT</u>	<u>Line Cost Of Goods Sold</u>
CD Players	30,519,126.00
Camcorders	299,487,742.00
Cameras	8,944,891.00
DVD	47,275,593.00
Digital Tape Recorders	36,157,863.00
PDA Devices	202,123,656.00
VCRs	22,056,033.00

### **Syntax:** How to Embed Graphics in a Compound Report

You can embed a graphic, such as a logo or a WebFOCUS graph captured as a GIF file, in a compound report. The graphic file must be embedded in a particular report within the set of compound reports.

To save a graph as a graphic image, include the following syntax in your graph request:

**HOLD FORMAT GIF**

For details on saving a graph as an image file, see [Creating a Graph](#) on page 1657.

To embed a graphic in a compound report, you must identify the image file in the StyleSheet declaration of the report in which you want to include it, along with size and position specifications if desired. For details about embedding and positioning graphics in reports, see [Adding an Image to a Report](#) on page 1376.

**Example:**    **Combining Report Formats and Graphs in a Compound Report**

This request generates a compound report from three different report types (PDF, HTML, and EXL2K), and embeds a graph in each report. Notice that each graph is saved as a GIF file in the graph request. The graph is then identified, sized, and positioned within the StyleSheet declaration (TYPE=REPORT, IMAGE=graphname...) of the report in which it is being embedded). Variations on the SET COMPOUND= syntax (OPEN, NOBREAK, CLOSE) combine the three reports on the same page. Key lines of code are highlighted in the following request.

**Report 1:**

```

SET GRMERGE = ON
GRAPH FILE SHORT
SUM PROJECTED_RETURN AS 'Return on Investment'
BY HOLDER
ACROSS CONTINENT
ON GRAPH SET LOOKGRAPH 3D_BAR
ON GRAPH SET GRAPHEDIT SERVER
ON GRAPH HOLD AS SLSGRPH1 FORMAT GIF
END

SET COMPOUND='OPEN NOBREAK'
TABLE FILE SHORT
SUM PROJECTED_RETURN AS 'Return on Investment'
BY CONTINENT
BY HOLDER
HEADING
"Investment Report"
" "

ON TABLE SET STYLE *
TYPE=DATA, BACKCOLOR=( BY=B2 'SILVER' 'WHITE' ), $
TYPE=HEADING, SIZE=14, STYLE=BOLD, $
TYPE=REPORT, IMAGE=SLSGRPH1.gif, POSITION=(4.5 0.5), SIZE=(3.5 2.5), $
ENDSTYLE
ON TABLE PCHOLD FORMAT PDF

END

```

**Report 2:**

```
GRAPH FILE TRADES
SUM AMOUNT
BY CONTINENT
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET GRAPHEDIT SERVER
ON GRAPH HOLD AS TRDSGR1 FORMAT GIF
END
```

```
SET COMPOUND=NOBREAK
TABLE FILE TRADES
SUM AMOUNT AS 'Amount'
BY CONTINENT AS 'Continent'
BY REGION AS 'Region'
HEADING
"Trades Report"
" "
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
  TYPE=DATA, BACKCOLOR=( BY=B2 'SILVER' 'WHITE' ), $
  TYPE=HEADING, SIZE=14, STYLE=BOLD, $
TYPE=REPORT, IMAGE=TRDSGR1.gif, POSITION=(4 3), SIZE=(4 2.5), $
ENDSTYLE
ON TABLE PCHOLD FORMAT HTML
END
```



**Report 3:**

```

GRAPH FILE SHORT
SUM BALANCE
BY CONTINENT
ON GRAPH SET LOOKGRAPH 3D_BAR
ON GRAPH SET GRAPHEDIT SERVER
ON GRAPH SET STYLE *
TYPE=DATA, COLOR=RED,$
ENDSTYLE
ON GRAPH HOLD AS BALGR1 FORMAT GIF
END

```

```

SET COMPOUND=CLOSE
TABLE FILE SHORT
SUM BALANCE AS 'Balance'
BY CONTINENT AS 'Continent'
BY REGION AS 'Region'
HEADING
"Balance by Region"
" "
ON TABLE SET STYLE *
  TYPE=DATA, BACKCOLOR=( BY=B2 'SILVER' 'WHITE' ), $
  TYPE=HEADING, SIZE=14, STYLE=BOLD, $
TYPE=REPORT, IMAGE=BALGR1.gif, POSITION=(4 6), SIZE=(4 2.5), $
ENDSTYLE
ON TABLE PCHOLD FORMAT EXL2K
END

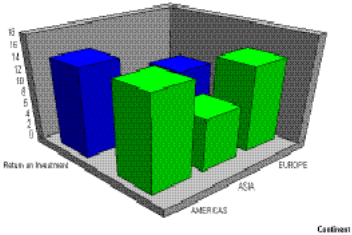
```

The output is:

PAGE 1

Investment Report

Continent	Instrument Holder	Return on Investment
AMERICAS	COMM	14.980
	GOVT	15.400
ASIA	COMM	8.400
	GOVT	8.680
EUROPE	COMM	9.850
	GOVT	14.280



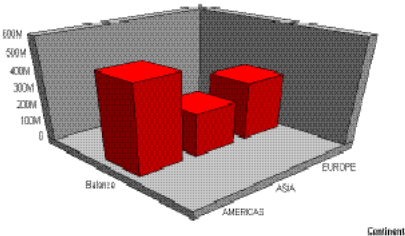
Trades Report

Continent	Region	Amount
AMERICAS	CENTRAL AMERICA	451,331,300
	NORTH AMERICA	1,370,452,600
	SOUTH AMERICA	454,334,100
ASIA	FAR EAST	954,384,400
	MID EAST	230,438,500
EUROPE	EAST EUROPE	436,090,610
	WEST EUROPE	1,042,079,900



Balance by Region

Continent	Region	Balance
AMERICAS	CENTRAL AMERICA	40,200,667
	NORTH AMERICA	367,610,683
	SOUTH AMERICA	87,661,381
ASIA	FAR EAST	202,660,842
	MIDDLE EAST	44,659,771
EUROPE	EASTERN EUROPE	121,977,597
	WESTERN EUROPE	210,550,374



## Creating a Compound Excel Report Using EXL2K

Excel Compound Reports generate multiple worksheet reports using the EXL2K output format.

The syntax of Excel Compound Reports is identical to that of PDF Compound Reports. By default, each of the component reports from the compound report is placed in a new Excel worksheet (analogous to a new page in PDF). If the NOBREAK keyword is used, the next report follows the current report on the same worksheet (analogous to starting the report on the same page in PDF).

Output, whether sent to the client using PCHOLD or saved in a file using HOLD, is generated in Microsoft Web Archive format. This format is labeled Single File Web Page in the Excel Save As dialog. Excel provides the conventionally given file suffixes: .mht or .mhtml. WebFOCUS uses the same .xht suffix that is used for EXL2K reports. Since the output is always a single file, it can be easily distributed using ReportCaster.

The components of an Excel compound report can be FORMULA or PIVOT reports (subject to the restrictions). They cannot be Table of Contents (TOC) reports.

**Note:** Excel 2002 (Office XP) or higher must be installed. Excel Compound Reports will not work with earlier versions of Excel since they do not support the Web Archive file format.

### **Reference:** Guidelines for Using the OPEN, CLOSE, and NOBREAK Keywords and SET COMPOUND

As with PDF, the keywords OPEN, CLOSE, and NOBREAK are used to control Excel compound reports. They can be specified with the HOLD or PCHOLD command or with a separate SET COMPOUND command.

- ☐ OPEN is used on the first report of a sequence of component reports to specify that a compound report be produced.
- ☐ CLOSE is used to designate the last report in a compound report.
- ☐ NOBREAK specifies that the next report be placed on the same worksheet as the current report. If it is not present, the default behavior is to place the next report on a separate worksheet.

NOBREAK may appear with OPEN on the first report, or alone on a report between the first and last reports. (Using CLOSE is irrelevant, since it refers to the placement of the next report, and no report follows the final report on which CLOSE appears.)

- ☐ When used with the HOLD/PCHOLD syntax, the compound report keywords OPEN, CLOSE, and NOBREAK must appear immediately after FORMAT EXL2K, and before any additional keywords, such as FORMULA or PIVOT. For example, you can specify:

☐ `ON TABLE PCHOLD FORMAT EXL2K OPEN`

- ☐ `ON TABLE HOLD AS MYHOLD FORMAT EXL2K OPEN NOBREAK`
- ☐ `ON TABLE PCHOLD FORMAT EXL2K NOBREAK FORMULA`
- ☐ `ON TABLE HOLD FORMAT EXL2K CLOSE PIVOT PAGEFIELDS COUNTRY`
- ☐ As with PDF compound reports, compound report keywords can be alternatively specified using SET COMPOUND:
  - ☐ `SET COMPOUND = OPEN`
  - ☐ `SET COMPOUND = 'OPEN NOBREAK'`
  - ☐ `SET COMPOUND = NOBREAK`
  - ☐ `SET COMPOUND = CLOSE`

### **Reference:** Guidelines for Producing Excel Compound Reports Using EXL2K

- ☐ **Pivot Tables and NOBREAK.** Pivot Table Reports may appear in compound reports, but they may not be combined with another report on the same worksheet using NOBREAK.
- ☐ **Naming of Worksheets.** The default worksheet tab names will be Sheet1, Sheet2, and so on. You have the option to specify a different worksheet tab name by using the TITLETEXT keyword in the stylesheet. For example:

```
TYPE=REPORT, TITLETEXT='Summary Report', $
```

Excel limits the length of worksheet titles to 31 characters. The following special characters cannot be used: ':', '?', '\*', and '/'.

- ☐ **File Names and Formats.** The output file name (AS name, or HOLD by default) is obtained from the first report of the compound report (the report with the OPEN keyword). Output file names on subsequent reports are ignored.

The HOLD FORMAT syntax used in the first component report in a compound report applies to all subsequent reports in the compound report, regardless of their format.

- ☐ **NOBREAK Behavior.** When NOBREAK is specified, the following report appears on the row immediately after the last row of the report with the NOBREAK. If additional spacing is required between the reports, a FOOTING or an ON TABLE SUBFOOT can be placed on the report with the NOBREAK, or a HEADING or an ON TABLE SUBHEAD can be placed on the following report. This allows the most flexibility, since if blank rows were added by default there would be no way to remove them.

**Example: Creating a Simple Compound Report Using EXL2K**

```

SET PAGE-NUM=OFF
TABLE FILE CAR
HEADING
"Sales Report"
" "
SUM SALES
BY COUNTRY
ON TABLE SET STYLE *
type=report, titletext='Sales Rpt', $
type=heading, size=18, $
ENDSTYLE
ON TABLE PCHOLD AS EX1 FORMAT EXL2K OPEN
END

TABLE FILE CAR
HEADING
"Inventory Report"
" "
SUM RC
BY COUNTRY
ON TABLE SET STYLE *
type=report, titletext='Inv. Rpt', $
type=heading, size=18, $
ENDSTYLE
ON TABLE HOLD AS EX1 FORMAT EXL2K
END

TABLE FILE CAR
HEADING
"Cost of Goods Sold Report"
" "
SUM DC
BY COUNTRY
ON TABLE SET STYLE *
type=report, titletext='Cost Rpt', $
type=heading, size=18, $
ENDSTYLE
ON TABLE HOLD AS EX1 FORMAT EXL2K CLOSE
END

```

The output for each tab in the Excel worksheet is:

A1 Sales Report		
	A	B
1	Sales Report	
2		
3	COUNTRY	SALES
4	ENGLAND	12000
5	FRANCE	0
6	ITALY	30200
7	JAPAN	78030
8	W GERMANY	88190
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		

Navigation: Sales Rpt / Inv. Rpt / Cost Rpt /

A1 Inventory Report		
	A	B
1	Inventory Report	
2		
3	COUNTRY	RETAIL_COST
4	ENGLAND	45,319
5	FRANCE	5,610
6	ITALY	51,065
7	JAPAN	6,478
8	W GERMANY	64,732
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		

Navigation: Sales Rpt / **Inv. Rpt** / Cost Rpt /

A1		fx	Cost of Goods Sold Report
	A	B	
1	Cost of Goods Sold Report		
2			
3	COUNTRY		DEALER_COST
4	ENGLAND		37,853
5	FRANCE		4,631
6	ITALY		41,235
7	JAPAN		5,512
8	W GERMANY		54,563
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			

**Example:** Creating a Compound Report With Pivot Tables and Formulas

```

SET PAGE-NUM=OFF
TABLE FILE CAR
HEADING
"Sales Report"
" "
PRINT RCOST
BY COUNTRY
ON TABLE SET STYLE *
type=report, titletext='Sales Rpt', $
type=heading, size=18, $
ENDSTYLE
ON TABLE PCHOLD AS PIV1 FORMAT EXL2K OPEN
END

```

```

TABLE FILE CAR
HEADING
"Inventory Report"
" "
PRINT SALES
BY COUNTRY
ON TABLE SET STYLE *
type=report, titletext='Inv. Rpt', $
type=heading, size=18, $
ENDSTYLE
ON TABLE HOLD AS PPPP FORMAT EXL2K PIVOT
PAGEFIELDS TYPE SEATS
CACHEFIELDS MODEL MPG RPM
END

```

```
TABLE FILE CAR
SUM RCOST
BY COUNTRY BY CAR BY MODEL BY TYPE BY SEATS SUMMARIZE
ON MODEL SUB-TOTAL
ON TABLE HOLD AS XFOCB FORMAT EXL2K FORMULA
END
```

```
TABLE FILE CAR
HEADING
"Cost of Goods Sold Report"
" "
PRINT DCOST
BY COUNTRY
ON TABLE SET STYLE *
type=report, titletext='Cost Rpt', $
type=heading, size=18, $
ENDSTYLE
ON TABLE HOLD AS ONE FORMAT EXL2K CLOSE PIVOT
PAGEFIELDS RCOST
CACHEFIELDS MODEL TYPE SALES ACCEL SEATS
END
```



The output for each tab in the Excel worksheet is:

A1		f <sub>x</sub>	Sales Report
A	B	C	
<b>Sales Report</b>			
COUNTRY	RETAIL COST		
ENGLAND	8,878		
	13,491		
	17,850		
	5,100		
FRANCE	5,610		
ITALY	5,925		
	6,820		
	6,820		
	31,500		
JAPAN	3,139		
	3,339		
W GERMANY	5,970		
	5,940		
	6,355		
	13,752		
	14,123		
	9,097		
	9,495		
▶ Sales Rpt    Inv Rpt    Sheet3    Cost Rpt			

[illegible]

## Creating a Compound Report

	A	B	C	D	E	F	G
1	COUNTRY	CAR	MODEL	BODYTYPE	SEATS	RETAIL_COST	
2	ENGLAND	JAGUAR	V12XKE AUTO	CONVERTIBLE	2	8,878	
3	*TOTAL SEATS 2					8,878	
4	*TOTAL BODYTYPE CONVERTIBLE					8,878	
5	*TOTAL MODEL V12XKE AUTO					8,878	
6			XJ12L AUTO	SEDAN	5	13,491	
7	*TOTAL SEATS 5					13,491	
8	*TOTAL BODYTYPE SEDAN					13,491	
9	*TOTAL MODEL XJ12L AUTO					13,491	
10	*TOTAL CAR JAGUAR					22,369	
11		JENSEN	INTERCEPTOR III	SEDAN	4	17,850	
12	*TOTAL SEATS 4					17,850	
13	*TOTAL BODYTYPE SEDAN					17,850	
14	*TOTAL MODEL INTERCEPTOR III					17,850	
15	*TOTAL CAR JENSEN					17,850	
16		TRIUMPH	TR7	HARDTOP	2	5,100	
17	*TOTAL SEATS 2					5,100	
18	*TOTAL BODYTYPE HARDTOP					5,100	
19	*TOTAL MODEL TR7					5,100	
20	*TOTAL CAR TRIUMPH					5,100	
21	*TOTAL COUNTRY ENGLAND					45,319	
22	FRANCE	PEUGEOT	504 4 DOOR	SEDAN	5	5,610	
23	*TOTAL SEATS 5					5,610	
24	*TOTAL BODYTYPE SEDAN					5,610	

[illegible]

**Example: Creating a Compound Report Using NOBREAK**

In this example, the first two reports are on the first worksheet, and the last two reports are on the second worksheet, since NOBREAK appears on both the first and third reports.

```
TABLE FILE GGSALES
HEADING
"Report 1: Coffee - Budget"
SUM BUDDOLLARS BUDUNITS COLUMN-TOTAL AS 'Total'
BY REGION
IF CATEGORY EQ Coffee
ON TABLE PCHOLD FORMAT EXL2K OPEN NOBREAK
ON TABLE SET STYLE *
type=report, font=Arial, size = 10, style=normal, $
type=title, style=bold, $
type=heading, size=12, style=bold, color=blue, $
type=grandtotal, style=bold, $
ENDSTYLE
END
```

```
TABLE FILE GGSALES
HEADING
" "
"Report 2: Coffee - Actual "
SUM DOLLARS UNITS COLUMN-TOTAL AS 'Total'
BY REGION
IF CATEGORY EQ Coffee
ON TABLE PCHOLD FORMAT EXL2K
ON TABLE SET STYLE *
type=report, font=Arial, size=10, style=normal, $
type=grandtotal, style=bold, $
type=heading, size=12, style=bold, color=blue, $
ENDSTYLE
END
```

```
TABLE FILE GGSALES
HEADING
"Report 3: Food - Budget"
SUM BUDDOLLARS BUDUNITS COLUMN-TOTAL AS 'Total'
BY REGION
IF CATEGORY EQ Food
ON TABLE PCHOLD FORMAT EXL2K NOBREAK
ON TABLE SET STYLE *
type=REPORT, font=Arial, size=10, style=normal, $
type=HEADING, style=bold, size=12, color=blue, $
type=title, style=bold, $
type=grandtotal, style=bold, $
ENDSTYLE
END
```

```
TABLE FILE GGSALES
HEADING
" "
"Report 4: Food - Actual"
SUM DOLLARS UNITS COLUMN-TOTAL AS 'Total'
BY REGION
IF CATEGORY EQ Food
ON TABLE PCHOLD FORMAT EXL2K CLOSE
ON TABLE SET STYLE *
type=report, font=Arial, size=10, $
type=title, style=bold, $
type=heading, size=12, style=bold, color=blue, $
type=grandtotal, style=bold, $
ENDSTYLE
END
```

The output is:

	A	B	C
1	<b>Report 1: Coffee - Budget</b>		
2	<b>Region</b>	<b>Budget Dollars</b>	<b>Budget Units</b>
3	Midwest	4086032	335526
4	Northeast	4252462	335920
5	Southeast	4431429	355693
6	West	4523963	358784
7	<b>Total</b>	<b>17293886</b>	<b>1385923</b>
8			
9	<b>Report 2: Coffee - Actual</b>		
10	<b>Region</b>	<b>Dollar Sales</b>	<b>Unit Sales</b>
11	Midwest	4178513	332777
12	Northeast	4164017	335778
13	Southeast	4415408	350948
14	West	4473517	356763
15	<b>Total</b>	<b>17231455</b>	<b>1376266</b>

	A	B	C
1	<b>Report 3: Food - Budget</b>		
2	<b>Region</b>	<b>Budget Dollars</b>	<b>Budget Units</b>
3	Midwest	4220721	339263
4	Northeast	4453907	351431
5	Southeast	4409288	351509
6	West	4183244	335361
7	<b>Total</b>	<b>17267160</b>	<b>1377564</b>
8			
9	<b>Report 4: Food - Actual</b>		
10	<b>Region</b>	<b>Dollar Sales</b>	<b>Unit Sales</b>
11	Midwest	4338271	341414
12	Northeast	4379994	353368
13	Southeast	4308731	349829
14	West	4202337	340234
15	<b>Total</b>	<b>17229333</b>	<b>1384845</b>

## Creating a PDF Compound Report With Drill Through Links

A common technique in business reporting is to create two related reports:

- ❑ **Summary Report.** Contains condensed information for a category such as a business account, with summed data such as total balances and total sales.
- ❑ **Detail Report.** For specified fields in the associated summary report, a detail report contains all the component values that contributed to each summary field value.

Drill Through provides a way to easily relate the data in these two types of reports. For example, a user scanning a summary report account may see an unusual figure in one of the accounts, requiring examination of the specific data behind that figure.

There are two forms of Compound Drill Through reports:

- ❑ **Legacy Drill Through Compound Reports.** These reports use the legacy compound report syntax (PCHOLD with the OPEN and CLOSE options) to create the compound report.
- ❑ **Drill Through Compound Layout Reports.** These reports use document syntax declarations to create the compound procedure and define which reports will be related through hyperlinks.

In both types of Compound Drill Through reports, the syntax for creating hyperlinks between reports within the PDF file are exactly the same.

### ***Reference:*** Drill Through and Drill Down Compared

Using Drill Down, you can construct a summary report in which clicking a hyperlink displays detail data. A Drill Down is implemented dynamically. Clicking a hyperlink causes a new report to run. The detail report typically displays only the detail data for a selected field on the summary report.

In contrast, Drill Through reports are static. Drill Through creates a PDF document that contains the summary report plus the detail report, with the detail report containing all the detail data for designated fields in the summary report. Clicking a Drill Through hyperlink navigates internally in the PDF file. No additional reports are run. You can save the PDF file to disk or distribute it using ReportCaster. When opened with Adobe Reader, it retains its full Drill Through functionality.

Drill Through provides flexibility in the appearance of reports and location of hyperlinks:

- ❑ Drill Through hyperlinks may appear in headings and subheadings, as well as, in rows of data.
- ❑ You can format the reports using a WebFOCUS StyleSheet.

- ❑ You can indicate a hyperlink by color, font, underlining, and so forth.
- ❑ You can mix conventional Drill Down hyperlinks freely in the same report with Drill Through hyperlinks.
- ❑ The PDF file created with Drill Through can consist of more than two reports.

**Reference: Use With Other Features**

You can use Drill Through with other WebFOCUS features:

- ❑ Compound reports that contain linked Drill Through reports may also contain unrelated reports, before or after the Drill Through reports. For example, you can add to the compound report package a PDF report that contains embedded graphs.
- ❑ You can add Drill Down and URL hyperlinks to a PDF report that contains Drill Through hyperlinks.
- ❑ Since Drill Through reports are standard PDF compound reports packaged into a single PDF file, you can distribute them using ReportCaster.
- ❑ Reports with DRILLTHROUGH syntax can be rendered in all other styled output formats: HTML, PostScript, EXL2K, and so on. In these other formats, the DRILLTHROUGH syntax is ignored. It is useful, for example, to generate a PostScript version of a Drill Through report, which is formatted identically to the PDF version, but which you can send directly to a PostScript printer using ReportCaster or operating system commands.
- ❑ Drill Through automates the process of navigating quickly and easily from general to specific information in related reports packaged in a single PDF compound report. Drill Through syntax sets up hyperlinks that take you from an item in a summary report to a corresponding item in a detail report.

**Procedure: How to Create a Drill Through in a PDF Compound Layout Report**

To create a Drill Through in a PDF Compound Layout report:

1. Create the summary report with a DRILLTHROUGH hyperlink.
2. Create the detail report with sort values that match the hyperlink field values.
3. Create the Compound Layout report with page layouts for each component report, and define a DRILLMAP attribute within the calling report to specify the targets of the drill through hyperlinks.

### ***Procedure:*** How to Create a Drill Through in a PDF Compound Report

To create a Drill Through in a PDF Compound Report:

1. Create the summary report.
2. Create the detail report.
3. Connect the reports with hyperlinks.
4. Merge the summary and detail reports into a PDF compound report.

### ***Syntax:*** How to Specify Drill Through Hyperlinks

```
TYPE=type, [element,] [styling_attributes,]  
    DRILLTHROUGH={DOWN|FIRST}(link_fields) , $
```

where:

*type*

Is one of the following StyleSheet types:

- ☐ DATA
- ☐ HEADING
- ☐ FOOTING
- ☐ SUBHEAD
- ☐ SUBFOOT
- ☐ SUBTOTAL
- ☐ RECAP

*element*

Is one or more identifying elements allowed in a WebFOCUS StyleSheet and Drill Through report. An element can describe a specific column (for example, COLUMN=PRODUCT) or heading item (for example, LINE=2, OBJECT=field, ITEM=3).

*styling\_attributes*

Optionally specify the appearance of the hyperlink (for example, COLOR=RED, STYLE=BOLD).

DOWN

Links to the next (the following) report.

FIRST

Links to the first Drill Through report in the sequence.



*link\_fields*

Specifies blank-delimited link field pairs with the following format:

*T1=S1 T2=S2 T3=S3...*

T1, T2, and T3 represent column references in the target (linked) report, and S1, S2, and S3 represent column references in the source (current) report. There may be more than three pairs.

A column reference can be the name of a field or any of the other symbols valid in WebFOCUS StyleSheets syntax (for example, Bn, Cn, Pn, Nn, An, subscripted field name, and so on).

The order of the syntax is similar to Drill Down syntax, in which the parameter pairs specify the column reference in the current (source) report on the right and the name of the Dialogue Manager variable in the Drill Down (target) procedure on the left.

If the column reference in the target report is identical to the column reference in the source report, you can use a single column reference, for example, COUNTRY instead of COUNTRY=COUNTRY.

**Example: Specifying Drill Through Hyperlinks**

The following StyleSheet declaration places a hyperlink on the PRODUCT field on each DATA line, specifies that the fields to link to the next report are CATEGORY and PRODUCT, specifies the action DOWN, so that clicking a hyperlink brings you to the location in the next report that has the corresponding values of the two link fields, and uses the default appearance for the hyperlinks, which is blue, underlined text. Since the target fields in the detail report have identical names in the summary report, you can use the notation CATEGORY rather than CATEGORY=CATEGORY.

```
TYPE=DATA, COLUMN=PRODUCT, DRILLTHROUGH=DOWN(CATEGORY PRODUCT), $
```

**Syntax: How to Specify Which Compound Layout Reports Will be Related Through Hyperlinks**

The target report is specified in the DRILLMAP attribute of the COMPONENT declaration for the calling report.

```
DRILLMAP=((L1 targetreport))
```

where:

*L1*

Is the link identifier.

*targetreport*

Is the component name of hyperlink destination.

**Note:** The double parentheses around the DRILLMAP values are required.

**Example:** **Sample Component Declarations With DRILLMAP Attributes**

The following COMPONENT declaration for REPORT1 specifies a DRILLMAP attribute that points to REPORT2:

```
COMPONENT='REPORT1', TEXT='REPORT1', TOC-LEVEL=2,
DRILLMAP=((L1 REPORT2)), POSITION=(0.750 1.083), DIMENSION=(7.000 3.167),
METADATA='Z-INDEX: 100; LEFT: 0.75in; OVERFLOW: auto; WIDTH: 7in;
POSITION: absolute; TOP: 1.083in; HEIGHT: 3.167in', $
```

The following COMPONENT declaration for REPORT2 specifies a DRILLMAP attribute that points to REPORT1:

```
COMPONENT='REPORT2', TEXT='REPORT2', TOC-LEVEL=2,
DRILLMAP=((L1 REPORT1)), POSITION=(0.500 0.667), DIMENSION=(7.417 7.000),
METADATA='Z-INDEX: 100; LEFT: 0.5in; OVERFLOW: auto; WIDTH: 7.417in;
POSITION: absolute; TOP: 0.667in; HEIGHT: 7in', $
```

**Reference:** **Usage Notes for Drill Through**

- ☐ As of Release 8.2 Version 01, individual component reports containing Drill Through designations can be run standalone. You will see a warning message in the Message Viewer indicating that outside of the compound report, the Drill Through will not be active.
- ☐ Only one Drill Through behavior can be specified per report.
- ☐ The field specified to contain a Drill Through behavior must also be present in the target report.
- ☐ The originating report containing the Drill Through link must be rendered (by the Reporting Server) prior to the target report. The order of reports must be handled by the user.
- ☐ Live Drill Through links are only generated for PDF output. Reports with DRILLTHROUGH syntax can be rendered in all other styled output formats: HTML, PostScript, EXL2K, and so on. In these other formats, the DRILLTHROUGH syntax is ignored. It is useful, for example, to generate a PostScript version of a Drill Through report, which is formatted identically to the PDF version, but which you can send directly to a PostScript printer using ReportCaster or operating system commands.
- ☐ Drill Through is only supported for reports (TABLE).

## Sample Drill Through PDF Compound Reports

The following examples illustrate how to use Drill Through syntax to create a compound report with a summary and detail report and navigate between them.

### *Example:* Creating the Summary Report (Step 1)

The following syntax generates a sample summary report:

```
TABLE FILE GGSales
SUM UNITS DOLLARS BY CATEGORY BY PRODUCT
ON TABLE PCHOLD FORMAT PDF
END
```

The output is:

PAGE		1	
<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

### Example: Creating the Detail Report (Step 2)

The following syntax generates a sample detail report:

The first page of the output is:

```
SET SQUEEZE=ON
TABLE FILE GGSALES
SUM UNITS BUDUNITS DOLLARS
BY CATEGORY NOPRINT BY PRODUCT NOPRINT
ON CATEGORY PAGE-BREAK
HEADING CENTER
"Category: <CATEGORY"
" "
ON PRODUCT SUBHEAD
"**** Product: <PRODUCT"
ON PRODUCT SUBFOOT
" "
"<25 **** Return to Summary ****"
ON PRODUCT PAGE-BREAK
BY REGION BY CITY
ON TABLE PCHOLD FORMAT PDF
END
```

PAGE	1				
Category: Coffee					
<u>Region</u>	<u>City</u>	<u>Unit Sales</u>	<u>Budget Units</u>	<u>Dollar Sales</u>	
**** Product: Capuccino					
Northeast	Boston	15358	15672	174344	
	New Haven	12386	11098	158995	
	New York	17041	17662	208756	
Southeast	Atlanta	27522	29171	352161	
	Memphis	21599	23090	274812	
	Orlando	24143	23092	317027	
West	Los Angeles	23222	22909	306468	
	San Francisco	23311	23555	279830	
	Seattle	24635	24121	309197	
**** Return to Summary ****					

**Example: Connecting the Reports With Hyperlinks (Step 3)**

The example illustrates the following:

- ❑ When you place a Drill Through hyperlink on a sort-break element, ensure the sort-break is at least at the level of the last sort field participating in the Drill Through. For example, in the second report, the Drill Through hyperlink is on the subfooting associated with PRODUCT rather than the heading (with a sort break) associated with CATEGORY.

Although the code can infer a value of PRODUCT for the CATEGORY heading (you can verify this by embedding the field <PRODUCT> in the heading), it is always the value of the first PRODUCT within that CATEGORY. Typically you want a Drill Through hyperlink for each value of PRODUCT within each CATEGORY.

You do not need to place the hyperlink on an embedded item. You can just as effectively place it on a text item. Any item in the subfooting is associated with the same values of CATEGORY and PRODUCT. Similarly, you can place a hyperlink on any field in a DATA line, and the values of the associated link fields will be identical. Conventional Drill Down hyperlinks also work this way.

The summary report:

- ❑ Places a hyperlink on the PRODUCT field on each DATA line.
- ❑ Specifies that the fields to link to the next report are CATEGORY and PRODUCT. Since the target fields in the detail report have identical names in the summary report, you can use the notation CATEGORY rather than CATEGORY=CATEGORY.
- ❑ Specifies the action DOWN, so that clicking a hyperlink brings you to the location in the next report that has the corresponding values of the two link fields.
- ❑ Uses the default appearance for the hyperlinks, which is blue, underlined text.

The summary report is:

```
TABLE FILE GGSales
SUM UNITS DOLLARS BY CATEGORY BY PRODUCT
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=DATA, COLUMN=PRODUCT, DRILLTHROUGH=DOWN(CATEGORY PRODUCT), $
ENDSTYLE
END
```

The detail report:

- ☐ Places a hyperlink in the subfooting associated with the link field PRODUCT. Since CATEGORY is a higher level BY field than PRODUCT, each PRODUCT subheading is also associated with a unique value of CATEGORY.
- ☐ Places a hyperlink on the first item of the second line of the subfooting, which is the text *Return to Summary*.
- ☐ Specifies action FIRST, so that clicking the hyperlink jumps to the line in the first (summary) report that contains the same values of the two link fields CATEGORY and PRODUCT.
- ☐ Uses the COLOR attribute to display the hyperlink as red, underlined text.

The detail report is:

```
SET SQUEEZE=ON
TABLE FILE GGSales
SUM UNITS BUDUNITS DOLLARS
BY CATEGORY NOPRINT BY PRODUCT NOPRINT
ON CATEGORY PAGE-BREAK
HEADING CENTER
"Category: <CATEGORY"
" "
ON PRODUCT SUBHEAD
"**** Product: <PRODUCT"
ON PRODUCT SUBFOOT
" "
"<25 **** Return to Summary ****"
ON PRODUCT PAGE-BREAK
BY REGION BY CITY
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=SUBFOOT, LINE=2, ITEM=1, DRILLTHROUGH=FIRST(CATEGORY PRODUCT),
COLOR=RED, $
ENDSTYLE
END
```

The next step is the only step that is different for creating a Compound Layout report or a legacy Compound Report.

### **Example:** Creating the Compound Layout Report (Step 4)

Perform this version of Step 4 if you are creating a Compound Layout report.

To create the Compound Layout report:

- ☐ Add a COMPOUND LAYOUT and SECTION declaration to the top of the procedure.
- ☐ Add PAGELAYOUT and COMPONENT declarations for the two reports. Add DRILLMAP attributes to the COMPONENT declarations.

- ☐ Add SET COMPONENT commands and the two reports.
- ☐ End the procedure with a COMPOUND END command:

```

SET HTMLARCHIVE=ON
COMPOUND LAYOUT PCHOLD FORMAT PDF
UNITS=IN, $
SECTION=section1, LAYOUT=ON, METADATA='0.5^0.5^0.5^0.5', MERGE=OFF,
    ORIENTATION=PORTRAIT, PAGESIZE=Letter, $
PAGELAYOUT=1, NAME='Page layout 1', text='Page layout 1', TOC-LEVEL=1,
    BOTTOMMARGIN=0.5, TOPMARGIN=0.5, METADATA='BOTTOMMARGIN=0.5,
    TOPMARGIN=0.5,LEFTMARGIN=0,RIGHTMARGIN=0, $
COMPONENT='REPORT1', TEXT='REPORT1', TOC-LEVEL=2,
DRILLMAP=((L1 REPORT2)), POSITION=(0.750 1.083), DIMENSION=(7.000 3.167),
    METADATA='Z-INDEX: 100; LEFT: 0.75in; OVERFLOW: auto; WIDTH: 7in;
    POSITION: absolute; TOP: 1.083in; HEIGHT: 3.167in', $
PAGELAYOUT=2, NAME='Page layout 2', text='Page layout 2', TOC-LEVEL=1,
    BOTTOMMARGIN=0.5, TOPMARGIN=0.5, METADATA='BOTTOMMARGIN=0.5,
    TOPMARGIN=0.5,LEFTMARGIN=0,RIGHTMARGIN=0, $
COMPONENT='REPORT2', TEXT='REPORT2', TOC-LEVEL=2,
DRILLMAP=((L1 REPORT1)), POSITION=(0.500 0.667), DIMENSION=(7.417 7.000),
    METADATA='Z-INDEX: 100; LEFT: 0.5in; OVERFLOW: auto; WIDTH: 7.417in;
    POSITION: absolute; TOP: 0.667in; HEIGHT: 7in', $
END

-* Add Report1 code and SET COMPONENT command
SET COMPONENT='REPORT1'
TABLE FILE GGSALES
SUM UNITS DOLLARS BY CATEGORY BY PRODUCT
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=DATA, COLUMN=PRODUCT, DRILLTHROUGH=DOWN(CATEGORY PRODUCT), $
ENDSTYLE
END

```

```
-* Add report2 code and SET COMPONENT command
SET COMPONENT='REPORT2'
SET SQUEEZE=ON
TABLE FILE GGSales
SUM UNITS BUDUNITS DOLLARS
BY CATEGORY NOPRINT BY PRODUCT NOPRINT
ON CATEGORY PAGE-BREAK
HEADING CENTER
"Category: <CATEGORY"
" "
ON PRODUCT SUBHEAD
"**** Product: <PRODUCT"
ON PRODUCT SUBFOOT
" "
"<25 **** Return to Summary ****"
ON PRODUCT PAGE-BREAK
BY REGION BY CITY
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=SUBFOOT, LINE=2, ITEM=1, DRILLTHROUGH=FIRST(CATEGORY PRODUCT),
COLOR=RED, $
ENDSTYLE
END
COMPOUND END
```

### ***Example:*** Merging Summary and Detail Reports Into a PDF Compound Report (Step 4)

Perform this version of Step 4 if you are creating a legacy compound report.

The next step is to combine the reports into a single PDF Compound Report. You can:

- ☐ Code the OPEN and CLOSE options on the [PC]HOLD FORMAT PDF command.
- ☐ Code the OPEN and CLOSE options on the SET COMPOUND command before the component report syntax.

Drill Through does not support the NOBREAK option, which displays compound reports without intervening page breaks.



This example uses the OPEN and CLOSE options on the PCHOLD FORMAT PDF command:

```
TABLE FILE GGSales
SUM UNITS DOLLARS BY CATEGORY BY PRODUCT
ON TABLE PCHOLD FORMAT PDF OPEN
ON TABLE SET STYLE *
TYPE=DATA, COLUMN=PRODUCT, DRILLTHROUGH=DOWN(CATEGORY PRODUCT), $
ENDSTYLE
END

SET SQUEEZE=ON
TABLE FILE GGSales
SUM UNITS BUDUNITS DOLLARS
BY CATEGORY NOPRINT BY PRODUCT NOPRINT
ON CATEGORY PAGE-BREAK
HEADING CENTER
"Category: <CATEGORY"
" "
ON PRODUCT SUBHEAD
"**** Product: <PRODUCT"
ON PRODUCT SUBFOOT
" "
"<25 **** Return to Summary ****"
ON PRODUCT PAGE-BREAK
BY REGION BY CITY
ON TABLE PCHOLD FORMAT PDF CLOSE
ON TABLE SET STYLE *
TYPE=SUBFOOT, LINE=2, ITEM=1, DRILLTHROUGH=FIRST(CATEGORY PRODUCT),
COLOR=RED, $
ENDSTYLE
END
```

**Example: Run the Drill Through Report (Step 5)**

Run the compound report. The first page of output has the summary report with the hyperlinks to the individual products in blue and underlined:

PAGE 1

<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	<a href="#"><u>Capuccino</u></a>	189217	2381590
	<a href="#"><u>Espresso</u></a>	308986	3906243
	<a href="#"><u>Latte</u></a>	878063	10943622
Food	<a href="#"><u>Biscotti</u></a>	421377	5263317
	<a href="#"><u>Croissant</u></a>	630054	7749902
	<a href="#"><u>Scone</u></a>	333414	4216114
Gifts	<a href="#"><u>Coffee Grinder</u></a>	186534	2337567
	<a href="#"><u>Coffee Pot</u></a>	190695	2449585
	<a href="#"><u>Mug</u></a>	360570	4522521
	<a href="#"><u>Thermos</u></a>	190081	2385829

Click the hyperlink *Croissant* for the category *Food*. You jump to that detail information. In the detail report, the hyperlink back to the summary report is in red and underlined:

PAGE 6

Category: Food

<u>Region</u>	<u>City</u>	<u>Unit Sales</u>	<u>Budget Units</u>	<u>Dollar Sales</u>
**** Product: Croissant				
Midwest	Chicago	43300	43271	549366
	Houston	46941	47050	587887
	St. Louis	48941	49327	613871
Northeast	Boston	41029	41351	497234
	New Haven	45847	46335	551489
	New York	50518	50178	622095
Southeast	Atlanta	53782	54126	661806
	Memphis	52499	51585	638477
	Orlando	50175	51437	602076
West	Los Angeles	66049	64432	800084
	San Francisco	65214	66025	824457
	Seattle	65759	64872	801060

[\\*\\*\\*\\* Return to Summary \\*\\*\\*\\*](#)

Click the hyperlink *Return to Summary* to return to the first page (summary report).

**Reference: Guidelines on Links For FIRST**

The following guidelines apply:

- ☐ The set of link fields used with FIRST must correspond to the set of link fields used with DOWN on the first report.
- ☐ The action of a FIRST hyperlink in the last report should return to the corresponding line in the first report. The chosen set of links must uniquely identify that line of the first report.
- ☐ The DOWN hyperlink on the line of the first report must uniquely identify that line of the first report to locate the matching line of the detail report. The set of links for the DOWN in the first report and the FIRST link in the last report are the same, since they both must uniquely identify a line in the first report.

**Reference: Rules For Drill Through Hyperlinks**

Reports linked with Drill Through must follow certain rules to ensure that the hyperlinks between them work correctly. The following are key concepts:

- ☐ **Source report.** The report from which you are linking. The source report contains hyperlinks to the target report.
- ☐ **Target report.** The report to which you are linking.
- ☐ **Link field.** One of a set of corresponding fields of the same data type that exist in both the source and target reports. Link fields locate the position in the target report to which a hyperlink in the source report jumps.

Source and target terminology refer to each pair of linked reports. For example, if there are three reports linked with Drill Through, the second report is generally the target report of the first report, and also the source report of the third report.

To process a report as a Drill Through, you must identify the link fields in the source report that relate to the target report:

- ☐ Choose meaningful link fields whose values match in the source and target reports. For example, if a field in the source report contains a part number and a field in the target report contains a Social Security number, the field values will not match and you cannot construct hyperlinks.

- ❑ Specify as many link fields as necessary to uniquely locate the position in the target report that corresponds to the link fields in the source report. For example, if the source report is sorted by STATE and CITY, specifying CITY alone as the link field will be problematic if different states contain a city with the same name.
- ❑ The link fields in the source and target reports must have the same internal (actual) format: the stopped data type and internal length must be identical. Formatting options, such as number of displayed digits and comma suppression, may differ. For example, an A20 field must link to another A20 field. However, an I6C field may link to an I8 field, since internally both are four-byte fields.
- ❑ The link fields must be sort fields or verb objects in both the source and target reports. You can include a NOPRINT (non-display) field, which is useful when constructing a report in which a field is embedded in heading text.
- ❑ Designing Drill Through reports is very similar to designing Drill Down reports. Choosing the link fields for a Drill Through report is similar to choosing the parameter fields in a Drill Down report. Likewise, the syntax of a Drill Through closely parallels the syntax of a Drill Down.
- ❑ Since Drill Through reports are linked by corresponding values of link fields, the hyperlinks must appear on report elements associated with a particular value of a link field. However, hyperlinks do not have to appear on any link itself.
- ❑ Not all line types are appropriate for placement of a Drill Through hyperlink. For example, if a page break occurs on a BY field that is also a Drill Through link, each page heading is clearly associated with a value in that field. However, if a page break occurs because of page overflow, avoid placing a Drill Through link in a heading. Similarly, subheadings, subfootings, subtotals, and recaps are associated only with the values of particular BY fields.

## Navigating Within an HTML Report

---

You can include the following navigation features within a report to control the report display:

- ❑ **Dynamic TOC.** You can add a multi-level table of contents to a report to enhance the viewing and navigation of data. The TOC can appear as an expandable list of clickable values or as a series of drop-down form controls. By clicking values in the TOC, you can toggle between views of the data and quickly locate specific values in the report.
- ❑ **WebFOCUS Viewer.** You can divide an HTML report into multiple webpages to speed the delivery of information to your browser. This feature, also known as on-demand paging, is implemented in the WebFOCUS Viewer, where you can navigate to subsequent, previous, or specific pages after your first page of output is displayed.
- ❑ **Hyperlinks between pages.** You can define automatic hyperlinks in a report that contains multiple pages, making it easy to link consecutive report pages together and navigate to the next or previous page.

For related information, see [Navigating Between Reports](#) on page 924.

### In this chapter:

- ❑ [Navigating Sort Groups From a Table of Contents](#)
  - ❑ [Adding the HTML Table of Contents Tree Control to Reports](#)
  - ❑ [Controlling the Display of Sorted Data With Accordion Reports](#)
  - ❑ [Navigating a Multi-Page Report With the WebFOCUS Viewer](#)
  - ❑ [Linking Report Pages](#)
- 

## Navigating Sort Groups From a Table of Contents

You can enhance navigation within a large HTML report by adding a dynamic HTML-based Table of Contents (TOC). To take advantage of this feature, the report must contain at least one vertical sort (BY) field. If you include more than one sort field in a report, the hierarchy is determined by the order in which the sort fields are specified in the request.

The TOC also enhances the display of groups of data. You can view one section (or page) of the report at a time, or you can view all sections at once. You can control this with a page break. For more information, see [Grouping Sort Fields for Display](#) on page 912.

The TOC displays all values of the first (highest-level) vertical sort field, as well as the values of any lower level BY fields that you designate for inclusion. These values are displayed as an expandable series of links or as a series of list controls. Unless otherwise specified in the request, a new page begins when the highest-level sort field changes.

The display of data for a lower level sort field is controlled by your selection of a higher-level sort field value. For example, in a report sorted first by country and then by car model, if you choose Italy from the TOC for country, you will only see a listing of Italian models in the TOC for cars. Cars produced in other countries are not displayed.

Using the TOC, you can:

- ☐ View any section of a report by clicking the associated link.
- ☐ Toggle between a single section and the entire report content by clicking *View Entire Report*.
- ☐ Remove the TOC by clicking *Remove Table of Contents*. This is beneficial when printing the report from the browser. Double-click anywhere in the report to restore it and continue navigation.

The TOC itself is an object that initially appears as an icon in the upper-left corner of the report or as one or more drop-down lists in a page heading or footing or a report heading or footing.

- ☐ **Heading Option.** To add the TOC to a heading or footing, you can use a StyleSheet. See [How to Add TOC Drop-down List Controls to a Heading](#) on page 917.
- ☐ **Report Option.** To add an HTML TOC object to the upper-left hand corner of a report, you can use a SET command or a PCHOLD command. See [How to Add a TOC Tree Control to a Report Using a SET Command](#) on page 907.

The sizing of tables within the HTML report is done by the browser. The columns are sized to fit the largest data value, and trailing spaces are automatically removed. In standard HTML reports, the data is presented in a single table, so the column widths are fixed for all data rows. Both HTML BYTOC (Report)/TOC (Heading) features place the data for each sort key value into individual tables or sections that are accessed using the HTML control. In the HTML BYTOC/TOC reports, the column widths in each individual table are determined by the data presented for each value, rather than the overall report. These different table sizes become apparent when the *View Entire Report* option is selected. Set SQUEEZE = OFF to define fixed column widths across all of the tables.

### **Reference:** Usage Notes for HTMLARCHIVE With HTML Table of Contents

WebFOCUS interactive reporting features must have a connection to the WebFOCUS client in order to access the components required to operate successfully.

HTMLARCHIVE can be used to create self-contained HTML pages with user-defined images when client access is not available.

To generate HTML pages containing user-defined images that can operate interactively, use one of the following commands:

```
SET HTMLREMBEDIMG=ON
SET HTMLARCHIVE=ON
```

Define BASEURL to point directly to the host machine where these files can be accessed using the following syntax:

```
SET BASEURL=http://{hostname:portnumber}
```

For more information on SET BASEURL, see [Specifying a Base URL](#) on page 813.

## Adding the HTML Table of Contents Tree Control to Reports

You can use three different types of syntax to add an HTML TOC object to a report.

### **Syntax:** How to Add a TOC Tree Control to a Report Using a SET Command.

Using a SET command, the syntax is

At the beginning of a request:

```
SET COMPOUND = 'BYTOC [n]'
```

In a request, use the syntax:

```
ON TABLE SET COMPOUND 'BYTOC [n]'
```

where:

*n*

Represents the number of vertical sort (BY) fields to include in the TOC, beginning with the first (highest-level) sort field in the request. The hierarchy of sort fields is determined by the order in which they are specified in the request.

The default value is 1, meaning that only the highest-level sort field and its values are displayed in the TOC.

By default, a section break is placed after the first (highest-level) sort field, unless otherwise specified in the request.

**Note:** Single quotation marks (') should be used when BYTOC is specified with a number in a SET command.

**Syntax:**      **How to Add a TOC Tree Control to a Report by Using the PCHOLD Command**

Using a PCHOLD command, the syntax is

```
ON TABLE PCHOLD FORMAT HTML BYTOC [n]
```

where:

*n*

Represents the number of vertical sort (BY) fields to include in the TOC, beginning with the first (highest-level) sort field in the request. The hierarchy of sort fields is determined by the order in which they are specified in the request.

The default value is 1, meaning that only the highest-level sort field and its values are displayed in the TOC.

By default, a section break is placed after the first (highest-level) sort field, unless otherwise specified in the request.

**Note:** Single quotation marks (') should not be used when BYTOC is specified in a PCHOLD command with a number.

**Syntax:**      **How to Add a TOC Tree Control to a Report Using a StyleSheet Declaration**

The following syntax enables the TOC tree control in the StyleSheet:

```
TYPE=REPORT, TOC=' n ', $
```

or

```
TYPE=REPORT, TOC=' sortfieldname ', $
```

where:

*n*

Represents the number of vertical sort (BY) fields to include in the TOC, beginning with the first (highest-level) sort field in the request. The hierarchy of sort fields in the TOC Tree is determined by the order in which they are listed in the request.

*sortfieldname*

Specifies the vertical sort (BY) column by its field name.

**Note:** Single quotation marks (') should be used when TOC is specified in the StyleSheet.

**Example:**      **Adding an HTML TOC as an Object in the Report (Report Option)**

You can add an HTML TOC as an icon to the upper-left corner of a report by preceding the request with a SET command, as illustrated in the following request. The TOC will list values of the first (highest level) vertical sort field, PLANT:



```
SET COMPOUND='BYTOC 2'
```

```
TABLE FILE CENTORD
HEADING
"SALES REPORT"
SUM LINEPRICE BY PLANT BY PRODCAT
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

You can also add an HTML TOC as an icon to the upper-left corner of a report by using a SET command within the request.

```
TABLE FILE CENTORD
HEADING
"SALES REPORT"
SUM LINEPRICE BY PLANT BY PRODCAT
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET COMPOUND 'BYTOC 2'
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The following example shows how you can use a PCHOLD command to run the request:

```
TABLE FILE CENTORD
HEADING
"SALES REPORT"
SUM LINEPRICE BY PLANT BY PRODCAT
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML BYTOC 2
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

In the following request, the TOC Tree control is enabled in the Report StyleSheet:

```
TABLE FILE CENTORD
HEADING
"SALES REPORT"
SUM LINEPRICE
BY PLANT BY PRODCAT
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=REPORT, TOC='PRODCAT', $
ENDSTYLE
END
```

**Note:** Single quotation marks (') should be used when TOC is specified in the StyleSheet.

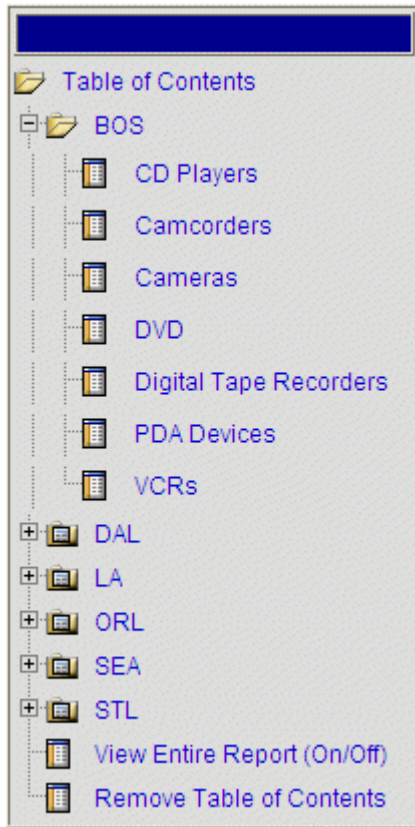
Run the report. The TOC object displays in the upper-left corner.



## SALES REPORT

<u>Manufacturing Plant</u>	<u>PRODCAT</u>	<u>Line Total</u>
BOS	CD Players	\$14,327,431.45
	Camcorders	\$119,144,483.32
	Cameras	\$4,142,193.13
	DVD	\$16,480,170.24
	Digital Tape Recorders	\$13,868,282.30
	PDA Devices	\$85,764,603.13
	VCRs	\$8,706,797.06

Double-click the TOC icon to open the Table of Contents Tree control. This displays the values of the sort fields in the report in the order in which they have been specified.



**Note:** You can move the TOC by clicking the blue area above Table of Contents and then dragging it to another area of the report, or double-click on a desired location in the report.

If you wish to display all available fields (the whole report), click the *View Entire Report (On/Off)* option.

**Tip:** You can also customize the look and feel of the TOC object by editing a .css file. It is recommended that you make a backup copy prior to editing.

❏ If you are working in a self service or Managed Reporting environment from a browser, go to the \ibi\WebFOCUS##\ibi\_apps\ibi\_html\javaassist\intl\xx directory, where ## is the release of WebFOCUS and xx is the language abbreviation. For English (EN) the .css file name is toc.css. For all other languages the .css file name is xxtoc.css, where xx is the language abbreviation.

**Note:** If you click *Remove Table of Contents* and then want to view the TOC again, simply double-click on a desired location in the report.

### **Reference:** Grouping Sort Fields for Display

Data in a TOC report is grouped into sections based on the sort fields. TOC reports only display one section at a time for easier viewing. Each section contains all of the values for its sort field. You can customize each section with a page break. By default, a page break is included in the first (highest level) sort field. You can add page breaks to create additional sections and group data that is based on lower level sort fields.

When adding a TOC to a heading, add additional page breaks for each lower level sort field. This ensures that the sorted data is correctly grouped and displayed.

### **Example:** Customizing Sections of the Report With a Page Break

```
TABLE FILE SHORT
PRINT PROJECTED_RETURN
BY CONTINENT
BY REGION
BY COUNTRY
BY HOLDER
BY TYPE
ON HOLDER PAGE-BREAK
ON TABLE SET PAGE-NUM OFF
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET COMPOUND 'BYTOC 5'
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE=REPORT,
GRID=OFF,
FONT='ARIAL',
SIZE=9,
COLOR='BLACK',
BACKCOLOR='NONE',
STYLE=NORMAL,
$
ENDSTYLE
END
```

One section of the report is displayed at a time.

The report is broken into sections based on the values for `HOLDER`. You will see the detail for each value of `HOLDER` in a single section.

<u>Continent</u>	<u>Region</u>	<u>Country</u>	<u>Instrument Holder</u>	Type of <u>Instrument</u>	Projected Annualized <u>Return</u>
AMERICAS	CENTRAL AMERICA	GUATEMALA	COMM	OVERNIGHT	.150
					.150
					.150
					.150
					.150
					.150
					.150
					.150
					.130
					.130
					.130
					.130
					.130
					.130
					.130

The screenshot shows a window titled "Table of Contents". It contains a list of items:

- Table of Contents
- AMERICAS
- ASIA
- EUROPE
- View Entire Report (On/Off)
- Remove Table of Contents

ST. NOTES

## Navigation Behavior in a Multi-Level TOC

If you select a value in the TOC, that value flashes (that is, it is highlighted in gray) to draw your attention to it in the browser window. Where the flash appears, and whether and how the screen display changes, is controlled by the following factors:

- ❑ When you change the highest level sort group from the TOC (either from the hierarchy above the report or from the first drop-down list in a heading or footing), the selected value flashes three times in the browser window.
- ❑ When you change a lower-level sort value within the current high-level sort group, the selected value flashes three times in window. This is because you are still within the same major sort group, and, therefore, within the same page-break. From the selected value at the top of the window: you can then scroll quickly to the related details.

If the selected lower level value is already viewable on the screen, and the remaining report will fit on the screen, the value flashes, but the report does not scroll.

**Example:**    **Navigating Sorted Data From a Multi-Level TOC**

This request adds a dynamic HTML TOC as an icon in the upper-left corner of the report by including a SET command in the request. The TOC displays a hierarchy consisting of *four* levels of sort fields, beginning with the first (highest-level). The sort fields are: CONTINENT, REGION, COUNTRY, and TYPE.

```
TABLE FILE SHORT
PRINT PROJECTED_RETURN
BY CONTINENT
BY REGION
BY COUNTRY
BY HOLDER
BY TYPE
ON TABLE SET PAGE-NUM OFF
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET COMPOUND 'BYTOC 5'
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE=REPORT,
GRID=OFF,
FONT='ARIAL',
SIZE=9,
STYLE=NORMAL,
$
ENDSTYLE
END
```

The output is displayed with the TOC object in the upper-left corner.

<u>Continent</u>	<u>Region</u>	<u>Country</u>	<u>Instrument Holder</u>	<u>Type of Instrument</u>	<u>Projected Annualized Return</u>
AMERICAS	CENTRAL AMERICA	GUATEMALA	COMM	OVERNIGHT	.150
					.150
					.150
					.150
					.150
					.150
				ST. NOTES	.130
					.130
					.130
					.130
					.130
					.130
			GOVT	CASH	.170
					.170
					.170

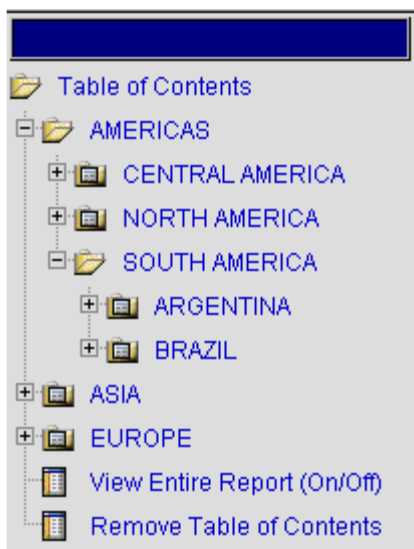
Double-click the object to expand the Table of Contents.



Select *View Entire Report*. Scroll down to see that the report contains data for all of the continents.

Scroll back to the top of the report window and reopen the TOC. This time select *Americas*. Your selection flashes to highlight it on the screen. Although the report display does not appear to change, if you scroll down now you will see that the report only contains values for the Americas.

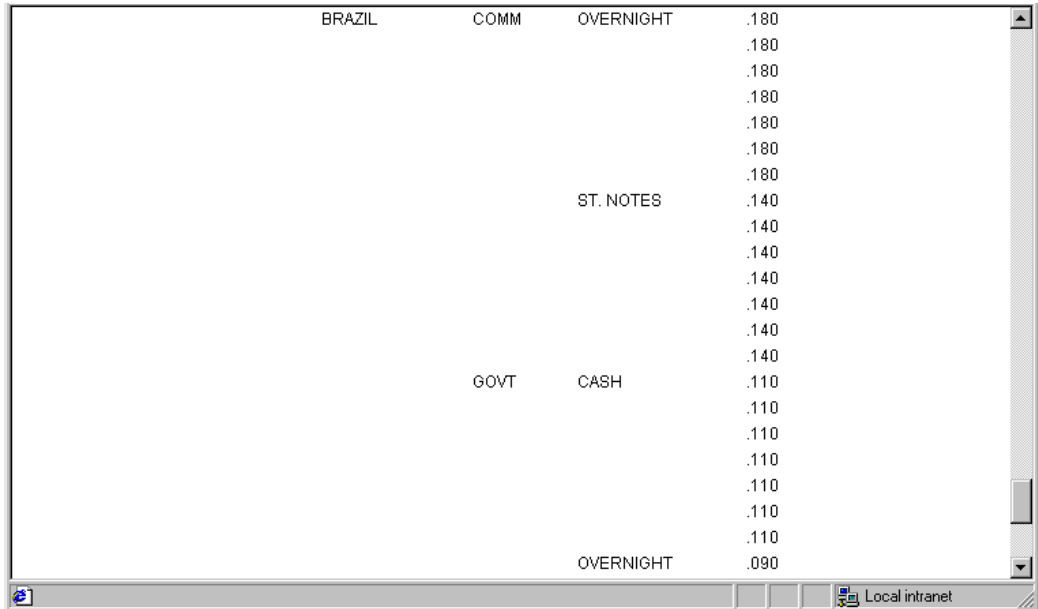
Scroll up again and double-click anywhere in the report to open the TOC. This time click the + sign next to Americas, then click the + sign next to South America.



The field values (Argentina and Brazil) are listed in the TOC. These are values of the field COUNTRY. If you wish to see the field name of a value in the TOC, hover over that value with your cursor.



Select *Brazil*. Your selection flashes and moves to the top of the window, as shown next.



BRAZIL	COMM	OVERNIGHT	.180
			.180
			.180
			.180
			.180
			.180
			.180
		ST. NOTES	.140
			.140
			.140
			.140
			.140
			.140
	GOVT	CASH	.110
			.110
			.110
			.110
			.110
			.110
			.110
		OVERNIGHT	.090

Scroll down to see the data for Brazil.

Continue to navigate to the detail you want to view by choosing values at any sort level in the TOC.

- ☐ Clicking a + sign expands the field to display its values in the TOC.
- ☐ Clicking an actual value (hyperlink) in the TOC momentarily highlights that value and, if necessary, adjusts the report display to move the value into view.

The TOC collapses to its icon when you click *Table of Contents*, but you can continue to scroll back, expand it, and make additional selections.

### **Syntax:** How to Add TOC Drop-down List Controls to a Heading

Include the following attribute in your StyleSheet declaration

`TYPE=heading, [subtype,] TOC=sort_column, $`

where:

*heading*

Is the type of heading or footing that contains the TOC.

Valid values are:

<code>TABHEADING</code>	Report heading.
<code>TABFOOTING</code>	Report footing.
<code>HEADING</code>	Page heading.
<code>FOOTING</code>	Page footing.

### *subtype*

Are attributes that identify the location in the heading or footing where each requested drop-down list will be displayed. These options can be used separately or in combination, depending upon the degree of specificity required to identify a component. Valid values are:

`LINE_#` identifies a line by its position in a heading or footing.

If a heading or footing has multiple lines and you apply a StyleSheet declaration that does not specify `LINE_#`, the declaration is applied to *all* lines. Blank lines are counted when interpreting the value of `LINE`.

`LINE=n` is required for a heading or footing that has multiple lines. Otherwise, you can omit it.

`OBJECT` identifies the TOC object in a heading or footing as a text string or field value. Valid values are `TEXT` or `FIELD`.

You can use a field and/or text as a placeholder for a TOC drop-down list. However, field is preferred. (If the TOC feature is not in effect, the field name is displayed in the report.)

`TEXT` may represent free text or a Dialogue Manager amper (&) variable.

It is not necessary to specify `OBJECT=TEXT` unless you are styling both text strings and embedded fields in the same heading or footing.

For related information, see `ITEM_#`.

`ITEM_#` which identifies an item by its position in a line.

To determine an `ITEM_#` for an `OBJECT`, follow these guidelines:

- ☐ When used with `OBJECT=TEXT`, count only the text strings from left to right.
- ☐ When used with `OBJECT=FIELD`, count only the fields from left to right.

If you apply a StyleSheet declaration that specifies `ITEM_#`, the number is counted from the beginning of each line in the heading or footing, not just from the beginning of the first line.

*sort\_column*

Identifies the vertical sort columns (BY fields) to include as TOCs. You can identify a column using the following notations:

*TOC=fieldname* specifies the sort column by its field name.

*TOC=Bn* specifies the sort column by its order in the request. For example, B2 denotes the second BY field (NOPRINT BY fields are included in the count).

*TOC=n* is the same as *TOC=Bn*.

**Note:** You must maintain the hierarchy of BY fields because the TOC objects in headings (the drop-down lists) are interdependent and corresponds with the hierarchy in the report.

**Example:**     **Adding HTML TOC Drop Down Lists in a Page Heading**

This request uses the required StyleSheet attributes to add a TOC to an HTML report. The drop-down TOC lists the values of the field CONTINENT, identified in the StyleSheet code as OBJECT=FIELD, ITEM=1, TOC=CONTINENT.

```
TABLE FILE SHORT
HEADING
"Projected Returns Report for Region:  <REGION in Continent:  <CONTINENT "
" "
SUM PROJECTED_RETURN
BY CONTINENT
BY REGION
BY COUNTRY
BY TYPE
ON TABLE SET PAGE-NUM OFF
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE=REPORT,
GRID=OFF,
FONT='ARIAL',
SIZE=9,
STYLE=NORMAL,
$
TYPE=HEADING, LINE=1, OBJECT=FIELD, ITEM=1, TOC=CONTINENT, $
ENDSTYLE
END
```

When you run the report. The TOC appears as a drop-down menu in the heading, in place of the field CONTINENT:

Projected Returns Report for Continent: AMERICAS 

<u>Continent</u>	<u>Region</u>	<u>Country</u>	Type of <u>Instrument</u>	Projected Annualized <u>Return</u>
AMERICAS	CENTRAL AMERICA	GUATEMALA	CASH	1.190
			OVERNIGHT	1.890
			ST. NOTES	1.680
		HONDURAS	CASH	1.190
			OVERNIGHT	1.890
			ST. NOTES	1.680
	NORTH AMERICA	CANADA	CASH	.280
			OVERNIGHT	2.240
			ST. NOTES	1.470
		MEXICO	CASH	.490
			OVERNIGHT	2.520
			ST. NOTES	2.030
		UNITED STATES	CASH	.000
			OVERNIGHT	2.030
			ST. NOTES	1.400
	SOUTH AMERICA	ARGENTINA	CASH	.770
			OVERNIGHT	1.890
			ST. NOTES	1.540
		BRAZIL	CASH	.770
			OVERNIGHT	1.890
			ST. NOTES	1.540

Click the TOC to see the list of sort values: AMERICAS, ASIA, EUROPE.

Click each continent to see the related information. The selected value flashes gray to highlight it in the window.

You can display all available fields (the whole report) by clicking the *View Entire Report* option. To remove the TOC, click the *Remove Table of Contents* option. To restore the TOC, double-click anywhere in the report or click the *Refresh* button in your browser.

**Example: Navigating a Multi-Level HTML TOC in a Page Heading**

This request uses a StyleSheet to add an HTML TOC that contains drop-down lists in the third line of the page heading for two sort (BY) fields specified in the request: CONTINENT and REGION. Each field becomes a place-holder for its TOC. (If the TOC features were not in effect, the field would display in the report.)

```
TABLE FILE SHORT
"Projected Return"
" "
"For:<CONTINENT For:<REGION "
SUM PROJECTED_RETURN
BY CONTINENT BY REGION BY COUNTRY BY TYPE
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLCSS ON
ON CONTINENT PAGE-BREAK
ON TABLE SET STYLE *
TYPE=REPORT,
GRID=OFF,
FONT='ARIAL',
SIZE=9,
STYLE=NORMAL,
$
TYPE=HEADING, LINE=1, STYLE=BOLD, $
TYPE=HEADING, LINE=3, OBJECT=FIELD, ITEM=1, TOC=B1,$
TYPE=HEADING, LINE=3, OBJECT=FIELD, ITEM=2, TOC=REGION,$
ENDSTYLE
END
```

The output is:

PAGE 1

### Projected Return

For:  For:

<u>Continent</u>	<u>Region</u>	<u>Country</u>	<u>Type of Instrument</u>	<u>Projected Annualized Return</u>
AMERICAS	CENTRAL AMERICA	GUATEMALA	CASH	1.190
			OVERNIGHT	1.890
			ST. NOTES	1.680
		HONDURAS	CASH	1.190
			OVERNIGHT	1.890
			ST. NOTES	1.680
		NORTH AMERICA	CASH	.280
			OVERNIGHT	2.240
			ST. NOTES	1.470
	SOUTH AMERICA	MEXICO	CASH	.490
			OVERNIGHT	2.520
			ST. NOTES	2.030
		UNITED STATES	CASH	.000
			OVERNIGHT	2.030
			ST. NOTES	1.400
		ARGENTINA	CASH	.770
			OVERNIGHT	1.890
			ST. NOTES	1.540
		BRAZIL	CASH	.770
			OVERNIGHT	1.890
			ST. NOTES	1.540

Click the arrow in the second TOC drop-down list and select *North America*. Keep in mind that the values in this drop-down list are related to those in the higher level drop-down list. They are all part of the same higher level sort group, and therefore, within the same section break. The selected value, North America, flashes and moves to the *top* of the browser window. From there, you can scroll to see the related data, as shown in the image below.

NORTH AMERICA	CANADA	ST. NOTES	1.680
		CASH	.280
		OVERNIGHT	2.240
	MEXICO	ST. NOTES	1.470
		CASH	.490
		OVERNIGHT	2.520
	UNITED STATES	ST. NOTES	2.030
		CASH	.000
		OVERNIGHT	2.030
SOUTH AMERICA	ARGENTINA	ST. NOTES	1.400
		CASH	.770

Note that if you select information already in your field of view, the value will be highlighted in gray and will flash, to draw your attention to it.

Next, scroll up and choose ASIA from the first TOC list. This selection changes your highest-level sort group, and affects all of the lists below it. ASIA flashes and moves to the top of the window, where you see information for the first country (Hong Kong) in the Far East region. The page number is now 2 since this is the second of the high-level sort groups in the TOC.

PAGE 2

### Projected Return

For:	ASIA	For:	FAR EAST	
<u>Continent</u>	<u>Region</u>	<u>Country</u>	<u>Type of Instrument</u>	<u>Projected Annualized Return</u>
ASIA	FAR EAST	HONG KONG	CASH	.770
			OVERNIGHT	2.170
			ST. NOTES	1.610
		JAPAN	CASH	.770
			OVERNIGHT	2.170
			ST. NOTES	1.610
	MIDDLE EAST	ISRAEL	CASH	.700
			OVERNIGHT	1.680
			ST. NOTES	1.400
		SAUDI ARABIA	CASH	.770
			OVERNIGHT	1.890
			ST. NOTES	1.540

Continue to experiment with other selections.

### **Reference:** Navigating Between Reports

Along with techniques and tools for navigating within a report, WebFOCUS provides several mechanisms for navigating between reports. With these features, a user initiates navigation from a report display. You can:

- ☐ Drill down to other reports, URLs, and JavaScript functions. See [Linking a Report to Other Resources](#) on page 763.
- ☐ Define frames and populate them with reports. See [Specifying a Target Frame](#) on page 814.



**Reference: HTML Table of Contents Limits**

The TOC feature:

- ☐ Applies to HTML output.
- ☐ Is supported with Internet Explorer. This feature may generate unexpected behavior when using other browsers. See the Web Browser Support Statement for WebFOCUS ([https://techsupport.informationbuilders.com/tech/wbf/wbf\\_tmo\\_027.html](https://techsupport.informationbuilders.com/tech/wbf/wbf_tmo_027.html)) for information on supported browsers for this and other WebFOCUS features.
- ☐ Does not support the Dialogue Manager command -HTMLFORM.
- ☐ Is not supported with Accordion reports.
- ☐ You cannot designate a TOC for a BY field without also specifying a TOC for its parent (BY) fields. The reason for this is that the TOC controls are interdependent and require the physical presence of each parent control to operate correctly. For example, if the request contains BY COUNTRY BY CAR BY MODEL, a report cannot include a TOC control for CAR without also including one for COUNTRY.
- ☐ The size of a TOC-enabled report is limited to the memory available on the WebFOCUS Client.
- ☐ If your request has both BYTOC Table of Contents and a Heading Table of Contents in the StyleSheet, the output will have the Heading TOC.

If you have installed ReportCaster, you can distribute a report with an HTML TOC by including the following commands in the report request:

- ☐ The SET BASEURL command set to the URL to connect to the application server on which WebFOCUS Client is installed. For information, see [Specifying a Base URL](#) on page 813.

**Controlling the Display of Sorted Data With Accordion Reports**

Accordion Reports provide a way to control the amount of sorted data that appears on an HTML report page. You can produce reports with expandable views for each vertical sort field in a request with multiple BY fields.

You can create two types of Accordion Reports:

- ❑ **Accordion By Row.** Accordion By Row reports present sort field values and their corresponding aggregated measures rolled up so that the highest level sort field and the grand totals are at the top of the report. A tree control can be used to open each dimension and view the associated aggregated values. Clicking the plus sign (+) next to a sort field value opens new rows that display the next lower-level sort field values and subtotals. The lowest-level sort field, when expanded, displays the aggregated data values. This type of Accordion Report is generated using the SET EXPANDBYROW command.

Using the Accordion By Row enhanced interface, navigation is easier when working with wide and large reports in a portal page, the data automatically resizes to fit the size of the container, and the column widths automatically adjust based on the largest data value or column title, whichever is larger. The SET EXPANDBYROWTREE=ON command in a procedure enables the enhanced Accordion by Row feature. For more information on the enhanced interface, see [How to Create an Accordion Report With the Enhanced Interface](#) on page 935.

**Note:** EXPANDBYROW is functionally stabilized. Any future enhancements will be done for EXPANDBYROWTREE.

- ❑ **Accordion By Column.** Accordion By Column reports present rolled up sort field and data values. However, they do not automatically display entire report rows. A plus sign appears to the left of each data value in the column under the highest-level sort heading. For data associated with lower-level sort fields, a plus sign is placed to the left of each data value, but the data does not appear unless manually expanded. Data values of the lowest-level sort field are not expandable. To expand your view of data for any expandable sort field, click a plus sign and all data associated with the next lower-level sort field appears. When you expand a data value under the next to lowest sort heading, all of the remaining associated data values in the report appear. This type of Accordion Report is generated using the SET EXPANDABLE command.

The use of horizontal sort fields coded with ACROSS phrases is supported with Accordion Reports. The ACROSS sort headings that appear above vertical sort headings in a standard HTML report do not display in an Accordion Report until at least one sorted data value has been manually expanded in each expandable sort column.

Two vertical sort fields coded with BY phrases are required when using Accordion Reports. If the command syntax does not contain at least two BY phrases, the Accordion Reports EXPANDABLE, EXPANDBYROW, or EXPANDBYROWTREE command is ignored, no message is generated, and a standard HTML report is created.

**Note:** Accordion Reports are only supported for HTML report output.

## Requirements for Accordion Reports

The following requirements must be taken into consideration when creating Accordion Reports:

- ❑ Adding a drill-down link to an Accordion Report requires that the TARGET parameter must be set to a value that specifies a new HTML frame.
- ❑ Once an Accordion Report is created and delivered to the user, there are no subsequent calls to the WebFOCUS Reporting Server required when the user is interacting with the report. However, the collapsible folder controls on the sort fields require JavaScript and images that reside on the WebFOCUS Client. The user must be connected to the WebFOCUS Web tier components in order to use this feature. For online, connected users of WebFOCUS, no change is required to the report.

However, for distribution of reports using ReportCaster, see the following Reference topic to ensure that the report is delivered correctly as an email attachment or as an archived report in the Report Library.

### **Reference:** Usage Notes for HTMLARCHIVE With Accordion Reports

WebFOCUS interactive reporting features must have a connection to the WebFOCUS client in order to access the components required to operate successfully.

HTMLARCHIVE can be used to create self-contained HTML pages with user-defined images when client access is not available.

To generate HTML pages containing user-defined images that can operate interactively, use one of the following commands:

```
SET HTMLRENDERING=ON
SET HTMLARCHIVE=ON
```

Define BASEURL to point directly to the host machine where these files can be accessed using the following syntax:

```
SET BASEURL=http://{hostname:portnumber}
```

For more information on SET BASEURL, see [Specifying a Base URL](#) on page 813.

### **Reference:** Distributing Accordion Reports With ReportCaster

Distributing Accordion Reports with ReportCaster requires the use of JavaScript components and images located on the WebFOCUS Client. To access the JavaScript components and images from a report distributed by ReportCaster, the scheduled procedure must contain the SET FOCHTMLURL command, which must be set to an absolute URL instead of the default value. For example,

```
SET FOCHTMLURL = http://hostname[:port]/ibi_apps/ibi_html
```

where:

`hostname[:port]`

Is the host name and optional port number (specified only if you are not using the default port number) where the WebFOCUS Web application is deployed.

`ibi_apps/ibi_html`

ibi\_apps is the site-customized web server alias pointing to the WEBFOCUS81/ibi\_apps directory (where ibi\_apps is the default value). ibi\_html is a directory within the path to the JavaScript files that are required to be accessible when viewing an Accordion report.

For more information about coding reports for use with ReportCaster, see the *Tips and Techniques for Coding a ReportCaster Report* appendix in the *ReportCaster* manual.

### Creating an Accordion By Row Report

Accordion By Row reports are HTML reports that offer an interactive interface to data aggregated at multiple levels by presenting the sort fields within an expandable tree. By default, the report will present the highest dimension or sort field (BY value) and the aggregated measures associated with each value. The tree control can be used to open or close each dimension and view the associated aggregated values. Clicking the plus sign (+) next to a sort field value opens new rows that display the next lower level sort field values and subtotals. The lowest level sort field, when expanded, displays the aggregated data values.

Using the SET EXPANDBYROW or SET EXPANDBYROWTREE command with HTMLCSS ON enables any HTML report to be turned into an Accordion By Row request. EXPANDBYROW and EXPANDBYROWTREE automatically invoke the SET SUBTOTALS=ABOVE command, which moves the subtotal rows above the subheading and data rows. A SUB-TOTAL command is automatically added for the next-to-last BY field.

When an Accordion By Row report uses the PRINT command, the innermost level of the resulting tree contains detail records from the data source. There can be many detail records for each combination of BY fields, so it may be unclear what distinguishes the various detail records within the display. In order to make the report more useful, include at least one field in the report that can be used to distinguish between the detail level rows.

When an Accordion By Row report uses the SUM command, each row, even at the innermost level of the tree, is actually a subtotal row and is completely described by the combination of BY fields in the request. Each level will be presented at the aggregated level, and the data values will represent the aggregation of the lowest level BY.

Styling an Accordion By Row report can be done using standard HTML report techniques, but it is important to keep the report structure in mind. All rows, except the lowest level, are actually SUBTOTAL rows and the lowest level contains the report DATA.

Accordion By Row reports display the grand total row as an anchor row below the data. This anchor row displays above both the report and page footings aligned to the left margin of the report. To generate Accordion By Row reports without the grand total anchor row, add ON TABLE NOTOTAL to the request.

Using the Accordion By Row enhanced interface, navigation is easier when working with wide and large reports in a portal page, the data automatically resizes to fit the size of the container, and the column widths automatically adjust based on the largest data value or column title, whichever is larger. The SET EXPANDBYROWTREE=ON command in a procedure enables the enhanced Accordion by Row feature. For more information on the enhanced interface, see [How to Create an Accordion Report With the Enhanced Interface](#) on page 935.

Accordion reports can also be created to be opened by column, instead of by row. See [How to Create an Accordion Report With the Enhanced Interface](#) on page 935 for information on how to create Accordion reports using the SET EXPANDABLE command.

### **Syntax:** How to Create Accordion Reports That Expand By Row

`SET EXPANDBYROW = {OFF|ON|n}`

`ON TABLE SET EXPANDBYROW {OFF|ON|n}`

where:

[OFF](#)

Does not create an accordion report. OFF is the default value.

[ON](#)

Creates an accordion report, which initially displays only the highest sort field level. To see rows on lower levels, click the plus sign (+) next to one of the displayed sort field values.

[ALL](#)

Creates an accordion report in which all sort field levels are initially expanded. To roll up a sort field level, click the minus sign (-) next to one of the sort field values on that level.

*n*

Creates an accordion report in which *n* sort field levels are initially expanded. To roll up an expanded sort field level, click the minus sign (-) next to one of the sort field values on that level.

**Note:**

- ❑ Accordion By Row reports require that the HTMLCSS parameter be set to ON.
- ❑ By default, a blank line is generated before a subtotal on the report output. You can eliminate these automatic blank lines by issuing the SET DROPBLNKLINE=ON command.

**Example: Creating an Accordion By Row SUM Report**

The following request against the GGSALES data source has four sort fields, REGION, ST, CATEGORY, and PRODUCT:

```
TABLE FILE GGSALES
SUM DOLLARS/D8MC
UNITS/D8C
BUDDOLLARS/D8MC BUDUNITS/D8C
BY REGION
BY ST
BY CATEGORY
BY PRODUCT
ON TABLE SET HTMLCSS ON
ON TABLE SET EXPANDBYROW ON
ON TABLE SET DROPBLNKLINE ON
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET STYLE *
TYPE=REPORT,
    COLOR=RGB(66 70 73),
    FONT='TREBUCHET MS',
    SIZE=9,
    SQUEEZE=ON,
    GRID=OFF,
$
TYPE=REPORT,
    GRID=OFF,
    FONT='TREBUCHET MS',
    COLOR=RGB(52 85 64),
$
```

```

TYPE=TITLE,
  COLOR='WHITE',
  BACKCOLOR=RGB(52 85 64),
  STYLE=-UNDERLINE,
$
TYPE=HEADING,
  COLOR='WHITE',
  BACKCOLOR=RGB(52 85 64),
$
TYPE=FOOTING,
  COLOR='WHITE',
  BACKCOLOR=RGB(52 85 64),
$
TYPE=SUBTOTAL,
  BACKCOLOR=RGB(72 118 91),
$
TYPE=SUBTOTAL,
  BY=1,
  COLOR='WHITE',
$
TYPE=SUBTOTAL,
  BY=2,
  COLOR='WHITE',
  BACKCOLOR=RGB(132 159 126),
$
TYPE=SUBTOTAL,
  BY=3,
  COLOR='WHITE',
  BACKCOLOR=RGB(158 184 153),
$
TYPE=GRANDTOTAL,
  COLOR='WHITE',
  BACKCOLOR=RGB(52 85 64),
  STYLE=BOLD,
$
ENDSTYLE
END

```

The initial output shows only the top level BY field (REGION), as shown in the following image.

	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
Midwest	\$11,400,665	905,045	\$11,194,373	907,107
Northeast	\$11,392,300	916,675	\$11,576,921	914,359
Southeast	\$11,710,379	935,232	\$11,807,971	942,247
West	\$11,652,946	932,039	\$11,641,513	930,781
<b>TOTAL</b>	<b>\$46,156,290</b>	<b>3,688,991</b>	<b>\$46,220,778</b>	<b>3,694,494</b>

Clicking the plus sign (+) next to the Midwest region opens the rows that show the states associated with that region, as shown in the following image.

	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
Midwest	\$11,400,665	905,045	\$11,194,373	907,107
IL	\$3,924,401	307,581	\$3,866,856	310,071
MO	\$3,761,286	297,727	\$3,646,838	297,081
TX	\$3,714,978	299,737	\$3,680,679	299,955
Northeast	\$11,392,300	916,675	\$11,576,921	914,359
Southeast	\$11,710,379	935,232	\$11,807,971	942,247
West	\$11,652,946	932,039	\$11,641,513	930,781
TOTAL	\$46,156,290	3,688,991	\$46,220,778	3,694,494

Clicking the plus sign (+) next to the state IL opens the rows that show the categories associated with that state, as shown in the following image.

	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
Midwest	\$11,400,665	905,045	\$11,194,373	907,107
IL	\$3,924,401	307,581	\$3,866,856	310,071
Coffee	\$1,398,779	109,581	\$1,366,264	111,431
Food	\$1,522,847	118,068	\$1,480,855	118,363
Gifts	\$1,002,775	79,932	\$1,019,737	80,277
MO	\$3,761,286	297,727	\$3,646,838	297,081
TX	\$3,714,978	299,737	\$3,680,679	299,955
Northeast	\$11,392,300	916,675	\$11,576,921	914,359
Southeast	\$11,710,379	935,232	\$11,807,971	942,247
West	\$11,652,946	932,039	\$11,641,513	930,781
TOTAL	\$46,156,290	3,688,991	\$46,220,778	3,694,494



Clicking the plus sign (+) next to the Coffee category shows the products associated with that category, as shown in the following image. This is the lowest level of the Accordion By Row report.

	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
[-] Midwest	\$11,400,665	905,045	\$11,194,373	907,107
[-] IL	\$3,924,401	307,581	\$3,866,856	310,071
[-] Coffee	\$1,398,779	109,581	\$1,366,264	111,431
Espresso	\$420,439	32,237	\$401,477	32,416
Latte	\$978,340	77,344	\$964,787	79,015
[+] Food	\$1,522,847	118,068	\$1,480,855	118,363
[+] Gifts	\$1,002,775	79,932	\$1,019,737	80,277
[+] MO	\$3,761,286	297,727	\$3,646,838	297,081
[+] TX	\$3,714,978	299,737	\$3,680,679	299,955
[+] Northeast	\$11,392,300	916,675	\$11,576,921	914,359
[+] Southeast	\$11,710,379	935,232	\$11,807,971	942,247
[+] West	\$11,652,946	932,039	\$11,641,513	930,781
TOTAL	\$46,156,290	3,688,991	\$46,220,778	3,694,494

### *Example:* Creating an Accordion By Row PRINT Report

The following request against the EMPLOYEE data source has two sort fields, DEPARTMENT and YEAR. It uses the PRINT display command.

```

SET EXPANDBYROW = ALL
DEFINE FILE EMPLOYEE
YEAR/YY = HIRE_DATE;
YEARMO/YYM = HIRE_DATE;
END
TABLE FILE EMPLOYEE
PRINT LAST_NAME AS 'Last,Name' FIRST_NAME AS 'First,Name'
CURR_SAL AS 'Current,Salary' ED_HRS AS 'Education,Hours'
BY DEPARTMENT BY YEAR
WHERE YEAR GT 1980
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE=REPORT,
  COLOR=RGB(66 70 73),
  FONT='TREBUCHET MS',
  SIZE=9,
  SQUEEZE=ON,
  GRID=OFF,
$

```

```
TYPE=TITLE,
  BACKCOLOR=RGB(102 102 102),
  COLOR=RGB(255 255 255),
  STYLE=-UNDERLINE+BOLD,
$
TYPE=DATA,
  BACKCOLOR=RGB(255 255 255),
$
TYPE=TITLE,
  COLOR='WHITE',
  BACKCOLOR=RGB(52 85 64),
  STYLE=-UNDERLINE,
$
TYPE=HEADING,
  COLOR='WHITE',
  BACKCOLOR=RGB(52 85 64),
$
TYPE=FOOTING,
  COLOR='WHITE',
  BACKCOLOR=RGB(52 85 64),
$
TYPE=SUBTOTAL,
  BACKCOLOR=RGB(72 118 91),
$
TYPE=SUBTOTAL,
  BY=1,
  COLOR='WHITE',
$
TYPE=SUBTOTAL,
  BY=2,
  COLOR='WHITE',
  BACKCOLOR=RGB(132 159 126),
$
TYPE=SUBTOTAL,
  BY=3,
  COLOR='WHITE',
  BACKCOLOR=RGB(158 184 153),
$
TYPE=GRANDTOTAL,
  COLOR='WHITE',
  BACKCOLOR=RGB(52 85 64),
  STYLE=BOLD,
$
ENDSTYLE
END
```

Including the fields LAST\_NAME and FIRST\_NAME in the report output distinguishes each detail line. However, those fields do not apply to the summary lines, so they are blank on the summary lines.

The output is:

	Last Name	First Name	Current Salary	Education Hours
<input type="checkbox"/> MIS			\$108,002.00	231.00
<input type="checkbox"/> 1981			\$58,742.00	81.00
	SMITH	MARY	\$13,200.00	36.00
	MCCOY	JOHN	\$18,480.00	.00
	CROSS	BARBARA	\$27,062.00	45.00
<input type="checkbox"/> 1982			\$49,260.00	150.00
	JONES	DIANE	\$18,480.00	50.00
	BLACKWOOD	ROSEMARIE	\$21,780.00	75.00
	GREENSPAN	MARY	\$9,000.00	25.00
<input type="checkbox"/> PRODUCTION			\$103,282.00	95.00
<input type="checkbox"/> 1982			\$103,282.00	95.00
	SMITH	RICHARD	\$9,500.00	10.00
	BANNING	JOHN	\$29,700.00	.00
	IRVING	JOAN	\$26,862.00	30.00
	ROMANS	ANTHONY	\$21,120.00	5.00
	MCKNIGHT	ROGER	\$16,100.00	50.00
<b>TOTAL</b>			<b>\$211,284.00</b>	<b>326.00</b>

**Syntax:** How to Create an Accordion Report With the Enhanced Interface

```
SET EXPANDBYROWTREE = {OFF|ON|ALL|n}
ON TABLE SET EXPANDBYROWTREE {OFF|ON|ALL|n}
```

where:

### OFF

Does not create an accordion report, with the enhanced interface. OFF is the default value.

### ON

Creates an accordion report, with the enhanced interface. This setting initially displays only the highest sort field level. To see rows on lower levels, click the plus sign (+) next to one of the displayed sort field values.

### ALL

Creates an accordion report, with the enhanced interface. This setting displays all sort field levels initially expanded. To roll up a sort field level, click the minus sign (-) row next to one of the sort field values on that level.

### n

Creates an accordion report, with the enhanced interface. This setting displays then sort field levels initially expanded. To roll up an expanded sort field level, click the minus sign (-) next to one of the sort field values on that level.

### **Example: Creating an Accordion Report With the Enhanced Interface**

The following request against the GGSALES data source has four sort fields, REGION, ST, CATEGORY, and PRODUCT. The request uses the default stylesheet and the default plus sign (+) and minus sign (-) to expand or collapse a row. In order to create the accordion report, with the enhanced interface, the SET EXPANDBYROWTREE command must be set to ON.

```
TABLE FILE GGSALES
SUM DOLLARS/D8MC
UNITS/D8C
BUDDOLLARS/D8MC BUDUNITS/D8C
BY REGION
BY ST
BY CATEGORY
BY PRODUCT
ON TABLE SET EXPANDBYROWTREE ON
ON TABLE SET DROPBLNKLINE ON
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
ENDSTYLE
END
```

The initial output shows only the top level BY field (REGION), as shown in the following image.

	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
+ Midwest	\$11,400,665	905,045	\$11,194,373	907,107
+ Northeast	\$11,392,300	916,675	\$11,576,921	914,359
+ Southeast	\$11,710,379	935,232	\$11,807,971	942,247
+ West	\$11,652,946	932,039	\$11,641,513	930,781
TOTAL	\$46,156,290	3,688,991	\$46,220,778	3,694,494

Clicking the plus sign (+) next to the Midwest region opens the rows that show the states associated with that region, as shown in the following image.

	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
- Midwest	\$11,400,665	905,045	\$11,194,373	907,107
+ IL	\$3,924,401	307,581	\$3,866,856	310,071
+ MO	\$3,761,286	297,727	\$3,646,838	297,081
+ TX	\$3,714,978	299,737	\$3,680,679	299,955
+ Northeast	\$11,392,300	916,675	\$11,576,921	914,359
+ Southeast	\$11,710,379	935,232	\$11,807,971	942,247
+ West	\$11,652,946	932,039	\$11,641,513	930,781
TOTAL	\$46,156,290	3,688,991	\$46,220,778	3,694,494

Clicking the plus sign (+) next to the state IL opens the rows that show the categories associated with that state, as shown in the following image.

	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
- Midwest	\$11,400,665	905,045	\$11,194,373	907,107
- IL	\$3,924,401	307,581	\$3,866,856	310,071
+ Coffee	\$1,398,779	109,581	\$1,366,264	111,431
+ Food	\$1,522,847	118,068	\$1,480,855	118,363
+ Gifts	\$1,002,775	79,932	\$1,019,737	80,277
+ MO	\$3,761,286	297,727	\$3,646,838	297,081
+ TX	\$3,714,978	299,737	\$3,680,679	299,955
+ Northeast	\$11,392,300	916,675	\$11,576,921	914,359
+ Southeast	\$11,710,379	935,232	\$11,807,971	942,247
+ West	\$11,652,946	932,039	\$11,641,513	930,781
TOTAL	\$46,156,290	3,688,991	\$46,220,778	3,694,494

Clicking the plus sign (+) next to the Coffee category shows the products associated with that category, as shown in the following image. This is the lowest level of the Accordion By Row report.

	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
- Midwest	\$11,400,665	905,045	\$11,194,373	907,107
- IL	\$3,924,401	307,581	\$3,866,856	310,071
- Coffee	\$1,398,779	109,581	\$1,366,264	111,431
Espresso	\$420,439	32,237	\$401,477	32,416
Latte	\$978,340	77,344	\$964,787	79,015
+ Food	\$1,522,847	118,068	\$1,480,855	118,363
+ Gifts	\$1,002,775	79,932	\$1,019,737	80,277
+ MO	\$3,761,286	297,727	\$3,646,838	297,081
+ TX	\$3,714,978	299,737	\$3,680,679	299,955
+ Northeast	\$11,392,300	916,675	\$11,576,921	914,359
+ Southeast	\$11,710,379	935,232	\$11,807,971	942,247
+ West	\$11,652,946	932,039	\$11,641,513	930,781
TOTAL	\$46,156,290	3,688,991	\$46,220,778	3,694,494

You can use the EBRT\_ANCHOR stylesheet attribute to change the default plus sign (+) and minus sign (-) to an arrow. Valid settings for the EBRT\_ANCHOR attribute are PLUSMINUS and ARROWS. The following request changes the default plus sign (+) and minus sign (-) to an arrow, and applies StyleSheet formatting to the request to change the color of the text to white and the background color to different shades of purple.

**Note:** The color of the arrows match the color of the SUBTOTAL line, in this case, white.

```

TABLE FILE GGSALES
SUM DOLLARS/D8MC
UNITS/D8C
BUDDOLLARS/D8MC BUDUNITS/D8C
BY REGION
BY ST
BY CATEGORY
BY PRODUCT
ON TABLE SET EXPANDBYROWTREE ON
ON TABLE SET DROPBLNKLINE ON
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET STYLE *
TYPE=REPORT, EBR1_ANCHOR=ARROWS,
COLOR=RGB(66 70 73), FONT='TREBUCHET MS', SIZE=9, SQUEEZE=ON,$
TYPE=REPORT, FONT='TREBUCHET MS', COLOR=RGB(151 43 153),$
TYPE=TITLE, COLOR='WHITE', BACKCOLOR=RGB(151 43 153), STYLE=-UNDERLINE,$
TYPE=HEADING, COLOR='WHITE', BACKCOLOR=RGB(151 43 153),$
TYPE=FOOTING, COLOR='WHITE', BACKCOLOR=RGB(151 43 153),$
TYPE=SUBTOTAL, COLOR=WHITE, BACKCOLOR=RGB(179 72 180),$
TYPE=SUBTOTAL, BY=2, BACKCOLOR=RGB(208 99 208),$
TYPE=SUBTOTAL, BY=3, BACKCOLOR=RGB(237 127 236),$
TYPE=GRANDTOTAL, COLOR='WHITE', BACKCOLOR=RGB(151 43 153), STYLE=BOLD,$
ENDSTYLE
END

```

The initial output shows only the top level BY field (REGION), as shown in the following image.

	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
► Midwest	\$11,400,665	905,045	\$11,194,373	907,107
► Northeast	\$11,392,300	916,675	\$11,576,921	914,359
► Southeast	\$11,710,379	935,232	\$11,807,971	942,247
► West	\$11,652,946	932,039	\$11,641,513	930,781
<b>TOTAL</b>	<b>\$46,156,290</b>	<b>3,688,991</b>	<b>\$46,220,778</b>	<b>3,694,494</b>



Clicking the arrow next to the Midwest region opens the rows that show the states associated with that region, as shown in the following image.

	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
▼ Midwest	\$11,400,665	905,045	\$11,194,373	907,107
▶ IL	\$3,924,401	307,581	\$3,866,856	310,071
▶ MO	\$3,761,286	297,727	\$3,646,838	297,081
▶ TX	\$3,714,978	299,737	\$3,680,679	299,955
▶ Northeast	\$11,392,300	916,675	\$11,576,921	914,359
▶ Southeast	\$11,710,379	935,232	\$11,807,971	942,247
▶ West	\$11,652,946	932,039	\$11,641,513	930,781
<b>TOTAL</b>	<b>\$46,156,290</b>	<b>3,688,991</b>	<b>\$46,220,778</b>	<b>3,694,494</b>

Clicking the right arrow next to the state IL opens the rows that show the categories associated with that state, as shown in the following image.

	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
▼ Midwest	\$11,400,665	905,045	\$11,194,373	907,107
▼ IL	\$3,924,401	307,581	\$3,866,856	310,071
▶ Coffee	\$1,398,779	109,581	\$1,366,264	111,431
▶ Food	\$1,522,847	118,068	\$1,480,855	118,363
▶ Gifts	\$1,002,775	79,932	\$1,019,737	80,277
▶ MO	\$3,761,286	297,727	\$3,646,838	297,081
▶ TX	\$3,714,978	299,737	\$3,680,679	299,955
▶ Northeast	\$11,392,300	916,675	\$11,576,921	914,359
▶ Southeast	\$11,710,379	935,232	\$11,807,971	942,247
▶ West	\$11,652,946	932,039	\$11,641,513	930,781
<b>TOTAL</b>	<b>\$46,156,290</b>	<b>3,688,991</b>	<b>\$46,220,778</b>	<b>3,694,494</b>

Clicking the right arrow next to the Coffee category shows the products associated with that category, as shown in the following image. This is the lowest level of the Accordion By Row report.

	Dollar Sales	Unit Sales	Budget Dollars	Budget Units
▼ Midwest	\$11,400,665	905,045	\$11,194,373	907,107
▼ IL	\$3,924,401	307,581	\$3,866,856	310,071
▼ Coffee	\$1,398,779	109,581	\$1,366,264	111,431
Espresso	\$420,439	32,237	\$401,477	32,416
Latte	\$978,340	77,344	\$964,787	79,015
► Food	\$1,522,847	118,068	\$1,480,855	118,363
► Gifts	\$1,002,775	79,932	\$1,019,737	80,277
► MO	\$3,761,286	297,727	\$3,646,838	297,081
► TX	\$3,714,978	299,737	\$3,680,679	299,955
► Northeast	\$11,392,300	916,675	\$11,576,921	914,359
► Southeast	\$11,710,379	935,232	\$11,807,971	942,247
► West	\$11,652,946	932,039	\$11,641,513	930,781
TOTAL	\$46,156,290	3,688,991	\$46,220,778	3,694,494

### **Reference:** Usage Notes for EXPANDBYROWTREE

- ❑ EXPANDBYROWTREE is not supported with OLAP. When both OLAP and EXPANDBYROWTREE are enabled, EXPANDBYROWTREE will be ignored. As a workaround, use EXPANDBYROW.
- ❑ EXPANDBYROWTREE is not supported with the AHTML output format. When using EXPANDBYROWTREE with an active report, EXPANDBYROWTREE will be ignored.

### **Accordion By Row Tooltips**

By default, EXPANDBYROW and EXPANDBYROWTREE reports display field information in tooltips activated when you hover the mouse over the values at each level of the tree. Since the column titles are not displayed above the tree control columns, as they would be in a standard HTML report, the field list for the tree is presented in the tooltip in the top-left corner of the report.

Pop-up field descriptions can also be enabled in Accordion By Row reports to present the field descriptions maintained within the Master File or DEFINE associated with the fields.

Titles can be customized by defining an AS name. To remove pop-up field descriptions from the expanding tree, define a blank AS name for the column title. In an Accordion By Row report, pop-up text boxes that display on mouse over present additional information about the fields and columns within the report. In standard Accordion reports, these pop-up text boxes display the column title or AS name for all of the BY values in the expandable tree.

As with standard HTML reports, the POPUPDESC parameter can be set ON to display field descriptions in these pop-up text boxes for all verb columns. Additionally, turning POPUPDESC ON will cause the BY field pop-up text to present the description value, if available.

The table below represents the order of precedence for descriptions displayed in tooltips when the EXPANDBYROW or EXPANDBYROWTREE setting is on.

Existing Field Information	Pop-Up Description Off	Pop-Up Description On
Description		1
AS Name	1	2
Column Title	2	3
Field Name	3	4

The color and size presentation of the tooltips and pop-up descriptions have been standardized for a uniform look throughout all reports.

***Example:* Creating an Accordion By Row Report Without Pop-Up Field Descriptions**

The following example demonstrates how pop-up text will display for the standard Accordion report in the default presentation, which means pop-up descriptions are not turned on.

```

DEFINE FILE GGSales.
UNITS/D12C DESCRIPTION ''=UNITS;
TOTALES/D12CM DESCRIPTION 'DOLLARS*UNITS'=DOLLARS*UNITS;
END
TABLE FILE GGSales
SUM DOLLARS UNITS AS 'Units'
TOTALES AS 'Total Sales'
BY REGION
BY CATEGORY AS ''
BY PRODUCT AS 'Product AS Name'
ON TABLE SET EXPANDBYROW ALL
ON TABLE SET DROPBLNKLINE ALL
ON TABLE SET STYLE *
TYPE=REPORT,
    COLOR=RGB(66 70 73),
    FONT='TREBUCHET MS',
    SIZE=9,
    SQUEEZE=ON,
    GRID=OFF,
$
TYPE=TITLE,
    BACKCOLOR=RGB(102 102 102),
    COLOR=RGB(255 255 255),
    STYLE=-UNDERLINE+BOLD,
$
TYPE=DATA,
    BACKCOLOR=RGB(255 255 255),
$
TYPE=SUBTOTAL,
    BACKCOLOR=RGB(200 200 200),
    STYLE=BOLD,
$
TYPE=GRANDTOTAL,
    BACKCOLOR=RGB(66 70 73),
    COLOR=RGB(255 255 255),
    STYLE=BOLD,
$
ENDSTYLE
END

```

### Fields as defined in the Master File:

```

FIELD=CATEGORY, ALIAS=E02, FORMAT=A11, INDEX=I, TITLE='Category',
    DESC='Product category',$
FIELD=PRODUCT, ALIAS=E04, FORMAT=A16, TITLE='Product', DESC='Product name',$
    FIELD=REGION, ALIAS=E05, FORMAT=A11, INDEX=I, TITLE='Region',
    DESC='Region code',$
FIELD=UNITS, ALIAS=E10, FORMAT=I08, TITLE='Unit Sales',
    DESC='Number of units sold',$

```

The following image shows the pop-up description for the tree control, located at the top-left corner of the table, displaying the list of column titles or AS name for the given BY column within the underlying tree control.

	Dollar Sales	Units	Total Sales
☐ Midwest	Region, Product AS Name	045	\$12,712,781,111
☐ Coffee	4178513	332,777	\$4,949,425,049
Espresso	1294947	101,154	\$1,473,777,907
Latte	2883566	231,623	\$3,475,647,142
☐ Food	4338271	341,414	\$5,339,526,754
Biscotti	1091727	86,105	\$1,351,488,188
Croissant	1751124	139,182	\$2,183,132,532
Scone	1495420	116,127	\$1,804,906,034

The following image shows the pop-up text that will display when the mouse hovers over any of the top level BY values that do not have an AS name, but do have a defined description and title. In this case, the column titles will display.

	Region	Units	Total Sales
☐ Midwest	11400665	905,045	\$12,712,781,111
☐ Coffee	4178513	332,777	\$4,949,425,049
Espresso	1294947	101,154	\$1,473,777,907
Latte	2883566	231,623	\$3,475,647,142

The pop-up text for a top level BY value that has an AS name and a defined description will display the AS name.

	Dollar Sales	Units	Total Sales
☐ Midwest	11400665	905,045	\$12,712,781,111
☐ Coffee	4178513	332,777	\$4,949,425,049
Espresso	Product AS Name	154	\$1,473,777,907
Latte	2883566	231,623	\$3,475,647,142

### **Example:** Creating an Accordion By Row Report With Pop-Up Field Descriptions

The following example demonstrates how pop-up text displays with pop-up descriptions turned on.

```

SET POPUPDESC = ON
DEFINE FILE GGSales.
UNITS/D12C DESCRIPTION ''=UNITS;
TOTSales/D12CM DESCRIPTION 'DOLLARS*UNITS'=DOLLARS*UNITS;
END
TABLE FILE GGSales
SUM DOLLARS UNITS AS 'Units'
TOTSales AS 'Total Sales'
BY REGION
BY CATEGORY AS ''
BY PRODUCT AS 'Product AS Name'
ON TABLE SET EXPANDBYROW ALL
ON TABLE SET DROPBLNKLIN ALL
ON TABLE SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
ENDSTYLE
END

```

As with all HTML reports, setting POPUPDESC=ON will activate a text box that displays the field descriptions for each of the verb column titles.

	Dollar Sales	Units	Total Sales	
Midwest	11400665	905,045	\$12,712,781,111	DOLLARS*UNITS
Coffee	4178513	332,777	\$4,949,425,049	
Espresso	1294947	101,154	\$1,473,777,907	
Latte	2883566	231,623	\$3,475,647,142	
Food	4338271	341,414	\$5,339,526,754	

Additionally, with POPUPDESC set ON, the field description will be presented for the BY elements within the tree. If no field description is defined, the column title or AS name will display.

With POPUPDESC=ON, the defined description will display in the pop-up text, as shown in the following image.

	Region code	Units	Total Sales
Midwest	11400665	905,045	\$12,712,781,111
Coffee	4178513	332,777	\$4,949,425,049
Espresso	1294947	101,154	\$1,473,777,907
Latte	2883566	231,623	\$3,475,647,142

The image that shows the description for the BY value appears in the pop-up text even though an AS name has been given to this field.

For additional information on pop-up field descriptions with HTML reports, see [Displaying Report Data](#) on page 43.

## Accordion By Row With NOPRINT

Hidden, NOPRINT BY fields, can be used in Accordion by Row reports. They allow the calculation of values for, and the sorting of data by, fields which are hidden. NOPRINT sort fields are included in the internal matrix and affect the sorting and aggregation of data in the Accordion report, even though they are not displayed in the report. These NOPRINT sort fields are defined using the BY *sortfield* NOPRINT phrase.

### Note:

- ❑ Hidden or NOPRINT fields are not displayed in tooltips or pop-up descriptions.
- ❑ When using empty or blank AS names, if spaces are added between the quotation marks, for example, BY *fieldname* ' ', the spaces will be removed and the functionality will be the same as BY *fieldname* ''.

### **Example:** Creating an Accordion By Row Report With an Explicit NOPRINT

The following request against the EMPLOYEE data source shows salary data for employees, grouped in categories. The output is sorted by the virtual field NAME\_SORT, which concatenates the LAST\_NAME and FIRST\_NAME fields. The NAME\_SORT field is hidden using NOPRINT on the sort phrase. To display employee names, the NAME\_DISPLAY virtual field is created, which concatenates the FIRST\_NAME field and the LAST\_NAME field.

```

DEFINE FILE EMPLOYEE
NAME_SORT/A50=EMPLOYEE.EMPINFO.LAST_NAME || ( ' , ' |
EMPLOYEE.EMPINFO.FIRST_NAME );
NAME_DISPLAY/A57=EMPLOYEE.EMPINFO.FIRST_NAME | EMPLOYEE.EMPINFO.LAST_NAME;
NAME_CODE/A1=EDIT(LAST_NAME, '9');
NAME_GROUP/A10=IF NAME_CODE LE 'G' THEN 'A-G' ELSE IF NAME_CODE LE 'P'
  THEN 'H-P' ELSE 'Q-Z';
END
TABLE FILE EMPLOYEE
SUM
EMPLOYEE.EMPINFO.CURR_SAL AS 'Current Salary'
BY NAME_GROUP AS 'Alphabetical Group'
BY LOWEST NAME_SORT NOPRINT
BY LOWEST NAME_DISPLAY AS 'Employee Name'
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE SET EXPANDBYROW 2
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET DROPBLNKLINE ON
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE=REPORT,
  COLOR=RGB(66 70 73),
  FONT='TREBUCHET MS',
  SIZE=9,
  SQUEEZE=ON,
  GRID=OFF,
$

```



```

TYPE=TITLE,
    BACKCOLOR=RGB(102 102 102),
    COLOR=RGB(255 255 255),
    STYLE=-UNDERLINE+BOLD,
$
TYPE=DATA,
    BACKCOLOR=RGB(255 255 255),
$
TYPE=SUBTOTAL
    BY=1,
    BACKCOLOR=RGB(200 200 200),
    STYLE=BOLD,
$
TYPE=SUBTOTAL,
    BY=2,
    BACKCOLOR=RGB(200 220 220),
    STYLE=BOLD,
$
TYPE=SUBTOTAL,
    BY=3,
    BACKCOLOR=RGB(220 220 200),
    STYLE=BOLD,
$
TYPE=GRANDTOTAL,
    BACKCOLOR=RGB(66 70 73),
    COLOR=RGB(255 255 255),
    STYLE=BOLD,
$
ENDSTYLE
END

```

If you hover the mouse over any value in the NAME\_DISPLAY column, the tooltip will display the AS name, Employee Name, as shown in the following image.

	Current Salary
☐ A-G	Employee Name 42.00
JOHN BANNING	\$29,700.00
ROSEMARIE BLACKWOOD	\$21,780.00
BARBARA CROSS	\$27,062.00
MARY GREENSPAN	\$9,000.00
☐ H-P	\$79,922.00
JOAN IRVING	\$26,862.00
DIANE JONES	\$18,480.00
JOHN MCCOY	\$18,480.00
ROGER MCKNIGHT	\$16,100.00
☐ Q-Z	\$54,820.00
ANTHONY ROMANS	\$21,120.00
MARY SMITH	\$13,200.00
RICHARD SMITH	\$9,500.00
ALFRED STEVENS	\$11,000.00
<b>TOTAL</b>	<b>\$222,284.00</b>

## Differences Between Reformatted and Redefined BY Fields

When a sort field is dynamically reformatted, both the original and reformatted fields are placed in the internal matrix. The original field is not displayed, but is used to sort or aggregate values.

When using a redefined field, the new column is used to display, sort, or aggregate values.

### *Example:* Creating an Accordion By Row Report With Dynamically Reformatted BY Fields

The following request against the EMPLOYEE data source shows employees and salaries by year of hire. The display fields, HIRE\_DATE and CURR\_SAL, are sorted by HIRE\_DATE reformatted with the format, YY, and by the virtual field NAME\_DISPLAY (employee name).

```

DEFINE FILE EMPLOYEE
NAME_SORT/A50=EMPLOYEE.EMPINFO.LAST_NAME || ( ' , ' |
  EMPLOYEE.EMPINFO.FIRST_NAME ); NAME_DISPLAY/
A57=EMPLOYEE.EMPINFO.FIRST_NAME | EMPLOYEE.EMPINFO.LAST_NAME;NAME_CODE/
A1=EDIT(LAST_NAME, '9'); NAME_GROUP/A10=IF NAME_CODE LE 'G'
  THEN 'A-G' ELSE IF NAME_CODE LE 'P' THEN 'H-P' ELSE 'Q-Z';
END
TABLE FILE EMPLOYEE
SUM
HIRE_DATE
EMPLOYEE.EMPINFO.CURR_SAL
BY HIRE_DATE/YY
BY LOWEST NAME_DISPLAY AS 'Employee Name'
ON TABLE SET PAGE-NUM NOLEAD
WHERE HIRE_DATE LT '820101';
ON TABLE SET EXPANDBYROW ALL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET DROPBLNKLINE ON
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
ENDSTYLE
END

```

In the report generated, there are multiple expandable groups or nodes for the same year, 1981. This occurs because the reformatted values are used for display, but the original values are still used for sorting and aggregating. In this report, 1981 is the common value used to represent two different dates, 81/07/01 and 81/11/02. The sorting takes place on the date, not on the year.

	HIRE_DATE	CURR_SAL
☐ 1980		\$11,000.00
ALFRED STEVENS	80/06/02	\$11,000.00
☐ 1981		\$31,680.00
JOHN MCCOY	81/07/01	\$18,480.00
MARY SMITH	81/07/01	\$13,200.00
☐ 1981		\$27,062.00
BARBARA CROSS	81/11/02	\$27,062.00
<b>TOTAL</b>		<b>\$69,742.00</b>

### **Example:** Creating an Accordion By Row Report With Redefined BY Fields

To sort your data on the reformatted field values instead of the original field values, create a virtual field containing the BY value with the new format applied. This will allow you to display, sort, and aggregate on the new redefined BY value.

```

DEFINE FILE EMPLOYEE
NAME_SORT/A50=EMPLOYEE.EMPINFO.LAST_NAME || ( ' , ' |
EMPLOYEE.EMPINFO.FIRST_NAME );
NAME_DISPLAY/A57=EMPLOYEE.EMPINFO.FIRST_NAME | EMPLOYEE.EMPINFO.LAST_NAME;
NAME_CODE/A1=EDIT(LAST_NAME, '9');
NAME_GROUP/A10=IF NAME_CODE LE 'G' THEN 'A-G' ELSE IF NAME_CODE LE 'P'
  THEN 'H-P' ELSE 'Q-Z';
DATE_HIRED/YY=HIRE_DATE;
END
TABLE FILE EMPLOYEE
SUM EMPLOYEE.EMPINFO.CURR_SAL
BY DATE_HIRED
BY LOWEST NAME_DISPLAY AS 'Employee Name'
ON TABLE SET PAGE-NUM NOLEAD
WHERE HIRE_DATE LT '820101';
ON TABLE SET EXPANDBYROW ALL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET DROPBLNKLINE ON
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
ENDSTYLE
END

```

In the above request, the verb fields, HIRE\_DATE and CURR\_SAL are sorted by the DEFINE field, DATE\_HIRED. The result is a report sorted and aggregated by the redefined date values, as shown in the following image.

	HIRE_DATE	CURR_SAL
☐ 1980		\$11,000.00
ALFRED STEVENS	80/06/02	\$11,000.00
☐ 1981		\$58,742.00
BARBARA CROSS	81/11/02	\$27,062.00
JOHN MCCOY	81/07/01	\$18,480.00
MARY SMITH	81/07/01	\$13,200.00
TOTAL		\$69,742.00

**Reference:** Usage for SET EXPANDBYROW and EXPANDBYROWTREE

The following features are not supported with Accordion By Row reports:

- ☐ HFREEZE
- ☐ HTML Table of Contents
- ☐ On Demand Paging

- ☐ TABLEF
- ☐ OVER
- ☐ ROW-TOTAL
- ☐ ON field RECAP
- ☐ FOR
- ☐ IN
- ☐ SEQUENCE
- ☐ PAGENUM
- ☐ SUBHEAD
- ☐ BORDER
- ☐ MULTILINES
- ☐ BY HIERARCHY
- ☐ Compound Reports
- ☐ Multiple requests in a single HTML report

In certain scenarios, a blank line is generated before a subtotal on the report output. You can eliminate these automatic blank lines by issuing the SET DROPBLNKLINE=ON command.

## Creating an Accordion By Column Report

Accordion By Column reports do not automatically display entire report rows. A plus sign appears to the left of each data value in the column under the highest-level sort heading. For data associated with lower-level sort fields, a plus sign is placed to the left of each data value, but the data does not appear unless manually expanded. Data values of the lowest-level sort field are not expandable. To expand your view of data for any expandable sort field, click a plus sign and all data associated with the next lower-level sort field appears. When you expand a data value under the next to lowest sort heading, all of the remaining associated data values in the report appear. This type of Accordion Report is generated using the SET EXPANDABLE command.

### **Reference:** Support for Accordion By Column Reports

The following commands are not supported when using Accordion Reports:

`BORDER`, `COLUMN`, `FOR`, `IN`, `OVER`, `PAGE-NUM`, `ROW-TOTAL`, `TOTAL`

Data Visualization, HTML BYTOC, OLAP, On Demand Paging (WebFOCUS Viewer), column freezing, and the ReportCaster burst feature are also not supported with Accordion Reports.

### ***Syntax:***      **How to Create Accordion by Column Reports**

To enable Accordion By Column Reports, specify the following

```
ON TABLE SET EXPANDABLE = {ON|OFF}
```

where:

ON

Enables Accordion By Column Reports.

OFF

Disables Accordion By Column Reports. OFF is the default value.

### ***Example:***      **Creating an Accordion By Column Report**

This example shows how to use an EXPANDABLE command to create an Accordion By Column Report.

```
TABLE FILE GGSales  
SUM UNITS DOLLARS  
BY REGION BY ST BY CITY BY CATEGORY  
ON TABLE SET EXPANDABLE ON  
END
```

The following image shows an Accordion by Column Report which displays all data associated with the first-level sort field, Region, by default. The expanded data values you see are the result of a report user clicking plus signs to the left of specific first, second- and third-level sort fields after the report is generated.

Region	State	City	Category	Unit Sales	Dollar Sales
⊕ Midwest					
⊖ Northeast	⊕ CT				
	⊕ MA				
	⊖ NY	⊖ New York	Coffee	116659	1459160
			Food	125473	1555165
			Gifts	70194	887940
⊕ Southeast					
⊖ West	⊖ CA	⊖ Los Angeles	Coffee	109082	1383924
			Food	110417	1381698
			Gifts	78571	1006381
		⊕ San Francisco			
	⊕ WA				

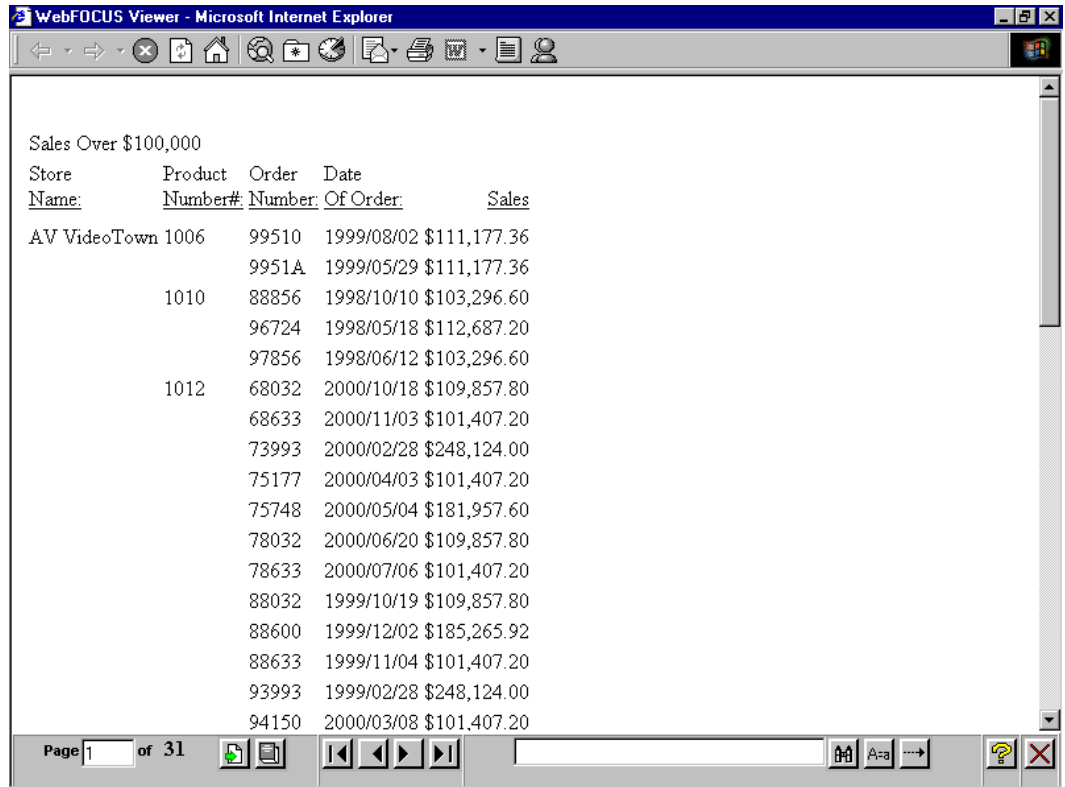
## Navigating a Multi-Page Report With the WebFOCUS Viewer

Normally, a web server returns an entire HTML report to a browser, which waits for all of the report before displaying it. On-demand paging, implemented in the WebFOCUS Viewer, returns one page of a report to a browser instead of the entire report. The web server holds the remaining pages until the user requests them. This feature shortens the time the user waits to see the first page and is especially useful for long reports. It also contains navigational features that enable you to move quickly among the pages of the report.

The WebFOCUS Viewer does not support the table of contents (BYTOC) option because the table of contents option requires all of the data to be on the same HTML page, even though it then filters and only exposes part of the page at a time. The WebFOCUS Viewer splits the output into many pages, only one of which is downloaded to the browser at a time. Accordion reports are also not supported with the WebFOCUS Viewer.

Note that you can use the HFREEZE StyleSheet option to display column titles on every page of output returned by the WebFOCUS Viewer.

The following is page 1 of a 31-page report displayed in the WebFOCUS Viewer.



Store Name	Product Number#	Order Number	Date	Sales
AV VideoTown	1006	99510	1999/08/02	\$111,177.36
		9951A	1999/05/29	\$111,177.36
1010		88856	1998/10/10	\$103,296.60
		96724	1998/05/18	\$112,687.20
		97856	1998/06/12	\$103,296.60
1012		68032	2000/10/18	\$109,857.80
		68633	2000/11/03	\$101,407.20
		73993	2000/02/28	\$248,124.00
		75177	2000/04/03	\$101,407.20
		75748	2000/05/04	\$181,957.60
		78032	2000/06/20	\$109,857.80
		78633	2000/07/06	\$101,407.20
		88032	1999/10/19	\$109,857.80
		88600	1999/12/02	\$185,265.92
		88633	1999/11/04	\$101,407.20
		93993	1999/02/28	\$248,124.00
		94150	2000/03/08	\$101,407.20

Notice that the WebFOCUS Viewer is divided into two frames:

- ☐ The Report Frame is the larger upper frame that contains one page of a report.
- ☐ The Control Frame contains the controls used to navigate the report and to search for a string in the report. The navigational controls allow you to display the next or previous page, the first or last page, or a specific page.

**Note:** You can control whether certain buttons display using SET commands. For more information, see *Controlling Button Display on the WebFOCUS Viewer* in the *Developing Reporting Applications* manual.

### **Reference:** Usage Notes for HTMLARCHIVE With the WebFOCUS Viewer

WebFOCUS interactive reporting features must have a connection to the WebFOCUS client in order to access the components required to operate successfully.



HTMLARCHIVE can be used to create self-contained HTML pages with user-defined images when client access is not available.

To generate HTML pages containing user-defined images that can operate interactively, use one of the following commands:

```
SET HTMLREMBEDIMG=ON
SET HTMLARCHIVE=ON
```

Define BASEURL to point directly to the host machine where these files can be accessed using the following syntax:

```
SET BASEURL=http://{hostname:portnumber}
```

For more information on SET BASEURL, see [Specifying a Base URL](#) on page 813.

### **Procedure: How to Navigate in the WebFOCUS Viewer**

The WebFOCUS Viewer Control Panel offers several ways to view pages in your report:

- ☐ To display the previous or the next page in sequence, click the *Previous* or *Next* arrow.
- ☐ To display the first or last page of the report, click the *First Page* or the *Last Page* arrow.
- ☐ To display a specific page:
  1. Enter a page number in the page input box.
  2. Click the *Go to Page* button.
- ☐ To download the entire report as a single document, click the *All Pages* button. The WebFOCUS Viewer displays the entire report without the Viewer Control Panel.

You can return to viewing a single page of your report by clicking the *Back* button on the browser toolbar.

- ☐ To locate a text string, enter the text in the input box and click the *Find* button.
 

To limit your search by case, toggle the *A=a* button.

To control the direction of your search, toggle the *->/<-* button.

## Using the WebFOCUS Viewer Search Option

The Viewer Control Panel contains controls that offer several ways to search your report. Using the Viewer search controls, you can select a string of information, such as a phrase that occurs in your report or a group of numbers, and search for each occurrence of that string. You can further customize your search by matching capitalization of words exactly (a case-sensitive search) or by controlling the direction of your search (either forward or backward from your starting point in the report).



When using the Search option:

1. The first search starts at top of the document.
2. The second and subsequent searches start from the last successfully found search string in the direction selected in the Viewer Control Panel. Note that the direction control is to the right of the Search entry field used to specify a search string.

## Linking Report Pages

With a StyleSheet, you can insert one or more navigational hyperlinks in a multi-page HTML report. This feature makes it easy for you to link consecutive report pages together without creating individual hyperlinks for each page.

You can define any report component as a hyperlink.

### **Syntax:** How to Link Report Pages

Use the following syntax in an HTML report

`URL=#_destination, $`

where:

*destination*

Is one of the following:

- `#_next` goes to the top of the next report page.
- `#_previous` goes to the top of the previous report page.
- `#_top` goes to the top of the current report page.
- `#_start` goes to the first page of the report.
- `#_end` goes to the last page of the report.

**Note:** In order to use these destinations in a self service applications, the following command must be issued at the beginning of the FOCEXEC:

```
SET BASEURL= ' '
```

**Example:**    **Linking Report Pages Through Images in a Heading**

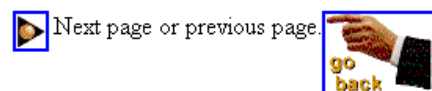
This request displays two images in the page heading of a long report. It creates a link between BULLET.GIF and the next page of the report, and GOBACK.GIF and the previous page of the report.

```
TABLE FILE GGORDER
ON TABLE SUBHEAD
"COFFEE GRINDER SALES BY STORE"
" "
HEADING
"Next page or previous page."
PRINT QUANTITY AS 'Ordered Units' BY STORE_CODE BY PRODUCT NOPRINT
BY ORDER_NUMBER
WHERE PRODUCT EQ 'Coffee Grinder'
ON STORE_CODE PAGE-BREAK
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=TABHEADING, STYLE=BOLD,$
TYPE=HEADING, IMAGE=/IBI_APPS/IBI_HTML/GGDEMO/BULLET, URL=#_next,
IMAGEALIGN=LEFT,$
TYPE=HEADING, IMAGE=/IBI_APPS/IBI_HTML/GGDEMO/GOBACK, URL=#_previous,
IMAGEALIGN=RIGHT,$
ENDSTYLE
END
```

The images display in each page heading.

PAGE 1

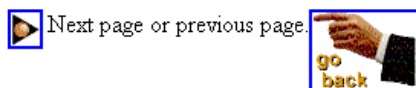
### COFFEE GRINDER SALES BY STORE



<u>Store Code</u>	<u>Order Number</u>	<u>Ordered Units</u>
R1019	9	265
	189	148
	369	382
	549	325
	729	381
	909	41
	1086	295
	1266	221

Click the image on the left of page 1 to display page 2.

PAGE 2

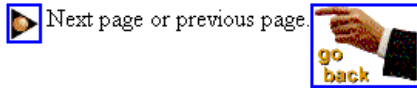


<u>Store Code</u>	<u>Order Number</u>	<u>Ordered Units</u>
R1020	24	133
	204	380
	384	279
	564	357
	744	189
	924	90
	1101	248
	1281	221
	1461	262
	1641	83
	1821	184

Click the "go back" image on page 2 to redisplay page 1.

PAGE 1

### COFFEE GRINDER SALES BY STORE



<u>Store Code</u>	<u>Order Number</u>	<u>Ordered Units</u>
R1019	9	265
	189	148
	369	382
	549	325
	729	381
	909	41
	1086	295
	1266	221

For details on including and positioning images in a report, see [Laying Out the Report Page](#) on page 1249.

Note that if this procedure is part of a self-service application, the following command must be issued at the start of the procedure:

```
SET BASEURL = ''
```

#### **Example:** Linking Pages Through Page Number and Heading Elements

This request creates hyperlinks from the page number to the next page in the report, and from the text of the page heading, which appears at the top of every report page, back to the previous page or to the first page.

```
TABLE FILE GGORDER
ON TABLE SUBHEAD
"COFFEE GRINDER SALES BY STORE"
" "
HEADING
"return to previous page"
"return to beginning"
PRINT QUANTITY AS 'Ordered Units' BY STORE_CODE BY PRODUCT NOPRINT
BY ORDER_NUMBER
WHERE PRODUCT EQ 'Coffee Grinder'
ON STORE_CODE PAGE-BREAK
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=TABHEADING, STYLE=BOLD,$
TYPE=PAGENUM, URL=#_next, $
TYPE=HEADING, LINE=1, URL=#_previous, $
TYPE=HEADING, LINE=2, URL=#_start, $
ENDSTYLE
END
```

The first page looks as follows:

### [PAGE 1](#)

#### **COFFEE GRINDER SALES BY STORE**

[return to previous page](#)

[return to beginning](#)

<u>Store</u> <u>Code</u>	<u>Order</u> <u>Number</u>	<u>Ordered Units</u>
R1019	9	265
	189	148
	369	382
	549	325
	729	381
	909	41
	1086	295
	1266	221
	1446	122

Click the page number three times to move to PAGE 4.

[PAGE 4](#)

[return to previous page](#)

[return to beginning](#)

<u>Store</u> <u>Code</u>	<u>Order</u> <u>Number</u>	<u>Ordered Units</u>
R1041	54	292
	234	165
	414	229
	594	208
	774	294
	953	276
	1131	137
	1311	278
	1491	289

Click *previous page* to return to PAGE 3. Click *return to beginning* to go directly to PAGE 1.

Note that if this procedure is part of a self-service application, the following command must be issued at the start of the procedure:

```
SET BASEURL = ''
```





## Bursting Reports Into Multiple HTML Files

Bursting separates a single report into multiple HTML files based on the value of the first sort field in your report.

### In this chapter:

- ❑ [Bursting Reports Overview](#)

### Bursting Reports Overview

When bursting separates a single report into multiple HTML files, each file contains all the requested information for one specific value of the sort field. By providing direct access to different sections of your report, this technique enables you to:

- ❑ Provide easier navigation of large reports. Bursting automatically creates an index file with direct links to each of the resulting files.
- ❑ Tailor the distribution of your report so that recipients can easily navigate to the section they need.
- ❑ Use standard heading and footing commands to label the index page and/or each of the resulting report files.

### **Syntax:** How to Burst Reports Into Multiple HTML Files

```
ON sortfield PCSEND LOCATION dir [AS burstname] FORMAT HTML
```

where:

*sortfield*

Specifies the sort field based upon which bursting will occur. Each burst file will contain report output for only one sort group.

You can burst reports based on the value of the first sort field only. You cannot burst reports based on the value of subsequent sort fields.

You can burst a report into a maximum of 10,000 separate files. Therefore, you can burst a report only if the number of individual values of the first sort field does not exceed 10,000.

### PCSEND

Initiates bursting. You can only use one PCSEND command in a request.

### *dir*

Specifies the location on the web server where the HTML index file and report files are stored. The LOCATION parameter is required and must specify a directory from which the web server reads HTML files. There is no default.

On UNIX, Windows, and OpenVMS platforms, the directory value must specify a fully qualified directory path.

**Note for z/OS Web390 users:** On z/OS platforms, the directory value must specify the ddname of an allocated PDS. No dynamic allocation of datasets is provided. The PDS is allocated to a ddname other than the Web390 standard WWWHTM. The alternate ddname must be WWWxxx, where xxx are any three alphanumeric characters. The Web390 web server requires an entry in its mime table to recognize the allocated PDS as having HTML output.

If the LOCATION parameter specifies an invalid directory or specifies a directory that cannot be written to, an error message is returned.

### *burstname*

Specifies the name of the HTML index file, which contains a list of hyperlinks to the bursted HTML report files. The hyperlinks to the bursted HTML report files will be numbered from 0 to 9999. If no file name is specified for the HTML index file, the default name is HOLD.

Each bursted HTML report file will use the first four characters of the index file name, followed by numerics 0000 through 9999.

**Note:** The report request can contain display fields with missing values. The report request can also contain NOPRINT fields. For details, see [Handling Records With Missing Field Values](#) on page 971.

### **Reference:** Rules for Headings and Footings on Index Pages and Bursted Reports

- ☐ To include a heading or footing on the HTML index page, use the commands:

ON TABLE SUBHEAD

and

`ON TABLE SUBFOOT`

These headings and footings can contain embedded fields. The heading or footing is not included in the HTML report output pages. Default styling is applied to the HTML index page.

- ❑ To include a standard heading or footing on all HTML report output pages, use the commands:

`HEADING`

and

`FOOTING`

These headings and footings can contain embedded fields. Default styling is applied to the HTML report output pages.

- ❑ To include headings and footings that change each time the sortfield changes, use the commands:

`ON sortfield SUBHEAD`

and

`ON sortfield SUBFOOT`

These subheadings or subfootings can include embedded fields, such as the sortfield used in the PCSEND command. Default styling is applied to the HTML report output pages.

For details on including heading and footings in reports, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.

**Example: Bursting a Report**

The following report procedure creates an HTML report output file for each different REGION value in the GGSALES data source. The report output files are named test0000.html, test0001.html, test0002.html, and so forth. The HTML index page is named test.html and contains a hyperlink for each REGION data value. The directory you select depends on where WebFOCUS is installed. In this example, the index page is stored in the directory e:\ibi\WebFOCUS82\temp.

```
TABLE FILE GGSALES
HEADING
"Regional Report"
SUM UNITS AND DOLLARS
BY REGION BY STCD BY CATEGORY
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Year-end Sales:"
"Regional Summary by Store"
ON REGION PCSEND LOCATION E:\IBI\WebFOCUS82\temp AS TEST FORMAT HTML
END
```

After running this request, no report output is returned, but the following message displays if the request was successful:

The bursted files were successfully created.

Separate HTML files are created for each value of the major sort field REGION and are stored in the location specified in the request.

The HTML index page created by the procedure follows:

Year-end Sales:  
Regional Summary by Store

**REGION**

1. [Midwest](#)
2. [Northeast](#)
3. [Southeast](#)
4. [West](#)

Selecting the Midwest hyperlink displays the following HTML report:

Regional Report

<u>Region</u>	<u>Store ID</u>	<u>Category</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Midwest	R1019	Coffee	113253	1393610
		Food	107615	1351523
		Gifts	78869	969845
	R1020	Coffee	109581	1398779
		Food	118068	1522847
		Gifts	79932	1002775
	R1250	Coffee	109943	1386124
		Food	115731	1463901
		Gifts	72053	911261

Notice that the headings specified in the ON TABLE SUBHEAD command are displayed on the HTML index page. See [Rules for Headings and Footings on Index Pages and Bursted Reports](#) on page 966.



## Handling Records With Missing Field Values

---

Missing data is defined as data that is missing from a report because it is not relevant or does not exist in the data source. Report output that involves averaging and counting calculations or the display of parent segment instances may be affected by missing data. Data can be missing from reports and calculations for the following reasons:

- ❑ Data is not relevant to a particular row and column in a report. See [Irrelevant Report Data](#) on page 971.
- ❑ A field in a segment instance does not have a data value. See [Missing Field Values](#) on page 972.
- ❑ A parent segment instance does not have child instances (missing segment instances). See [Handling a Missing Segment Instance](#) on page 992.

**Note:** To run the examples in this topic, you must run the stored procedures EMPMISS and SALEMISS to add missing data to the EMPLOYEE and SALES data sources, respectively.

### In this chapter:

- ❑ [Irrelevant Report Data](#)
  - ❑ [Missing Field Values](#)
  - ❑ [Handling a Missing Segment Instance](#)
  - ❑ [Setting the NODATA Character String](#)
- 

### Irrelevant Report Data

Data can be missing from a report row or column because it is not relevant. The missing or inapplicable value is indicated by the NODATA default character, a period (.).

**Tip:** You may specify a more meaningful NODATA value by issuing the SET NODATA command (see [Setting the NODATA Character String](#) on page 1002).

**Example: Irrelevant Report Data**

The following request shows how the default NODATA character displays missing data in a report.

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL
BY LAST_NAME
BY FIRST_NAME
ACROSS DEPARTMENT
END
```

The output is:

LAST_NAME	FIRST_NAME	DEPARTMENT	
		MIS	PRODUCTION
BANNING	JOHN	.	\$29,700.00
BLACKWOOD	ROSEMARIE	\$21,780.00	.
CROSS	BARBARA	\$27,062.00	.
GREENSPAN	MARY	\$9,000.00	.
IRVING	JOAN	.	\$26,862.00
JONES	DIANE	\$18,480.00	.
MCCOY	JOHN	\$18,480.00	.
MCKNIGHT	ROGER	.	\$16,100.00
ROMANS	ANTHONY	.	\$21,120.00
SMITH	MARY	\$13,200.00	.
		.	\$9,050.00
STEVENS	ALFRED	.	\$11,000.00

The salary for an employee working in the production department displays in the PRODUCTION column. The salary for an employee working in the MIS department displays in the MIS column. The corresponding value in the PRODUCTION or MIS column, respectively, is missing because the salary displays only under the department where the person is employed.

**Missing Field Values**

Missing values within segment instances occur when the instances exist, but some of the fields lack values.

When fields in instances lack values, numeric fields are assigned the value 0, and alphanumeric fields, the value blank. These default values appear in reports and are used in all calculations performed by the SUM and COUNT display commands, DEFINE commands, and prefix operators such as MAX. and AVE.



To prevent the use of these default values in calculations (which might then give erroneous results), you can add the MISSING attribute to the field declaration in the Master File, for either a real or a virtual field. When the MISSING attribute is set to ON, the missing values are marked with a special internal code to distinguish them from blanks or zeros, and the missing values are ignored in calculations. In reports, the internal code is represented by the SET NODATA value, a period (.), by default. See [Setting the NODATA Character String](#) on page 1002.

For example, missing data for a field in a segment instance may occur when the data values are unknown, as in the following scenario. Suppose that the employees recorded in the EMPLOYEE data source are due for a pay raise by a certain date, but the amount of the raise has not yet been determined. The company enters the date for each employee into the data source without the salary amounts; the salaries will be entered later. Each date is an individual instance in the salary history segment, but the new salary for each date instance is missing. Suppose further that a report request averages the SALARY field (SUM AVE.SALARY). The accuracy of the resulting average depends on whether the missing values for the SALARY field are treated as zeros (MISSING=OFF), or as internal codes (MISSING=ON).

**Note:** When all of the field values used in the calculation of a numeric summary value, such as a subtotal, are missing, the summary value is assigned the missing data value, *not* the value zero (0). This includes summary values produced by the operators ST. and CT. used in a subfooting.

#### **Example:** Counting With Missing Values

Suppose the CURR\_SAL field appears in 12 segment instances. In three of those instances, the field was given no value. Nevertheless, the display command

```
COUNT CURR_SAL
```

counts 12 occurrences of the CURR\_SAL field. This occurs because the MISSING attribute is OFF by default, so the missing values are included in the count. If you wanted to exclude the missing data from the count, you could set MISSING ON.

#### **Example:** Averaging With Missing Values

Suppose you have the following records of data for a field:

```
.
.
1
3
```

The numeric values in the first two records are missing (indicated by the periods). The last two records have values of 1 and 3. If you average these fields without the MISSING attribute (MISSING OFF), the value 0 is supplied for the two records that are missing values. Thus, the average of the records is  $(0+0+1+3)/4$ , or 1. If you use the MISSING ON attribute, the two missing values are ignored, calculating the average as  $(1+3)/2$ , or 2.

### MISSING Attribute in the Master File

In some applications, the default values (blanks and zeros) may represent valid data rather than the absence of information. However, if this is *not* the case, you can include the MISSING attribute after the field format in the Master File declaration for the field with the missing values. The MISSING attribute can be used with an actual field in the data source, or a virtual field that you are defining in the Master File.

For example, the following field declaration specifies the MISSING attribute for the RETURNS field:

```
FIELDNAME=RETURNS, ALIAS=RTN, FORMAT=I4, MISSING=ON,$
```

The next declaration specifies the MISSING attribute for a virtual field called PROFIT:

```
DEFINE PROFIT/D7 MISSING ON NEEDS SOME DATA = RETAIL_COST - DEALER_COST;$
```

To ensure that missing values are handled properly for virtual fields, you can set the MISSING attribute ON for the virtual field in the DEFINE command, and specify whether you want to apply the calculation if some or all values are missing. For related information on the SOME and ALL phrases, see [How to Specify Missing Values in a DEFINE or COMPUTE Command](#) on page 976.

When the MISSING attribute is set to ON in a field declaration, the field containing no data is marked with a special internal code, rather than with blanks or zeros. During report generation, the SUM and COUNT commands and all prefix operators (for example, AVE., MAX., MIN.) exclude the missing data in their computations. For related information about the MISSING attribute and field declarations, see the *Describing Data With WebFOCUS Language* manual.

#### Note:

- ☐ You may add MISSING field attributes to the Master File at any time. However, MISSING attributes only affect data entered into the data source after the attributes were added.
- ☐ Key fields are needed to identify a record. Therefore, key fields should not be identified as missing.

**Example: Handling Missing Values With the MISSING Attribute**

This example illustrates the difference between a field with MISSING ON and one without. In it a virtual field, X>Returns, without the MISSING attribute, is set to equal a real field, RETURNS, with the MISSING attribute declared in the Master File. When the field with the MISSING attribute (RETURNS) is missing a value, the corresponding value of X>Returns is 0, since a data source field that is missing a value is evaluated as 0 (or blank) for the purpose of computation (see *MISSING Attribute in a DEFINE or COMPUTE Command* on page 975).

The following request defines the virtual field:

```
DEFINE FILE SALES
X>Returns/I4 = RETURNS;
END
```

Now issue the following report request:

```
TABLE FILE SALES
SUM CNT.X>Returns CNT.RETURNS AVE.X>Returns AVE.RETURNS
END
```

Remember that the field X>Returns has the same value as RETURNS except when RETURNS is missing a value, in which case, the X>Returns value is 0.

The output is:

X>Returns	RETURNS	AVE	AVE
<u>COUNT</u>	<u>COUNT</u>	<u>X&gt;Returns</u>	<u>RETURNS</u>
22	20	2	3

The count for the RETURNS field is lower than the count for X>Returns and the average for RETURNS is higher than for X>Returns because the missing values in RETURNS are not part of the calculations.

**MISSING Attribute in a DEFINE or COMPUTE Command**

You can set the MISSING attribute ON in a DEFINE or COMPUTE command to enable a temporary field with missing values to be interpreted and represented correctly in reports.

An expression used to derive the values of the temporary field can contain real fields that have missing values. However, when used to derive the value of a temporary field, a data source field that is missing a value is evaluated as 0 or blank for computational purposes, even if the MISSING attribute has been set to ON for that field in the Master File.

To ensure that missing values are handled properly for temporary fields, you can set the MISSING attribute ON for the virtual field in the DEFINE or COMPUTE command, and specify whether you want to apply the calculation if some or all values are missing. See [How to Specify Missing Values in a DEFINE or COMPUTE Command](#) on page 976.

### **Syntax:** How to Specify Missing Values in a DEFINE or COMPUTE Command

```
field[/format] MISSING {ON|OFF} [NEEDS] {SOME|ALL} [DATA] = expression;
```

where:

*field*

Is the name of the virtual field created by the DEFINE command.

*/format*

Is the format of the virtual field. The default is D12.2.

#### MISSING

[ON](#) enables the value of the temporary field to be interpreted as missing (that is, distinguished by the special internal code from an intentionally entered zero or blank), and represented by the NODATA character in reports.

[OFF](#) treats missing values for numeric fields as zeros, and missing values for alphanumeric fields as blanks. This is the default value.

#### NEEDS

Is optional. It helps to clarify the meaning of the command.

#### SOME

Indicates that if at least one field in the expression has a value, the temporary field has a value (the missing values of the field are evaluated as 0 or blank in the calculation). If all of the fields in the expression are missing values, the temporary field is missing its value. SOME is the default value.

#### ALL

Indicates that if all the fields in the expression have values, the temporary field has a value. If at least one field in the expression has a missing value, the temporary field also has a missing value.

#### DATA

Is optional. It helps to clarify the meaning of the command.

*expression*

Is a valid expression from which the temporary field derives its value.

**Note:** You can also use the SET MISS\_ON command to set a default value for MISSING ON in DEFINE and COMPUTE.

**Example:** Handling Missing Values for a Virtual Field With MISSING OFF

The following request illustrates the use of two fields, RETURNS and DAMAGED, to define the NO\_SALE field. Both the RETURNS and DAMAGED fields have the MISSING attribute set to ON in the SALES Master File, yet whenever one of these fields is missing a value, that field is evaluated as 0.

```
DEFINE FILE SALES
NO_SALE/I4 = RETURNS + DAMAGED;
END
TABLE FILE SALES
PRINT RETURNS AND DAMAGED AND NO_SALE
BY CITY BY DATE BY PROD_CODE
END
```

The output is:

CITY	DATE	PROD_CODE	RETURNS	DAMAGED	NO_SALE
----	----	-----	-----	-----	-----
NEW YORK	10/17	B10	2	3	5
		B17	2	1	3
		B20	0	1	1
		C13	.	6	6
		C14	4	.	4
		C17	0	0	0
		D12	3	2	5
		E1	4	7	11
		E2	.	.	0
		E3	4	2	6
		B10	1	1	2
		B12	1	0	1
		B10	10	6	16
STAMFORD	12/12	B12	3	3	6
		B17	2	1	3
		C13	3	0	3
		C7	5	4	9
		D12	0	0	0
		E2	9	4	13
		E3	8	9	17
		B20	1	1	2
		C7	0	0	0
UNIONDALE	10/18	B20	1	1	2
		C7	0	0	0

Notice that the products C13, C14, and E2 in the New York section all show missing values for either RETURNS or DAMAGED, because the MISSING ON attribute has been set in the Master File. However, the calculation that determines the value of NO\_SALE interprets these missing values as zeros, because MISSING ON has not been set for the virtual field.

***Example:* Handling Missing Values for Virtual Fields With SOME and ALL**

The following request illustrates how to use the DEFINE command with the MISSING attribute to specify that if either some or all of the field values referenced in a DEFINE command are missing, the virtual field should also be missing its value.

The SOMEDATA field contains a value if either the RETURNS or DAMAGED field contains a value. Otherwise, SOMEDATA is missing its value. The ALLDATA field contains a value only if both the RETURNS and DAMAGED fields contain values. Otherwise, ALLDATA is missing its value.

```
DEFINE FILE SALES
SOMEDATA/I5 MISSING ON NEEDS SOME=RETURNS + DAMAGED;
ALLDATA/I5 MISSING ON NEEDS ALL=RETURNS + DAMAGED;
END
```

```
TABLE FILE SALES
PRINT RETURNS AND DAMAGED SOMEDATA ALLDATA
BY CITY BY DATE BY PROD_CODE
END
```

The output is:

CITY	DATE	PROD_CODE	RETURNS	DAMAGED	SOMEDATA	ALLDATA
NEW YORK	10/17	B10	2	3	5	5
		B17	2	1	3	3
		B20	0	1	1	1
		C13	.	6	6	.
		C14	4	.	4	.
		C17	0	0	0	0
		D12	3	2	5	5
		E1	4	7	11	11
		E2	.	.	.	.
		E3	4	2	6	6
NEWARK	10/18	B10	1	1	2	2
	10/19	B12	1	0	1	1
STAMFORD	12/12	B10	10	6	16	16
		B12	3	3	6	6
		B17	2	1	3	3
		C13	3	0	3	3
		C7	5	4	9	9
		D12	0	0	0	0
		E2	9	4	13	13
		E3	8	9	17	17
		E3	8	9	17	17
UNIONDALE	10/18	B20	1	1	2	2
		C7	0	0	0	0

### ***Syntax:***

### **How to Setting MISSING ON Behavior for DEFINE and COMPUTE**

When a virtual field or calculated value can have missing values, you can specify whether all or some of the field values used in the expression that creates the DEFINE or COMPUTE field must be missing to make the result field missing. If you do not specify ALL or SOME for a DEFINE or COMPUTE with MISSING ON, the default value is SOME.

The SET parameter MISS\_ON enables you to specify whether SOME or ALL should be used for MISSING ON in a DEFINE or COMPUTE that does not specify which to use.

```
SET MISS_ON = {SOME|ALL}
```

where:

#### [SOME](#)

Indicates that if at least one field in the expression has a value, the temporary field has a value (the missing values of the field are evaluated as 0 or blank in the calculation). If all of the fields in the expression are missing values, the temporary field has a missing value. SOME is the default value.

ALL

Indicates that if all the fields in the expression have values, the temporary field has a value. If at least one field in the expression has a missing value, the temporary field has a missing value.

*Example:*     **Setting a Default Value for MISSING ON in DEFINE and COMPUTE**

The following request creates three virtual fields that all have MISSING ON. Field AAA has all missing values. Field BBB is missing only when the category is Gifts and has the value 100 otherwise. Field CCC is the sum of AAA and BBB.

```
SET MISS_ON = SOME
DEFINE FILE GGSales
AAA/D20 MISSING ON = MISSING;
BBB/D20 MISSING ON = IF CATEGORY EQ 'Gifts' THEN MISSING ELSE 100;
CCC/D20 MISSING ON = AAA + BBB;
END
TABLE FILE GGSales
SUM
AAA
BBB
CCC
BY CATEGORY
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
END
```

Running the request with SET MISS\_ON=SOME (the default) shows that CCC has a value unless both AAA and BBB are missing.

<u>Category</u>	<u>AAA</u>	<u>BBB</u>	<u>CCC</u>
Coffee	.	144,000	144,000
Food	.	144,000	144,000
Gifts	.	.	.



Changing SET MISS\_ON to ALL, produces the following output. CCC is assigned a missing value because one of the fields used to calculate it is always missing.

<u>Category</u>	<u>AAA</u>	<u>BBB</u>	<u>CCC</u>
Coffee	.	144,000	.
Food	.	144,000	.
Gifts	.	.	.

### Testing for Missing Values in IF-THEN-ELSE Expressions

In order to optimize the efficiency of IF-THEN-ELSE expression processing, the entire expression may not be parsed when the result (true or false) can be determined without parsing the entire expression. For example, consider the following IF-THEN-ELSE expression.

```
COMPUTE CCC/D9.1 MISSING ON = IF AAA EQ 0 OR BBB EQ 0 THEN 0 ELSE AAA / BBB
* 100;
```

#### Note:

- ❑ AAA is a missable field (MISSING ON) that is set to MISSING.
- ❑ BBB is a field with MISSING ON that is set to 100, so it has a value that is not MISSING.

For evaluation in expressions, a field that is MISSING is treated as zero (0).

CCC is a numeric field that has MISSING ON computed using an IF-THEN-ELSE expression. The IF clause contains two tests connected by the OR operator. If either one of these two tests is TRUE, the value in the THEN clause will be passed to the final MISSING check. If neither is TRUE, the value in the ELSE clause will be passed to the MISSING check.

In this case, AAA EQ 0 is true (since MISSING is treated as 0 when used in a comparison). So the THEN value, which is zero, is accepted. Since CCC has MISSING ON, which defaults to NEEDS SOME values, at least one data value evaluated in the IF clause must *not* be missing.

Only AAA was evaluated to get the TRUE condition. BBB was not tested and, therefore, is not included in the fields that are checked for missing values. Since AAA is MISSING, CCC (since it NEEDS SOME and does not have any) must be set to MISSING. Note that this processing represents a change from the processing in prior releases and, while it increases processing efficiency, it may produce different results.

Since the test for a value and the test for existence or presence (MISSING) are not the same, the best practice is always to use the test that is appropriate. When testing for MISSING, using IS MISSING is preferred to using EQ 0, as it is more direct and does not result in the same behavior change from previous releases.

Consider the following request. CCC uses EQ 0 in the IF-THEN-ELSE test, and DDD uses IS MISSING.

```
DEFINE FILE GGSALES
AAA/D20 MISSING ON = MISSING;
BBB/D20 MISSING ON = 100;
END
TABLE FILE GGSALES
SUM
  AAA
  BBB
  COMPUTE CCC/D9.1 MISSING ON = IF AAA EQ 0 OR BBB EQ 0 THEN 0 ELSE AAA /
BBB * 100;
  COMPUTE DDD/D9.1 MISSING ON = IF AAA IS MISSING OR BBB IS MISSING THEN 0
ELSE AAA / BBB * 100;
BY CATEGORY
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
END
```

The output is shown in the following image. While CCC evaluates to a missing value, DDD evaluates to 0 because the both AAA and BBB were evaluated to get the TRUE condition and BBB is not missing, satisfying the NEEDS SOME VALUES definition for DDD.

<u>Category</u>	<u>AAA</u>	<u>BBB</u>	<u>CCC</u>	<u>DDD</u>
Coffee	.	144,000	.	.0
Food	.	144,000	.	.0
Gifts	.	143,700	.	.0

**Reference:** Using SET MISSINGTEST With IF-THEN-ELSE Expressions

In prior releases, by default, when an IF-THEN-ELSE expression was used to calculate a result and the IF expression evaluated to zero (for numeric expressions) or blank (for alphanumeric expressions), the left hand side was checked to see if it had MISSING ON. If it did, and only some values were needed (NEEDS SOME), the result of the IF expression was MISSING, not true or false. The outcome returned was also MISSING, not the result of evaluating the THEN or ELSE expression. The SET MISSINGTEST = NEW command eliminates the missing test for the IF expression so that either the THEN expression or the ELSE expression will be evaluated and returned as the result. This is the default behavior.

The syntax is:

```
SET MISSINGTEST = {NEW|OLD|SPECIAL}
```

where:

#### NEW

Excludes the IF expression from the missing values evaluation so that the IF expression results in either true or false, not MISSING. If it evaluates to true, the THEN expression is used to calculate the result. If it evaluates to false, the ELSE expression is used to calculate the result. This is the default value.

#### OLD

Includes the IF expression in the missing values evaluation. If the IF expression evaluates to MISSING and the missing field only needs some missing values, the result is also MISSING.

#### SPECIAL

Is required for passing parameters to RStat.

### **Example:** Using SET MISSINGTEST With IF-THEN-ELSE Expressions

The following request defines a field named MISS\_FIELD that contains a missing value for the country name Austria. In the TABLE request there are two calculated values, CALC1 and CALC2 that test this field in IF-THEN-ELSE expressions. Both of these fields have MISSING ON and need only some missing values to be missing:

```
SET MISSINGTEST = OLD
DEFINE FILE wf_retail_lite
MISS_FIELD/A10 MISSING ON = IF COUNTRY_NAME NE 'Austria' THEN 'DATAEXISTS'
ELSE MISSING;
END

TABLE FILE wf_retail_lite
SUM COGS_US MISS_FIELD
COMPUTE CALC1/A7 MISSING ON = IF ((MISS_FIELD EQ '') OR (MISS_FIELD EQ
MISSING)) THEN 'THEN' ELSE 'ELSE';
COMPUTE CALC2/A7 MISSING ON = IF ((MISS_FIELD EQ MISSING) OR (MISS_FIELD EQ
'')) THEN 'THEN' ELSE 'ELSE';
BY COUNTRY_NAME
WHERE BUSINESS_REGION EQ 'EMEA'
WHERE COUNTRY_NAME LT 'E'
ON TABLE SET NODATA 'MISSING'
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

Running the request with MISSINGTEST=OLD produces the output shown in the following image:

Customer				
<u>Country</u>	<u>Cost of Goods</u>	<u>MISS FIELD</u>	<u>CALC1</u>	<u>CALC2</u>
Austria	\$426,570.00	MISSING	MISSING	THEN
Belgium	\$3,800.00	DATAEXISTS	ELSE	ELSE
China	\$563.00	DATAEXISTS	ELSE	ELSE
Denmark	\$1,346.00	DATAEXISTS	ELSE	ELSE

Note that for Austria, MISS\_FIELD is MISSING.

- ❑ In CALC1, the expression MISS\_FIELD EQ ' ' is evaluated to true. MISS\_FIELD IS MISSING is not evaluated at all because evaluation stops when it can be determined that the result of the expression is true. However, since the expression compared the field to blank, it is checked to see if the result field supports missing values. Since it does, the final result is MISSING.
- ❑ In CALC2, the expression MISS\_FIELD EQ MISSING is true. MISS\_FIELD EQ ' ' is not evaluated at all because evaluation stops when it can be determined that the result of the expression is true. No missing check is needed, so the result of the IF expression is TRUE, and the THEN expression is evaluated and returned as the result.

Changing the SET command to SET MISSINGTEST=NEW and rerunning the request produces the output shown in the following image. The IF expressions in CALC1 and CALC2 both evaluate to true because neither expression is checked to see if the result field supports missing, so the THEN expression is evaluated and returned as the result in both cases.

Customer				
<u>Country</u>	<u>Cost of Goods</u>	<u>MISS FIELD</u>	<u>CALC1</u>	<u>CALC2</u>
Austria	\$426,570.00	MISSING	THEN	THEN
Belgium	\$3,800.00	DATAEXISTS	ELSE	ELSE
China	\$563.00	DATAEXISTS	ELSE	ELSE
Denmark	\$1,346.00	DATAEXISTS	ELSE	ELSE

## Testing for a Segment With a Missing Field Value

You can specify WHERE criteria to identify segment instances with missing field values.

You cannot use these tests to identify missing instances. See [Handling a Missing Segment Instance](#) on page 992.

**Syntax:**      **How to Test for a Segment With a Missing Field Value**

To test for a segment with missing field values, the syntax is:

```
WHERE field {IS|EQ} MISSING
```

To test for the presence of field values, the syntax is:

```
WHERE field {NE|IS-NOT} MISSING
```

A WHERE criterion that tests a numeric field for 0 or an alphanumeric field for blanks also retrieves instances for which the field has a missing value.

**Example:**      **Testing for a Missing Field Value**

The following request illustrates the use of MISSING to display grocery items (by code) for which the number of packages returned by customers is missing.

```
TABLE FILE SALES
PRINT RETURNS
BY CITY BY DATE BY PROD_CODE
WHERE RETURNS IS MISSING
END
```

The output is:

CITY	DATE	PROD_CODE	RETURNS
----	----	-----	-----
NEW YORK	10/17	C13	.
		E2	.

**Example:**    **Testing for an Existing Field Value**

The following request illustrates the use of MISSING to display only those grocery items for which the number of packages returned by customers is not missing.

```
TABLE FILE SALES
PRINT RETURNS
BY CITY BY DATE BY PROD_CODE
WHERE RETURNS IS-NOT MISSING
END
```

The output is:

CITY	DATE	PROD_CODE	RETURNS
NEW YORK	10/17	B10	2
		B17	2
		B20	0
		C14	4
		C17	0
		D12	3
		E1	4
		E3	4
NEWARK	10/18	B10	1
	10/19	B12	1
STAMFORD	12/12	B10	10
		B12	3
		B17	2
		C13	3
		C7	5
		D12	0
		E2	9
		E3	8
UNIONDALE	10/18	B20	1
		C7	0

**Example:**    **Testing for a Blank or Zero**

The following request displays grocery items that either were never returned or for which the number of returned packages was never recorded:

```
TABLE FILE SALES
PRINT RETURNS
BY CITY BY DATE BY PROD_CODE
WHERE RETURNS EQ 0
END
```

The output is:

CITY	DATE	PROD_CODE	RETURNS
NEW YORK	10/17	B20	0
		C13	.
		C17	0
		E2	.
STAMFORD	12/12	D12	0
UNIONDALE	10/18	C7	0

**Example:** Excluding Missing Values From a Test

To display only those items that have not been returned by customers, you need two WHERE criteria. The first to restrict the number of returns to 0, the other to exclude missing values, as in the following request.

```
TABLE FILE SALES
PRINT RETURNS
BY CITY BY DATE BY PROD_CODE
WHERE RETURNS EQ 0
WHERE RETURNS IS-NOT MISSING
END
```

The output is:

CITY	DATE	PROD_CODE	RETURNS
NEW YORK	10/17	B20	0
		C17	0
		E2	0
STAMFORD	12/12	D12	0
UNIONDALE	10/18	C7	0

## Preserving Missing Data Values in an Output File

The ability to distinguish between missing data and default values (blanks and zeros) in fields can be carried over into output files. If the retrieved and processed information displayed the NODATA string in a report, by default the NODATA string can be stored in the output file. Using the SET HNODATA command, you can change the NODATA value used for alphanumeric output files. You can also use the SET HOLDMISS command to store the missing values rather than the NODATA character in an output file. For related information, see [Saving and Reusing Your Report Output](#) on page 471.

**Syntax:**      **How to Distinguish Missing Data in an Extract File**

```
ON TABLE {HOLD|SAVE|SAVB} MISSING {ON|OFF}
```

where:

**HOLD**

Creates an extract file for use in subsequent reports. The default for MISSING is ON.

**SAVE**

Creates a text extract file for use in other programs. The default for MISSING is OFF.

**SAVB**

Creates a binary extract file for use in other programs. The default for MISSING is OFF.

HOLD files can be created with both the MISSING and FORMAT ALPHA options, specified in any order. For example:

```
ON TABLE HOLD FORMAT ALPHA MISSING OFF
ON TABLE HOLD MISSING OFF FORMAT ALPHA
```

**Example:**      **Incorporating MISSING Values in an Extract File**

The following request specifies MISSING ON in the HOLD phrase:

```
TABLE FILE SALES
SUM RETURNS AND HOLD FORMAT ALPHA MISSING ON
BY CITY BY DATE BY PROD_CODE
END
```

The MISSING=ON attribute for the RETURNS field is propagated to the HOLD Master File. In addition, the missing data symbols are propagated to the HOLD file for the missing field values:

```
FILENAME=HOLD      , SUFFIX=FIX      , $
SEGMENT=HOLD, SEGTYPE=S3, $
  FIELDNAME=CITY, ALIAS=E01, USAGE=A15, ACTUAL=A15, $
  FIELDNAME=DATE, ALIAS=E02, USAGE=A4MD, ACTUAL=A04, $
  FIELDNAME=PROD_CODE, ALIAS=E03, USAGE=A3, ACTUAL=A03, $
  FIELDNAME=RETURNS, ALIAS=E04, USAGE=I3, ACTUAL=A03, $
  MISSING=ON, $
```

With MISSING OFF in the HOLD phrase, the MISSING=ON attribute is not propagated to the HOLD Master File and the missing data symbols are replaced with default values.



**Syntax:**      **How to Store Missing Data in HOLD Files**

```
SET HOLDMISS={ON|OFF}
ON TABLE SET HOLDMISS {ON|OFF}
```

where:

ON

Allows you to store missing data in a HOLD file. When TABLE generates a default value for data not found, it generates missing values.

OFF

Does not allow you to store missing data in a HOLD file. OFF is the default value.

**Reference:**      **Usage Notes for Holding Missing Values**

- ☐ Setting HOLDMISS ON adds the MISSING=ON attribute to every field in the extract file.
- ☐ Data is not found if:
  - ☐ ALL is set to ON.
  - ☐ The request is multi-path.
  - ☐ An ACROSS statement has been issued.

**Example:**      **Holding Missing Values Using HOLDMISS**

```
SET HOLDMISS=ON
TABLE FILE MOVIES
  SUM WHOLESALEPR
  BY CATEGORY ACROSS RATING
  ON TABLE HOLD AS HLDM
END
TABLE FILE HLDM
  PRINT *
END
```

The output is:

CATEGORY	WHOLESALEPR	WHOLESALEPR	WHOLESALEPR	WHOLESALEPR	WHOLESALEPR
ACTION	.	.	20.98	.	34.48
CHILDREN	54.49	51.38	.	.	.
CLASSIC	40.99	160.80	.	.	.
COMEDY	.	.	46.70	30.00	13.75
DRAMA	.	.	.	.	10.00
FOREIGN	13.25	.	62.00	.	70.99
MUSICALS	15.00	.	13.99	9.99	13.99
MYSTERY	.	9.00	18.00	9.00	80.97
SCI/FI	.	.	.	35.99	43.53
TRAIN/EX	.	.	.	.	.
60.98	.	.	.	.	.

Propagating Missing Values to Reformatted Fields in a Request

When a field is reformatted in a request (for example, SUM field/format), an internal COMPUTE field is created to contain the reformatted field value and display on the report output. If the original field has a missing value, that missing value can be propagated to the internal field by setting the COMPMISS parameter ON. If the missing value is not propagated to the internal field, it displays a zero (if it is numeric) or a blank (if it is alphanumeric). If the missing value is propagated to the internal field, it displays the missing data symbol on the report output.

Syntax: How to Control Missing Values in Reformatted Fields

SET COMPMISS = {ON|OFF}

where:

ON

Propagates a missing value to a reformatted field. ON is the default value.

OFF

Displays a blank or zero for a reformatted field.

**Note:** The COMPMISS parameter cannot be set in an ON TABLE command.

Example: Controlling Missing Values in Reformatted Fields

The following procedure prints the RETURNS field from the SALES data source for store 14Z. With COMPMISS OFF, the missing values display as zeros in the column for the reformatted field value.

**Note:** Before trying this example, you must make sure that the SALEMISS procedure, which adds missing values to the SALES data source, has been run.

```

SET COMPMISS = OFF
TABLE FILE SALES
PRINT RETURNS RETURNS/D12.2 AS 'REFORMATTED,RETURNS'
BY STORE_CODE
WHERE STORE_CODE EQ '14Z'
END

```

The output is:

STORE_CODE	RETURNS	REFORMATTED RETURNS
-----	-----	-----
14Z	2	2.00
	2	2.00
	0	.00
	.	.00
	4	4.00
	0	.00
	3	3.00
	4	4.00
	.	.00
	4	4.00

With COMPMISS ON, the column for the reformatted version of RETURNS displays the missing data symbol when a value is missing:

```

SET COMPMISS = ON
TABLE FILE SALES
PRINT RETURNS RETURNS/D12.2 AS 'REFORMATTED,RETURNS'
BY STORE_CODE
WHERE STORE_CODE EQ '14Z'
END

```

The output is:

STORE_CODE	RETURNS	REFORMATTED RETURNS
-----	-----	-----
14Z	2	2.00
	2	2.00
	0	.00
	.	.
	4	4.00
	0	.00
	3	3.00
	4	4.00
	.	.
	4	4.00

### **Reference: Usage Notes for SET COMPMISS**

If you create a HOLD file with COMPMISS ON, the HOLD Master File for the reformatted field indicates MISSING = ON (as does the original field). With COMPMISS = OFF, the reformatted field does NOT have MISSING = ON in the generated Master File.

## **Handling a Missing Segment Instance**

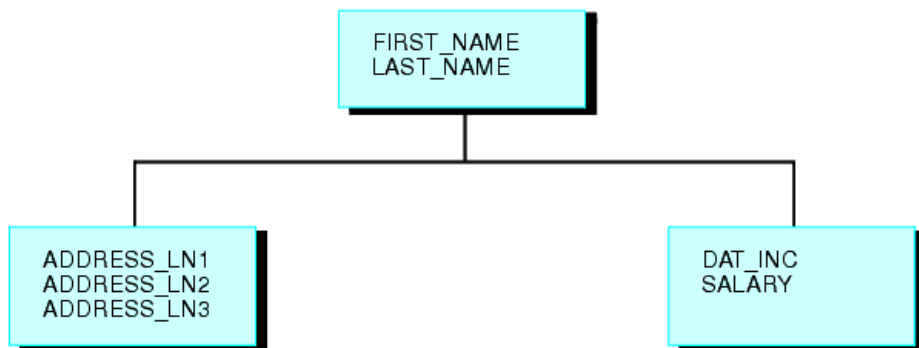
In multi-segment data sources, when an instance in a parent segment does not have descendant instances, the nonexistent descendant instances are called missing instances.

When you write a request from a data source that has missing segment instances, the missing instances affect the report. For example, if the request names fields in a segment and its descendants, the report omits parent segment instances that have no descendants. It makes no difference whether fields are display fields or sort fields.

When an instance is missing descendants in a child segment, the instance, its parent, the parent of its parent, and so on up to the root segment, is called a short path. Unique segments are never considered to be missing.

For example, consider the following subset of the EMPLOYEE data source.

- ☐ The top segment contains employee names.
- ☐ The left segment contains addresses.
- ☐ The right segment contains the salary history of each employee: the date the employee was granted a new salary, and the amount of the salary.



Suppose some employees are paid by an outside agency. None of these employees have a company salary history. Instances referring to these employees in the salary history segment are missing.

Nonexistent descendant instances affect whether parent segment instances are included in report results. The SET ALL parameter and the ALL. prefix enable you to include parent segment data in reports.

For illustrations of how missing segment instances impact reporting, see [Reporting Against Segments Without Descendant Instances](#) on page 993 and [Reporting Against Segments With Descendant Instances](#) on page 994.

**Example: Reporting Against Segments Without Descendant Instances**

The following request displays the salary histories for each employee.

```
TABLE FILE EMPLOYEE
PRINT SALARY
BY LAST_NAME BY FIRST_NAME
BY DAT_INC
END
```

However, two employees, Davis and Gardner, are omitted from the following report because the LAST\_NAME and FIRST\_NAME fields belong to the root segment, and the DAT\_INC and SALARY fields belong to the descendant salary history segment. Since Davis and Gardner have no descendant instances in the salary history segment, they are omitted from the report.

The output is:

LAST_NAME	FIRST_NAME	DAT_INC	SALARY
-----	-----	-----	-----
BANNING	JOHN	82/08/01	\$29,700.00
BLACKWOOD	ROSEMARIE	82/04/01	\$21,780.00
CROSS	BARBARA	81/11/02	\$25,775.00
		82/04/09	\$27,062.00
GREENSPAN	MARY	82/04/01	\$8,650.00
		82/06/11	\$9,000.00
IRVING	JOAN	82/01/04	\$24,420.00
		82/05/14	\$26,862.00
JONES	DIANE	82/05/01	\$17,750.00
		82/06/01	\$18,480.00
MCCOY	JOHN	82/01/01	\$18,480.00
MCKNIGHT	ROGER	82/02/02	\$15,000.00
		82/05/14	\$16,100.00
ROMANS	ANTHONY	82/07/01	\$21,120.00
SMITH	MARY	82/01/01	\$13,200.00
	RICHARD	82/01/04	\$9,050.00
		82/05/14	\$9,500.00
STEVENS	ALFRED	81/01/01	\$10,000.00
		82/01/01	\$11,000.00

**Example:**    **Reporting Against Segments With Descendant Instances**

The following request displays the course codes and expenses for employees in the EMPDATA and TRAIN2 data sources. The report output displays all employees that have instances in the COURSECODE or EXPENSES fields. The employees that are missing instances for either of those fields are omitted from the report. For those employees that have instances for only one of the fields, the designator for missing data displays in the respective column. In this example, Henry Chisolm has taken two courses but only has expenses for one. Therefore, the designator for missing instances displays in the EXPENSES column.

```
JOIN EMPDATA.PIN IN EMPDATA TO ALL TRAINING.PIN IN TRAIN2 AS JOIN1
TABLE FILE EMPDATA
PRINT LASTNAME AND FIRSTNAME AND COURSECODE AND EXPENSES
BY PIN
END
```

The output is:

PIN	LASTNAME	FIRSTNAME	COURSECODE	EXPENSES
000000010	VALINO	DANIEL	PDR740	2,300.00
000000030	CASSANOVA	LOIS	NAMA730	2,600.00
	CASSANOVA	LOIS	EDP090	2,300.00
.				
.				
.				
000000350	FERNSTEIN	ERWIN	SSI220	1,850.00
	FERNSTEIN	ERWIN	MC90	1,730.00
	FERNSTEIN	ERWIN	UMI720	3,350.00
000000360	CHISOLM	HENRY	EDP090	.00
	CHISOLM	HENRY	EDP690	3,000.00
000000370	WANG	JOHN	UMI710	2,050.00
000000380	ELLNER	DAVID	EDP090	.
	ELLNER	DAVID	UNI780	3,350.00
000000410	CONTI	MARSHALL	EDP690	3,100.00

**Note:** The report output has been truncated for demonstration purposes.

## Including Missing Instances in Reports With the ALL. Prefix

If a request excludes parent segment instances that lack descendants, you can include the parent instances by attaching the ALL. prefix to one of the fields in the parent segment.

Note that if the request contains WHERE or IF criteria that screen fields in segments that have missing instances, the report omits parent instances even when you use the ALL. prefix. To include these instances, use the SET ALL=PASS command described in [Including Missing Segment Instances With the ALL. Prefix](#) on page 995.

### **Example:** Including Missing Segment Instances With the ALL. Prefix

The following request displays the salary history of each employee. Although employees Elizabeth Davis and David Gardner have no salary histories, they are included in the report.

```
TABLE FILE EMPLOYEE
PRINT SALARY
BY ALL.LAST_NAME BY FIRST_NAME
BY DAT_INC
END
```

The output is:

LAST_NAME	FIRST_NAME	DAT_INC	SALARY
-----	-----	-----	-----
BANNING	JOHN	82/08/01	\$29,700.00
BLACKWOOD	ROSEMARIE	82/04/01	\$21,780.00
CROSS	BARBARA	81/11/02	\$25,775.00
		82/04/09	\$27,062.00
DAVIS	ELIZABETH	.	.
GARDNER	DAVID	.	.
GREENSPAN	MARY	82/04/01	\$8,650.00
		82/06/11	\$9,000.00
IRVING	JOAN	82/01/04	\$24,420.00
		82/05/14	\$26,862.00
JONES	DIANE	82/05/01	\$17,750.00
		82/06/01	\$18,480.00
MCCOY	JOHN	82/01/01	\$18,480.00
MCKNIGHT	ROGER	82/02/02	\$15,000.00
		82/05/14	\$16,100.00
ROMANS	ANTHONY	82/07/01	\$21,120.00
SMITH	MARY	82/01/01	\$13,200.00
	RICHARD	82/01/04	\$9,050.00
		82/05/14	\$9,500.00
STEVENS	ALFRED	81/01/01	\$10,000.00
		82/01/01	\$11,000.00

## Including Missing Instances in Reports With the SET ALL Parameter

You can control how parent instances with missing descendants are processed by issuing the SET ALL command before executing the request. In a join, issuing the SET ALL = ON command controls left outer join processing.

**Note:** A request with WHERE or IF criteria, which screen fields in a segment that has missing instances, omits instances in the parent segment even if you use the SET ALL=ON command. To include these instances, use the SET ALL=PASS command.

In WebFOCUS, the command SET ALL = ON or JOIN LEFT\_OUTER specifies a left outer join. With a left outer join, all records from the host file display on the report output. If a cross-referenced segment instance does not exist for a host segment instance, the report output displays missing values for the fields from the cross-referenced segment.

If there is a screening condition on the dependent segment, those dependent segment instances that do not satisfy the screening condition are omitted from the report output, and so are their corresponding host segment instances. With missing segment instances, tests for missing values fail because the fields in the segment have not been assigned missing values.

When a relational engine performs a left outer join, it processes host records with missing cross-referenced segment instances slightly differently from the way WebFOCUS processes those records when both of the following conditions apply:

- ❑ There is a screening condition on the cross-referenced segment.
- ❑ A host segment instance does not have a corresponding cross-referenced segment instance. This is called a *short path*.

When these two conditions are true, WebFOCUS omits the host record from the report output, while relational engines supply null values for the fields from the dependent segment and then apply the screening condition. If the missing values pass the screening condition, the entire record is retained on the report output. This type of processing is useful for finding or counting all host records that do not have matching records in the cross-referenced file or for creating a DEFINE-based join from the cross-referenced segment with the missing instance to another dependent segment.

If you want WebFOCUS to assign null values to the fields in a missing segment instance when a left outer join is in effect, you can issue the command SET SHORTPATH=SQL.

### **Syntax:** How to Include a Parent Instance With Missing Descendants

```
SET ALL= {OFF|ON|PASS}
```

where:

[OFF](#)

Omits parent instances that are missing descendants from the report. OFF is the default value.



**ON**

Includes parent instances that are missing descendants in the report. However, if a test on a missing segment fails, this causes the parent to be omitted from the output. It is comparable to the ALL. prefix.

**PASS**

Includes parent instances that are missing descendants, even if WHERE or IF criteria exist to screen fields in the descendant segments that are missing instances (that is, a test on a missing segment passes).

***Example:* Including Missing Segment Instances With SET ALL**

The following request displays all employees, regardless of whether they have taken a course or not since the ALL=PASS command is set.

If the ALL=ON command had been used, employees that had not taken courses would have been omitted because of the WHERE criteria.

```
JOIN EMPDATA.PIN IN EMPDATA TO ALL TRAINING.PIN IN TRAINING AS JOIN1
SET ALL = PASS
TABLE FILE EMPDATA
PRINT LASTNAME AND FIRSTNAME AND COURSECODE AND EXPENSES
BY PIN
WHERE EXPENSES GT 3000
END
```

The output is:

PIN	LASTNAME	FIRSTNAME	COURSECODE	EXPENSES
000000020	BELLA	MICHAEL	.	.
000000040	ADAMS	RUTH	EDP750	3,400.00
000000050	ADDAMS	PETER	UMI720	3,300.00
000000060	PATEL	DORINA	.	.
000000070	SANCHEZ	EVELYN	.	.
000000080	SO	PAMELA	EDP690	3,200.00
	SO	PAMELA	BIT420	3,350.00
000000090	PULASKI	MARIANNE	.	.
000000100	ANDERSON	TIM	NAMA930	3,100.00
000000130	CVEK	MARCUS	.	.
000000140	WHITE	VERONICA	BIT420	3,600.00
000000150	WHITE	KARL	UNI780	3,400.00
000000170	MORAN	WILLIAM	.	.
000000190	MEDINA	MARK	EDP690	3,150.00
000000220	LEWIS	CASSANDRA	.	.
000000230	NOZAWA	JIM	.	.
000000300	SOPENA	BEN	.	.
000000340	GOTLIEB	CHRIS	SSI670	3,300.00
	GOTLIEB	CHRIS	EDP750	3,450.00
000000350	FERNSTEIN	ERWIN	UMI720	3,350.00
000000380	ELLNER	DAVID	UNI780	3,350.00
000000390	GRAFF	ELAINE	.	.
000000400	LOPEZ	ANNE	.	.
000000410	CONTI	MARSHALL	EDP690	3,100.00

### **Syntax:** How to Control Short Path Processing In a Left Outer Join

```
SET SHORTPATH = {FOCUS | SQL}
```

where:

#### [FOCUS](#)

Omits a host segment from the report output when it has no corresponding cross-referenced segment and the report has a screening condition on the cross-referenced segment.

#### [SQL](#)

Supplies missing values for the fields in a missing cross-referenced segment in an outer join. Applies screening conditions against this record and retains the record on the report output if it passes the screening test.

**Note:** There must be an outer join in effect, either as a result of the SET ALL=ON command or a JOIN LEFT\_OUTER command (either inside or outside of the Master File).

**Reference: Usage Notes for SET SHORTPATH = SQL**

A FOCUS data source is supported as the host file in a join used with SET SHORTPATH = SQL, but not as the cross-referenced file.

**Example: Controlling Outer Join Processing**

The following procedure creates two Oracle tables, ORAEMP and ORAEDUC, that will be used in a join.

```
TABLE FILE EMPLOYEE
SUM LAST_NAME FIRST_NAME CURR_SAL CURR_JOBCODE DEPARTMENT
BY EMP_ID
ON TABLE HOLD AS ORAEMP FORMAT SQLORA
END
-RUN
TABLE FILE EDUCFILE
SUM COURSE_CODE COURSE_NAME
BY EMP_ID BY DATE_ATTEND
ON TABLE HOLD AS ORAEDUC FORMAT SQLORA
END
```

The following request joins the two Oracle tables and creates a left outer join (SET ALL = ON).

```
JOIN EMP_ID IN ORAEMP TO ALL EMP_ID IN ORAEDUC AS J1
SET ALL = ON
TABLE FILE ORAEMP
PRINT COURSE_CODE COURSE_NAME
BY EMP_ID
END
```

Since the join is an outer join, all ORAEMP rows display on the report output. ORAEMP rows with no corresponding ORAEDUC row display the missing data symbol for the fields from the ORAEDUC table.

EMP_ID	COURSE_CODE	COURSE_NAME
-----	-----	-----
071382660	101	FILE DESCRPT & MAINT
112847612	101	FILE DESCRPT & MAINT
	103	BASIC REPORT PREP FOR PROG
117593129	101	FILE DESCRPT & MAINT
	103	BASIC REPORT PREP FOR PROG
	201	ADVANCED TECHNIQUES
	203	FOCUS INTERNALS
119265415	108	BASIC RPT NON-DP MGRS
119329144	.	.
123764317	.	.
126724188	.	.
219984371	.	.
326179357	104	FILE DESC & MAINT NON-PROG
	106	TIMESHARING WORKSHOP
	102	BASIC REPORT PREP NON-PROG
	301	DECISION SUPPORT WORKSHOP
	202	WHAT'S NEW IN FOCUS
451123478	101	FILE DESCRPT & MAINT
543729165	.	.
818692173	107	BASIC REPORT PREP DP MGRS

The following request adds a screening condition on the ORAEDUC segment. To satisfy the screening condition, the course name must either contain the characters *BASIC* or be missing.

```

JOIN CLEAR
JOIN EMP_ID IN ORAEMP TO ALL EMP_ID IN ORAEDUC AS J1
SET ALL = ON
TABLE FILE ORAEMP
PRINT COURSE_CODE COURSE_NAME
BY EMP_ID
WHERE COURSE_NAME CONTAINS 'BASIC' OR COURSE_NAME IS MISSING
END

```

However, with SET ALL = ON, the rows with missing values are not retained on the report output.

EMP_ID	COURSE_CODE	COURSE_NAME
-----	-----	-----
112847612	103	BASIC REPORT PREP FOR PROG
117593129	103	BASIC REPORT PREP FOR PROG
119265415	108	BASIC RPT NON-DP MGRS
326179357	102	BASIC REPORT PREP NON-PROG
818692173	107	BASIC REPORT PREP DP MGRS

The following request adds the SET SHORTPATH = SQL command.

```
JOIN CLEAR
JOIN EMP_ID IN ORAEMP TO ALL EMP_ID IN ORAEDUC AS J1
SET ALL = ON
SET SHORTPATH=SQL
TABLE FILE ORAEMP
PRINT COURSE_CODE COURSE_NAME
BY EMP_ID
WHERE COURSE_NAME CONTAINS 'BASIC' OR COURSE_NAME IS MISSING
END
```

The report output now displays both the records containing the characters *BASIC* and those with missing values.

EMP_ID	COURSE_CODE	COURSE_NAME
-----	-----	-----
112847612	103	BASIC REPORT PREP FOR PROG
117593129	103	BASIC REPORT PREP FOR PROG
119265415	108	BASIC RPT NON-DP MGRS
119329144	.	.
123764317	.	.
126724188	.	.
219984371	.	.
326179357	102	BASIC REPORT PREP NON-PROG
543729165	.	.
818692173	107	BASIC REPORT PREP DP MGRS

### **Example:** Finding Host Records That Have No Matching Cross-Referenced Records

The following request counts and lists those employees who have taken no courses.

```
JOIN LEFT_OUTER EMP_ID IN ORAEMP TO ALL EMP_ID IN ORAEDUC AS J1
SET ALL = ON
SET SHORTPATH=SQL
TABLE FILE ORAEMP
COUNT EMP_ID
LIST EMP_ID LAST_NAME FIRST_NAME
WHERE COURSE_NAME IS MISSING
END
```

The output is:

EMP_ID				
COUNT	LIST	EMP_ID	LAST_NAME	FIRST_NAME
-----	-----	-----	-----	-----
5	1	119329144	BANNING	JOHN
	2	123764317	IRVING	JOAN
	3	126724188	ROMANS	ANTHONY
	4	219984371	MCCOY	JOHN
	5	543729165	GREENSPAN	MARY

Testing for Missing Instances in FOCUS Data Sources

You can use the ALL PASS parameter to produce reports that include only parent instances with missing descendant values. To do so, write the request to screen out all existing instances in the segment with missing instances. After you set the ALL parameter to PASS, the report displays only the parent instances that are missing descendants.

*Example:*    **Testing for a MISSING Instance in a FOCUS Data Source**

The following request tests for missing instances in the COURSECODE field. Since no COURSECODE can equal 'XXXX', only employees with missing instances in COURSECODE display in the report output.

```
JOIN EMPDATA.PIN IN EMPDATA TO ALL TRAINING.PIN IN TRAINING AS JOIN1
SET ALL = PASS
TABLE FILE EMPDATA
PRINT LASTNAME AND FIRSTNAME AND COURSECODE AND EXPENSES
BY PIN
WHERE COURSECODE EQ 'XXXX'
END
```

The output is:

PIN	LASTNAME	FIRSTNAME	COURSECODE	EXPENSES
000000020	BELLA	MICHAEL	.	.
000000060	PATEL	DORINA	.	.
000000070	SANCHEZ	EVELYN	.	.
000000090	PULASKI	MARIANNE	.	.
000000130	CVEK	MARCUS	.	.
000000170	MORAN	WILLIAM	.	.
000000220	LEWIS	CASSANDRA	.	.
000000230	NOZAWA	JIM	.	.
000000300	SOPENA	BEN	.	.
000000390	GRAFF	ELAINE	.	.
000000400	LOPEZ	ANNE	.	.

Setting the NODATA Character String

In a report, the NODATA character string indicates no data or inapplicable data. The default NODATA character is a period. However, you can change this character designation.

**Syntax:** How to Set the NODATA String

```
SET NODATA = string
```

where:

*string*

Is the character string used to indicate missing data in reports. The default string is a period (.). The string may be a maximum of 11 characters. Common choices are NONE, N/A, and MISSING.

**Example:** Setting NODATA Not to Display Characters

If you do not want any characters, issue the command:

```
SET NODATA = ' '
```

**Example:** Setting the NODATA Character String

In the following request, the NODATA character string is set to MISSING. The word MISSING displays on the report instead of the default period.

```
SET NODATA=MISSING
```

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL BY LAST_NAME BY FIRST_NAME
ACROSS DEPARTMENT
END
```

The output is:

LAST_NAME	FIRST_NAME	DEPARTMENT MIS	PRODUCTION
BANNING	JOHN	MISSING	\$29,700.00
BLACKWOOD	ROSEMARIE	\$21,780.00	MISSING
CROSS	BARBARA	\$27,062.00	MISSING
GREENSPAN	MARY	\$9,000.00	MISSING
IRVING	JOAN	MISSING	\$26,862.00
JONES	DIANE	\$18,480.00	MISSING
MCCOY	JOHN	\$18,480.00	MISSING
MCKNIGHT	ROGER	MISSING	\$16,100.00
ROMANS	ANTHONY	MISSING	\$21,120.00
SMITH	MARY	\$13,200.00	MISSING
	RICHARD	MISSING	\$9,500.00
STEVENS	ALFRED	MISSING	\$11,000.00





## Joining Data Sources

---

You can join two or more related data sources to create a larger integrated data structure from which you can report in a single request. The joined structure is virtual. It is a way of accessing multiple data sources as if they were a single data source. Up to 1023 joins can be in effect at one time, for a total of 1024 segments, depending on the number of active segments and the number and length of the fields (there is a 32K limit on the length of all fields).

For details about data sources you can use in a join, see [Data Sources You Can and Cannot Join](#) on page 1007.

**In this chapter:**

- ❑ [Types of Joins](#)
  - ❑ [How the JOIN Command Works](#)
  - ❑ [Creating an Equijoin](#)
  - ❑ [Using a Conditional Join](#)
  - ❑ [Full Outer Joins for Relational Data Sources](#)
  - ❑ [Reporting Against a Multi-Fact Cluster Synonym](#)
  - ❑ [Invoking Context Analysis for a Star Schema With a Fan Trap](#)
  - ❑ [Adding DBA Restrictions to the Join Condition: SET DBAJJOIN](#)
  - ❑ [Preserving Virtual Fields During Join Parsing](#)
  - ❑ [Displaying Joined Structures](#)
  - ❑ [Clearing Joined Structures](#)
-

## Types of Joins

When you join two data sources, some records in one of the files may lack corresponding records in the other file. When a report omits records that are not in both files, the join is called an inner join. When a report displays all matching records, plus all records from the host file that lack corresponding cross-referenced records, the join is called a left outer join. When a report displays all matching records plus all records from both files that lack corresponding records in the other file, the join is called a full outer join. Full outer joins are supported for relational data sources only.

The SET ALL command globally determines how all joins are implemented. If the SET ALL=ON command is issued, all joins are treated as outer joins. With SET ALL=OFF, the default, all joins are treated as inner joins.

Each JOIN command can specify explicitly which type of join to perform, locally overruling the global setting. This syntax is supported for FOCUS, XFOCUS, Relational, VSAM, IMS, and Adabas. If you do not specify the type of join in the JOIN command, the ALL parameter setting determines the type of join to perform.

You can also join data sources using one of two techniques for determining how to match records from the separate data sources. The first technique is known as an equijoin and the second is known as a conditional join. When deciding which of the two join techniques to use, it is important to know that when there is an equality condition between two data sources, it is more efficient to use an equijoin rather than a conditional join.

You can use an equijoin structure when you are joining two or more data sources that have two fields, one in each data source, with formats (character, numeric, or date) and values in common. Joining a product code field in a sales data source (the host file) to the product code field in a product data source (the cross-referenced file) is an example of an equijoin. For more information on using equijoins, see [Creating an Equijoin](#) on page 1018.

The conditional join uses WHERE-based syntax to specify joins based on WHERE criteria, not just on equality between fields. Additionally, the host and cross-referenced join fields do not have to contain matching formats. Suppose you have a data source that lists employees by their ID number (the host file), and another data source that lists training courses and the employees who attended those courses (the cross-referenced file). Using a conditional join, you could join an employee ID in the host file to an employee ID in the cross-referenced file to determine which employees took training courses in a given date range (the WHERE condition). For more information on conditional joins, see [Using a Conditional Join](#) on page 1034.

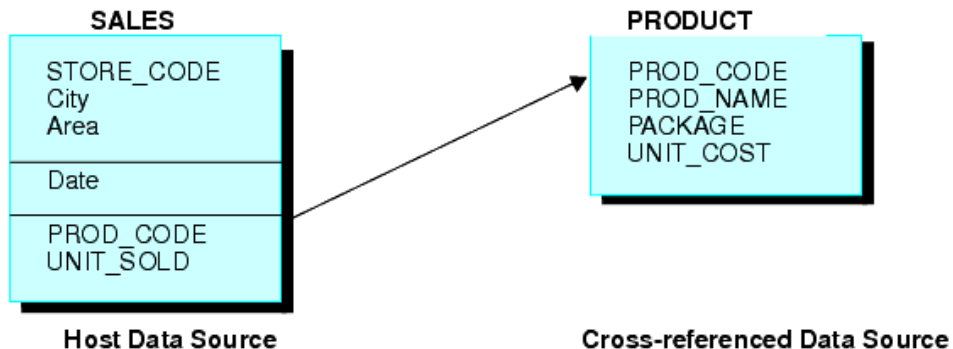
Joins can also be unique or non-unique. A unique, or one-to-one, join structure matches one value in the host data source to one value in the cross-referenced data source. Joining an employee ID in an employee data source to an employee ID in a salary data source is an example of a unique equijoin structure.

A non-unique, or one-to-many, join structure matches one value in the host data source to multiple values in the cross-referenced field. Joining an employee ID in a company's employee data source to an employee ID in a data source that lists all the training classes offered by that company results in a listing of all courses taken by each employee, or a joining of the one instance of each ID in the host file to the multiple instances of that ID in the cross-referenced file.

For more information on unique and non-unique joins, see [Unique and Non-Unique Joined Structures](#) on page 1008.

### **Example:** Joined Data Structure

Consider the SALES and PRODUCT data sources. Each store record in SALES may contain many instances of the PROD\_CODE field. It would be redundant to store the associated product information with each instance of the product code. Instead, PROD\_CODE in the SALES data source is joined to PROD\_CODE in the PRODUCT data source. PRODUCT contains a single instance of each product code and related product information, thus saving space and making it easier to maintain product information. The joined structure, which is an example of an equijoin, is illustrated below:



### **Reference:** Data Sources You Can and Cannot Join

The use of data sources as host files and cross-referenced files in joined structures depends on the types of data sources you are joining:

- ☐ Typically, joins can be established between any FOCUS-readable files.

- ❑ Data sources protected by DBA security may be joined, with certain restrictions. For details, see [Notes on DBA Security for Joined Data Structures](#) on page 1008.
- ❑ Conditional joins are supported only for FOCUS, VSAM, ADABAS, IMS, and all relational data sources.

### **Reference:** Notes on DBA Security for Joined Data Structures

- ❑ You can join a data source with DBA protection to another data source with DBA protection, as long as they use the same password.
- ❑ In addition, you can join DBA protected data sources with different passwords by adding the DBAFILE attribute to your security definition. The DBAFILE attribute names a central Master File that contains different passwords and restrictions for several Master Files. If you use a DBAFILE, a user can set separate passwords for each file using the syntax:

```
SET PASS = pswd1 IN file1, pswd2 IN file2
```

Individual DBA information remains in effect for each file in the JOIN. For details about the DBAFILE attribute, see the *Describing Data With WebFOCUS Language* manual.

- ❑ You can also join a DBA-protected host file to an unprotected cross-referenced file. The DBA information is taken from the host file.

### **Unique and Non-Unique Joined Structures**

In a unique joined structure, one value in the host field corresponds to one value in the cross-referenced field. In a non-unique joined structure, one value in the host field corresponds to multiple values in the cross-referenced field.

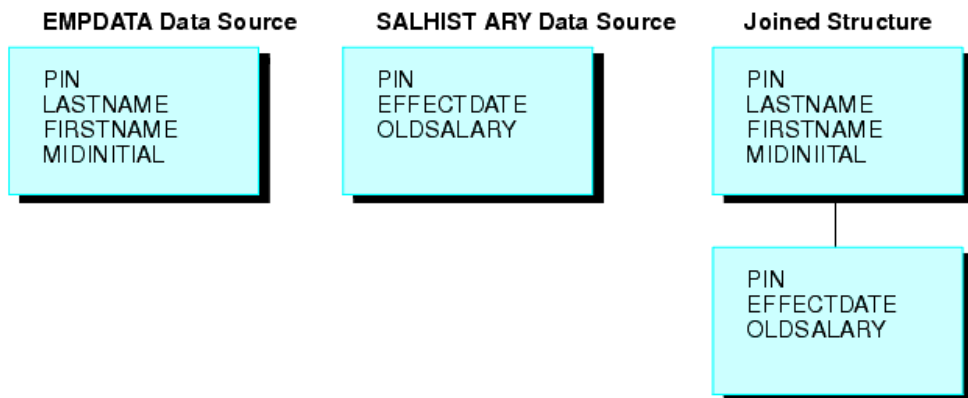
The ALL parameter in a JOIN command indicates that the joined structure is non-unique.

- ❑ Omit the ALL parameter only when you are sure that the joined structure is unique. Omitting the ALL parameter reduces overhead.
- ❑ The ALL parameter does not interfere with the proper creation of the joined structure even if it is unique. Use the ALL parameter if you are not sure whether the joined structure is unique. This ensures that your reports contain all relevant data from the cross-referenced file, regardless of whether the structure is unique.

**Example: A Unique Equijoin Structure**

The following example illustrates a unique joined structure. Two FOCUS data sources are joined together: an EMPDATA data source and a SALHIST data source. Both data sources are organized by PIN, and they are joined on a PIN field in the root segments of both files. Each PIN has one segment instance in the EMPDATA data source, and one instance in the SALHIST data source. To join these two data sources, issue this JOIN command:

```
JOIN PIN IN EMPDATA TO PIN IN SALHIST
```

**Example: A Non-Unique Equijoin Structure**

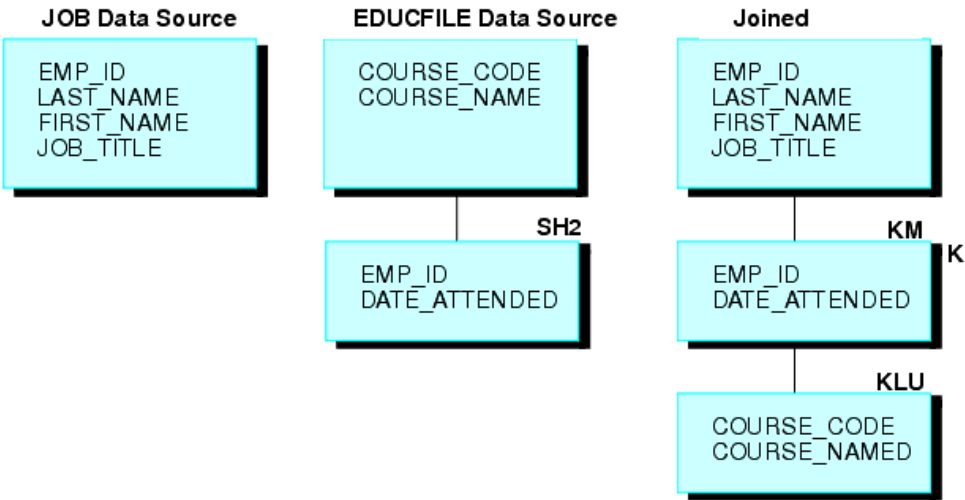
If a field value in the host file can appear in many segment instances in the cross-referenced file, then you should include the ALL phrase in the JOIN syntax. This structure is called a non-unique joined structure.

For example, assume that two FOCUS data sources are joined together: the JOB data source and an EDUCFILE data source which records employee attendance at in-house courses. The joined structure is shown in the following diagram.

The JOB data source is organized by employee, but the EDUCFILE data source is organized by course. The data sources are joined on the EMP\_ID field. Since an employee has one position but can attend several courses, the employee has one segment instance in the JOB data source but can have as many instances in the EDUCFILE data source as courses attended.

To join these two data sources, issue the following JOIN command, using the ALL phrase:

```
JOIN EMP_ID IN JOB TO ALL EMP_ID IN EDUCFILE
```



**Syntax:**      **How to Correct for Lagging Values With a Unique Join**

If a parent segment has two or more unique child segments that each have multiple children, the report may incorrectly display a missing value. The remainder of the child values may then be misaligned in the report. These misaligned values are called lagging values. The JOINOPT parameter ensures proper alignment of your output by correcting for lagging values.

```
SET JOINOPT={NEW|OLD|GNTINT}
```

where:

**NEW**

Specifies that segments be retrieved from left to right and from top to bottom, which results in the display of all data for each record, properly aligned. Missing values only occur when they exist in the data.

**OLD**

Specifies that segments be retrieved as unique segments, which results in the display of missing data in a report where all records should have values. This might cause lagging values. OLD is the default value.

**GNTINT**

Specifies that segments be retrieved from left to right and from top to bottom, which results in the display of all data for each record, properly aligned. Missing values only occur when they exist in the data.

**Note:** The value GNTINT both corrects for lagging values and enables joins between different numeric data types, as described in [Joining Fields With Different Numeric Data Types](#) on page 1033.

**Example:** **Correcting for Lagging Values in a Procedure With Unique Segments and Multiple Children**

This example is a hypothetical scenario in which you would use the JOINOPT parameter to correct for lagging values. Lagging values display missing data such that each value appears off by one line.

A single-segment host file (ROUTES) is joined to two files (ORIGIN and DEST), each having two segments. The files are joined to produce a report that shows each train number, along with the city that corresponds to each station.

The following request prints the city of origin (OR\_CITY) and the destination city (DE\_CITY). Note that missing data is generated, causing the data for stations and corresponding cities to lag, or be off by one line.

```
TABLE FILE ROUTES
PRINT TRAIN_NUM
OR_STATION OR_CITY
DE_STATION DE_CITY
END
```

The output is:

TRAIN_NUM	OR_STATION	OR_CITY	DE_STATION	DE_CITY
-----	-----	-----	-----	-----
101	NYC	NEW YORK	ATL	.
202	BOS	BOSTON	BLT	ATLANTA
303	DET	DETROIT	BOS	BALTIMORE
404	CHI	CHICAGO	DET	BOSTON
505	BOS	BOSTON	STL	DETROIT
505	BOS	.	STL	ST. LOUIS

Issuing SET JOINOPT=NEW enables segments to be retrieved in the expected order (from left to right and from top to bottom), without missing data.

```
SET JOINOPT=NEW
TABLE FILE ROUTES
PRINT TRAIN_NUM
OR_STATION OR_CITY
DE_STATION DE_CITY
END
```

The correct report has only 5 lines instead of 6, and the station and city data is properly aligned. The output is:

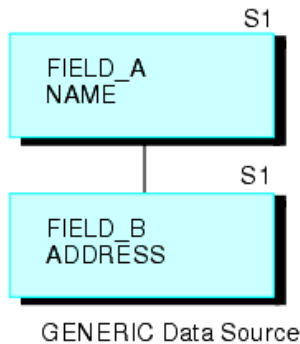
TRAIN_NUM	OR_STATION	OR_CITY	DE_STATION	DE_CITY
101	NYC	NEW YORK	ATL	ATLANTA
202	BOS	BOSTON	BLT	BALTIMORE
303	DET	DETROIT	BOS	BOSTON
404	CHI	CHICAGO	DET	DETROIT
505	BOS	BOSTON	STL	ST. LOUIS

Recursive Joined Structures

You can join a FOCUS or IMS data source to itself, creating a recursive structure. In the most common type of recursive structure, a parent segment is joined to a descendant segment, so that the parent becomes the child of the descendant. This technique (useful for storing bills of materials, for example) enables you to report from data sources as if they have more segment levels than is actually the case.

Example: Understanding Recursive Joined Structures

For example, the GENERIC data source shown below consists of Segments A and B.

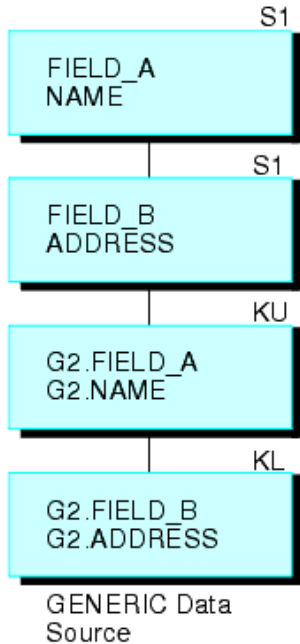


The following request creates a recursive structure:

```
JOIN FIELD_B IN GENERIC TAG G1 TO FIELD_A IN GENERIC TAG G2 AS RECURSIV
```



This results in the joined structure (shown below).



Note that the two segments are repeated on the bottom. To refer to the fields in the repeated segments (other than the field to which you are joining), prefix the tag names to the field names and aliases and separate them with a period, or append the first four characters of the JOIN name to the field names and aliases. In the above example, the JOIN name is RECURSIV. You should refer to FIELD\_B in the bottom segment as G2.FIELD\_B (or RECUFIELD\_B). For related information, see [Usage Notes for Recursive Joined Structures](#) on page 1013.

**Reference:** Usage Notes for Recursive Joined Structures

- ❑ You must either specify a unique JOIN name, or use tag names in the JOIN command. Otherwise, you will not be able to refer to the fields in the repeated segments at the bottom of the join structure.
- ❑ If you use tag names in a recursive joined structure, note the following guidelines:
  - ❑ If tag names are specified in a recursive join, duplicate field names must be qualified with the tag name.
  - ❑ If a join name is specified and tag names are not specified in a recursive join, duplicate field names must be prefixed with the first four characters of the join name.

- ☐ If both a join name and a tag name are specified in a recursive join, the tag name must be used as a qualifier.
- ☐ The tag name must be used as the field name qualifier in order to retrieve duplicate field names in a non-recursive join. If you do not qualify the field name, the first occurrence is retrieved.
- ☐ You may use a DEFINE-based join (see [How to Join From a Virtual Field to a Real Field](#) on page 1028) to join a virtual field in a descendant segment to a field in the parent segment.
- ☐ You can extend a recursive structure by issuing multiple JOIN commands from the bottom repeat segment in the structure to the parent segment, creating a structure up to 16 levels deep.
- ☐ For FOCUS data sources, the field in the parent segment to which you are joining must be indexed.
- ☐ For IMS data sources, the following applies:
  - ☐ The parent segment must be the root segment of the data source.
  - ☐ The field to which you are joining must be both a key field and a primary or secondary index.
- ☐ You need a duplicate PCB in the PSB for every recursive join you create.

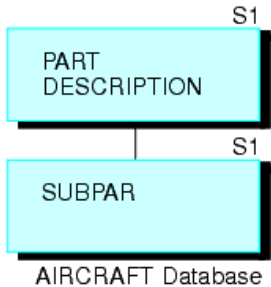
### ***Example:*** Using Recursive Joined Structures

This example explains how to use recursive joins to store and report on a bill of materials. Suppose you are designing a data source called AIRCRAFT that contains the bill of materials for an aircraft manufactured by a company. The data source records data on three levels of airplane parts:

- ☐ Major divisions, such as the cockpit or cabin.
- ☐ Parts of divisions, such as instrument panels and seats.
- ☐ Subparts, such as nuts and bolts.

The data source must record each part, the part description, and the smaller parts composing the part. Some parts, such as nuts and bolts, are common throughout the aircraft. If you design a three-segment structure, one segment for each level of parts, descriptions of common parts are repeated in every place they are used.

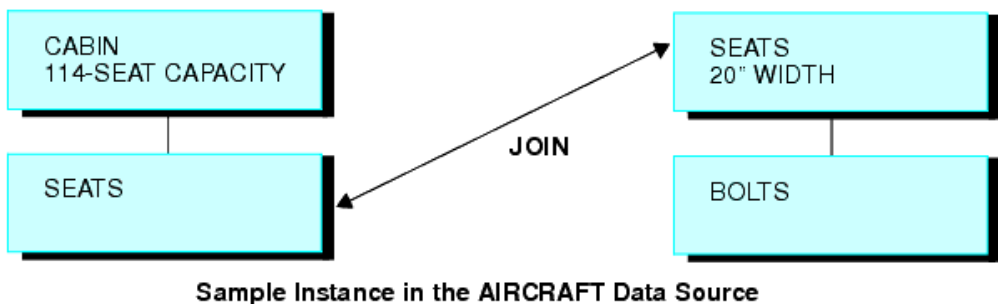
To reduce this repetition, design a data source that has only two segments (shown in the following diagram). The top segment describes each part of the aircraft, large and small. The bottom segment lists the component parts without descriptions.



Every part (except for the largest divisions) appears in both the top segment, where it is described, and in the bottom segment, where it is listed as one of the components of a larger part. (The smallest parts, such as nuts and bolts, appear in the top segment without an instance of a child in the bottom segment.) Note that each part, no matter how often it is used in the aircraft, is described only once.

If you join the bottom segment to the top segment, the descriptions of component parts in the bottom segment can be retrieved. The first-level major divisions can also be related to third-level small parts, going from the first level to the second level to the third level. Thus, the structure behaves as a three-level data source, although it is actually a more efficient two-level source.

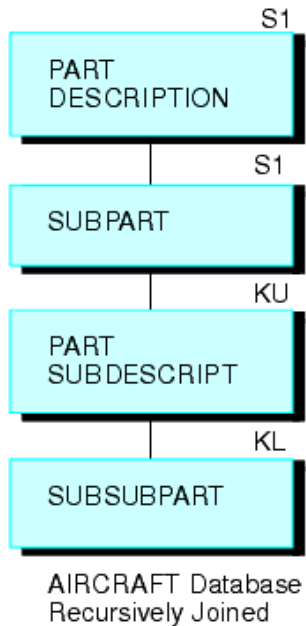
For example, CABIN is a first-level division appearing in the top segment. It lists SEATS as a component in the bottom segment. SEATS also appears in the top segment. It lists BOLTS as a component in the bottom segment. If you join the bottom segment to the top segment, you can go from CABIN to SEATS and from SEATS to BOLTS.



Join the bottom segment to the top segment with this JOIN command:

```
JOIN SUBPART IN AIRCRAFT TO PART IN AIRCRAFT AS SUB
```

This creates the following recursive structure.



You can then produce a report on all three levels of data with this TABLE command (the field SUBDESCRIPT describes the contents of the field SUBPART):

```
TABLE FILE AIRCRAFT
PRINT SUBPART BY PART BY SUBPART BY SUBDESCRIPT
END
```

## How the JOIN Command Works

The JOIN command enables you to report from two or more related data sources with a single request. Joined data sources remain physically separate, but are treated as one. Up to 1023 joins can be in effect at any one time.

When two data sources are joined, one is called the host file; the other is called the cross-referenced file. Each time a record is retrieved from the host file, the corresponding fields in the cross-referenced file are identified if they are referenced in the report request. The records in the cross-referenced file containing the corresponding values are then retrieved.

Two data sources can be joined using a conditional join whenever you can define an expression that determines how to relate records in the host file to records in the cross-referenced file. Two data sources can be joined using an equijoin when they have fields in each data source with formats (character, numeric, or date) and values in common. The common formats ensure proper interpretation of the values. For example, suppose that you need to read data from two data sources: one named JOB, containing job information, and a second named SALARY, containing salary information. You can join these two data sources if each has a field identifying the same group of employees in the same way: by last name, serial number, or social security number. The join becomes effective when common values (for example, common social security numbers) are retrieved for the joined fields.

After you issue the JOIN command, you can issue a single TABLE, TABLEF, MATCH FILE, or GRAPH request to read the joined data source. You only need to specify the first data source (host file) to produce a report from two or more data sources. For example, assume you are writing a report from the JOB and SALARY data sources. You execute the following equijoin:

```
JOIN EMP_ID IN JOB TO ALL EMP_ID IN SALARY
```

This command joins the field EMP\_ID in the JOB file to the field EMP\_ID in the SALARY file. JOB is the host file and SALARY is the cross-referenced file. You then execute this report request:

```
TABLE FILE JOB
PRINT SALARY AND JOB_TITLE BY EMP_ID
END
```

The first record retrieved is a JOB file record for employee #071382660. Next, all records in the SALARY data source containing employee #071382660 are retrieved. This process continues until all the records have been read.

You can base your join on:

- ☐ Real fields that have been declared in the Master Files of the host and cross-referenced data sources, respectively. See [How to Join Real Fields](#) on page 1018.
- ☐ A virtual field in the host data source (that has either been defined in the Master File or with a DEFINE command) and a real field that has been declared in the Master File of the cross-referenced data source. See [How to Join From a Virtual Field to a Real Field](#) on page 1028.
- ☐ A condition you specify in the JOIN command itself. See [How to Create a Conditional JOIN](#) on page 1035.

## **Reference:** Increasing Retrieval Speed in Joined Data Sources

You can increase retrieval speed in joined structures by using an external index. However, the target segment for the index cannot be a cross-referenced segment. For related information, see [Improving Report Processing](#) on page 1839.

## Creating an Equijoin

The most common joined structures are based on real fields that have been declared in the Master Files of the host and cross-referenced data sources, respectively.

### **Syntax:** How to Join Real Fields

The following JOIN syntax requires that the fields you are using to join the files are real fields declared in the Master File. This join may be a simple one based on one field in each file to be joined, or a multi-field join for data sources that support this type of behavior. The following syntax describes the simple and multi-field variations:

```
JOIN [LEFT_OUTER|INNER] hfld1 [AND hfld2 ...] IN hostfile [TAG tag1]
    TO [UNIQUE|MULTIPLE]
    crfield [AND crfld2 ...] IN crfile [TAG tag2] [AS joinname]
END
```

where:

**JOIN hfld1**

Is the name of a field in the host file containing values shared with a field in the cross-referenced file. This field is called the host field.

**AND hfld2...**

Can be an additional field in the host file, with the caveats noted below. The phrase beginning with AND is required when specifying multiple fields.

- ❑ When you are joining two FOCUS data sources you can specify up to four alphanumeric fields in the host file that, if concatenated, contain values shared with the cross-referenced file. You may not specify more than one field in the cross-referenced file when the suffix of the file is FOC. For example, assume the cross-referenced file contains a phone number field with an area code-prefix-exchange format. The host file has an area code field, a prefix field, and an exchange field. You can specify these three fields to join them to the phone number field in the cross-referenced file. The JOIN command treats the three fields as one. Other data sources do not have this restriction on the cross-referenced file.
- ❑ For data adapters that support multi-field and concatenated joins, you can specify up to 16 fields. See your data adapter documentation for specific information about supported join features. Note that FOCUS data sources do not support these joins.

**INNER**

Specifies an inner join. If you do not specify the type of join in the JOIN command, the ALL parameter setting determines the type of join to perform.

**LEFT OUTER**

Specifies a left outer join. If you do not specify the type of join in the JOIN command, the ALL parameter setting determines the type of join to perform.

Note that in a left outer join, host records with a missing cross-referenced instance are included in the report output. To control how tests against missing cross-referenced segment instances are processed, use the SET SHORTPATH command described in [Handling Records With Missing Field Values](#) on page 971.

**IN *hostfile***

Is the name of the host file.

**TAG *tag1***

Is a tag name of up to 66 characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in the host file.

The tag name for the host file must be the same in all the JOIN commands of a joined structure.

**TO [UNIQUE | MULTIPLE] *crfld1***

Is the name of a field in the cross-referenced file containing values that match those of *hfld1* (or of concatenated host fields). This field is called the cross-referenced field.

**Note:** Unique returns only one instance and, if there is no matching instance in the cross-referenced file, it supplies default values (blank for alphanumeric fields and zero for numeric fields).

Use the MULTIPLE parameter when *crfld1* may have multiple instances in common with one value in *hfld1*. Note that ALL is a synonym for MULTIPLE, and omitting this parameter entirely is a synonym for UNIQUE. See [Unique and Non-Unique Joined Structures](#) on page 1008 for more information.

**AND *crfld2...***

Is the name of a field in the cross-referenced file with values in common with *hfld2*.

**Note:** *crfld2* may be qualified. This field is only available for data adapters that support multi-field joins.

**IN *crfile***

Is the name of the cross-referenced file.

**TAG *tag2***

Is a tag name of up to 66 characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in cross-referenced files. In a recursive join structure, if no tag name is provided, all field names and aliases are

prefixed with the first four characters of the join name. For related information, see [Usage Notes for Recursive Joined Structures](#) on page 1013.

The tag name for the host file must be the same in all the JOIN commands of a joined structure.

### **AS** *joinname*

Is an optional name of up to eight characters that you may assign to the join structure. You must assign a unique name to a join structure if:

- ☐ You want to ensure that a subsequent JOIN command does not overwrite it.
- ☐ You want to clear it selectively later.
- ☐ The structure is recursive. See [Recursive Joined Structures](#) on page 1012.

**Note:** If you do not assign a name to the join structure with the AS phrase, the name is assumed to be blank. A join without a name overwrites an existing join without a name.

### **END**

Required when the JOIN command is longer than one line. It terminates the command. It must be on a line by itself.

### **Example:** Creating a Simple Unique Joined Structure

An example of a simple unique join is shown below:

```
JOIN JOBCODE IN EMPLOYEE TO JOBCODE IN JOBFILE AS JJOIN
```

### **Example:** Creating an Inner Join

The following procedure creates three FOCUS data sources:

- ☐ EMPINFO, which contains the fields EMP\_ID, LAST\_NAME, FIRST\_NAME, and CURR\_JOBCODE from the EMPINFO segment of the EMPLOYEE data source.
- ☐ JOBINFO, which contains the JOBCODE and JOB\_DESC fields from the JOBFILE data source.
- ☐ EDINFO, which contains the EMP\_ID, COURSE\_CODE, and COURSE\_NAME fields from the EDUCFILE data source.



The procedure then adds an employee to EMPINFO named Fred Newman who has no matching record in the JOBINFO or EDINFO data sources.

```
TABLE FILE EMPLOYEE
SUM LAST_NAME FIRST_NAME CURR_JOBCODE
BY EMP_ID
ON TABLE HOLD AS EMPINFO FORMAT FOCUS INDEX EMP_ID CURR_JOBCODE
END
-RUN
```

```
TABLE FILE JOBFIL
SUM JOB_DESC
BY JOBCODE
ON TABLE HOLD AS JOBINFO FORMAT FOCUS INDEX JOBCODE
END
-RUN
```

```
TABLE FILE EDUCFILE
SUM COURSE_CODE COURSE_NAME
BY EMP_ID
ON TABLE HOLD AS EDINFO FORMAT FOCUS INDEX EMP_ID
END
-RUN
```

```
MODIFY FILE EMPINFO
FREEFORM EMP_ID LAST_NAME FIRST_NAME CURR_JOBCODE
MATCH EMP_ID
ON NOMATCH INCLUDE
ON MATCH REJECT
DATA
111111111, NEWMAN, FRED, C07,$
END
```

The following request prints the contents of EMPINFO. Note that Fred Newman has been added to the data source:

```
TABLE FILE EMPINFO
PRINT *
END
```

The output is:

EMP_ID	LAST_NAME	FIRST_NAME	CURR_JOBCODE
-----	-----	-----	-----
071382660	STEVENS	ALFRED	A07
112847612	SMITH	MARY	B14
117593129	JONES	DIANE	B03
119265415	SMITH	RICHARD	A01
119329144	BANNING	JOHN	A17
123764317	IRVING	JOAN	A15
126724188	ROMANS	ANTHONY	B04
219984371	MCCOY	JOHN	B02
326179357	BLACKWOOD	ROSEMARIE	B04
451123478	MCKNIGHT	ROGER	B02
543729165	GREENSPAN	MARY	A07
818692173	CROSS	BARBARA	A17
111111111	NEWMAN	FRED	C07

The following JOIN command creates an inner join between the EMPINFO data source and the JOBINFO data source.

```
JOIN CLEAR *
JOIN INNER CURR_JOBCODE IN EMPINFO TO MULTIPLE JOBCODE IN JOBINFO AS J0
```

Note that the JOIN command specifies a multiple join. In a unique join, the cross-referenced segment is never considered missing, and all records from the host file display on the report output. Default values (blank for alphanumeric fields and zero for numeric fields) display if no actual data exists.

The following request displays fields from the joined structure:

```
TABLE FILE EMPINFO
PRINT LAST_NAME FIRST_NAME JOB_DESC
END
```

Fred Newman is omitted from the report output because his job code does not have a match in the JOBINFO data source:

LAST_NAME	FIRST_NAME	JOB_DESC
-----	-----	-----
STEVENS	ALFRED	SECRETARY
SMITH	MARY	FILE QUALITY
JONES	DIANE	PROGRAMMER ANALYST
SMITH	RICHARD	PRODUCTION CLERK
BANNING	JOHN	DEPARTMENT MANAGER
IRVING	JOAN	ASSIST.MANAGER
ROMANS	ANTHONY	SYSTEMS ANALYST
MCCOY	JOHN	PROGRAMMER
BLACKWOOD	ROSEMARIE	SYSTEMS ANALYST
MCKNIGHT	ROGER	PROGRAMMER
GREENSPAN	MARY	SECRETARY
CROSS	BARBARA	DEPARTMENT MANAGER

**Example: Creating a Left Outer Join**

The following JOIN command creates a left outer join between the EMPINFO data source and the EDINFO data source:

```
JOIN CLEAR *
JOIN LEFT_OUTER EMP_ID IN EMPINFO TO MULTIPLE EMP_ID IN EDINFO AS J1
```

The following request displays fields from the joined structure:

```
TABLE FILE EMPINFO
PRINT LAST_NAME FIRST_NAME COURSE_NAME
END
```

All employee records display on the report output. The records for those employees with no matching records in the EDINFO data source display the missing data character (.) in the COURSE\_NAME column. If the join were unique, blanks would display instead of the missing data character.

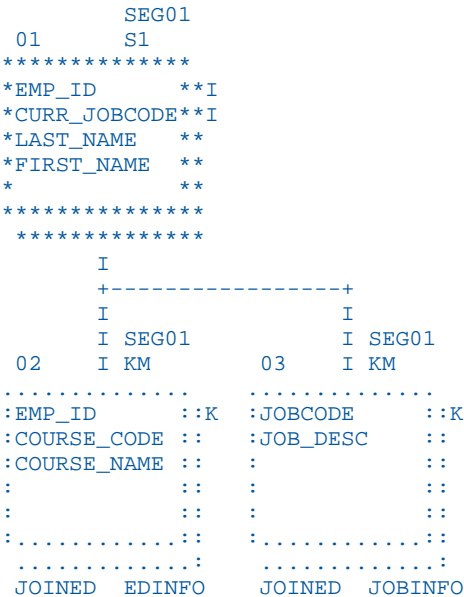
LAST_NAME	FIRST_NAME	COURSE_NAME
-----	-----	-----
STEVENS	ALFRED	FILE DESCRPT & MAINT
SMITH	MARY	BASIC REPORT PREP FOR PROG
JONES	DIANE	FOCUS INTERNALS
SMITH	RICHARD	BASIC RPT NON-DP MGRS
BANNING	JOHN	.
IRVING	JOAN	.
ROMANS	ANTHONY	.
MCCOY	JOHN	.
BLACKWOOD	ROSEMARIE	DECISION SUPPORT WORKSHOP
MCKNIGHT	ROGER	FILE DESCRPT & MAINT
GREENSPAN	MARY	.
CROSS	BARBARA	HOST LANGUAGE INTERFACE
NEWMAN	FRED	.

**Example: Creating Two Inner Joins With a Multipath Structure**

The following JOIN commands create an inner join between the EMPINFO and JOBINFO data sources and an inner join between the EMPINFO and EDINFO data sources:

```
JOIN CLEAR *
JOIN INNER CURR_JOBCODE IN EMPINFO TO MULTIPLE JOBCODE IN JOBINFO AS J0
JOIN INNER EMP_ID IN EMPINFO TO MULTIPLE EMP_ID IN EDINFO AS J1
```

The structure created by the two joins has two independent paths:



The following request displays fields from the joined structure:

```

SET MULTIPATH=SIMPLE
TABLE FILE EMPINFO
PRINT LAST_NAME FIRST_NAME IN 12 COURSE_NAME JOB_DESC
END
```

With MULTIPATH=SIMPLE, the independent paths create independent joins. All employee records accepted by either join display on the report output. Only Fred Newman (who has no matching record in either of the cross-referenced files) is omitted:

LAST_NAME	FIRST_NAME	COURSE_NAME	JOB_DESC
STEVENS	ALFRED	FILE DESCRPT & MAINT	SECRETARY
SMITH	MARY	BASIC REPORT PREP FOR PROG	FILE QUALITY
JONES	DIANE	FOCUS INTERNALS	PROGRAMMER ANALYST
SMITH	RICHARD	BASIC RPT NON-DP MGRS	PRODUCTION CLERK
BANNING	JOHN	.	DEPARTMENT MANAGER
IRVING	JOAN	.	ASSIST.MANAGER
ROMANS	ANTHONY	.	SYSTEMS ANALYST
MCCOY	JOHN	.	PROGRAMMER
BLACKWOOD	ROSEMARIE	DECISION SUPPORT WORKSHOP	SYSTEMS ANALYST
MCKNIGHT	ROGER	FILE DESCRPT & MAINT	PROGRAMMER
GREENSPAN	MARY	.	SECRETARY
CROSS	BARBARA	HOST LANGUAGE INTERFACE	DEPARTMENT MANAGER

With MULTIPATH=COMPOUND, only employees with matching records in both of the cross-referenced files display on the report output:

LAST_NAME	FIRST_NAME	COURSE_NAME	JOB_DESC
-----	-----	-----	-----
STEVENS	ALFRED	FILE DESCRPT & MAINT	SECRETARY
SMITH	MARY	BASIC REPORT PREP FOR PROG	FILE QUALITY
JONES	DIANE	FOCUS INTERNALS	PROGRAMMER ANALYST
SMITH	RICHARD	BASIC RPT NON-DP MGRS	PRODUCTION CLERK
BLACKWOOD	ROSEMARIE	DECISION SUPPORT WORKSHOP	SYSTEMS ANALYST
MCKNIGHT	ROGER	FILE DESCRPT & MAINT	PROGRAMMER
CROSS	BARBARA	HOST LANGUAGE INTERFACE	DEPARTMENT MANAGER

### **Reference:** Requirements for Cross-Referenced Fields in an Equijoin

The cross-referenced fields used in a JOIN must have the following characteristics in specific data sources:

- ☐ In relational data sources and in a CA-DATACOM/DB data source, the cross-referenced field can be any field.
- ☐ In FOCUS and XFOCUS data sources, the cross-referenced field must be indexed. Indexed fields have the attribute FIELDTYPE=I or INDEX=I or INDEX=ON in the Master File. If the cross-referenced field does not have this attribute, append the attribute to the field declaration in the Master File and rebuild the file using the REBUILD utility with the INDEX option. This adds an index to your FOCUS or XFOCUS data source.
 

**Note:** The indexed fields can be external. See the *Describing Data With WebFOCUS Language* manual for more information about indexed fields and the Rebuild tool.
- ☐ In IMS data sources, the cross-referenced field must be a key field in the root segment. It can be a primary or secondary index.
- ☐ In fixed format or delimited sequential files, any field can be a cross-referenced field. However, both the host and cross-referenced file must be retrieved in ascending order on the named (key) fields, if the command ENGINE INT CACHE SET OFF is in effect. In this situation, if the data is not in the same sort order, errors are displayed and a many-to-many join is not supported. However, if ENGINE INT CACHE SET ON is in effect, the files do not have to be in ascending order and a many-to-many join is supported. ON is the default value. A delimited file used as the cross-referenced file in the join must consist of only one segment. If the join is based on multiple fields, a fixed format sequential file must consist of a single segment. If the cross-referenced fixed format sequential file contains only one segment, the host file must have a segment declaration.

### **Reference: Restrictions on Group Fields**

When group fields are used in a joined structure, the group in the host file and the group in the cross-referenced file must have the same number of elements:

- ❑ In ISAM data sources, the field must be the full primary key if you issue a unique join, or an initial subset of the primary key if you issue a non-unique join. In the Master File, the primary key is described by a key GROUP; the initial subset is the first field in that group.
- ❑ In VSAM KSDS data sources, the field must be the full primary or alternate key if you issue a unique join, or an initial subset of the primary or alternate key if you issue a non-unique join. In the Master File, the primary key is described by a key GROUP. The initial subset is the first field in that group.

In VSAM ESDS data sources, the field can be any field, as long as the file is already sorted on that field.

- ❑ In Model 204 data sources, the field must be a key field. In the Access File, the types of key fields are alphanumeric (KEY), ordered character (ORA), ordered numeric (ORN), numeric range (RNG), invisible (IVK), and invisible range (IVR).
- ❑ In ADABAS data sources, the field must be a descriptor field, a superdescriptor defined with the .SPR or .NOP field name suffix, or a subdescriptor defined with the .NOP field name suffix. The field description in the Master File must contain the attribute FIELDTYPE=I.

In the Access File, the cross-referenced segment must specify ACCESS=ADBS and either CALLTYPE=Find or CALLTYPE=RL. If CALLTYPE=RL, the host field can be joined to the high-order portion of a descriptor, superdescriptor, or subdescriptor, if the high-order portion is longer than the host field.

- ❑ In CA-IDMS/DB data sources, the field must be an indexed field on a network record identified by the attribute FIELDTYPE=I in the Master File, a CA-IDMS/DB CALC field on a network record identified by the CLCFLD phrase in the Access File, or any field on an LRF or ASF record.
- ❑ For a partial key join using fixed format or delimited sequential files, the setting ENGINE INT CACHE SET OFF must be in effect.

### **Reference: Usage Notes for Inner and Outer JOIN Command Syntax**

- ❑ The SET ALL and SET CARTESIAN commands are ignored by the syntax.
- ❑ The ALL. parameter is not supported. If the ALL. parameter is used, the following message displays:

(FOC32452) Use of ALL. with LEFT\_OUTER/INNER not allowed

- ❑ If you define multiple joins, the resulting structure can be a single path or multi-path data source.
- ❑ If the setting MULTIPATH=SIMPLE is in effect and the report is based on multiple paths, each of the individual joins is constructed separately without regard to the other joins, and the matching records from each of the separate paths displays on the report output. Therefore, the output can contain records that would have been omitted if only one of the joins was in effect.
- ❑ If the setting MULTIPATH=COMPOUND is in effect with a multi-path report, or if the report displays data only from a single path, the report output displays only those records that satisfy all of the joins.

## Joining From a Virtual Field to a Real Field Using an Equijoin

You can use DEFINE-based JOIN syntax to create a virtual host field that you can join to a real cross-referenced field. The DEFINE expression that creates the virtual host field may contain only fields in the host file and constants. (It may not contain fields in the cross-referenced file.) You can do more than one join from a virtual field.

You can create the virtual host field in a separate DEFINE command or in a Master File. For information on Master Files, see the *Describing Data With WebFOCUS Language* manual.

The same report request can use JOIN-based virtual fields, and virtual fields unrelated to the join.

Note that if you are creating a virtual field in a DEFINE command, you must issue the DEFINE after the JOIN command, but before the TABLE request since a JOIN command clears all fields created by DEFINE commands for the host file and the joined structure. Virtual fields defined in Master Files are not cleared.

**Tip:** If a DEFINE command precedes the JOIN command, you can set KEEPDEFINES ON to reinstate virtual fields during the parsing of a subsequent JOIN command. For more information, see [Preserving Virtual Fields Using KEEPDEFINES](#) on page 1065.

**Syntax:**      **How to Join From a Virtual Field to a Real Field**

The DEFINE-based JOIN command enables you to join a virtual field in the host file to a real field in the cross-referenced file. The syntax is:

```
JOIN [LEFT_OUTER|INNER] deffld WITH host_field ...  
    IN hostfile [TAG tag1]  
    TO [UNIQUE|MULTIPLE]  
    cr_field IN crfile [TAG tag2] [AS joinname]  
END
```

where:

**JOIN** *deffld*

Is the name of a virtual field for the host file (the host field). The virtual field can be defined in the Master File or with a DEFINE command. For related information, see [Notes on Using Virtual Fields With Joined Data Sources](#) on page 1030.

**WITH** *host\_field*

Is the name of any real field in the host segment with which you want to associate the virtual field. This association is required to locate the virtual field.

**Note:** The WITH field referenced in the JOIN command must be in the same segment as the WITH field referenced in the DEFINE that creates the virtual field or no output will be produced.

The WITH phrase is required unless the KEEPDEFINES parameter is set to ON and *deffld* was defined prior to issuing the JOIN command.

To determine which segment contains the virtual field, use the ? DEFINE query after issuing the DEFINE command. See the *Developing Reporting Applications* manual for details about Query commands.

INNER

Specifies an inner join. If you do not specify the type of join in the JOIN command, the ALL parameter setting determines the type of join to perform.

**LEFT\_OUTER**

Specifies a left outer join. If you do not specify the type of join in the JOIN command, the ALL parameter setting determines the type of join to perform.

**IN** *hostfile*

Is the name of the host file.

**TAG** *tag1*

Is a tag name of up to 66 characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in host files.

The tag name for the host file must be the same in all JOIN commands of a joined structure.



**TO** `[UNIQUE|MULTIPLE] crfld1`

Is the name of a real field in the cross-referenced data source whose values match those of the virtual field. This must be a real field declared in the Master File.

**Note:** UNIQUE returns only one instance and, if there is no matching instance in the cross-referenced file, it supplies default values (blank for alphanumeric fields and zero for numeric fields).

Use the MULTIPLE parameter when *crfld1* may have multiple instances in common with one value in *hfld1*. Note that ALL is a synonym for MULTIPLE, and omitting this parameter entirely is a synonym for UNIQUE. See [Unique and Non-Unique Joined Structures](#) on page 1008 for more information.

**IN** `crfile`

Is the name of the cross-referenced file.

**TAG** `tag2`

Is a tag name of up to 66 characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in cross-referenced files. In a recursive joined structure, if no tag name is provided, all field names and aliases are prefixed with the first four characters of the join name. For related information, see [Unique and Non-Unique Joined Structures](#) on page 1008.

The tag name for the host file must be the same in all JOIN commands of a joined structure.

**AS** `joinname`

Is an optional name of up to eight characters that you may assign to the joined structure. You must assign a unique name to a join structure if:

- ☐ You want to ensure that a subsequent JOIN command does not overwrite it.
- ☐ You want to clear it selectively later.
- ☐ The structure is recursive, and you do not specify tag names. See [Recursive Joined Structures](#) on page 1012.

If you do not assign a name to the joined structure with the AS phrase, the name is assumed to be blank. A join without a name overwrites an existing join without a name.

**END**

Required when the JOIN command is longer than one line. It terminates the command. It must be on a line by itself.

### **Reference:** Notes on Using Virtual Fields With Joined Data Sources

Requests reading joined data sources can contain virtual fields that are defined either:

- ❑ In the Master File of the host data source.
- ❑ In a DEFINE command, in which the syntax

```
DEFINE FILE hostfile
```

identifies the host data source in the joined structure.

**Note:** The expression defining the host field for the join can use only host fields and constants.

All other virtual fields can contain real fields from the host file and the cross-referenced file.

**Tip:** Since issuing the JOIN command clears all DEFINE commands for the host file and the joined structure, you must issue the DEFINE command after the JOIN or turn KEEPDEFINES ON to preserve the virtual fields. For more information, see [Preserving Virtual Fields During Join Parsing](#) on page 1064.

### **Example:** Creating a Virtual Host Field for a Joined Structure

Suppose that a retail chain sends four store managers to attend classes. Each person, identified by an ID number, manages a store in a different city. The stores and the cities in which they are located are contained in the SALES data source. The manager IDs, the classes, and dates the managers attended are contained in the EDUCFILE data source.

The following procedure lists the courses that the managers attended, identifying the managers by the cities in which they work. Note the three elements in the procedure:

- ❑ The JOIN command joins the SALES data source to the EDUCFILE data source, based on the values common to the ID\_NUM field (which contains manager IDs) in SALES and the EMP\_ID field in EDUCFILE. Note that the ID\_NUM field does not exist yet and will be created by the DEFINE command.
- ❑ The DEFINE command creates the ID\_NUM field, assigning to it the IDs of the managers working in the four cities.
- ❑ The TABLE command produces the report.

The procedure is:

```
JOIN ID_NUM WITH CITY IN SALES TO ALL EMP_ID IN EDUCFILE AS SALEDUC

DEFINE FILE SALES
ID_NUM/A9 = DECODE CITY ('NEW YORK' 451123478 'NEWARK' 119265415
                        'STAMFORD' 818692173 'UNIONDALE' 112847612);
END

TABLE FILE SALES
PRINT DATE_ATTEND BY CITY BY COURSE_NAME
END
```

The output is:

CITY	COURSE_NAME	DATE_ATTEND
----	-----	-----
NEW YORK	FILE DESCRIPT & MAINT	81/11/15
NEWARK	BASIC RPT NON-DP MGRS	82/08/24
STAMFORD	BASIC REPORT PREP DP MGRS	82/08/02
	HOST LANGUAGE INTERFACE	82/10/21
UNIONDALE	BASIC REPORT PREP FOR PROG	81/11/16
	FILE DESCRIPT & MAINT	81/11/15

## Join Modes in an Equijoin

The JOIN\_LENGTH\_MODE (JOINLM) parameter controls processing of equality joined field pairs for record-based non-SQL Adapters, such as DFIX, VSAM, and FIX. This setting controls processing when two alphanumeric fields of different lengths or two numeric fields of different data types and precisions are joined.

For SQL data sources, joins are normally either optimized (sent to the SQL engine for processing) or managed to comply with SQL processing rules.

There are two supported modes of handling compatible, but not identical, joined fields:

- ☐ **SQL compliance mode.** The JOIN command processor assures strict value equality of joined fields. Detected truncation of significant characters during host to cross-referenced conversion generates *atarget not found* condition, in which case the join is not done. If a shorter host field is joined to a longer cross-referenced file, the shorter host field value is extended to the length of the cross-referenced field with non-significant characters, according to the data type, and the join is processed.
- ☐ **FOCUS reporting mode.** The JOIN command processor assures partial value equality of joined fields.
  - ☐ When joining a shorter to a longer field, a search range is created to find all cross-referenced values that are prefixed with the host value.

- ❑ When joining a longer to a shorter field, the host value is unconditionally truncated to the cross referenced field length. If the truncation removes non-blank characters, the match will not be done and the comparison will fail, rejecting the records.

### **Syntax:**      **How to Control the Join Mode for Record-Based Data Sources**

```
SET JOIN_LENGTH_MODE {JOINLM} = {SQL|RANGE}
```

where:

#### [SQL](#)

Sets SQL compliant mode. which assures strict equality between host and cross-referenced field values. This is the default value.

#### [RANGE](#)

Sets FOCUS reporting mode, which supports partial key joins.

### **Data Formats of Shared Fields**

Generally, the fields containing the shared values in the host and cross-referenced files must have the same data formats.

If you specify multiple host file fields, the JOIN command treats the fields as one concatenated field. Add the field format lengths to obtain the length of the concatenated field. You must observe the following rules:

- ❑ If the host field is alphanumeric, the cross-referenced field must also be alphanumeric and have the same length.

The formats may have different edit options.

Note that a text field cannot be used to join data sources.

- ❑ If the host field is a numeric field, the host field format, as specified by the USAGE (or FORMAT) attribute in the Master File, must agree in type (I, P, F, or D) with the format of the cross-referenced field as specified by the USAGE (or FORMAT) attribute. For details, see [Joining Fields With Different Numeric Data Types](#) on page 1033.

The edit options may differ. The length may also differ, but with the following effect:

- ❑ If the format of the host field (as specified by the USAGE attribute) is packed decimal (P) or integer (I) and is longer than the cross-referenced field format (specified by the USAGE attribute for FOCUS data sources or the ACTUAL attribute for other data sources), only the length of the cross-referenced field format is compared, using only the right-most digits of the shorter field. For example, if a five-digit packed decimal format field is joined to a three-digit packed decimal format field, when a host record with a five-digit number is retrieved, all cross-referenced records with the last three digits of that number are also retrieved.
- ❑ If the format of the host field is double precision (D), the left-most eight bytes of each field are compared.
- ❑ If the host field is a date field, the cross-referenced field must also be a date field. Date and date-time fields must have the same components, not necessarily in the same order.
- ❑ The host and cross-referenced fields can be described as groups in the Master File if they contain the same number of component fields. The corresponding component fields in each group (for example, the first field in the host group and the first field in the cross-referenced group) must obey the above rules. For related information, see [Restrictions on Group Fields](#) on page 1026.

If the host field is not a group field, the cross-referenced field can still be a group. If the host field is a group, the cross-referenced field must also be a group.

## Joining Fields With Different Numeric Data Types

You can join two or more data sources containing different numeric data types. For example, you can join a field with a short packed decimal format to a field with a long packed decimal format, or a field with an integer format to a field with a packed decimal format. This provides enormous flexibility for creating reports from joined data sources.

- ❑ When joining a shorter field to a longer field, the cross-referenced value is padded to the length of the host field, adding spaces (for alpha fields) or hexadecimal zeros (for numeric fields). This new value is used for searches in the cross-referenced file.
- ❑ When joining a longer field to a shorter field, the FROM value is truncated. If part of your value is truncated due to the length of the USAGE in the cross-referenced file, only records matching the truncated value will be found in the cross-referenced file.

### Note:

- ❑ For comparison on packed decimal fields to be accomplished properly, all signs for positive values are converted to hex C and all signs for negative values are converted to hex D.

- ❑ The JOINOPT parameter also corrects for lagging values in a unique join. For information, see [How to Correct for Lagging Values With a Unique Join](#) on page 1010.

### **Syntax:**      **How to Enable Joins With Data Type Conversion**

To enable joins with data type conversion, issue the command

```
SET JOINOPT = [GNTINT|OLD]
```

where:

[GNTINT](#)

Enables joins with data type conversion.

[OLD](#)

Disables joins with data type conversion. This value is the default.

### **Example:**      **Issuing Joins With Data Type Conversion**

Since you can join a field with a short packed decimal format to a field with a long packed decimal format, a join can be defined in the following Master Files:

```
FILE=PACKED,SUFFIX=FIX,$
  SEGNAME=ONE,SEGTYPE=S0
  FIELD=FIRST,,P8,P4,INDEX=I,$

FILE=PACKED2,SUFFIX=FIX,$
  SEGNAME=ONE,SEGTYPE=S0
  FIELD=PFIRST,,P31,P16,INDEX=I,$
```

The JOIN command might look like this:

```
JOIN FIRST IN PACKED TO ALL PFIRST IN PACKED2 AS J1
```

When joining packed fields, the preferred sign format of X'C' for positive values and X'D' for negative values is still required. All other non-preferred signs are converted to either X'C' or X'D'.

## **Using a Conditional Join**

Using conditional JOIN syntax, you can establish joins based on conditions other than equality between fields. In addition, the host and cross-referenced join fields do not have to contain matching formats, and the cross-referenced field does not have to be indexed.

The conditional join is supported for FOCUS and for VSAM, ADABAS, IMS, IDMS, and all relational data sources. Because each data source differs in its ability to handle complex WHERE criteria, the optimization of the conditional JOIN syntax differs depending on the specific data sources involved in the join and the complexity of the WHERE criteria.

The standard ? JOIN command lists every join currently in effect, and indicates any that are based on WHERE criteria.

### **Syntax:**      **How to Create a Conditional JOIN**

The syntax of the conditional (WHERE-based) JOIN command is

```
JOIN [LEFT_OUTER|INNER] FILE hostfile AT hfld1      [WITH hfld2] [TAG tag1]
  TO {UNIQUE|MULTIPLE}
  FILE crfile AT crfld [TAG tag2] [AS joinname]
  [WHERE expression1;
   [WHERE expression2;
   ...]
END
```

where:

#### INNER

Specifies an inner join. If you do not specify the type of join in the JOIN command, the ALL parameter setting determines the type of join to perform.

#### LEFT\_OUTER

Specifies a left outer join. If you do not specify the type of join in the JOIN command, the ALL parameter setting determines the type of join to perform.

#### *hostfile*

Is the host Master File.

#### AT

Links the correct parent segment or host to the correct child or cross-referenced segment. The field values used as the AT parameter are not used to cause the link. They are simply used as segment references.

#### *hfld1*

Is a field name in the host Master File whose segment will be joined to the cross-referenced data source. The field name must be at the lowest level segment in its data source that is referenced.

#### *tag1*

Is the optional tag name of up to 66 characters that is used as a unique qualifier for fields and aliases in the host data source.

*hfld2*

Is a data source field with which to associate a DEFINE-based conditional JOIN. For a DEFINE-based conditional join, the KEEPDEFINES setting must be ON, and you must create the virtual fields before issuing the JOIN command.

**MULTIPLE**

Specifies a one-to-many relationship between *hostfile* and *crfile*. Note that ALL is a synonym for MULTIPLE.

**UNIQUE**

Specifies a one-to-one relationship between *hostfile* and *crfile*. Note that ONE is a synonym for UNIQUE.

**Note:** Regardless of the character of the JOIN—INNER or LEFT\_OUTER—the join to UNIQUE will return only one instance of the cross-referenced file, and if this instance does not match based on the evaluation of the WHERE expression, default values (spaces for alphanumeric fields and 0 for numerical fields) are returned. There are never short paths or missing values in the cross-referenced file.

*crfile*

Is the cross-referenced Master File.

*crfld*

Is a field name in the cross-referenced Master File. It can be any field in the segment.

*tag2*

Is the optional tag name of up to 66 characters that is used as a unique qualifier for fields and aliases in the cross-referenced data source.

*joinname*

Is the name associated with the joined structure.

*expression1, expression2*

Are any expressions that are acceptable in a DEFINE FILE command. All fields used in the expressions must lie on a single path.

You must include the connection between the tables in the WHERE conditions. The AT references do not actually perform a JOIN between the fields as with a standard JOIN.

If you do not include any WHERE conditions in the join, a cartesian product is generated.

**END**

The END command is required to terminate the command and must be on a line by itself.



**Note:** Single line JOIN syntax is not supported.

**Example:** **Using a Conditional Join**

The following example joins the VIDEOTRK and MOVIES data sources on the conditions that:

- ☐ The transaction date (in VIDEOTRK) is more than ten years after the release date (in MOVIES).
- ☐ The movie codes match in both data sources.

The join is performed at the segment that contains MOVIECODE in the VIDEOTRK data source, because the join must occur at the lowest segment referenced.

The following request displays the title, most recent transaction date, and release date for each movie in the join, and computes the number of years between this transaction date and the release date:

```
JOIN FILE VIDEOTRK AT MOVIECODE TAG V1 TO ALL
      FILE MOVIES   AT RELDATE   TAG M1 AS JW1
      WHERE DATEDIF(RELDATE, TRANSDATE, 'Y') GT 10;
      WHERE V1.MOVIECODE EQ M1.MOVIECODE;
END
TABLE FILE VIDEOTRK
  SUM TITLE/A25 AS 'Title'
      TRANSDATE AS 'Last,Transaction'
      RELDATE AS 'Release,Date'
  COMPUTE YEARS/I5 = (TRANSDATE - RELDATE)/365; AS 'Years,Difference'
  BY TITLE NOPRINT
  BY HIGHEST 1 TRANSDATE NOPRINT
END
```

The output is:

Title	Last Transaction	Release Date	Years Difference
-----	-----	-----	-----
ALICE IN WONDERLAND	91/06/22	51/07/21	39
ALIEN	91/06/18	80/04/04	11
ALL THAT JAZZ	91/06/25	80/05/11	11
ANNIE HALL	91/06/24	78/04/16	13
BAMBI	91/06/22	42/07/03	49
BIRDS, THE	91/06/23	63/09/27	27
CABARET	91/06/25	73/07/14	17
CASABLANCA	91/06/27	42/03/28	49
CITIZEN KANE	91/06/22	41/08/11	49
CYRANO DE BERGERAC	91/06/20	50/11/09	40
DEATH IN VENICE	91/06/26	73/07/27	17
DOG DAY AFTERNOON	91/06/23	76/04/04	15
EAST OF EDEN	91/06/20	55/01/12	36
GONE WITH THE WIND	91/06/24	39/06/04	52
JAWS	91/06/27	78/05/13	13
MALTESE FALCON, THE	91/06/19	41/11/14	49
MARTY	91/06/19	55/10/26	35
NORTH BY NORTHWEST	91/06/21	59/02/09	32
ON THE WATERFRONT	91/06/24	54/07/06	36
PHILADELPHIA STORY, THE	91/06/21	40/05/06	51
PSYCHO	91/06/17	60/05/16	31
REAR WINDOW	91/06/17	54/12/15	36
SHAGGY DOG, THE	91/06/25	59/01/09	32
SLEEPING BEAUTY	91/06/24	75/08/30	15
TIN DRUM, THE	91/06/17	80/03/01	11
VERTIGO	91/06/27	58/11/25	32

Full Outer Joins for Relational Data Sources

The WebFOCUS join command and conditional join command have a FULL OUTER join option.

A full outer join returns all rows from the source data source and all rows from the target data source. Where values do not exist for the rows in either data source, null values are returned. WebFOCUS substitutes default values on the report output (blanks for alphanumeric columns, the NODATA symbol for numeric columns).

The full outer join is only supported for use with those relational data sources that support this type of join, in which case the WebFOCUS join syntax is optimized (translated to the full outer join SQL syntax supported by the RDBMS). Use of this syntax for any data source that does not support a full outer join, or the failure of the request to be optimized to the engine, produces an error message.

*Syntax:*      **How to Specify a Full Outer Join**

The following syntax generates a full outer equijoin based on real fields:

```
JOIN FULL_OUTER hfld1 [AND hfld2 ...] IN table1 [TAG tag1] TO {UNIQUE|
MULTIPLE} crfld [AND crfld2 ...] IN table2 [TAG tag2] [AS joinname]
END
```

where:

*hfld1*

Is the name of a field in the host table containing values shared with a field in the cross-referenced table. This field is called the host field.

AND *hfld2*...

Can be an additional field in the host table. The phrase beginning with AND is required when specifying multiple fields.

- ❑ For relational adapters that support multi-field and concatenated joins, you can specify up to 16 fields. See your adapter documentation for specific information about supported join features.

IN *table1*

Is the name of the host table.

TAG *tag1*

Is a tag name of up to 66 characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in the host table.

The tag name for the host table must be the same in all the JOIN commands of a joined structure.

TO [UNIQUE|MULTIPLE] *crfld1*

Is the name of a field in the cross-referenced table containing values that match those of *hfld1* (or of concatenated host fields). This field is called the cross-referenced field.

**Note:** UNIQUE returns only one instance and, if there is no matching instance in the cross-referenced table, it returns null values.

Use the MULTIPLE parameter when *crfld1* may have multiple instances in common with one value in *hfld1*. Note that ALL is a synonym for MULTIPLE, and omitting this parameter entirely is a synonym for UNIQUE.

AND *crfld2*...

Is the name of a field in the cross-referenced table with values in common with *hfld2*.

**Note:** *crfld2* may be qualified. This field is only available for adapters that support multi-field joins.

*IN crfile*

Is the name of the cross-referenced table.

*TAG tag2*

Is a tag name of up to 66 characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in cross-referenced tables. In a recursive join structure, if no tag name is provided, all field names and aliases are prefixed with the first four characters of the join name.

The tag name for the host table must be the same in all the JOIN commands of a joined structure.

*AS joinname*

Is an optional name of up to eight characters that you may assign to the join structure. You must assign a unique name to a join structure if:

- ☐ You want to ensure that a subsequent JOIN command does not overwrite it.
- ☐ You want to clear it selectively later.
- ☐ The structure is recursive.

**Note:** If you do not assign a name to the join structure with the AS phrase, the name is assumed to be blank. A join without a name overwrites an existing join without a name.

*END*

Required when the JOIN command is longer than one line. It terminates the command and must be on a line by itself.

The following syntax generates a DEFINE-based full outer join:

```
JOIN FULL_OUTER deffld WITH host_field ...  
  IN table1 [TAG tag1]  
  TO [UNIQUE|MULTIPLE]  
  cr_field IN table2 [TAG tag2] [AS joinname]  
END
```

where:

*deffld*

Is the name of a virtual field for the host file (the host field). The virtual field can be defined in the Master File or with a DEFINE command.

*WITH host\_field*

Is the name of any real field in the host segment with which you want to associate the virtual field. This association is required to locate the virtual field.

The WITH phrase is required unless the KEEPDEFINES parameter is set to ON and *deffld* was defined prior to issuing the JOIN command.

To determine which segment contains the virtual field, use the ? DEFINE query after issuing the DEFINE command.

**IN** *table1*

Is the name of the host table.

**TAG** *tag1*

Is a tag name of up to 66 characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in host tables.

The tag name for the host table must be the same in all JOIN commands of a joined structure.

**TO** [**UNIQUE**|**MULTIPLE**] *crfld1*

Is the name of a real field in the cross-referenced table whose values match those of the virtual field. This must be a real field declared in the Master File.

**Note:** UNIQUE returns only one instance and, if there is no matching instance in the cross-referenced table, it returns null values.

Use the MULTIPLE parameter when *crfld1* may have multiple instances in common with one value in *hfld1*. Note that ALL is a synonym for MULTIPLE, and omitting this parameter entirely is a synonym for UNIQUE.

**IN** *crfile*

Is the name of the cross-referenced table.

**TAG** *tag2*

Is a tag name of up to 66 characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in cross-referenced tables. In a recursive joined structure, if no tag name is provided, all field names and aliases are prefixed with the first four characters of the join name.

The tag name for the host file must be the same in all JOIN commands of a joined structure.

**AS** *joinname*

Is an optional name of up to eight characters that you may assign to the joined structure. You must assign a unique name to a join structure if:

- ☐ You want to ensure that a subsequent JOIN command does not overwrite it.

- ☐ You want to clear it selectively later.
- ☐ The structure is recursive, and you do not specify tag names.

If you do not assign a name to the joined structure with the AS phrase, the name is assumed to be blank. A join without a name overwrites an existing join without a name.

END

Required when the JOIN command is longer than one line. It terminates the command and must be on a line by itself.

The following syntax generates a full outer conditional join:

```
JOIN FULL_OUTER FILE table1 AT hfld1 [WITH hfld2] [TAG tag1]  
  TO {UNIQUE|MULTIPLE}  
  FILE table2 AT crfld [TAG tag2] [AS joinname]  
  [WHERE expression1;  
  [WHERE expression2;  
  ...]  
END
```

where:

*table1*

Is the host Master File.

AT

Links the correct parent segment or host to the correct child or cross-referenced segment. The field values used as the AT parameter are not used to cause the link. They are used as segment references.

*hfld1*

Is the field name in the host Master File whose segment will be joined to the cross-referenced table. The field name must be at the lowest level segment in its data source that is referenced.

*tag1*

Is the optional tag name that is used as a unique qualifier for fields and aliases in the host table.

*hfld2*

Is a table column with which to associate a DEFINE-based conditional JOIN. For a DEFINE-based conditional join, the KEEPDEFINES setting must be ON, and you must create the virtual fields before issuing the JOIN command.

**MULTIPLE**

Specifies a one-to-many relationship between *table1* and *table2*. Note that ALL is a synonym for MULTIPLE.

**UNIQUE**

Specifies a one-to-one relationship between *table1* and *table2*. Note that ONE is a synonym for UNIQUE.

**Note:** The join to UNIQUE will return only one instance of the cross-referenced table, and if this instance does not match based on the evaluation of the WHERE expression, null values are returned.

*crfile*

Is the cross-referenced Master File.

*crfld*

Is the join field name in the cross-referenced Master File. It can be any field in the segment.

*tag2*

Is the optional tag name that is used as a unique qualifier for fields and aliases in the cross-referenced table.

*joinname*

Is the name associated with the joined structure.

*expression1, expression2*

Are any expressions that are acceptable in a DEFINE FILE command. All fields used in the expressions must lie on a single path.

**END**

The END command is required to terminate the command and must be on a line by itself.

**Example: Optimizing a Full Outer Join of Microsoft SQL Server Tables**

The following requests generate two Microsoft SQL Server tables to join, and then issues a request against the join. The tables are generated using the wf\_retail sample that you can create using the WebFOCUS - Retail Demo tutorial in the server Web Console.

The following request generates the WF\_SALES table. The field ID\_PRODUCT will be used in the full outer join command. The generated table will contain ID\_PRODUCT values from 2150 to 4000:

```
TABLE FILE WF_RETAIL_LITE
SUM GROSS_PROFIT_US PRODUCT_CATEGORY PRODUCT_SUBCATEG
BY ID_PRODUCT
WHERE ID_PRODUCT FROM 2150 TO 4000
ON TABLE HOLD AS WF_SALES FORMAT SQLMSS
END
```

The following request generates the WF\_PRODUCT table. The field ID\_PRODUCT will be used in the full outer join command. The generated table will contain ID\_PRODUCT values from 3000 to 5000:

```
TABLE FILE WF_RETAIL_LITE
SUM PRICE_DOLLARS PRODUCT_CATEGORY PRODUCT_SUBCATEG PRODUCT_NAME
BY ID_PRODUCT
WHERE ID_PRODUCT FROM 3000 TO 5000
ON TABLE HOLD AS WF_PRODUCT FORMAT SQLMSS
END
```

The following request issues the JOIN command and displays values from the joined tables:

```
SET TRACEUSER=ON
SET TRACESTAMP=OFF
SET TRACEOFF=ALL
SET TRACEON = STMTRACE//CLIENT
JOIN FULL_OUTER ID_PRODUCT IN WF_PRODUCT TAG T1 TO ALL ID_PRODUCT IN
WF_SALES TAG T2
END
TABLE FILE WF_PRODUCT
PRINT T1.ID_PRODUCT AS 'Product ID'
PRICE_DOLLARS AS Price
T2.ID_PRODUCT AS 'Sales ID'
GROSS_PROFIT_US
BY T1.ID_PRODUCT NOPRINT
ON TABLE SET PAGE NOPAGE
END
```

The trace shows that the full outer join was optimized (translated to SQL) so that SQL Server could process the join:

```
SELECT
T1."ID_PRODUCT",
T1."PRICE_DOLLARS",
T2."ID_PRODUCT",
T2."GROSS_PROFIT_US"
FROM
( WF_PRODUCT T1
FULL OUTER JOIN WF_SALES T2
ON T2."ID_PRODUCT" = T1."ID_PRODUCT" )
ORDER BY
T1."ID_PRODUCT" ;
```



The output has a row for each ID\_PRODUCT value that is in either table. Rows with ID\_PRODUCT values from 2150 to 2167 are only in the WF\_SALES table, so the columns from WF\_PRODUCT display the NODATA symbol. Rows with ID\_PRODUCT values above 4000 are only in the WF\_PRODUCT table, so the columns from WF\_SALES display the NODATA symbol. Rows with ID\_PRODUCT values from 2000 to 4000 are in both tables, so all columns have values, as shown in the following image.

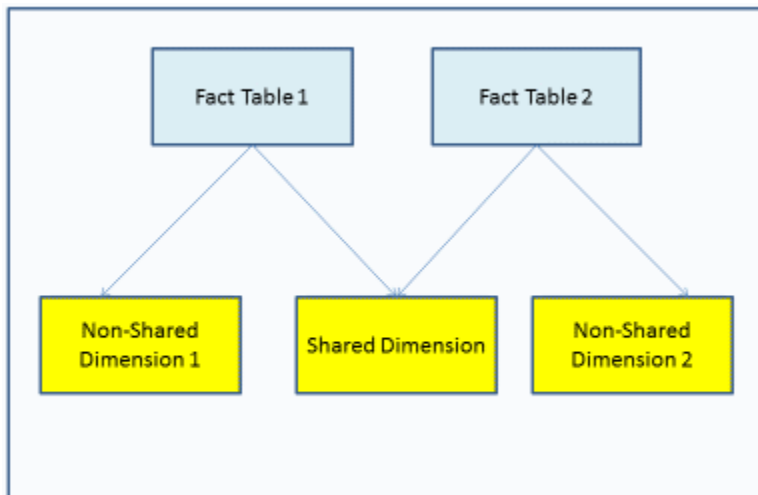
Product ID	Price	Sales ID	Gross Profit
.	.	2151	\$669.81
.	.	2152	\$238.15
.	.	2153	\$837.65
.	.	2154	\$1,929.30
.	.	2155	\$1,019.84
.	.	2156	\$1,596.00
.	.	2157	\$1,109.71
.	.	2158	\$826.10
.	.	2160	\$218.40
.	.	2161	\$819.15
.	.	2162	\$997.77
.	.	2163	\$2,004.80
.	.	2164	\$579.80
.	.	2165	\$891.66
.	.	2166	\$1,051.79
.	.	2167	\$351.96
3003	6,798.00	3003	\$1,986.00
3005	2,097.00	3005	\$527.10
3006	1,299.90	3006	\$61.91
3007	3,843.00	3007	\$1,490.20
3008	5,699.81	3008	\$2,677.72
3009	4,186.00	3009	\$2,145.55
3010	2,392.00	3010	\$764.90
3011	3,239.82	3011	\$1,835.77
3012	2,327.00	3012	\$1,429.75
3013	1,790.00	3013	\$1,358.00
3014	2,039.88	3014	\$2,021.81
3015	3,399.00	3015	\$2,080.62
4012	3,999.85	.	.
4014	3,996.00	.	.
4016	1,399.96	.	.
4017	7,199.91	.	.
4018	4,990.00	.	.
5000	1,399.95	.	.

## Reporting Against a Multi-Fact Cluster Synonym

A cluster synonym is a synonym in which each segment is added to the cluster by reference using a CRFILE attribute that points to the base synonym. Child segments are joined to their parents using a JOIN WHERE attribute. A cluster Master File can have multiple root segments. In this case, the root segments are usually fact tables and the child segments are usually dimension tables, as found in a star schema. This type of structure is called a multi-fact cluster.

A dimension table can be a child of multiple fact tables (called a shared dimension) or be a child of a single fact table (called a non-shared dimension). In most cases, the fact tables are used for aggregation and the dimension tables are used for sorting.

The following image shows a simple multi-fact structure.



For information about creating a multi-fact cluster Master File, see the *Describing Data With WebFOCUS Language* manual.

The following list shows the rules for creating a report request against a multi-fact cluster Master File.

- ☐ You can report against only the fact tables.
- ☐ The first sort field in the request must be from a shared dimension.
- ☐ Any number of shared dimensions can be referenced in the request.
- ☐ Multiple non-shared dimensions can be included in the request, as long as they have the same parent. More than one non-shared dimension from different parents cannot be referenced in a request.

- ❑ The MATCH FILE command is not supported for reporting against a multi-fact synonym.

### **Example: Reporting Against a Multi-Fact Cluster Synonym**

The following request against the WF\_RETAIL\_LITE multi-fact cluster synonym sums the COGS\_US measure from the WF\_RETAIL\_SALES segment and the DAYSDELAYED measure from the WF\_RETAIL\_SHIPMENTS segment. The first BY field, BRAND, is in the shared dimension WF\_RETAIL\_PRODUCT. The second BY field, TIME\_QTR, is from the non-shared dimension WF\_RETAIL\_TIME\_DELIVERED.

```
TABLE FILE WF_RETAIL_LITE
SUM COGS_US DAYSDELAYED
BY BRAND
BY WF_RETAIL_TIME_DELIVERED.TIME_QTR
WHERE BRAND EQ 'Denon' OR 'Grado'
WHERE DAYSDELAYED GT 1
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
END
```

The output is shown in the following image. The sum of DAYSDELAYED is totaled for each value of the shared dimension and, within each value of the shared dimension, for each value of the non-shared dimension.

Brand	Cost of Goods	Quarter	Days Delayed
Denon	\$25,970.00		12
		1	3
		2	4
		3	5
Grado	\$21,930.00		5
		2	5

### **Adding a New Fact To Multi-Fact Synonyms: JOIN AS\_ROOT**

The JOIN AS\_ROOT command adds a new fact table as an additional root to an existing fact-based cluster (star schema). The source Master File has a parent fact segment and at least one child dimension segment. The JOIN AS\_ROOT command supports a unique join from a child dimension segment (at any level) to an additional fact parent.

**Syntax:**      **How to Add an Additional Parent Segment**

```
JOIN AS_ROOT sfld1 [AND sfld2 ...] IN [app1/]sfile TO UNIQUE tfld1 [AND  
tfld2 ...] IN [app2/]tfile AS jname  
END
```

where:

*sfld1* [AND *sfld2* ...]

Are fields in the child (dimension) segment of the source file that match values of fields in the target file.

[*app1/*]*sfile*

Is the source file.

TO UNIQUE *tfld1* [AND *tfld2* ...]

Are fields in the target file that match values of fields in the child segment of the source file. The join must be unique.

[*app2/*]*tfile*

Is the target file.

*jname*

Is the join name.

END

Is required to end the JOIN command.

**Example:**      **Joining AS\_ROOT From the WebFOCUS Retail Data Source to an Excel File**

The following request joins the product category and product subcategory fields in the WebFOCUS Retail data source to an Excel file named PROJECTED.

To generate the WebFOCUS Retail data source in the Web Console, click *Tutorials* from the Applications page.

### Create Tutorial Framework

1 Tutorial	WebFOCUS - Retail Demo
2 Select DBMS:	MS SQL Server
3 Select Connection:	wf_retail
4 Prefix for SQL Tables	wrd wrd or mylib/wrd or xxx.wrd
5 Data Security (DBA)	None
<input checked="" type="checkbox"/> Limit Tutorial Data (from 285M to 9M)	
6 Application	wfretail
<input type="button" value="Create"/>	

Select *WebFOCUS - Retail Demo*. Select your configured relational adapter (or select the flat file option if you do not have a relational adapter configured), check *Limit Tutorial Data*, and then click *Create*.

The Master File for the Excel File is:

```
FILENAME=PROJECTED, SUFFIX=DIREXCEL,
  DATASET=app2/projected.xlsx, $
  SEGMENT=PROJECTED, SEGTYPE=S0, $
    FIELDNAME=PRODUCT_CATEGORY, ALIAS='Product Category', USAGE=A16V,
  ACTUAL=A16V,
    MISSING=ON,
    TITLE='Product Category',
    WITHIN='*PRODUCT', $
    FIELDNAME=PRODUCT_SUBCATEGORY, ALIAS='Product Subcategory',
  USAGE=A25V, ACTUAL=A25V,
    MISSING=ON,
    TITLE='Product Subcategory',
    WITHIN=PRODUCT_CATEGORY, $
    FIELDNAME=PROJECTED_COG, ALIAS='Projected COG',
  USAGE=P15.2C, ACTUAL=A15,
    MISSING=ON,
    TITLE='Projected COG', MEASURE_GROUP=PROJECTED,
    PROPERTY=MEASURE, $
    FIELDNAME=PROJECTED_SALE_UNITS, ALIAS='Projected Sale
Units', USAGE=I9, ACTUAL=A11,
    MISSING=ON,
    TITLE='Projected Sale Units', MEASURE_GROUP=PROJECTED,
    PROPERTY=MEASURE, $
  MEASUREGROUP=PROJECTED, CAPTION='PROJECTED', $
  DIMENSION=PRODUCT, CAPTION='Product', $
  HIERARCHY=PRODUCT, CAPTION='Product', HRY_DIMENSION=PRODUCT,
  HRY_STRUCTURE=STANDARD, $
```

The following image shows the data in the Excel file.

	A	B	C	D
1	Product Category	Product Subcategory	Projected COG	Projected Sale Units
2	Accessories	Charger	\$2,068,508.00	75,279
3	Accessories	Headphones	\$52,061,301.00	163,152
4	Accessories	Universal Remote Controls	\$36,297,267.00	127,286
5	Camcorder	Handheld	\$20,733,053.00	178,704
6	Camcorder	Professional	\$35,440,708.00	9,095
7	Camcorder	Standard	\$49,442,067.00	137,489
8	Computers	Smartphone	\$44,420,201.00	146,858
9	Computers	Tablet	\$26,047,885.00	105,053
10	Media Player	Blu Ray	\$182,459,862.00	485,131
11	Media Player	DVD Players	\$3,756,254.00	13,346
12	Media Player	DVD Players - Portable	\$306,576.00	3,981
13	Media Player	Streaming	\$5,108,342.00	48,630
14	Stereo Systems	Boom Box	\$840,373.00	6,687
15	Stereo Systems	Home Theater Systems	\$56,829,817.00	285,041
16	Stereo Systems	Receivers	\$40,620,030.00	107,537
17	Stereo Systems	Speaker Kits	\$81,962,756.00	174,036
18	Stereo Systems	iPod Docking Station	\$26,310,783.00	221,723
19	Televisions	CRT TV	\$1,928,416.00	3,268
20	Televisions	Flat Panel TV	\$59,540,624.00	66,119
21	Televisions	Portable TV	\$545,348.00	5,696
22	Video Production	Video Editing	\$40,380,803.00	142,594

The following request joins from the wf\_retail\_product segment of the wf\_retail data source to the excel file as a new root and reports from both parent segments:

```

JOIN AS_ROOT PRODUCT_CATEGORY AND PRODUCT_SUBCATEG IN WF_RETAIL
  TO UNIQUE PRODUCT_CATEGORY AND PRODUCT_SUBCATEGORY IN PROJECTED
  AS J1
END
TABLE FILE WF_RETAIL
SUM PROJECTED_SALE_UNITS REVENUE_US
BY PRODUCT_CATEGORY
ON TABLE SET PAGE NOPAGE
END

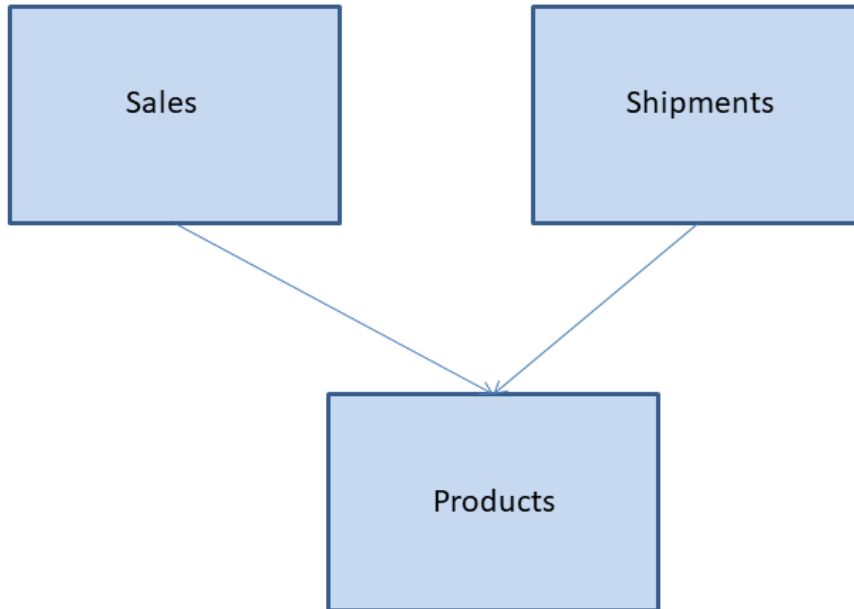
```

The output is:

Product Category	Projected Sale Units	Revenue
Accessories	365717	\$41,817.34
Camcorder	325288	\$49,768.33
Computers	251911	\$18,298.75
Media Player	551088	\$118,615.17
Stereo Systems	795024	\$111,768.71
Televisions	75083	\$27,209.29
Video Production	142594	\$25,837.37

## Generating Outer Joins of Cluster Synonym Contexts

Reporting against multiple root segments and a shared dimension generates multiple contexts in a cluster synonym. For example, in the following image Sales and Products form one context, while Shipments and Products form a second context.



When a request contains fields from both contexts, by default, an inner join is passed to the SQL engine. This retrieves only matching values of the shared dimension fields from both contexts.

You can use the `BLEND-MODE` parameter to generate a full outer join instead of an inner join and retrieve all values from both contexts.



**Syntax:**      **How to Control Join Processing of Cluster Synonym Contexts**

You can set the blend mode parameter from the server Web Console and store the setting in a profile or procedure. On the Adapters page, click *Change Common Adapter Settings* on the ribbon, and select *Select all values* from the BLEND-MODE drop-down list in the Request Transformation Settings section, as shown in the following image.

The screenshot shows the 'Request transformation settings' section of the Web Console. At the top, there are fields for 'Save settings in' (set to 'Profile') and 'Select' (set to 'edasprof'). Below this, there are two main sections: 'Customize data type mappings' and 'Request transformation settings'. The 'Request transformation settings' section contains several settings, including 'GEO\_UNIFIED\_ROLE', 'GEO\_MAP\_PROVIDER', 'GEO\_ALIASING', 'GEO\_ALIASING\_APP', 'GEO\_ALIASING\_FIPS', 'GEO\_COUNTRY', 'GEO\_ROLE\_2\_MBRLEVEL', 'FCA', 'BLEND-MODE', 'FCA-HOLD', and 'MFACT-OUTPUT'. The 'BLEND-MODE' setting is highlighted, showing a dropdown menu with three options: 'Select all values' (selected), 'Select only common values', and 'Select all values'. The description for 'BLEND-MODE' is 'Mode for processing blended structures'.

You can also use the following syntax to set the blend mode parameter.

```
ENGINE INT SET BLEND-MODE {COMMON-VALUES|ALL-VALUES}
```

where:

COMMON-VALUES

Generates an inner join of cluster synonym contexts and returns only matching values of the shared dimension fields. This is the default value.

ALL-VALUES

Generates a full outer join of cluster synonym contexts and returns all values of the shared dimension fields. Missing values are returned for fields from contexts that do not have a matching value of the shared dimension fields.

**Example: Controlling Join Processing of Cluster Synonym Contexts**

The following Excel file (excelroot.xlsx) will be uploaded to the server using the Adapter for Excel and joined as a root to the WF\_RETAIL Master File, creating two contexts. A report request will then be issued against the two roots and the shared dimension.

	A	B	C	D
1	Product Category	Product Subcategory	Projected COG	Projected Sale Units
2	Accessories	Charger	\$2,068,508.00	75,279
3	Accessories	Headphones	\$52,061,301.00	163,152
4	Accessories	Universal Remote Controls	\$36,297,267.00	127,286
5	Computers	Smartphone	\$44,420,201.00	146,858
6	Computers	Tablet	\$26,047,885.00	105,053
7	Media Player	Blu Ray	\$182,459,862.00	485,131
8	Media Player	DVD Players	\$3,756,254.00	13,346
9	Media Player	DVD Players - Portable	\$306,576.00	3,981
10	Media Player	Streaming	\$5,108,342.00	48,630
11	Displays	CRT TV	\$1,928,416.00	3,268
12	Televisions	Flat Panel TV	\$59,540,624.00	66,119
13	Televisions	Portable TV	\$545,348.00	142,594

Note that this file has no data for product categories Camcorder, Stereo Systems, and Video Production. It has a product category named Displays that does not exist in WF\_RETAIL.

The following is the Master File generated for this Excel file.

```

FILENAME=EXCELROOT, SUFFIX=DIREXCEL,
  DATASET=ibisamp/excelroot.xlsx, $
  SEGMENT=EXCELROOT, SEGTYPE=S0, $
    FIELDNAME=PRODUCT_CATEGORY, ALIAS='Product Category', USAGE=A15V,
ACTUAL=A15V,
  MISSING=ON,
  TITLE='Product Category', $
    FIELDNAME=PRODUCT_SUBCATEGORY, ALIAS='Product Subcategory', USAGE=A31V,
ACTUAL=A31V,
  MISSING=ON,
  TITLE='Product Subcategory', $
    FIELDNAME=PROJECTED_COG, ALIAS='Projected COG', USAGE=D15.2:C,
ACTUAL=A64V,
  MISSING=ON,
  TITLE='Projected COG',
  CURRENCY_DISPLAY=LEFT_FLOAT, CURRENCY_ISO_CODE=USD, $
    FIELDNAME=PROJECTED_SALE_UNITS, ALIAS='Projected Sale Units', USAGE=I9,
ACTUAL=A11V,
  MISSING=ON,
  TITLE='Projected Sale Units', $

```

The following request joins the Excel file as a root and generates a report that contains fields from both roots and the shared dimension. Using the default value for BLEND-MODE produces an inner join that returns only common values of PRODUCT\_CATEGORY.

```
JOIN AS_ROOT PRODUCT_CATEGORY AND PRODUCT_SUBCATEG IN ibisamp/WF_RETAIL
  TO PRODUCT_CATEGORY AND PRODUCT_SUBCATEGORY IN ibisamp/EXCELROOT
  AS J1
END
TABLE FILE ibisamp/WF_RETAIL
SUM COGS_US PROJECTED_SALE_UNITS
BY PRODUCT_CATEGORY
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
GRID=OFF,$
END
```

The output is shown in the following image.

<u>Product</u> <u>Category</u>	<u>Cost of Goods</u>	<u>Projected Sale Units</u>
Accessories	\$342,877.00	365717
Computers	\$109,281.00	251911
Media Player	\$779,593.00	551088
Televisions	\$227,820.00	208713

The following version of the request issues the ENGINE INT SET BLEND-MODE ALL-VALUES command to produce a full outer join that returns all values of PRODUCT\_CATEGORY.

```
ENGINE INT SET BLEND-MODE ALL-VALUES
JOIN AS_ROOT PRODUCT_CATEGORY AND PRODUCT_SUBCATEG IN ibisamp/WF_RETAIL
  TO PRODUCT_CATEGORY AND PRODUCT_SUBCATEGORY IN ibisamp/EXCELROOT
  AS J1
END
TABLE FILE ibisamp/WF_RETAIL
SUM COGS_US PROJECTED_SALE_UNITS
BY PRODUCT_CATEGORY
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
GRID=OFF,$
END
```

The output is shown in the following image. Note the missing value indicators:

- ❑ For the Projected Sale Units field in the rows that correspond to product categories Camcorder, Stereo Systems, and Video Production, which are not represented in the Excel file.

- ❑ For the Cost of Goods field in the row that corresponds to product category Displays, which is not represented in the WF\_RETAIL data source.

<u>Product Category</u>	<u>Cost of Goods</u>	<u>Projected Sale Units</u>
Accessories	\$89,753,898.00	1964918
Camcorder	\$104,866,857.00	.
Computers	\$69,807,664.00	2519110
Displays	.	3268
Media Player	\$190,240,481.00	10585039
Stereo Systems	\$205,113,863.00	.
Televisions	\$61,551,109.00	1033209
Video Production	\$40,105,657.00	.

## Joining From a Multi-Fact Synonym

Multi-parent synonyms are now supported as the source for a join to a single segment in a target synonym.

A join from a multi-parent synonym is subject to the following conditions:

- ❑ Conditional joins are not supported (JOIN WHERE).
- ❑ The join must be unique. That is, the TO ALL or TO MULTIPLE phrase is not supported.
- ❑ The target of the join cannot be a multi-parent synonym.
- ❑ The target of the JOIN must be a single segment, either in a single segment synonym or one segment in a single parent, multi-segment synonym.
- ❑ All fields in the JOIN must be FROM/TO a single segment. Any single segment in the source synonym can be used in the join.

**Example: Joining From a Multi-Fact Synonym**

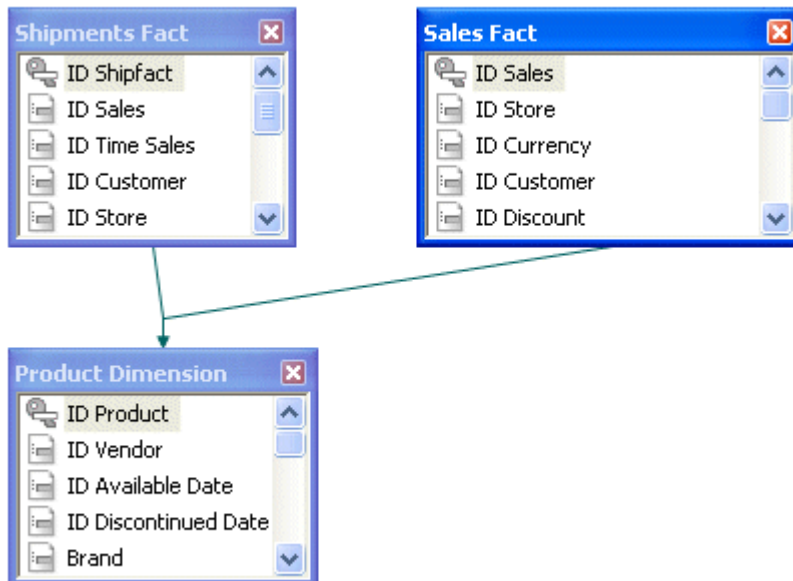
The following Master File describes a multi-parent structure based on the WebFOCUS Retail tutorial. The two fact tables wf\_retail\_sales and wf\_retail\_shipments are parents of the dimension table wf\_retail\_product.

```

FILENAME=WF_RETAIL_MULTI_PARENT, $
  SEGMENT=WF_RETAIL_SHIPMENTS, CRFILE=WFRETAIL/FACTS/WF_RETAIL_SHIPMENTS,
CRINCLUDE=ALL,
  DESCRIPTION='Shipments Fact', $
  SEGMENT=WF_RETAIL_SALES, PARENT=., CRFILE=WFRETAIL/FACTS/WF_RETAIL_SALES,
CRINCLUDE=ALL,
  DESCRIPTION='Sales Fact', $
  SEGMENT=WF_RETAIL_PRODUCT, CRFILE=WFRETAIL/DIMENSIONS/WF_RETAIL_PRODUCT,
CRINCLUDE=ALL,
  DESCRIPTION='Product Dimension', $
  PARENT=WF_RETAIL_SHIPMENTS, SEGTYPE=KU,
  JOIN_WHERE=WF_RETAIL_SHIPMENTS.ID_PRODUCT EQ
WF_RETAIL_PRODUCT.ID_PRODUCT; $
  PARENT=WF_RETAIL_SALES, SEGTYPE=KU,
  JOIN_WHERE=WF_RETAIL_SALES.ID_PRODUCT EQ WF_RETAIL_PRODUCT.ID_PRODUCT; ,
$

```

The following image shows the joins between these tables in the Synonym Editor of the Data Management Console (DMC).



The following request joins the product segment to the dimension table wf\_retail\_vendor based on the vendor ID and issues a request against the joined structure:

```
JOIN ID_VENDOR IN WF_RETAIL_MULTI_PARENT TO ID_VENDOR IN WF_RETAIL_VENDOR
AS J1
TABLE FILE WF_RETAIL_MULTI_PARENT
SUM COGS_US DAYSDELAYED
BY PRODUCT_CATEGORY
BY VENDOR_NAME
WHERE PRODUCT_CATEGORY LT 'S'
ON TABLE SET PAGE NOPAGE
END
```

The output is:

Product Category	Vendor Name	Cost of Goods	Days Delayed
Accessories	Audio Technica	\$38,000.00	28
	Denon	\$25,970.00	17
	Grado	\$21,930.00	15
	Logitech	\$61,432.00	114
	Niles Audio	\$73,547.00	150
	Pioneer	\$16,720.00	71
	Samsung	\$5,405.00	83
	Sennheiser	\$78,113.00	128
	Sony	\$21,760.00	157
Camcorder	Canon	\$110,219.00	97
	JVC	\$72,292.00	75
	Panasonic	\$22,356.00	91
	Sanyo	\$31,590.00	179
	Sony	\$216,748.00	333
Computers	Samsung	\$33,129.00	156
	Sony	\$76,152.00	186
Media Player	JVC	\$87,057.00	267
	LG	\$3,830.00	13
	Panasonic	\$143,600.00	171
	Pioneer	\$169,810.00	206
	Roku	\$10,248.00	85
	Samsung	\$151,620.00	191
	Sharp	\$66,024.00	157
	Sony	\$142,190.00	121
	Toshiba	\$5,214.00	7

## Invoking Context Analysis for a Star Schema With a Fan Trap

When a star schema contains a segment with aggregated facts and a lower-level segment with the related detail-level facts, a request that performs aggregation on both levels and returns them sorted by the higher level can experience the multiplicative effect. This means that the fact values that are already aggregated may be re-aggregated and, therefore, return multiplied values.

When the adapter detects the multiplicative effect, it turns optimization off in order to handle the request processing and circumvent the multiplicative effect. However, performance is degraded when a request is not optimized.

A new context analysis process has been introduced in this release that detects the multiplicative effect and generates SQL script commands that retrieve the correct values for each segment context. These scripts are then passed to the RDBMS as subqueries in an optimized SQL statement.

To activate the context analysis feature, click *Change Common Adapter Settings* on the Adapters page of the Web Console. Then select Yes for the *FCA* parameter in the *Miscellaneous Settings* section and click *Save*, as shown in the following image.

### Change Settings for Common Adapter

Save settings in 

Profile

 Select 

edasprof

Customize data type mappings

DECOMPOSE-DATE

No

Decompose Date fields into components

DATEFMT

Adapter specific default

Format for Date fields

Miscellaneous settings

FCA

Yes

Activate Foctransform Context Analysis

BLEND-MODE

Select only common values

Mode for processing blended structures

LITE-MODE

No

Simplified mode driven by configuration parameters

ETL-TRG-DBMS

Default ETL Target DBMS

Save

Cancel



## Adding DBA Restrictions to the Join Condition: SET DBAJJOIN

When DBA restrictions are applied to a request on a multi-segment structure, by default, the restrictions are added as WHERE conditions in the report request. When the DBAJJOIN parameter is set ON, DBA restrictions are treated as internal to the file or segment for which they are specified, and are added to the join syntax.

**Note:** DBA restrictions with DBAJJOIN OFF apply to the entire record instance that is being retrieved. Therefore, the entire record instance is suppressed when any part of that instance is restricted. DBAJJOIN ON applies the DBA only to the segment where the data value appears, allowing the rest of the record instance to be displayed, if applicable.

This difference is important when the file or segment being restricted has a parent in the structure and the join is an outer or unique join.

When restrictions are treated as report filters, lower-level segment instances that do not satisfy them are omitted from the report output, along with their host segments. Since host segments are omitted, the output does not reflect a true outer or unique join.

When the restrictions are treated as join conditions, lower-level values from segment instances that do not satisfy them are displayed as missing values, and the report output displays all host rows.

### *Syntax:* How to Add DBA Restrictions to the Join Condition

```
SET DBAJJOIN = {OFF | ON}
```

where:

OFF

Treats DBA restrictions as WHERE filters in the report request. OFF is the default value.

ON

Treats DBA restrictions as join conditions.

**Example:**     **Using the DBAJJOIN Setting With Relational Tables**

The following request creates two tables, EMPINFOSQL and EDINFOSQL:

```
TABLE FILE EMPLOYEE
SUM LAST_NAME FIRST_NAME CURR_JOBCODE
BY EMP_ID
ON TABLE HOLD AS EMPINFOSQL FORMAT SQLMSS
END
-RUN
TABLE FILE EDUCFILE
SUM COURSE_CODE COURSE_NAME
BY EMP_ID
ON TABLE HOLD AS EDINFOSQL FORMAT SQLMSS
END
```

Add the following DBA attributes to the end of the generated EMPINFOSQL Master File. With the restrictions listed, USER2 cannot retrieve course codes of 300 or above:

```
END
DBA=USER1,$
USER=USER2, ACCESS = R, $
FILENAME=EDINFOSQL,$
USER=USER2, ACCESS = R, RESTRICT = VALUE, NAME=SYSTEM, VALUE=COURSE_CODE LT
300;,$
```

Add the following DBA attributes to the end of the generated EDINFOSQL Master File:

```
END
DBA=USER1,DBAFILE=EMPINFOSQL,$
```

Issue the following request:

```
SET USER=USER2
SET DBAJJOIN=OFF
JOIN LEFT_OUTER EMP_ID IN EMPINFOSQL TO MULTIPLE EMP_ID IN EDINFOSQL AS J1
TABLE FILE EMPINFOSQL
PRINT LAST_NAME FIRST_NAME COURSE_CODE COURSE_NAME
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
GRID=OFF,$
END
```

On the report output, all host and child rows with course codes 300 or above have been omitted, as shown in the following image:

<u>LAST_NAME</u>	<u>FIRST_NAME</u>	<u>COURSE_CODE</u>	<u>COURSE_NAME</u>
STEVENS	ALFRED	101	FILE DESCRPT & MAINT
SMITH	MARY	103	BASIC REPORT PREP FOR PROG
JONES	DIANE	203	FOCUS INTERNALS
SMITH	RICHARD	108	BASIC RPT NON-DP MGRS
MCKNIGHT	ROGER	101	FILE DESCRPT & MAINT

In the generated SQL the DBA restriction has been added to the WHERE predicate in the SELECT statement:

```
SELECT
  T1."EID",
  T1."LN",
  T1."FN",
  T2."CC",
  T2."CD"
  FROM
    EMPINFOSQL T1,
    EDINFOSQL T2
  WHERE
    (T2."EID" = T1."EID") AND
    (T2."CC" < '300;');
```

Rerun the request with SET DBAJJOIN=ON. The output now displays all host rows, with missing values substituted for lower-level segment instances that did not satisfy the DBA restriction, as shown on the following image:

<u>LAST_NAME</u>	<u>FIRST_NAME</u>	<u>COURSE_CODE</u>	<u>COURSE_NAME</u>
STEVENS	ALFRED	101	FILE DESCRIPT & MAINT
SMITH	MARY	103	BASIC REPORT PREP FOR PROG
JONES	DIANE	203	FOCUS INTERNALS
SMITH	RICHARD	108	BASIC RPT NON-DP MGRS
BANNING	JOHN	.	.
IRVING	JOAN	.	.
ROMANS	ANTHONY	.	.
MCCOY	JOHN	.	.
BLACKWOOD	ROSEMARIE	.	.
MCKNIGHT	ROGER	101	FILE DESCRIPT & MAINT
GREENSPAN	MARY	.	.
CROSS	BARBARA	.	.

In the generated SQL, the DBA restriction has been added to the join, and there is no WHERE predicate:

```
SELECT
  T1."EID",
  T1."LN",
  T1."FN",
  T2."EID",
  T2."CC",
  T2."CD"
FROM
  ( EMPINFOSQL T1
    LEFT OUTER JOIN EDINFOSQL T2
      ON T2."EID" = T1."EID" AND
         (T2."CC" < '300;' ) );
```

## Preserving Virtual Fields During Join Parsing

There are two ways to preserve virtual fields during join parsing. One way is to use KEEPDEFINES, and the second is to use DEFINE FILE SAVE and DEFINE FILE RETURN.

## Preserving Virtual Fields Using KEEPDEFINES

The KEEPDEFINES parameter determines if a virtual field created by the DEFINE command for a host or joined structure is retained or cleared after the JOIN command is run. It applies when the DEFINE command precedes the JOIN command.

The prior virtual fields constitute what is called a context. Each new context creates a new layer or command environment. When you first enter the new environment, all virtual fields defined in the previous layer are available in the new layer. Overwriting or clearing a virtual field definition affects only the current layer. When you return to the previous layer, its virtual field definitions are intact.

New DEFINE fields issued after the JOIN command constitute another context, and by so doing generate a stack of contexts. In each context, all virtual fields of all prior contexts are accessible.

- ☐ By default the KEEPDEFINES setting is OFF. With this setting, a JOIN command removes prior virtual fields.
- ☐ When KEEPDEFINES is set to ON, virtual fields are reinstated during the parsing of a subsequent JOIN command.

A JOIN CLEAR *as\_name* command removes all the contexts that were created after the JOIN *as\_name* was issued.

For DEFINE-based conditional joins, the KEEPDEFINES setting must be ON. You then must create all virtual fields before issuing the DEFINE-based conditional JOIN command. This differs from traditional DEFINE-based joins in which the virtual field is created after the JOIN command. In addition, a virtual field may be part of the JOIN syntax or WHERE syntax.

DEFINE commands issued after the JOIN command do not replace or clear the virtual fields created before the join, since a new file context is created.

### ***Syntax:*** How to Use KEEPDEFINES

```
SET KEEPDEFINES = {ON|OFF}
```

where:

ON

Retains the virtual field after a JOIN command is run.

OFF

Clears the virtual field after a JOIN command is run. This value is the default.

**Reference: Usage Notes for KEEPDEFINES**

Virtual fields defined prior to setting KEEPDEFINES ON are not preserved after a JOIN command.

**Example: Preserving Virtual Fields During Join Parsing With KEEPDEFINES**

The first virtual field, DAYSKEPT, is defined prior to issuing any joins, but after setting KEEPDEFINES to ON. DAYSKEPT is the number of days between the return date and rental date for a videotape:

```
SET KEEPDEFINES = ON
DEFINE FILE VIDEOTRK
DAYSKEPT/I5 = RETURNDATE - TRANSDATE;
END
```

The ? DEFINE query command shows that this is the only virtual field defined at this point:

```
? DEFINE

FILE      FIELD NAME                FORMAT  SEGMENT  VIEW      TYPE
VIDEOTRK  DAYSKEPT                  I5      4
```

The following request prints all transactions in which the number of days kept is two:

```
TABLE FILE VIDEOTRK
PRINT MOVIECODE TRANSDATE RETURNDATE DAYSKEPT
COMPUTE ACTUAL_DAYS/I2 = RETURNDATE-TRANSDATE;
WHERE DAYSKEPT EQ 2
END
```

The first few lines of output show that each return date is two days after the transaction date:

MOVIECODE	TRANSDATE	RETURNDATE	DAYSKEPT	ACTUAL_DAYS
001MCA	91/06/27	91/06/29	2	2
692PAR	91/06/27	91/06/29	2	2
259MGM	91/06/19	91/06/21	2	2

Now, the VIDEOTRK data source is joined to the MOVIES data source. The ? DEFINE query shows that the join did not clear the DAYSKEPT virtual field:

```
JOIN MOVIECODE IN VIDEOTRK TO ALL MOVIECODE IN MOVIES AS J1
? DEFINE

FILE      FIELD NAME                FORMAT  SEGMENT  VIEW      TYPE
VIDEOTRK  DAYSKEPT                  I5      4
```

Next a new virtual field, YEARS, is defined for the join between VIDEOTRK and MOVIES:

```
DEFINE FILE VIDEOTRK
YEARS/I5 = (TRANSDATE - RELDATE)/365;
END
```

The ? DEFINE query shows that the virtual field created prior to the join was not cleared by this new virtual field because it was in a separate context:

```
? DEFINE
```

FILE	FIELD NAME	FORMAT	SEGMENT	VIEW	TYPE
VIDEOTRK	DAYSKEPT	I5	4		
VIDEOTRK	YEARS	I5	5		

Next, the field DAYSKEPT is re-defined so that it is the number of actual days plus one:

```
DEFINE FILE VIDEOTRK
DAYSKEPT/I5 = RETURNDATE - TRANSDATE + 1;
END
```

The ? DEFINE query shows that there are two versions of the DAYSKEPT virtual field. However, YEARS was cleared because it was in the same context (after the join) as the new version of DAYSKEPT, and the DEFINE command did not specify the ADD option:

```
? DEFINE
```

FILE	FIELD NAME	FORMAT	SEGMENT	VIEW	TYPE
VIDEOTRK	DAYSKEPT	I5	4		
VIDEOTRK	DAYSKEPT	I5	4		

The same request now uses the new definition for DAYSKEPT. Note that the number of days between the return date and transaction date is actually one day, not two because of the change in the definition of DAYSKEPT:

MOVIECODE	TRANSDATE	RETURNDATE	DAYSKEPT	ACTUAL_DAYS
040ORI	91/06/20	91/06/21	2	1
505MGM	91/06/21	91/06/22	2	1
710VES	91/06/26	91/06/27	2	1

Now, J1 is cleared. The redefinition for DAYSKEPT is also cleared:

```
JOIN CLEAR J1
? DEFINE
```

FILE	FIELD NAME	FORMAT	SEGMENT	VIEW	TYPE
VIDEOTRK	DAYSKEPT	I5	4		

The report output shows that the original definition for DAYSKEPT is now in effect:

MOVIECODE	TRANSDATE	RETURNDATE	DAYSKEPT	ACTUAL_DAYS
-----	-----	-----	-----	-----
001MCA	91/06/27	91/06/29	2	2
692PAR	91/06/27	91/06/29	2	2
259MGM	91/06/19	91/06/21	2	2

Preserving Virtual Fields Using DEFINE FILE SAVE and RETURN

The DEFINE FILE SAVE command forms a new context for virtual fields, which can then be removed with DEFINE FILE RETURN. For details, see [Creating Temporary Fields](#) on page 277.

*Example:* Preserving Virtual Fields With DEFINE FILE SAVE and RETURN

The following command enables you to preserve virtual fields within a file context:

```
SET KEEPDEFINES=ON
```

The following command defines virtual field A for the VIDEOTRK data source and places it in the current context:

```
DEFINE FILE VIDEOTRK
  A/A5='JAWS';
END
```

The following command creates a new context and saves virtual field B in this context:

```
DEFINE FILE VIDEOTRK SAVE
  B/A5='ROCKY';
END
? DEFINE
```

The output of the ? DEFINE query lists virtual fields A and B:

FILE	FIELD NAME	FORMAT	SEGMENT	VIEW	TYPE
VIDEOTRK	A	A5			
VIDEOTRK	B	A5			

The following DEFINE command creates virtual field C. All previously defined virtual fields are cleared because the ADD option was not used in the DEFINE command:

```
DEFINE FILE VIDEOTRK
  C/A10='AIRPLANE';
END
? DEFINE
```

The output of the ? DEFINE query shows that C is the only virtual field defined:

FILE	FIELD NAME	FORMAT	SEGMENT	VIEW	TYPE
VIDEOTRK	C	A10			



The following JOIN command creates a new context. Because KEEPDEFINES is set to ON, virtual field C is not cleared by the JOIN command:

```
JOIN MOVIECODE IN VIDEOTRK TAG V1 TO MOVIECODE IN MOVIES TAG M1 AS J1
? DEFINE
```

The output of the ? DEFINE query shows that field C is still defined:

FILE	FIELD NAME	FORMAT	SEGMENT	VIEW	TYPE
VIDEOTRK	C	A10			

The next DEFINE command creates virtual field D in the new context created by the JOIN command:

```
DEFINE FILE VIDEOTRK SAVE
  D/A10='TOY STORY';
END
? DEFINE
```

The output of the ? DEFINE query shows that virtual fields C and D are defined:

FILE	FIELD NAME	FORMAT	SEGMENT	VIEW	TYPE
VIDEOTRK	C	A10			
VIDEOTRK	D	A10			

The DEFINE FILE RETURN command clears virtual field D created in the current context (after the JOIN):

```
DEFINE FILE VIDEOTRK RETURN
END
? DEFINE
```

The output of the ? DEFINE query shows that virtual field D was cleared, but C is still defined:

FILE	FIELD NAME	FORMAT	SEGMENT	VIEW	TYPE
VIDEOTRK	C	A10			

The following DEFINE FILE RETURN command does not clear virtual field C because field C was not created using a DEFINE FILE SAVE command:

```
DEFINE FILE VIDEOTRK RETURN
END
? DEFINE
```

The output of the ? DEFINE query shows that virtual field C is still defined:

FILE	FIELD NAME	FORMAT	SEGMENT	VIEW	TYPE
VIDEOTRK	C	A10			

**Note:** DEFINE FILE RETURN is only activated when a DEFINE FILE SAVE is in effect.

### Screening Segments With Conditional JOIN Expressions

The conditional JOIN command can reference any and all fields in the joined segment and any and all fields in the parent segment, or higher on the parent's path.

When active, these join expressions screen the segment on which they reside (the child or joined segment). That is, if no child segment passes the test defined by the expression, the join follows the rules of SET ALL=OFF, or SET ALL=ON when no child segment exists. Unlike WHERE phrases in TABLE commands, JOIN\_WHERE screening does not automatically screen the parent segment when SET ALL=ON.

### Parsing WHERE Criteria in a Join

WHERE criteria take effect in a join only when a TABLE request reference is made to a cross-referenced segment or its children. If no such reference is made, the WHERE has no effect.

The AT attribute is used to link the correct parent segment or host to the correct child or cross-referenced segment. The field values used as the AT parameter are not used to cause the link. They are used simply as segment references.

**Note:** If no WHERE criteria are in effect, you receive a Cartesian product.

### Displaying Joined Structures

When you join two data sources together, they are subsequently treated as one logical structure. This structure results from appending the structure of the cross-referenced file to the structure of the host file. The segment in the cross-referenced file containing the shared value field becomes the child of the segment in the host file with the shared value field.

#### **Syntax:** How to Display a Joined Structure

To display the joined structure, issue the following command:

```
CHECK FILE hostfile PICTURE
```

where:

*hostfile*

Is the name of the host file.

**Example: Displaying a Joined Structure**

Notice that the segments belonging to the host file appear as regular segments outlined by asterisks. The segments belonging to the cross-referenced file appear as virtual segments outlined by dots. The segments of the cross-referenced file are also labeled with the cross-referenced file name below each segment.

```
JOIN PIN IN EMPDATA TO PIN IN SALHIST
CHECK FILE EMPDATA PICTURE
0 NUMBER OF ERRORS=      0
  NUMBER OF SEGMENTS=    2  ( REAL=    1  VIRTUAL=    1 )
  NUMBER OF FIELDS=     14  INDEXES=    1  FILES=     2
  NUMBER OF DEFINES=     1
  TOTAL LENGTH OF ALL FIELDS= 132
1SECTION 01.01
                        STRUCTURE OF FOCUS      FILE EMPDATA  ON 03/05/01 AT 12.22.49
```

```

      EMPDATA
01      S1
*****
*PIN          **I
*LASTNAME     **
*FIRSTNAME    **
*MIDINITIAL   **
*             **
*****
      I
      I
      I
      I SLHISTRY
02      I KU
.....
:PIN          :K
:EFFECTDATE   :
:OLDSALARY    :
:             :
:             :
:.....:
JOINED  SALHIST
```

The top segment of the cross-referenced file structure is the one containing the shared-value field. If this segment is not the root segment, the cross-referenced file structure is inverted, as in an alternate file view.

The cross-referenced file segment types in the joined structure are the following:

- ❑ In unique join structures, the top cross-referenced file segment has the segment type KU. Its unique child segments have segment type KLU. Non-unique child segments have segment type KL.
- ❑ In non-unique join structures, the top cross-referenced file segment has the segment type KM. Its unique child segments have segment type KLU. Non-unique child segments have segment type KL.

The host file structure remains unchanged. The cross-referenced file may still be used independently.

**Syntax:**      **How to List Joined Structures**

To display a list of joined data sources, issue the following command:

```
? JOIN
```

This displays every JOIN command currently in effect. For example:

JOINS CURRENTLY ACTIVE								
HOST				CROSSREFERENCE				
FIELD	FILE	TAG		FIELD	FILE	TAG	AS	ALL WH
----	----	---		----	----	---	--	--- --
JOBCODE	EMPLOYEE			JOBCODE	JOBFILE			N N

If the joined structure has no join name, the AS phrase is omitted. If two data sources are joined by multiple JOIN commands, only the first command you issued is displayed. The N in the WH column indicates that the join is not conditional. A Y indicates that the join is conditional.

**Clearing Joined Structures**

You can clear specific join structures, or all existing structures. Clearing deactivates the designated joins. If you clear a conditional join, all joins issued subsequently to that join using the same host file are also cleared.

**Tip:** If you wish to list the current joins before clearing or see details about all active joined structures, issue the query command ? JOIN. For details and illustrations, see [How to List Joined Structures](#) on page 1072.

**Syntax:**      **How to Clear a Join**

To clear a joined structure, issue this command:

```
JOIN CLEAR {joinname|*}
```

where:

*joinname*

Is the AS name of the joined structure you want to clear.

\*

Clears all joined structures.

**Clearing a Conditional Join**

You can clear a join by issuing the JOIN CLEAR command. The effect of the JOIN CLEAR command depends on whether any conditional join exists.

- ❑ If conditional joins are found and were issued after the join you wish to clear, or if the join you wish to clear is a conditional join, then the JOIN CLEAR *as\_name* command removes all joins issued after the specified join.
- ❑ If no conditional joins were issued after the join you wish to clear, only the join you specify is cleared. Any virtual fields saved in the context of a join that is cleared are also cleared. Normal joins may or may not be cleared, depending on the position of the conditional join. The JOIN CLEAR \* command clears every join issued, along with its associated virtual fields. However, all virtual fields in the null context remain untouched.

**Note:** The null context is the context of the data source prior to any joins being issued.

**Example:**      **Clearing Joins**

The following request creates three joins using VIDEOTRK as the host data source. The first two are conditional (JW1, JW2), and the third join is unconditional (J1):

```
JOIN FILE VIDEOTRK AT PRODCODE TO ALL
      FILE GGSales AT PCD AS JW1
WHERE PRODCODE NE PCD;
END
JOIN FILE VIDEOTRK AT TRANSDATE TO ALL
      FILE MOVIES AT RElDATE AS JW2
WHERE (TRANSDATE - RElDATE)/365 GT 10;
END
JOIN MOVIECODE IN VIDEOTRK TO MOVIECODE IN MOVIES AS J1
```

The next request creates a conditional join (JW3) using MOVIES as the host data source:

```
JOIN  FILE MOVIES      AT MOVIECODE TO ONE
      FILE VIDEOTRK AT TRANSDATE AS JW3
WHERE (TRANSDATE - RELDATE)/365 LT 2;
END
```

The last request creates a third conditional join (JW4) that uses VIDEOTRK as the host data source:

```
JOIN  FILE VIDEOTRK AT LASTNAME  TO ALL
      FILE EMPLOYEE AT LAST_NAME AS JW4
WHERE LASTNAME GE LAST_NAME;
END
```

Following is the output of the ? JOIN query after executing these joins:

```
? JOIN
JOINS CURRENTLY ACTIVE
```

HOST			CROSSREFERENCE					
FIELD	FILE	TAG	FIELD	FILE	TAG	AS	ALL	WH
----	----	---	----	----	---	--	---	--
PRODCODE	VIDEOTRK		PCD	GGSales		JW1	Y	Y
TRANSDATE	VIDEOTRK		RELDATE	MOVIES		JW2	Y	Y
MOVIECODE	VIDEOTRK		MOVIECODE	MOVIES		J1	N	N
MOVIECODE	MOVIES		TRANSDATE	VIDEOTRK		JW3	N	Y
LASTNAME	VIDEOTRK		LAST_NAME	EMPLOYEE		JW4	Y	Y

Clearing JW2 clears all joins that were issued after JW2 and that use the same host data source. JW1 remains because it was issued prior to JW2, and JW3 remains because it uses a different host data source:

```
JOIN CLEAR JW2
? JOIN
JOINS CURRENTLY ACTIVE
```

HOST			CROSSREFERENCE					
FIELD	FILE	TAG	FIELD	FILE	TAG	AS	ALL	WH
----	----	---	----	----	---	--	---	--
PRODCODE	VIDEOTRK		PCD	GGSales		JW1	Y	Y
MOVIECODE	MOVIES		TRANSDATE	VIDEOTRK		JW3	N	Y

## Merging Data Sources

---

You can gather data for your reports by merging the contents of data structures with the MATCH command, or concatenating data sources with the MORE phrase, and reporting from the combined data.

**In this chapter:**

- ☐ [Merging Data](#)
  - ☐ [Types of MATCH Processing](#)
  - ☐ [MATCH Processing With Common High-Order Sort Fields](#)
  - ☐ [Fine-Tuning MATCH Processing](#)
  - ☐ [Universal Concatenation](#)
  - ☐ [Merging Concatenated Data Sources](#)
  - ☐ [Cartesian Product](#)
- 

### Merging Data

You can merge two or more data sources, and specify which records to merge and which to sort out, using the MATCH command. The command creates a new data source (a HOLD file), into which it merges fields from the selected records. You can report from the new data source and use it as you would use any other HOLD file.

You select the records to be merged into the new data source by specifying sort fields in the MATCH command. You specify one set of sort fields (using the BY phrase), for the first data source, and a second set of sort fields for the second data source. The MATCH command compares all sort fields that have been specified in common for both data sources, and then merges all records from the first data source whose sort values match those in the second data source into the new HOLD file. You can specify up to 128 sort sets. This includes the number of common sort fields.

In addition to merging data source records that share values, you can merge records based on other relationships. For example, you can merge all records in each data source whose sort values are not matched in the other data source. Yet another type of merge combines all records from the first data source with any matching records from the second data source.

You can merge up to 16 sets of data in one Match request. For example, you can merge different data sources, or data from the same data source.

**Note:** The limit of 16 applies to the most complex request. Simpler requests may be able to merge more data sources.

### **Syntax:**      **How to Merge Data Sources**

The syntax of the MATCH command is similar to that of the TABLE command:

```
MATCH FILE file1
.
.
.
RUN
FILE file2
.
.
.
[AFTER MATCH merge_phrase]
RUN
FILE file3
.
.
.
[AFTER MATCH merge_phrase]
END
```

where:

*file1*

Is the first data source from which MATCH retrieves requested records.

*merge\_phrase*

Specifies how the retrieved records from the files are to be compared. For details, see [Merge Phrases](#) on page 1083.

*file2/file3*

Are additional data sources from which MATCH retrieves requested records.

Note that a RUN command must follow each AFTER MATCH command (except for the last one). The END command must follow the final AFTER MATCH command.

MATCH generates a HOLD file. You can print the contents of the HOLD file using the PRINT command with the wildcard character (\*).



## Types of MATCH Processing

There are two types of MATCH processing, grouped and ungrouped. Grouped processing is the newer type of MATCH processing, and is the processing used by default. Ungrouped processing is the legacy MATCH processing. If you need to invoke legacy processing, you can use the SET MATCHCOLUMNORDER = UNGROUPED command.

In all MATCH requests:

- ❑ The two sides of the merge, the OLD and the NEW, are matched together based on their common high-order BY fields. In order for actual matching to take place between the OLD and NEW files, the high-order BY fields have to be the same. If there are no common high-order BY fields, the records are concatenated on a record-by-record basis.
- ❑ The output selected from the OLD and NEW sides is based on the AFTER MATCH command.

The output stage of the MATCH differs for grouped and ungrouped processing.

- ❑ With grouped processing, the output file has each common sort key (BY field) followed its display fields in the order specified in the request. As a result, ungrouped processing is limited to flat file output.
- ❑ In contrast, with grouped processing, fields in the request are grouped with their highest common sort fields in the output file. This enables you to generate multi-segment hierarchical output files in FOCUS or XFOCUS format by using multiple display commands on each side of the merge. In fact, you can create the output file using any format that has a corresponding Master File.

For standard MATCH requests that use the same sort keys on both sides of the MATCH (OLD and NEW), grouped and ungrouped processing produce the same output.

However, in requests that use multiple display commands or differing sort fields on each side of the merge, the grouping of fields with their sort keys can produce output files in which the field order is different from the legacy processing. In any case where the new behavior generates output that is different from previous results and not desirable, the SET MATCHCOLUMNORDER command is available to return the legacy results.

For example, if you use MATCH to create output that includes a list of products with columns of aggregations based on differing sorts, MATCHCOLUMNORDER=UNGROUPED will ensure that the sequence of the column output will remain what it was in the past.

The way MATCH merges data depends on the order in which you name data sources in the request, the BY fields, display commands, the type of processing, and the merge phrases you use. In general, however, processing is as follows:

1. MATCH retrieves requested records from the first data source you name, and writes them to a temporary work area.
2. MATCH retrieves requested records from the second data source you name, and writes them to a temporary work area.
3. It compares the common high-order sort fields from the retrieved records as specified in the merge phrase (for example, OLD-OR-NEW). For more information, see [Merge Phrases](#) on page 1083.
4. If the default grouped processing is in effect, it may re-order the fields to group them under their common sort fields.
5. It writes the merged results of the comparison to a temporary data source (if there are more MATCH operations). It cycles through all data sources named until END is encountered.
6. It writes final records to the HOLD file.

### **Syntax:** How to Controlling MATCH Processing

`SET MATCHCOLUMNORDER = {GROUPED|UNGROUPED}`

where:

#### [GROUPED](#)

Groups fields in the output file under their common high-order sort fields. This is the default value.

#### [UNGROUPED](#)

Does not group fields in the output file with their common high-order sort fields, but lays them out as specified in the MATCH request.

### **Reference:** Usage Notes for Match Requests

- ❑ With ungrouped processing, you cannot specify a format for the HOLD file generated by MATCH. It will be created as a single-segment BINARY or ALPHA HOLD file, depending on the value of the HOLDFORMAT parameter. The merge process does not change the original data sources.

- ❑ Alias names are assigned sequentially (E01, E02, ...) in the HOLD Master File that results from the MATCH request. When the same field name is used multiple times in the MATCH, users distinguish between them in requests against the HOLD file by referencing these alias names instead of the field names.

With grouped processing, fields are rearranged in the Master File, and this causes the alias names to represent different fields from the same alias names assigned with ungrouped processing. This can produce different results if you switch from one type of processing to the other.

To avoid using alias names, use the AS phrase in your MATCH request to create distinct field names (except for the common high-order BY fields, which have to be the same), and use those field names in requests against the HOLD file.

- ❑ The ACROSS, BY HIGHEST/LOWEST  $n$ , IN-GROUPS-OF, WHERE TOTAL, and IF TOTAL phrases, and the COMPUTE command, are not permitted in a MATCH request. You can, however, use the DEFINE command.
- ❑ Up to 128 BY phrases and the maximum number of display fields can be used in each MATCH request. The count of sort sets includes the number of common sort fields. The maximum number of display fields is determined by a combination of factors.

For details, see [Displaying Report Data](#) on page 43.

- ❑ You must specify at least one BY field for each file used in the MATCH request.
- ❑ When used with MATCH, the SET HOLDLIST parameter behaves as if HOLDLIST were set to ALL.
- ❑ The following prefix operators are not supported in MATCH requests: DST., DST.CNT., RNK., ST., and CT.

**Example: Merging Data Sources**

In the following request, the high-order sort field is the same for both files, so the result is the same using grouped and ungrouped processing.

```
MATCH FILE EDUCFILE
SUM COURSE_CODE
BY EMP_ID
RUN
FILE EMPLOYEE
SUM LAST_NAME AND FIRST_NAME
BY EMP_ID BY CURR_SAL
AFTER MATCH HOLD OLD-OR-NEW
END
_*****
-* PRINT CONTENTS OF HOLD FILE
_*****
TABLE FILE HOLD
PRINT *
END
```

The merge phrase used in this example was OLD-OR-NEW. This means that records from both the first (old) data source plus the records from the second (new) data source appear in the HOLD file.

The output is:

EMP_ID	COURSE_CODE	CURR_SAL	LAST_NAME	FIRST_NAME
-----	-----	-----	-----	-----
071382660	101	\$11,000.00	STEVENS	ALFRED
112847612	103	\$13,200.00	SMITH	MARY
117593129	203	\$18,480.00	JONES	DIANE
119265415	108	\$9,500.00	SMITH	RICHARD
119329144		\$29,700.00	BANNING	JOHN
123764317		\$26,862.00	IRVING	JOAN
126724188		\$21,120.00	ROMANS	ANTHONY
212289111	103	\$ .00		
219984371		\$18,480.00	MCCOY	JOHN
315548712	108	\$ .00		
326179357	301	\$21,780.00	BLACKWOOD	ROSEMARIE
451123478	101	\$16,100.00	MCKNIGHT	ROGER
543729165		\$9,000.00	GREENSPAN	MARY
818692173	302	\$27,062.00	CROSS	BARBARA

**Example: Comparing Grouped and Ungrouped Processing**

The following MATCH request has two SUM commands and one PRINT command for each file, with all sort fields common to both files. The SET MATCHCOLUMNORDER = UNGROUPED command is issued to invoke legacy processing.

```

SET MATCHCOLUMNORDER = UNGROUPED
MATCH FILE GGSales
SUM DOLLARS
BY ST
SUM BUDDOLLARS BY ST BY CITY
PRINT UNITS BY ST BY CITY BY CATEGORY
RUN
FILE GGSales
SUM DOLLARS
BY ST
SUM BUDDOLLARS BY ST BY CITY
PRINT BUDUNITS BY ST BY CITY BY CATEGORY
AFTER MATCH HOLD OLD-OR-NEW
END
TABLE FILE HOLD
PRINT *
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, SIZE=9,$
ENDSTYLE
END

```

The HOLD Master File follows. Since the sort fields are common to both files, the two files were merged based on those fields. However, note that the order of fields in the Master File follows the order in the request, the highest-level sort field followed by its display field, then the next sort field followed by its display fields, and so on.

```

FILENAME=HOLD, SUFFIX=FIX, IOTYPE=BINARY, $
SEGMENT=HOLD, SEGTYPE=S1, $
  FIELDNAME=ST, ALIAS=E01, USAGE=A02, ACTUAL=A04, $
  FIELDNAME=DOLLARS, ALIAS=E02, USAGE=I08, ACTUAL=I04, $
  FIELDNAME=CITY, ALIAS=E03, USAGE=A20, ACTUAL=A20, $
  FIELDNAME=BUDDOLLARS, ALIAS=E04, USAGE=I08, ACTUAL=I04, $
  FIELDNAME=CATEGORY, ALIAS=E05, USAGE=A11, ACTUAL=A12, $
  FIELDNAME=UNITS, ALIAS=E06, USAGE=I08, ACTUAL=I04, $
  FIELDNAME=DOLLARS, ALIAS=E07, USAGE=I08, ACTUAL=I04, $
  FIELDNAME=BUDDOLLARS, ALIAS=E08, USAGE=I08, ACTUAL=I04, $
  FIELDNAME=BUDUNITS, ALIAS=E09, USAGE=I08, ACTUAL=I04, $

```

The partial output is shown in the following image.

<u>ST</u>	<u>DOLLARS</u>	<u>CITY</u>	<u>BUDDOLLARS</u>	<u>CATEGORY</u>	<u>UNITS</u>	<u>DOLLARS</u>	<u>BUDDOLLARS</u>	<u>BUDUNITS</u>
CA	7642261	Los Angeles	3669484	Coffee	1667	7642261	3669484	1464
CA	7642261	Los Angeles	3669484	Coffee	1663	7642261	3669484	1464
CA	7642261	Los Angeles	3669484	Coffee	1519	7642261	3669484	1464
CA	7642261	Los Angeles	3669484	Coffee	1452	7642261	3669484	1520
CA	7642261	Los Angeles	3669484	Coffee	1663	7642261	3669484	1463
CA	7642261	Los Angeles	3669484	Coffee	1508	7642261	3669484	1539
CA	7642261	Los Angeles	3669484	Coffee	1615	7642261	3669484	1759
CA	7642261	Los Angeles	3669484	Coffee	1449	7642261	3669484	1494
CA	7642261	Los Angeles	3669484	Coffee	1208	7642261	3669484	1038
CA	7642261	Los Angeles	3669484	Coffee	1272	7642261	3669484	1512

Changing the UNGROUPED setting to GROUPED produces the following Master File. The fields that have the same common sort fields from both files are moved to be under those sort fields in the Master File.

```
FILENAME=HOLD, SUFFIX=FIX      , IOTYPE=BINARY, $
SEGMENT=HOLD, SEGTYPE=S1, $
  FIELDNAME=ST, ALIAS=E01, USAGE=A02, ACTUAL=A04, $
  FIELDNAME=DOLLARS, ALIAS=E02, USAGE=I08, ACTUAL=I04, $
  FIELDNAME=DOLLARS, ALIAS=E03, USAGE=I08, ACTUAL=I04, $
  FIELDNAME=CITY, ALIAS=E04, USAGE=A20, ACTUAL=A20, $
  FIELDNAME=BUDDOLLARS, ALIAS=E05, USAGE=I08, ACTUAL=I04, $
  FIELDNAME=BUDDOLLARS, ALIAS=E06, USAGE=I08, ACTUAL=I04, $
  FIELDNAME=CATEGORY, ALIAS=E07, USAGE=A11, ACTUAL=A12, $
  FIELDNAME=UNITS, ALIAS=E08, USAGE=I08, ACTUAL=I04, $
  FIELDNAME=BUDUNITS, ALIAS=E09, USAGE=I08, ACTUAL=I04, $
```

The partial output is shown in the following image.

<u>ST</u>	<u>DOLLARS</u>	<u>DOLLARS</u>	<u>CITY</u>	<u>BUDDOLLARS</u>	<u>BUDDOLLARS</u>	<u>CATEGORY</u>	<u>UNITS</u>	<u>BUDUNITS</u>
CA	7642261	7642261	Los Angeles	3669484	3669484	Coffee	1667	1464
CA	7642261	7642261	Los Angeles	3669484	3669484	Coffee	1663	1464
CA	7642261	7642261	Los Angeles	3669484	3669484	Coffee	1519	1464
CA	7642261	7642261	Los Angeles	3669484	3669484	Coffee	1452	1520
CA	7642261	7642261	Los Angeles	3669484	3669484	Coffee	1663	1463
CA	7642261	7642261	Los Angeles	3669484	3669484	Coffee	1508	1539
CA	7642261	7642261	Los Angeles	3669484	3669484	Coffee	1615	1759
CA	7642261	7642261	Los Angeles	3669484	3669484	Coffee	1449	1494
CA	7642261	7642261	Los Angeles	3669484	3669484	Coffee	1208	1038
CA	7642261	7642261	Los Angeles	3669484	3669484	Coffee	1272	1512

For the request that uses the GROUPED value for MATCHCOLUMNORDER, you can change the HOLD command to produce a FORMAT FOCUS output file, as follows.

```
AFTER MATCH HOLD FORMAT FOCUS OLD-OR-NEW
```

The following hierarchical multi-segment Master File is generated.

```

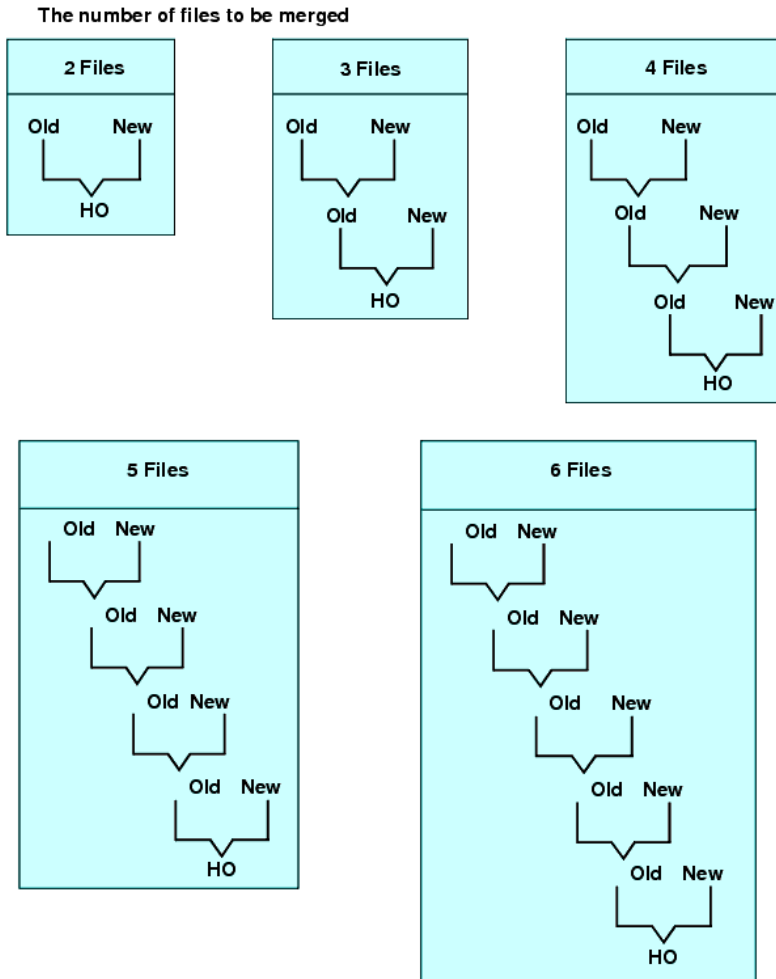
FILENAME=HOLD      , SUFFIX=FOC      , $
  SEGMENT=SEG01, SEGTYPE=S1, $
    FIELDNAME=ST, ALIAS=E01, USAGE=A02,
      TITLE='State', DESCRIPTION='State', $
    FIELDNAME=DOLLARS, ALIAS=E02, USAGE=I08,
      TITLE='Dollar Sales', DESCRIPTION='Total dollar amount of reported
sales', $
    FIELDNAME=DOLLARS, ALIAS=E03, USAGE=I08,
      TITLE='Dollar Sales', DESCRIPTION='Total dollar amount of reported
sales', $
  SEGMENT=SEG02, SEGTYPE=S1, PARENT=SEG01, $
    FIELDNAME=CITY, ALIAS=E04, USAGE=A20,
      TITLE='City', DESCRIPTION='City', $
    FIELDNAME=BUDDOLLARS, ALIAS=E05, USAGE=I08,
      TITLE='Budget Dollars', DESCRIPTION='Total sales quota in dollars', $
    FIELDNAME=BUDDOLLARS, ALIAS=E06, USAGE=I08,
      TITLE='Budget Dollars', DESCRIPTION='Total sales quota in dollars', $
  SEGMENT=SEG03, SEGTYPE=S2, PARENT=SEG02, $
    FIELDNAME=CATEGORY, ALIAS=E07, USAGE=A11,
      TITLE='Category', DESCRIPTION='Product category', $
    FIELDNAME=FOCLIST, ALIAS=E08, USAGE=I5, $
    FIELDNAME=UNITS, ALIAS=E09, USAGE=I08,
      TITLE='Unit Sales', DESCRIPTION='Number of units sold', $
  SEGMENT=SEG04, SEGTYPE=S1, PARENT=SEG03, $
    FIELDNAME=FOCLIST, ALIAS=E10, USAGE=I5, $
    FIELDNAME=BUDUNITS, ALIAS=E11, USAGE=I08,
      TITLE='Budget Units', DESCRIPTION='Number of units budgeted', $

```

### ***Reference:*** Merge Phrases

MATCH logic depends on the concept of old and new data sources. Old refers to the first data source named in the request, and new refers to the second data source. The result of each merge creates a HOLD file until the END command is encountered.

The following diagram illustrates the general merge process:



### **Syntax:** How to Specify Merge Phrases

```
AFTER MATCH HOLD [AS 'name'] mergetype
```

where:

```
AS 'name'
```

Specifies the name of the extract data source created by the MATCH command. The default is HOLD.

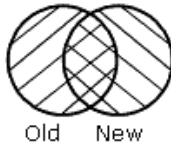


*mergetype*

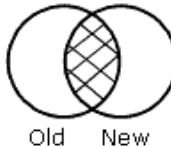
Specifies how the retrieved records from the files are to be compared.

The results of each phrase are graphically represented using Venn diagrams. In the diagrams, the left circle represents the old data source, the right circle represents the new data source, and the shaded areas represent the data that is written to the HOLD file.

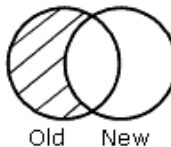
**OLD-OR-NEW** specifies that all records from both the old data source and the new data source appear in the HOLD file. This is the default if the AFTER MATCH line is omitted.



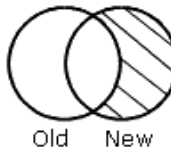
**OLD-AND-NEW** specifies that records that appear in both the old and new data sources appear in the HOLD file. (The intersection of the sets.)



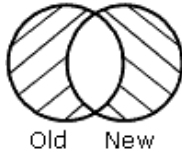
**OLD-NOT-NEW** specifies that records that appear only in the old data source appear in the HOLD file.



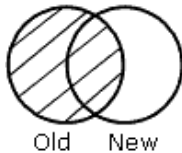
**NEW-NOT-OLD** specifies that records that appear only in the new data source appear in the HOLD file.



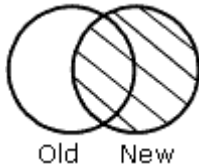
**OLD-NOR-NEW** specifies that only records that are in the old data source but not in the new data source, or in the new data source but not in the old, appear in the HOLD file (the complete set of non-matching records from both data sources).



**OLD** specifies that all records from the old data source, and any matching records from the new data source, are merged into the HOLD file.



**NEW** specifies that all records from the new data source, and any matching records from the old data source, are merged into the HOLD file.



## MATCH Processing With Common High-Order Sort Fields

When you construct your MATCH so that the first sort (BY) field (called the common high-order sort field) used for both data sources is the same, the match compares the values of the common high-order sort fields. If the entire sequence of sort fields is common to both files, all are compared.

At least one pair of sort fields is required. Field formats must be the same. In some cases, you can redefine a field format using the DEFINE command. If the field names differ, use the AS phrase to rename the second sort field to match the first. When the AS phrase is used in a MATCH request, the specified field is automatically renamed in the resulting HOLD file.

When you are merging files with common sort fields, the following assumptions are made:

- ❑ If one of the sort fields is a subset of the other, a one-to-many relationship is assumed.

- ❑ If neither of the sort fields is a subset of the other, a one-to-one relationship is assumed. At most, one matching record is retrieved.

**Example: MATCH Processing With Common High-Order Sort Fields**

To understand common high-order sort fields more clearly, consider some of the data from the following data sources

EMPLOYEE Data Source		EDUCFILE Data Source	
071382660	STEVENS	071382660	101
119329144	BANNING	212289111	103
112847612	SMITH	112847612	103

and this MATCH request:

```
MATCH FILE EMPLOYEE
SUM LAST_NAME BY EMP_ID
RUN
FILE EDUCFILE
SUM COURSE_CODE BY EMP_ID
AFTER MATCH HOLD OLD-OR-NEW
END
```

MATCH processing occurs as follows:

- ❑ Since there is a common high-order sort field (EMP\_ID), the MATCH logic begins by matching the EMP\_ID values in records from the EMPLOYEE and EDUCFILE files.
- ❑ There are records from both files with an EMP\_ID value of 071382660. Since there is a match, this record is written to the HOLD file:

```
Record n: 071382660 STEVENS 101
```

- ❑ There are records from both files with an EMP\_ID value of 112847612. Since there is a match, this record is written to the HOLD file:

```
Record n: 112847612 SMITH 103
```

- ❑ The records do not match where a record from the EMPLOYEE file has an EMP\_ID value of 119329144 and a record from the EDUCFILE file has an EMP\_ID value of 212289111. The record with the lower value is written to the HOLD file and a space is inserted for the missing value:

Record *n*: 119329144 BANNING

- Similarly, the 212289111 record exists only in the EDUCFILE file, and is written as:

Record *n*: 212289111 103

The following code produces a report of the records in the HOLD file:

```
TABLE FILE HOLD
PRINT *
END
```

The output is:

<u>EMP_ID</u>	<u>LAST_NAME</u>	<u>COURSE_CODE</u>
071382660	STEVENS	101
112847612	SMITH	103
117593129	JONES	203
119265415	SMITH	108
119329144	BANNING	
123764317	IRVING	
126724188	ROMANS	
212289111		103
219984371	MCCOY	
315548712		108
326179357	BLACKWOOD	301
451123478	MCKNIGHT	101
543729165	GREENSPAN	
818692173	CROSS	302

**Example: Merging With a Common High-Order Sort Field**

This request combines data from the EMPLOYEE and EMPDATA data sources. The sort fields are EID and PIN.

```

MATCH FILE EMPLOYEE
PRINT LN FN DPT
BY EID
RUN
FILE EMPDATA
PRINT LN FN DEPT
BY PIN
AFTER MATCH HOLD OLD-OR-NEW
END

TABLE FILE HOLD
PRINT *
END

```

**Example: Merging Without a Common High-Order Sort Field**

If there are no common high-order sort fields, a match is performed on a record-by-record basis. The following request matches the data and produces the HOLD file:

```

MATCH FILE EMPLOYEE
PRINT LAST_NAME AND FIRST_NAME
BY EMP_ID
RUN
FILE EMPDATA
PRINT PIN
BY LASTNAME
BY FIRSTNAME
AFTER MATCH HOLD OLD-OR-NEW
END

TABLE FILE HOLD
PRINT *
END

```

The retrieved records from the two data sources are written to the HOLD file; no values are compared. The output is:

<u>EMP_ID</u>	<u>LAST_NAME</u>	<u>FIRST_NAME</u>	<u>LASTNAME</u>	<u>FIRSTNAME</u>	<u>PIN</u>
071382660	STEVENS	ALFRED	ADAMS	RUTH	000000040
112847612	SMITH	MARY	ADDAMS	PETER	000000050
117593129	JONES	DIANE	ANDERSON	TIM	000000100
119265415	SMITH	RICHARD	BELLA	MICHAEL	000000020

## Fine-Tuning MATCH Processing

---

119329144	BANNING	JOHN	CASSANOVA	LOIS	000000030
123764317	IRVING	JOAN	CASTALANETT A	MARIE	000000270
126724188	ROMANS	ANTHONY	CHISOLM	HENRY	000000360
219984371	MCCOY	JOHN	CONRAD	ADAM	000000250
326179357	BLACKWOOD	ROSEMARIE	CONTI	MARSHALL	000000410
451123478	MCKNIGHT	ROGER	CVEK	MARCUS	000000130
543729165	GREENSPAN	MARY	DONATELLO	ERICA	000000320
818692173	CROSS	BARBARA	DUBOIS	ERIC	000000210
			ELLNER	DAVID	000000380
			FERNSTEIN	ERWIN	000000350
			GORDON	LAURA	000000180
			GOTLIEB	CHRIS	000000340
			GRAFF	ELAINE	000000390
			HIRSCHMAN	ROSE	000000160
			KASHMAN	YOLANDA	000000240
			LASTRA	KAREN	000000200
			LEWIS	CASSANDR A	000000220
.					
.					
.					

## Fine-Tuning MATCH Processing

You can fine-tune the MATCH process using the PRINT and SUM commands. To understand their difference, you should have an understanding of the one-to-many relationship: SUM generates one record from many, while PRINT displays each individual record. Through proper choices of BY fields, it is possible to use only the SUM command and get the same result that PRINT would produce.

**Example: Using Display Commands in MATCH Processing**

To illustrate the effects of PRINT and SUM on the MATCH process, consider data sources A and B and the series of requests that follow:

A			B		
F1	F2	F3	F1	F4	F5
1	x	100	1	a	10
2	y	200	1	b	20
			2	c	30
			2	d	40

**Request 1:** This request sums the fields F2 and F3 from file A, sums the fields F4 and F5 from file B, and uses F1 as the common high-order sort field.

```
MATCH FILE A
SUM F2 AND F3 BY F1
RUN
FILE B
SUM F4 AND F5 BY F1
AFTER MATCH HOLD OLD-OR-NEW
END
```

The HOLD file contains the following data:

F1	F2	F3	F4	F5
1	x	100	b	30
2	y	200	d	70

Note that the resulting file contains only 1 record for each common high-order sort field.

**Request 2:** This request sums fields F2 and F3 from file A, prints fields F4 and F5 from file B, and uses F1 as the common high-order sort field.

```
MATCH FILE A
SUM F2 AND F3 BY F1
RUN
FILE B PRINT F4 AND F5 BY F1
AFTER MATCH HOLD OLD-OR-NEW
END
```

The HOLD file contains:

F1	F2	F3	F4	F5
1	x	100	a	10
1	x	100	b	20
2	y	200	c	30
2	y	200	d	40

Note that the records from file A are duplicated for each record from file B.

**Request 3:** This request prints fields F2 and F3 from file A, sums fields F4 and F5 from file B, and uses F1 as the common high-order sort field.

```
MATCH FILE A
PRINT F2 AND F3 BY F1
RUN
FILE B
SUM F4 AND F5 BY F1
AFTER MATCH HOLD OLD-OR-NEW
END
```

The HOLD file contains:

F1	F2	F3	F4	F5
1	x	100	b	30
2	y	200	d	70

Note that each record from file A is included, but only the last record from file B for each common high-order sort field is included.

**Request 4:** This request prints fields F2 and F3 from file A, prints fields F4 and F5 from file B, and uses F1 as the common high-order sort field.

```
MATCH FILE A
PRINT F2 AND F3 BY F1
RUN
FILE B PRINT F4 AND F5 BY F1
AFTER MATCH HOLD OLD-OR-NEW
END
```

The HOLD file contains:

F1	F2	F3	F4	F5
1	x	100	a	10
1		0	b	20
2	y	200	c	30
2		0	d	40

Note the blank value for F2 and the 0 for F3.



**Request 5:** This request sums the fields F2 and F3 from file A, sums the field F5 from file B and sorts it by field F1, the common high-order sort field, and by F4.

```
MATCH FILE A
SUM F2 AND F3 BY F1
RUN
FILE B
SUM F5 BY F1 BY F4
AFTER MATCH HOLD OLD-OR-NEW
END
```

The HOLD file contains:

F1	F2	F3	F4	F5
1	x	100	a	10
1	x	100	b	20
2	y	200	c	30
2	y	200	d	40

Note that the records for file A are printed for every occurrence of the record in file B.

## Universal Concatenation

With universal concatenation, you can retrieve data from unlike data sources in a single request; all data, regardless of source, appears to come from a single file. The MORE phrase can concatenate all types of data sources (such as, FOCUS, DB2, IMS, and VSAM), provided they share corresponding fields with the same format. You can use WHERE and IF selection tests in conjunction with MORE. For related information, see [Selecting Records for Your Report](#) on page 219.

To use MORE, you must divide your request into:

- ❑ One main request that retrieves the first data source and defines the data fields, sorting criteria, and output format for all data.
- ❑ Subrequests that define the data sources and fields to be concatenated to the data of the main request. The fields printed and sorted by the main request must exist in each concatenated data source. If they do not, you must create them as virtual fields.

During retrieval, data is gathered from each data source in turn, then all data is sorted and the output formatted as specified in the main request.

**Syntax:**      **How to Concatenate Data Sources**

The MORE phrase, which is accessible within the TABLE and MATCH commands, specifies how to concatenate data from sources with dissimilar Master Files.

```
{TABLE|MATCH}  FILE file1main request
MORE
FILE file2
    subrequest
MORE
FILE file3
    subrequest
MORE
.
.
.
{END|RUN}
```

where:

**TABLE|MATCH**

     Begins the request that concatenates the data sources.

*file1*

     Is the name of the first data source.

*main request*

     Is a request, without END or RUN, that retrieves the first data source and defines the data fields, sorting criteria, and output format for all data. WHERE and IF criteria in the main request apply only to *file1*.

     When concatenating files within the TABLE command, you can also define calculated values for the first data source.

**MORE**

     Begins a subrequest. There is no limit to the number of subrequests, other than available memory.

**FILE *file2***

     Defines *file2* as the second data source for concatenation.

*subrequest*

     Is a subrequest. Subrequests can only include WHERE and IF phrases.

END | RUN

Ends the request.

### **Example:** Concatenating Data Sources

Both the EMPLOYEE and the EXPERSON data sources contain employee information. You can concatenate their common data into a single file:

❑ EMPLOYEE contains the field values EMP\_ID=123456789 and CURR\_SAL=50.00.

❑ EXPERSON contains the field values SSN=987654321 and WAGE=100.00.

The following annotated request concatenates the two data sources:

```

DEFINE FILE EXPERSON
1. EMP_ID/A9 = SSN;
   CURR_SAL/D12.2 = WAGE;
   END
2. TABLE FILE EMPLOYEE
   PRINT CURR_SAL
   BY EMP_ID
3. MORE
   FILE EXPERSON
   END

```

1. The request must re-map the field names and formats in the EXPERSON data source to match those used in the main request.
2. The main request names the first data source in the concatenation, EMPLOYEE. It also defines the print and sort fields for both data sources.
3. The MORE phrase starts the subrequest that concatenates the next data source, EXPERSON. No display commands are allowed in the subrequest. IF and WHERE criteria are the only report components permitted in a subrequest.

Field Name and Format Matching

All fields referenced in the main request must either exist with the same names and formats in all the concatenated files, or be remapped to those names and formats using virtual fields. Referenced fields include those used in COMPUTE commands, headings, aggregation phrases, sort phrases, and the PRINT, LIST, SUM, COUNT, WRITE, or ADD commands.

A successful format match means that:

Usage Format Type	Correspondence
A	Format type and length must be equal.
I, F, D	Format type must be the same.
P	Format type and scale must be equal.
DATE (new)	Format information (type, length, components, and order) must always correspond.
DATE (old)	Edit options must be the same.
DATE -TIME	Format information (type, length, components, and order) must always correspond.

Text (TX) fields and CLOB fields (if supported) cannot be concatenated.

**Example: Matching Field Names and Formats**

The following annotated example concatenates data from the EMPDATA and SALHIST data sources.

```

DEFINE FILE EMPDATA
1. NEWID/All=EDIT (ID, '999-99-9999');
END

DEFINE FILE SALHIST
2. NEWID/All=EDIT (ID, '999-99-9999');
   CSAL/D12.2M=OLDSALARY;
END

3. TABLE FILE EMPDATA
   HEADING
   "EMPLOYEE SALARIES"
   " "
   PRINT CSAL
   BY NEWID
4. WHERE CSAL GT 65000
5. MORE
   FILE SALHIST
6. WHERE OLDSALARY GT 65000
   END

```

1. Defines NEWID in the EMPDATA data source with the same name and format as the sort field referenced in the main request.
2. Defines NEWID in the SALHIST data source with the same name and format as the sort field referenced in the main request.
3. The main request. This contains all the formatting for the resulting report and names the first file to be concatenated. It also contains all printing and sorting information. The fields printed and the sort fields must exist as real or DEFINE fields in each file.
4. The WHERE criterion in the main request applies only to the EMPDATA data source.
5. The MORE phrase concatenates the SALHIST data source to the EMPDATA data source.
6. This WHERE criterion applies only to the SALHIST data source. Notice that it references a field that is not defined in the EMPDATA data source.

The output is:

EMPLOYEE SALARIES

NEWID

000-00-0030

SALARY

\$70,000.00

	\$70,000.00
000-00-0070	\$83,000.00
	\$83,000.00
	\$79,100.00
000-00-0200	\$115,000.00
	\$115,000.00
	\$102,500.00
	\$89,500.00
000-00-0230	\$80,500.00
	\$80,500.00
	\$75,000.00
	\$70,800.00
000-00-0300	\$79,000.00
	\$79,000.00
	\$75,000.00
	\$70,000.00

When you concatenate data, record sets are simply appended, not grouped or aggregated across files. Therefore, if duplicate sort fields exist, they show up twice in the report output.

Merging Concatenated Data Sources

You can use the MORE phrase in a MATCH request to merge up to 16 sets of concatenated data sources.

You must meet all MATCH requirements in the main request. All data sources to be merged must be sorted by at least one field with a common format.

The MATCH request results in a HOLD file containing the merged data. You can specify how you want each successive file merged using an AFTER MATCH command. For example, you can retain:

- ☐ All records from both files (OLD-OR-NEW). This is the default.
- ☐ Only records common to both files (OLD-AND-NEW).
- ☐ Records from the first file with no match in the second file (OLD-NOT-NEW).

- ☐ Records from the second file with no match in the first file (NEW-NOT-OLD).
- ☐ All non-matching records from both files; that is, records that were in either one of the files but not in both (OLD-NOR-NEW).
- ☐ All records from the first file with all matching records from the second file (OLD).
- ☐ All records from the second file with all matching records from the first file (NEW).

**Syntax:****How to Merge Concatenated Data Sources**

```

1. MATCH FILE file1main request
   MORE
2. FILE file2subrequest
   MORE
3. FILE file3subrequest
   RUN
4. FILE file4main request
5. [AFTER MATCH merge_phrase]
   MORE
6. FILE file5subrequest
   MORE
7. FILE file6subrequest
   RUN
8. FILE file7main request
9. [AFTER MATCH merge_phrase]
   MORE
10. FILE file8subrequest
    MORE
11. FILE file9subrequest
    END

```

1. Starts the first answer set in the MATCH. File1 is the first data source in the first answer set.
2. Concatenates file2 to file1 in the first MATCH answer set.
3. Concatenates file3 to file1 and file2 in the first MATCH answer set.
4. Starts the second answer set in the MATCH. File4 is the first data source in the second answer set.
5. All data concatenated in the first answer set is merged with the data concatenated in the second answer set using the AFTER MATCH merge\_phrase in the second answer set.
6. Concatenates file5 to file4 in the second MATCH answer set.
7. Concatenates file6 to file4 and file5 in the second MATCH answer set.
8. Starts the third answer set in the MATCH. File7 is the first data source in the third answer set.

9. All merged data from the first and second answer sets, now a HOLD file, is merged with the data concatenated in the third answer set using the AFTER MATCH merge\_phrase in the third answer set. This final set of merged data is stored in a HOLD file.

10. Concatenates file8 to file7 in the third MATCH answer set.

11. Concatenates file9 to file7 and file8 in the third MATCH answer set.

### Using Sort Fields in MATCH Requests

If the data sources in the MATCH share common high-order sort fields with identical names and formats, the MATCH process merges records with matching sort field values from each of the files. If the two data sources in the MATCH have the same sort field with different names, you can change one of the names with an AS phrase.

If the files in the MATCH do not share a high-order sort field, the fields are not compared. Instead, the fields from the first record in each data source are merged to create the first record in the HOLD file, and so on for all remaining records.



**Example: Merging Concatenated Data Sources With Common High-Order Sort Fields**

The following annotated sample stored procedure illustrates MATCH with MORE, using a common sort field:

```

1. DEFINE FILE EMPDATA
   CURR_SAL/D12.2M = CSAL;
   FIRST_NAME/A10 = FN;
   EID/A9 = PIN;
   END

   -*Start MATCH.

2. MATCH FILE EMPLOYEE
   SUM CURR_SAL AS 'CURRENT'
   FIRST_NAME AS 'FIRST'
   BY EID AS 'SSN'
   -*Concatenate file EMPDATA to EMPLOYEE to form first MATCH answer set.
3. MORE
   FILE EMPDATA
   RUN
   -*Second MATCH answer set:

4. FILE TRAINING
   PRINT EXPENSES
5. BY PIN AS 'SSN'
6. AFTER MATCH HOLD OLD-OR-NEW
   END

   -*Print merged file:

7. TABLE FILE HOLD
   PRINT *
   END

```

1. Defines the EMPDATA fields needed for concatenating it to EMPLOYEE.
2. Starts the MATCH and the main request in the concatenation. The main request defines all printing and sorting for the concatenated files. The sort field is called SSN in the resulting file.
3. Concatenates file EMPDATA to EMPLOYEE. This concatenated file becomes the OLD file in the MATCH.
4. Creates the NEW file in the MATCH.
5. Uses an AS phrase to change the name of the sort field in the NEW file to the same name as the sort field in the OLD file.
6. Defines the merge procedure. All records from the NEW file, the OLD file, and both files are included in the final HOLD file.
7. Prints the values from the merged file.

The first page of output is:

SSN ---	CURRENT -----	FIRST -----	EXPENSES -----
000000010	\$55,500.00	DANIEL	2,300.00
000000020	\$62,500.00	MICHAEL	.
000000030	\$70,000.00	LOIS	2,600.00
000000030	\$70,000.00	LOIS	2,300.00
000000040	\$62,500.00	RUTH	3,400.00
000000050	\$54,100.00	PETER	3,300.00
000000060	\$55,500.00	DORINA	.
000000070	\$83,000.00	EVELYN	.
000000080	\$43,400.00	PAMELA	3,200.00
000000080	\$43,400.00	PAMELA	3,350.00
000000090	\$33,000.00	MARIANNE	.
000000100	\$32,400.00	TIM	3,100.00
000000110	\$19,300.00	ANTHONY	1,800.00
000000110	\$19,300.00	ANTHONY	2,500.00
000000110	\$19,300.00	ANTHONY	2,400.00
000000120	\$49,500.00	KATE	2,200.00
000000130	\$62,500.00	MARCUS	.

**Example: Merging Concatenated Data Sources Without a Common Sort Field**

In this example, the merged data sources do not share a sort field:

```

DEFINE FILE EMPDATA
CURR_SAL/D12.2M = CSAL;
FIRST_NAME/A10 = FN;
EID/A9 = PIN;
END

-*Start MATCH

MATCH FILE EMPLOYEE
SUM CURR_SAL AS 'CURRENT'
    FIRST_NAME AS 'FIRST'
BY EID AS 'SSN'

-*Concatenate EMPDATA to EMPLOYEE to form the first MATCH answer set

MORE
FILE EMPDATA
RUN

-*Second MATCH answer set:

FILE TRAINING
PRINT EXPENSES
BY PIN AS 'EID'
AFTER MATCH HOLD OLD-OR-NEW
END

-*Print merged file:

TABLE FILE HOLD
PRINT *
END

```

The AS phrase changes the answer set. Since the sort fields no longer have the same names, the fields are merged with no regard to matching records.

The first page of output is:

SSN	CURRENT	FIRST	EID	EXPENSES
---	-----	-----	---	-----
000000010	\$55,500.00	DANIEL	000000010	2,300.00
000000020	\$62,500.00	MICHAEL	000000030	2,600.00
000000030	\$70,000.00	LOIS	000000030	2,300.00
000000040	\$62,500.00	RUTH	000000040	3,400.00
000000050	\$54,100.00	PETER	000000050	3,300.00
000000060	\$55,500.00	DORINA	000000080	3,200.00
000000070	\$83,000.00	EVELYN	000000080	3,350.00
000000080	\$43,400.00	PAMELA	000000100	3,100.00
000000090	\$33,000.00	MARIANNE	000000110	1,800.00
000000100	\$32,400.00	TIM	000000110	2,500.00
000000110	\$19,300.00	ANTHONY	000000110	2,400.00
000000120	\$49,500.00	KATE	000000120	2,200.00
000000130	\$62,500.00	MARCUS	000000140	3,600.00
000000140	\$62,500.00	VERONICA	000000150	3,400.00
000000150	\$40,900.00	KARL	000000160	1,000.00
000000160	\$62,500.00	ROSE	000000180	1,250.00
000000170	\$30,800.00	WILLIAM	000000190	3,150.00

Cartesian Product

Cartesian product enables you to generate a report containing all combinations of non-related records or data instances in a multi-path request. This means that if a parent segment has three child instances on one path and two child instances on another path, when CARTESIAN is ON a request that references the parent segment and both children generates 16 records. When CARTESIAN is OFF, the same request generates only three records.

For related information about controlling how selection tests are applied to child segments on independent paths, see [Selecting Records for Your Report](#) on page 219.

Syntax: How to Enable/Disable Cartesian Product

SET CARTESIAN = {[OFF](#)|[ON](#)}

where:

[OFF](#)

Disables Cartesian product. OFF is the default setting.

[ON](#)

Enables Cartesian product and generates all possible combinations of non-related records.

SET CARTESIAN may also be issued within a request.

Reference: Usage Notes for Cartesian Product

- ❑ Cartesian product is performed on the lowest segment common to all paths, whether or not a field in that segment is referenced.

- ☐ Short paths do not display in requests with Cartesian product.
- ☐ The SET CARTESIAN parameter is disabled when ACROSS is specified, and a warning message is issued.
- ☐ The SUM display command and the TOT. prefix operator have no effect on Cartesian product.
- ☐ SUM, COMPUTE, and WITHIN in combination with the PRINT display command are performed on the Cartesian product.
- ☐ ON TABLE COLUMN-TOTAL is automatically generated on the Cartesian product.
- ☐ NOSPLIT is disabled if specified in combination with the SET CARTESIAN parameter, and no warning message is issued.
- ☐ MATCH is not supported with the SET CARTESIAN parameter. A warning message is not issued if MATCH is requested, and the request is processed as if CARTESIAN is set to OFF.
- ☐ TABLEF is not supported with the SET CARTESIAN parameter.

**Example: Reporting With Cartesian Product**

When CARTESIAN is set to ON, the following multi-path request produces a report containing all possible combinations of models and standards for each car:

```
SET CARTESIAN=ON
TABLE FILE CAR
PRINT MODEL STANDARD
BY CAR
IF CAR EQ 'JAGUAR'
END
```

The output in an ASCII environment is:

<u>CAR</u>	<u>MODEL</u>	<u>STANDARD</u>
JAGUAR	V12XKE AUTO	4 WHEEL DISC BRAKES
	V12XKE AUTO	POWER STEERING
	V12XKE AUTO	RECLINING BUCKET SEATS
	V12XKE AUTO	WHITEWALL RADIAL PLY TIRES
	V12XKE AUTO	WRAP AROUND BUMPERS
	XJ12L AUTO	4 WHEEL DISC BRAKES
	XJ12L AUTO	POWER STEERING

XJ12L AUTO	RECLINING BUCKET SEATS
XJ12L AUTO	WHITEWALL RADIAL PLY TIRES
XJ12L AUTO	WRAP AROUND BUMPERS

When CARTESIAN is set to OFF (the default), the same request results in a report from the CAR data source containing a list of models and standards without logical relationships.

The output in an ASCII environment is:

<u>CAR</u>	<u>MODEL</u>	<u>STANDARD</u>
JAGUAR	V12XKE AUTO	4 WHEEL DISC BRAKES
	XJ12L AUTO	POWER STEERING
	.	RECLINING BUCKET SEATS
	.	WHITEWALL RADIAL PLY TIRES
	.	WRAP AROUND BUMPERS

## Formatting Reports: An Overview

---

To create an effective report, you need to account for:

- ❑ **Content.** The data in your report.
- ❑ **Formatting.** How to present that content to a reader in a way that achieves maximum impact.

There are many formatting options that you can use to give your report a professional appearance and affect how people read and interpret it. For example, you can control:

- ❑ The appearance of the data, which you can change to emphasize important values.
- ❑ The headings, footings, and other text with which you "frame" the data to give it context.
- ❑ The layout of the report on the page or screen, which you can adjust for different display environments and for audiences with different vision needs.

The following topics provide an overview of formatting tabular reports (including Financial Modeling Language reports) and free-form reports. For information about formatting graphs, see [Creating a Graph](#) on page 1657.

### In this chapter:

- ❑ [What Kinds of Formatting Can I Do?](#)
  - ❑ [How to Specify Formatting in a Report](#)
  - ❑ [Standard and Legacy Formatting](#)
  - ❑ [Techniques for Quick and Easy Formatting](#)
  - ❑ [Navigating From a Report to Other Resources](#)
- 

## What Kinds of Formatting Can I Do?

There are many kinds of formatting that you can apply to a report:

- ❑ **The appearance of the report data**, such as its font, size, style (italic, bold, or underlined), color (of the foreground and the background), position, and justification. You can also draw boxes or lines around data. You can use these properties to emphasize critical values and to draw attention to important data relationships.

You can also select which character to use to mark decimal position, using either a period (.) or a comma (,), to match the convention of the country in which the report will be read. You can even choose which character to use to represent a null value and missing data. For more information, see [Formatting Report Data](#) on page 1611.

- ❑ **Providing context for data by "framing" it** with headings, footings, and customized column titles. You can include fields and images within headings and footings. As with data, you can specify a heading, footing, and column title font, size, style, color, position, and justification, as well as enclose it within boxes or lines. You can use these framing devices to explain the context of the data and to engage the interest of the reader. For more information, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.
- ❑ **Laying out the report** on the screen or printed page. You can choose the report margins, where to place headings and footings, where to place background images (watermarks), and how to arrange the report columns (adjusting the space around and between columns, adjusting column width and column order, and even stacking one column above another to reduce report width). You can visually distinguish between different columns, rows, or sort groups using color and lines. If you wish, you can draw borders around parts of a report or around the entire report.

You can lay out the report to optimize it for different display environments such as screens of different sizes and resolutions, and printed pages of different sizes. You can create multiple report panes on a single page to print labels. You can even combine several reports into a single file to display or print them as a group. For more information, see [Laying Out the Report Page](#) on page 1249.

- ❑ **Conditionally formatting** a report based on the report data. You specify a condition that, at run time, is automatically evaluated for each instance of the report component you specify, such as each value of a sort column. The formatting option is applied to each instance of the report component for which the condition is true. For example, in a sales report, you can draw attention to sales staff who exceeded quota by making their names bold and using a different color. For more information, see [Controlling Report Formatting](#) on page 1139.
- ❑ **Choosing a display format, such as HTML (the default), PDF (Adobe Acrobat Portable Document Format), Excel 2000, or PostScript**, to suit the viewing and processing needs of the readers. For more information and a list of all the display formats available to you, see [Choosing a Display Format](#) on page 565.



- ❑ **Making a report accessible to all users** regardless of their physical abilities, their browser type, or their screen settings. For example, you can design a report fonts, colors, layout, and other formatting to make it easier to read by audiences with special vision needs, and provide text descriptions of tables and graphics to make their information accessible to people who use speech-based or Braille-based browsers. You can ensure that a report conforms to any accessibility guidelines, such as Section 508 of the U.S. Rehabilitation Act, to which the report is subject.

### *Example:* Advantages of Formatting a Report

The following pair of reports shows order number, order date, and total order revenue for Century Corporation in the third quarter of 2000. Compare the formatted version (on the left) with the unformatted version (on the right):



page 1

PAGE 1

<u>Date</u> <u>Of Order:</u>	<u>Order</u> <u>Number:</u>	<u>Order</u> <u>Total:</u>	<u>Date</u> <u>Of Order:</u>	<u>Order</u> <u>Number:</u>	<u>Line</u> <u>Total</u>
Oct 1, 00	79613	\$1,417.29	2000/10/01	79613	\$1,417.29
Oct 2, 00	78360	\$5,485.05	2000/10/02	78360	\$5,485.05
Oct 3, 00	79327	\$628,241.37	2000/10/03	79327	\$628,241.37
Oct 4, 00	78373	\$2,650.47	2000/10/04	78373	\$2,650.47
Oct 6, 00	74577	\$209,286.00	2000/10/06	74577	\$209,286.00
Oct 7, 00	88950	\$3,670.68	2000/10/07	88950	\$3,670.68
Oct 9, 00	68840	\$4,251.87	2000/10/09	68840	\$4,251.87
Oct 10, 00	68922	\$867,979.35	2000/10/10	68922	\$867,979.35
Oct 11, 00	68974	\$122,298.40	2000/10/11	68974	\$122,298.40

Consider how the formatting applied to the version on the left:

- ❑ **Catches the interest of the reader** with a heading and use of color.
- ❑ **Makes the significance of the report clearer** using the heading, and by changing the last column title from the default "Line Total" to "Order Total."
- ❑ **Makes the report easier and more appealing to read** by increasing the space between rows, by reformatting the order date, and by using proportional fonts.

- ❑ **Draws the attention of the reader to important data.** In this case, to orders exceeding \$500,000, by conditionally formatting these rows with background color, font color, and (for the order total) bold font style.

## How to Specify Formatting in a Report

You can specify your report formatting using a style sheet. A style sheet is a set of declarations that defines the appearance of a report. For some types of formatting, you may need to supplement style sheets with other features, such as SET parameters and TABLE commands. In each case, this manual describes everything required to achieve a given kind of formatting.

**Benefits of using style sheets.** For some types of formatting you can choose between using a style sheet or a different feature. Style sheets are usually preferred because they enable you to centralize and reuse formatting logic. This provides you with several advantages:

- ❑ **Productivity.** By using just a few lines of code (a single style sheet), you can format dozens of reports, reducing the development time of each report.
- ❑ **Easy maintenance.** You can change formatting for dozens of reports at one time by editing a single style sheet.
- ❑ **Consistent appearance.** Your enterprise can guarantee a consistent look for its reports by assigning the same style sheet(s) to them.
- ❑ **Rapid reformatting.** You can change the appearance of a report quickly and easily by switching the style sheet assigned to it.
- ❑ **Prioritizing.** You can focus on your first priority (report content), because you can quickly address report presentation by applying an existing style sheet.

There are different kinds of style sheets that you can use to format a report. You can learn about them and how to choose between them in [How to Choose a Type of Style Sheet](#) on page 1113.

**Example: Specifying Formatting for the Order Revenue Report**

This report displays the order number, order date, and total order revenue for Century Corporation for the third quarter of 2000:



page 1

<u>Date Of Order:</u>	<u>Order Number:</u>	<u>Order Total:</u>
Oct 1, 00	79613	\$1,417.29
Oct 2, 00	78360	\$5,485.05
Oct 3, 00	79327	\$628,241.37
Oct 4, 00	78373	\$2,650.47
Oct 6, 00	74577	\$209,286.00
Oct 7, 00	88950	\$3,670.68
Oct 9, 00	68840	\$4,251.87
Oct 10, 00	68922	\$867,979.35
Oct 11, 00	68974	\$122,298.40

The report is formatted by a WebFOCUS StyleSheet and by formatting commands in the report procedure itself. The procedure, Revenue.fex, is shown below, followed by the StyleSheet file, OrderRev.sty:

**Revenue.fex**

```
TABLE FILE CENTORD
1. HEADING
1. " "
1. "Century Corporation"
1. " "
1. "Order Revenue - 2000 Q3"
1. " "
1. "page <TABPAGENO"
1. " "
2. SUM ORDER_DATE/MtdY ORDER_NUM LINEPRICE AS 'Order,Total:'
   BY LOWEST 9 ORDER_DATE NOPRINT
   WHERE (ORDER_DATE GE '2000/10/01') AND (ORDER_DATE LE '2000/12/31');
   ON TABLE SET ONLINE-FMT PDF
3. ON TABLE SET SQUEEZE ON
4. ON TABLE SET STYLESHEET OrderRev
END
```

### OrderRev.sty

```
5. TYPE=Report, GRID=Off, UNITS=Inches, TOPGAP=0.06, BOTTOMGAP=0.06, $
6. TYPE=Data, FONT='Times', $
7. TYPE=Data, BACKCOLOR=Aqua, COLOR=Navy,
7.   WHEN=LinePrice GT 500000, $
7. TYPE=Data, COLUMN=LINEPRICE, BACKCOLOR=Aqua, COLOR=Navy, STYLE=Bold,
7.   WHEN=LinePrice GT 500000, $
8. TYPE=Title, FONT='Helvetica', $
9. TYPE=Heading, FONT='Helvetica', STYLE=Bold, SIZE=14, JUSTIFY=Center,
9.   BACKCOLOR=Dark Turquoise, COLOR=White, $
9. TYPE=Heading, LINE=6, BACKCOLOR=White, COLOR=Dark Turquoise, $
9. TYPE=Heading, LINE=7, BACKCOLOR=White, $
```

1. Adds a page heading to the report.
2. Reformats the order date from (for example) 2000/10/07 to Oct. 7, 00.
3. Aligns the heading with the report margins instead of the page margins.
4. Identifies a StyleSheet file to format the report.
5. Increases spacing between report lines.
6. Uses a proportional serif font for the report data.
7. Highlights each order that totals more than \$500,000 by applying a navy font and an aqua background, and by bolding the order total.
8. Uses a proportional sans serif font for the report column titles.
9. Formats the report heading by centering it, applying a larger sans serif font, coloring most of it with a dark turquoise background and white lettering, and applying the inverse coloring to the page number (the sixth line of the heading).

This is only a summary of what these formatting instructions do. You can find complete explanations in the topics that describe each formatting feature.

The formatting logic that you apply to your own reports may be briefer or more extensive than this example, depending on the report and on what formatting you choose to apply.

## How to Choose a Type of Style Sheet

You can choose between two types of style sheets to format a report:

- ❑ **WebFOCUS StyleSheets** (often abbreviated to "StyleSheets"), the native WebFOCUS style sheet language. These provide you with the flexibility to format reports in many display formats, including HTML, PDF, Excel 2000, and PostScript. You can choose between saving the StyleSheet as a separate file, which you can assign to multiple reports, or saving it within one report request.

If you are generating a report in HTML format, you can boost its performance, and increase the number of formatting options available to it, by having the WebFOCUS StyleSheet dynamically generate an "internal" cascading style sheet (CSS). (CSS is the standard style sheet language designed for HTML documents. The *internal* CSS generated by WebFOCUS is internal to the report output, instead of being saved as a separate file.) For more information about generating an internal cascading style sheet, see [Generating an Internal Cascading Style Sheet for HTML Reports](#) on page 1140.

- ❑ **External cascading style sheets**, the standard style sheet language designed for HTML documents. You can apply an external cascading style sheet to any WebFOCUS report in HTML format. (An *external* cascading style sheet is one that is saved as a separate file, instead of within the document it formats, and so is "external" to the document.)

How do you choose between the two types of style sheets? Consider choosing:

- ❑ **A WebFOCUS StyleSheet** if:

**You want to display a report in different display formats**, such as PDF and Excel 2000. WebFOCUS StyleSheets are supported for many kinds of display formats, but cascading style sheets work for reports in HTML format only.

- ❑ **An external cascading style sheet** for any of the following reasons:

**Your enterprise already uses cascading style sheets** to format HTML documents, and it wants reports to conform to these same presentation guidelines.

**You want to apply the same formatting to other kinds of HTML documents** in your enterprise.

## Standard and Legacy Formatting

New releases of WebFOCUS often introduce improved ways of formatting reports. Some of these new features are advances over earlier features that performed similar formatting, but with fewer options or less functionality. The new feature becomes the standard, and the earlier one is then considered a "legacy" feature.

When you create a new application and have a choice between using a standard or a legacy feature, we encourage you to use the standard feature. When you are maintaining an earlier application that incorporates a legacy feature, you may choose to retain the legacy feature to save time, or to convert the application to the standard feature in order to leverage its new functionality.

As an example of the difference between standard and legacy formatting, consider the standard and legacy methods of laying out a report on a page:

- ❑ **The standard way** is to specify the page margins using TOPMARGIN, BOTTOMMARGIN, LEFTMARGIN, and RIGHTMARGIN. (You can apply these keywords as StyleSheet attributes or SET command parameters.) You can specify the margins in inches, centimeters, or points, as determined by UNITS (which you can also issue as a StyleSheet attribute or a SET command parameter). This is simple, and enables you to design reports the same way that you design other kinds of documents.
- ❑ **The legacy way** is to specify the height of the report output on the page, measured in report lines (using the LINES parameter of the SET command); and the width of the report, measured in characters (using the WIDTH parameter of the SET command). The top and bottom margins will each be half the difference of the page height and the report height, measured in character lines; the left and right margins will each be half the difference of the page width and the report width, measured in characters. This legacy method limits you to using a monospace font, such as Courier.

## Techniques for Quick and Easy Formatting

You can apply several formatting techniques to save yourself time and effort. Most of these techniques enable you to use code provided for you by WebFOCUS, or to leverage code that you write yourself:

- ❑ **Inheritance and overrides.** Each report component inherits StyleSheet attributes from its "parent" report component. This powerful feature lets you define common formatting in a single declaration for a parent component, and lets descendant components automatically inherit the formatting, while enabling you to override the inherited values when you wish. By designing your StyleSheet to take advantage of inheritance, you can write less code and can quickly update formatting for multiple report components.

For example, if you declare all the report data to be blue, all data in all columns will be displayed as blue. If you also declare all vertical sort (BY) columns to be orange, this will override the blue for sort columns, which will be displayed as orange. If you also declare the EMP\_ID sort column green, it will override the orange and be displayed as green. For more information, see [WebFOCUS StyleSheet Attribute Inheritance](#) on page 1127.

- ❑ **Macros.** If you are going to specify the same attribute and value in several declarations in a StyleSheet, you can create a "macro" that enables you to apply the attribute repeatedly throughout the StyleSheet without coding it each time. Then, if you ever need to change the value, you can change it once (in the macro), and have the change applied automatically throughout the StyleSheet.

For example, if there are several parts of a report that you wish to emphasize (such as, titles of important columns, data values that exceed a threshold, and sort headings), and you want all of these to be bold and purple, you could define a macro that sets font style to bold and color to purple, and then apply the macro to all of these report components. For more information, see [Reusing WebFOCUS StyleSheet Declarations With Macros](#) on page 1124.

- ❑ **Defaults.** Many WebFOCUS StyleSheet attributes have default values. Instead of explicitly specifying every StyleSheet attribute, you can omit some and accept their defaults. For example, you can accept the default font instead of specifying a font. You can find each default value of an attribute documented where its syntax is described.

## Navigating From a Report to Other Resources

You can enable someone reading a report to navigate to other reports and Internet resources, and even to navigate within the report itself. Although navigation is not considered formatting, you can support some kinds of navigation by using a StyleSheet.

You can enable someone reading a report to:

- ❑ **Drill down** to a related report. For more information, see [Linking to Another Report](#) on page 764.
- ❑ **Link to a webpage**, compose and send an email message, and connect to other kinds of Internet resources. For more information, see [Linking to a URL](#) on page 769.
- ❑ **Execute JavaScript functions** to perform additional analysis of report data. For more information, see [Linking to a JavaScript Function](#) on page 776.

- ❑ **Use a table of contents to jump** directly to the data that interests that reader. The report generates the table of contents dynamically based on sort values, and enables the reader to see any report section, or the entire report, simply by selecting it from the table of contents. For more information, see [Navigating Sort Groups From a Table of Contents](#) on page 905.
- ❑ **Jump from one report page to the next**, to the top of the current report page, or to the beginning or end of the report. For more information, see [Linking Report Pages](#) on page 958.



A StyleSheet enables you to format and produce attractive reports that highlight key information. With StyleSheets, you can specify various characteristics of your report and format report components individually.

You can use a StyleSheet to:

- ☐ Format report components individually.
- ☐ Incorporate graphical elements.
- ☐ Define dynamic hyperlinks.
- ☐ Format data that meets specified conditions.
- ☐ Create macros that enable you to streamline your formatting specifications.

You can also use external cascading style sheets and you can enable internal cascading style sheets for HTML reports. For details, see [Using an External Cascading Style Sheet](#) on page 1211 and [Controlling Report Formatting](#) on page 1139.

Unless otherwise noted, all StyleSheet references in this chapter refer to WebFOCUS StyleSheets.

**In this chapter:**

- ☐ [Creating a WebFOCUS StyleSheet](#)
  - ☐ [General WebFOCUS StyleSheet Syntax](#)
  - ☐ [Reusing WebFOCUS StyleSheet Declarations With Macros](#)
  - ☐ [WebFOCUS StyleSheet Attribute Inheritance](#)
  - ☐ [Creating Reports With the ENWarm StyleSheet](#)
- 

## Creating a WebFOCUS StyleSheet

You can create a StyleSheet:

- ☐ **Within a report request**, as an inline StyleSheet. This is useful when you need to apply a StyleSheet to only one report. For details, see [Creating a WebFOCUS StyleSheet Within a Report Request](#) on page 1118.

- ❑ **Outside of a report request**, as a separate file. This enables you to apply one StyleSheet to multiple reports. For details, see [Creating and Applying a WebFOCUS StyleSheet File](#) on page 1120.

**Note:** You can also include a StyleSheet file in another StyleSheet. This enables you to apply the styles in the included StyleSheet file, but override specific attributes. For information, see [How to Include a StyleSheet File in Another StyleSheet](#) on page 1119.

### Creating a WebFOCUS StyleSheet Within a Report Request

You can create a StyleSheet within a report request. This enables you to create and maintain the formatting for your report directly in the report request. This type of StyleSheet is known as an inline StyleSheet.

#### **Syntax:** How to Create a WebFOCUS StyleSheet Within a Report Request

```
ON TABLE SET STYLE[SHEET] *  
declaration  
[declaration]  
.  
.  
.  
[ENDSTYLE]
```

where:

**SHEET**

Can be omitted to make the command shorter, and has no effect on its behavior.

**declaration**

Is a StyleSheet declaration. StyleSheet declarations usually specify the report component you want to format and the formatting you want to apply. For more information about declarations, see [General WebFOCUS StyleSheet Syntax](#) on page 1122.

**ENDSTYLE**

Indicates the end of an inline StyleSheet. You can omit ENDSTYLE if it is followed immediately by END in the report request.

#### **Example:** Creating a WebFOCUS StyleSheet Within a Report Request

The following illustrates an inline StyleSheet. The StyleSheet is highlighted in the request.

```

TABLE FILE GGSALES
SUM UNITS DOLLARS BY CATEGORY BY PRODUCT
HEADING
"Sales Report"
FOOTING CENTER
"***End of Report***"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=HEADING, FONT=ARIAL, SIZE=12, STYLE=BOLD, $
TYPE=TITLE, STYLE=ITALIC, $
TYPE=DATA, COLUMN=N1, STYLE=BOLD, COLOR=BLUE, $
TYPE=FOOTING, COLOR=RED, STYLE=BOLD, $
ENDSTYLE
END

```

The output is:

## Sales Report

<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
<b>Coffee</b>	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
<b>Food</b>	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
<b>Gifts</b>	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

**\*\*\*End of Report\*\*\***

**Syntax:** How to Include a StyleSheet File in Another StyleSheet

```
INCLUDE = stysheet,$
```

where:

*stysheet*

Is the StyleSheet file to include.

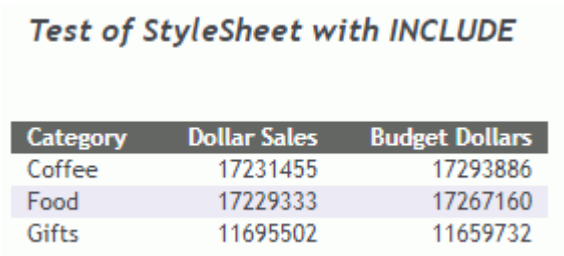
StyleSheet declarations are applied in the order in which they are found in the StyleSheet. Therefore, if you want to include a StyleSheet file and then override some of the attributes within it, place the INCLUDE statement first, then the declarations that override specific attributes below it.

**Example:** Including a StyleSheet File in Another StyleSheet

The following request includes one of the distributed WebFOCUS StyleSheet Sin the inline report StyleSheet and overrides the heading style to be bold and italic.

```
HEADING CENTER
"Test of Stylesheet with Include"
" "
SUM DOLLARS BUDDOLLARS
BY CATEGORY
ON TABLE HOLD AS STYLE2 FORMAT HTML
ON TABLE SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/
ENIADefault_combine.sty,$
TYPE=HEADING, STYLE = BOLD+ITALIC,$
END
```

The output is shown in the following image.



Test of StyleSheet with INCLUDE		
Category	Dollar Sales	Budget Dollars
Coffee	17231455	17293886
Food	17229333	17267160
Gifts	11695502	11659732

**Creating and Applying a WebFOCUS StyleSheet File**

You can create a StyleSheet as a separate file and apply it to as many reports as you wish. A StyleSheet file contains only declarations and optional comments. Unlike an inline StyleSheet, a StyleSheet file does not contain the ON TABLE SET STYLESHEET and ENDSTYLE commands. You can apply a StyleSheet file to a report using the SET STYLESHEET command, as described in [How to Apply a WebFOCUS StyleSheet File to a Report](#) on page 1121. For information about StyleSheet declarations, see [General WebFOCUS StyleSheet Syntax](#) on page 1122.

As an alternative to creating a new StyleSheet file, you can use one of the sample StyleSheet files provided with WebFOCUS as a template.

Whether you create a StyleSheet file, or copy and customize an existing one, you need to store it in the correct location, as described in [Naming and Storing a WebFOCUS StyleSheet File](#) on page 1121.

**Reference:** **Naming and Storing a WebFOCUS StyleSheet File**

When you create a StyleSheet file to be used in:

- ❑ **WebFOCUS**, upload the file to the WebFOCUS repository to a folder in which you are permitted to create content, and that users running procedures that reference the StyleSheet are permitted to read.

If you create a StyleSheet for a self-service application, you can deploy your StyleSheet file to the apps\baseapp directory where it can be shared by multiple applications. You can also deploy the StyleSheet file to the same location as the report procedure it works with if you are only applying the file to that particular procedure.

You should name a StyleSheet file *filename.sty*, where *filename* can include letters, numbers, and underscores (\_), and otherwise must be valid for the operating environments on which it resides.

**Syntax:** **How to Apply a WebFOCUS StyleSheet File to a Report**

To apply your StyleSheet file at the beginning of your report request, use

```
SET STYLE[SHEET] = stylesheet
```

To apply your StyleSheet file within your report request use

```
ON TABLE SET STYLE[SHEET] stylesheet
```

where:

*SHEET*

Can be omitted to make the command shorter, and has no effect on its behavior.

*stylesheet*

Is the name of the StyleSheet file. Do not include the file extension.

## General WebFOCUS StyleSheet Syntax

A StyleSheet consists of declarations that identify the report components you wish to format and the formatting you wish to apply. A declaration usually begins with the TYPE attribute and is followed by the *attribute=value* pairs you assign to the report component. You can also include comments that provide context for your StyleSheet. Comments do not affect StyleSheet behavior. For details, see [Adding a Comment to a WebFOCUS StyleSheet](#) on page 1124.

For information about identifying a report component, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

### **Syntax:** How to Specify a WebFOCUS StyleSheet Declaration

Each StyleSheet declaration specifies a series of attributes in the form

```
attribute = value, [attribute = value, ...] $
```

where:

*attribute*

Is the attribute you are specifying, such as TYPE, COLUMN, COLOR, or FONT.

*value*

Is the value you assign to the attribute.

### **Example:** Sample WebFOCUS StyleSheet

Following is a request that includes an inline StyleSheet. The StyleSheet begins with ON TABLE SET STYLE \* and ends with ENDSTYLE.

```

TABLE FILE CENTORD
HEADING
" "
"Century Corporation"
" "
"Order Revenue - 2000 Q3"
" "
"page <TABPAGE NO"
" "
SUM ORDER_DATE/MtDY ORDER_NUM LINEPRICE AS 'Order,Total:'
BY LOWEST 9 ORDER_DATE NOPRINT
WHERE (ORDER_DATE GE '2000/10/01') AND (ORDER_DATE LE '2000/12/31');
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET HTMLCSS ON

ON TABLE SET STYLESHEET *
TYPE=Report, GRID=Off, UNITS=Inches, $
TYPE=Data, FONT='Times', $
TYPE=Data, BACKCOLOR=Aqua, COLOR=Navy,
    WHEN=LinePrice GT 500000, $
TYPE=Data, COLUMN=LinePrice, BACKCOLOR=Aqua, COLOR=Navy, STYLE=Bold,
    WHEN=LinePrice GT 500000, $
TYPE=Title, FONT='Helvetica', $
TYPE=Heading, FONT='Helvetica', STYLE=Bold, SIZE=14, JUSTIFY=Center,
    COLOR=White, BACKCOLOR=Dark Turquoise, $
TYPE=Heading, LINE=6, BACKCOLOR=White, COLOR=Dark Turquoise, $
TYPE=Heading, LINE=7, BACKCOLOR=White, $
ENDSTYLE

END

```

## Improving WebFOCUS StyleSheet Readability

There are many ways to structure your StyleSheet declarations in order to make the StyleSheet easy to read. You can do any one, or a combination, of the following:

- ☐ Begin a declaration in any column using blank spaces or tabs.
- ☐ Include blank lines between declarations.
- ☐ Create declarations in all uppercase, all lowercase, or mixed case.
- ☐ Use more than one declaration to format a single report component.
- ☐ Include blank spaces or tabs in between the attribute, equal sign (=), value, comma, and dollar sign (\$).

- ❑ Split a single declaration across a line. The declaration will continue to be processed until the terminating dollar sign. For example, you can split a declaration like this:

```
TYPE=HEADING, FONT=ARIAL,  
SIZE=14, STYLE=BOLD, $
```

- ❑ Split an *attribute=value* pair across a line. Use the backslash (\) character as continuation syntax at the end of the first line if you are splitting an attribute or value in a declaration across a line. For example:

```
TYPE=TITLE, COLUMN=N2, STY\  
LE=BOLD+ITALIC, COLOR=BLUE, $
```

### Adding a Comment to a WebFOCUS StyleSheet

You can add comments to a StyleSheet to give context to a declaration. Comments do not affect StyleSheet behavior.

You can add a comment:

- ❑ **On a declaration line.** Add the desired text after the dollar sign (\$). For example,

```
TYPE=HEADING, STYLE=BOLD, COLOR=BLUE, SIZE=14, $ Sample comment
```

- ❑ **On its own line.** Begin the line with either a dollar sign (\$), or a hyphen and an asterisk (-\*), followed by the desired text. For example,

```
-* This is a sample comment  
$ This is another sample comment
```

**Note:** You can add comments anywhere in your request, not only in StyleSheets.

### Reusing WebFOCUS StyleSheet Declarations With Macros

If you frequently use a group of attributes within a StyleSheet declaration, you can create a StyleSheet macro that groups the sequence of attributes together, enabling you to apply them repeatedly throughout the StyleSheet without recoding them.

### Defining a WebFOCUS StyleSheet Macro

A StyleSheet macro must be defined in the StyleSheet that references it and the macro definition must precede its use in the StyleSheet.

To define a macro, use the DEFMACRO attribute followed by the desired styling attributes.



**Syntax:**      **How to Define a WebFOCUS StyleSheet Macro**

```
DEFMACRO = macroname, attribute1 = value1, [attribute2 = value2,]... $
```

where:

*macroname*

Is the name you assign to the macro you are creating.

*attribute*

Is any StyleSheet attribute, such as an attribute to format a report component, insert a graphic, define a hyperlink, or apply a condition for conditional formatting (WHEN).

*value*

Is the value you want to assign to the attribute.

**Applying a WebFOCUS StyleSheet Macro**

A StyleSheet macro applies all the formatting defined in the macro to the report component specified in the declaration. To apply a macro, use the MACRO attribute. You can apply one macro per declaration.

When applying a StyleSheet macro to a report component, you can override any attribute defined in the macro by specifying the same attribute with the new value in that declaration, following the MACRO attribute.

**Syntax:**      **How to Apply a WebFOCUS StyleSheet Macro**

```
TYPE=type, [subtype,] MACRO=macroname, [condition,] $
```

where:

*type*

Is the report component you wish to affect. You can specify any report component.

*subtype*

Are any additional attributes, such as COLUMN, ACROSS, or ITEM, that are needed to identify the report component to which you are applying the macro. For information about how to specify different types of report components, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

*macroname*

Is the name of the macro to apply to the specified report component. The macro must be defined in the same StyleSheet.

*condition*

Is an optional WHEN attribute that you can specify if you wish to make this declaration conditional. For information about conditional declarations, see [Controlling Report Formatting](#) on page 1139.

**Example: Defining, Applying, and Overriding a WebFOCUS StyleSheet Macro**

The following annotated example illustrates how to define, apply, and override macros in your StyleSheet:

```
TABLE FILE GGSALES
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT
HEADING
"Sales Report"
FOOTING
"Sales Report - Page <TABPAGENO>"
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF,$
1.  DEFMACRO=A, STYLE=BOLD, SIZE=12, $
2.  DEFMACRO=BI, STYLE=BOLD+ITALIC, COLOR=PURPLE, $
3.  TYPE=HEADING, MACRO=A, $
4.  TYPE=FOOTING, MACRO=BI, COLOR=BLACK, $
5.  TYPE=DATA, COLUMN=N1, MACRO=BI, $
ENDSTYLE
END
```

1. Defines the A macro.
2. Defines the BI macro.
3. Illustrates how the A macro is applied to the heading.
4. Illustrates how the BI macro is applied to the footing and is partially overridden by the attribute value pair COLOR=BLACK.
5. Illustrates how the BI macro is applied to the data in the BY sort field CATEGORY (specified by TYPE=DATA, COLUMN=N1).

The output is:

## Sales Report

<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
<i>Coffee</i>	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
<i>Food</i>	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
<i>Gifts</i>	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

*Sales Report - Page 1*

## WebFOCUS StyleSheet Attribute Inheritance

Each report component inherits StyleSheet attributes from its parent component. You can override an inherited attribute by explicitly specifying the same attribute with a different value in the declaration for the child component. Since each component inherits automatically, you need specify only those attributes that differ from, or that augment, the inherited attributes of a component.

Inheritance enables you to define common formatting in a single declaration, and to apply it automatically to all child components, except for those components for which you specify different attribute values to override the inherited values. You benefit from less coding and a more concise StyleSheet.

For example, you could specify that all report titles should be blue and bold:

```
TYPE=TITLE, COLOR=BLUE, STYLE=BOLD, $
```

Each column title will inherit this formatting, appearing in blue and bold by default. However, you can choose to format one column differently, allowing it to inherit the blue color, but specifying that it override the bold style and that it add a yellow background color:

```
TYPE=TITLE, COLUMN=N2, STYLE=-BOLD, BACKCOLOR=YELLOW, $
```

### **Reference:** WebFOCUS StyleSheet Inheritance Hierarchy

Report components inherit StyleSheet attributes according to a hierarchy. The root of the hierarchy is the entire report, specified in a StyleSheet declaration by TYPE=REPORT. Declarations that omit TYPE default to TYPE=REPORT and are also applied to the entire report. Attributes that are unspecified for the entire report default to values that are determined according to the display format of the report, such as HTML or PDF.

Each report component inherits from its parent component. Component X is a parent of component Y if X is specified by a subset of all the "type" attributes that specify Y, and if those shared type attributes have the same values. For example,

- ❑ A component specified by TYPE=x, subtype=y, elementtype=z is a child of the component specified by TYPE=x, subtype=y and inherits attributes from it.
- ❑ The component specified by TYPE=x, subtype=y is a child of the component specified by TYPE=x, and inherits from it.
- ❑ The component specified by TYPE=x, where x is any value other than REPORT, is a child of the entire report (TYPE=REPORT) and inherits from it.
- ❑ The component specified by TYPE=DATA, subtype=z does not inherit from TYPE=REPORT, subtype=z. The rule is that a style for individual components (subtype=z) can only inherit from a style with the same type. The only exception is that any style can inherit from the top-level TYPE=REPORT component (the default style that has only TYPE=REPORT, with no other components, such as COLUMN, and so on).

For example, in the following syntax, by defining specific styling for the column at the DATA level, the overall style from TYPE=REPORT will be applied (FONT=TAHOMA), but not TYPE=REPORT, COLUMN=N2 (COLOR=GREEN). To apply the color green to the data in column 2, you have to explicitly make it green.

```
TYPE=REPORT, FONT=TAHOMA, $
TYPE=REPORT, COLUMN=N2, COLOR=GREEN, $
TYPE=DATA, COLUMN=N2, STYLE=+UNDERLINE+BOLD, $
```

When you use an external cascading style sheet (CSS), a report component inherits formatting from parent HTML elements, not from a parent report component. For more information, see [Inheritance and External Cascading Style Sheets](#) on page 1231.

### **Example:** Augmenting Inherited WebFOCUS StyleSheet Attributes

The following illustrates how to augment inherited StyleSheet attributes. The StyleSheet declarations discussed in this example are highlighted in the report request.

The page heading in this report has two lines. The first StyleSheet declaration identifies the report component HEADING to be formatted in bold and have 12-point font size. This will format both lines of the heading with these styles.

To augment the format for the second line of the heading, a second declaration has been added that specifies the heading line number and the additional style characteristic. In this case we have added the declaration TYPE=HEADING, LINE=2, STYLE=ITALIC. The second line of the heading will inherit the bold style and 12-point font size from the first HEADING declaration, and will also receive the italic style defined in the second declaration.

```
TABLE FILE GGSALES
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT
HEADING
"Sales Report:"
"First Quarter"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=HEADING, STYLE=BOLD, SIZE=12, $
TYPE=HEADING, LINE=2, STYLE=ITALIC, $
ENDSTYLE
END
```

The output is:

## Sales Report:

### *First Quarter*

<u>Category Product</u>		<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

**Example: Overriding Inherited WebFOCUS StyleSheet Attributes**

The following illustrates how to override StyleSheet inheritance. The StyleSheet declarations discussed in this example are highlighted in the report request.

```
TABLE FILE GGSALES
HEADING
"Sales Report"
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT BY DATE NOPRINT
WHERE DATE GE 19960101 AND DATE LE 19960401
ON TABLE SET STYLEMODE PAGED
ON TABLE SET LINES 20
FOOTING
"Page <TABPAGENO of <TABLASTPAGE"
ON TABLE SET STYLE *
    TYPE=REPORT, GRID=OFF, $
1. TYPE=REPORT, BACKCOLOR=BLUE, COLOR=WHITE, $
2. TYPE=HEADING, BACKCOLOR=WHITE, COLOR=BLACK, STYLE=BOLD, SIZE=12, $
3. TYPE=FOOTING, SIZE=11, STYLE=BOLD+ITALIC, BACKCOLOR=WHITE,
    COLOR=BLACK, $
4. TYPE=FOOTING, OBJECT=FIELD, ITEM=1, STYLE=-ITALIC, $
ENDSTYLE
END
```

1. Formats the entire report (all components) to appear with a blue background and white font.
2. Overrides the inherited format for the page heading (defined in the TYPE=REPORT declaration) by specifying the background color as white and the font as black.
3. Formats the page footing as font size 11, with a bold and italic style and overrides the report color by specifying BACKCOLOR=WHITE and COLOR=BLACK.
4. Since the <TABPAGENO system variable is part of the page footing, it inherits all of the formatting specified in the first TYPE=FOOTING declaration. This declaration overrides the inherited format for the page footing by specifying OBJECT=FIELD, ITEM=1, and removing the italic style (STYLE=-ITALIC). Note that ITEM=1 needs to be specified since there are two embedded fields in the footing.

The output is:

## Sales Report

Category	Product	Unit Sales	Dollar Sales
Coffee	Capuccino	5507	71728
		5219	67429
		6343	87290
		4170	58692
	Espresso	14499	182900
		14015	169023
		14871	184474
		11613	156396
	Latte	31469	401873
		38725	490645
		42717	520948
		36374	468137

***Page 1 of 4***

## Creating Reports With the ENWarm StyleSheet

The ENWarm StyleSheet (ENWarm.sty) is the new default style sheet for the WebFOCUS toolset. Using a clean and simple layout and design, you can take advantage of this style sheet when working with styling options for your reports and charts. Consistent text and color schemes, as well as a common look and feel, provide you with predictability across all of your reporting and charting activities.

In addition, there is an ENFlat StyleSheet, which offers the look and feel of ENWarm, with a different color scheme. It is recommend that you use this StyleSheet with active reports.

This section explains the specifications of the ENWarm StyleSheet and how it applies to report styling, and active reports.

Report Styling

The following topic explains the specifications of the ENWarm StyleSheet and how it applies to report styling.

Data, Report, and Title Styling

Styling for the data, report, and title elements are shown in the following image.

Product Category	Product Subcategory	Revenue	Quantity Sold	Gross Profit
<a href="#">Accessories</a>	Charger	\$4,022,834.91	105,257	\$1,970,123.91
	Headphones	\$76,186,587.97	228,349	\$24,523,023.97
	Universal Remote Controls	\$49,398,915.65	178,061	\$13,361,292.65
<a href="#">Camcorder</a>	Handheld	\$41,970,570.97	250,167	\$21,393,654.97
	Professional	\$44,053,830.75	12,872	\$8,835,522.75
	Standard	\$68,441,300.52	192,205	\$19,369,667.52

Title

☐ Bold

Report

☐ Arial

☐ 9 pt

☐ Font color: RGB (20, 20, 20)

☐ Title line skip

☐ Hyperlink color: RGB (51, 102, 255)

☐ Page color: White

Data

☐ Top border: RGB (219, 219, 219)

☐ Bottom border: RGB (219, 219, 219)

☐ Top gap: .05

☐ Bottom gap: .05



## Headings and Footings Styling

The headers and footers are defined in your report and pages, as shown in the following image.

Product Category	Product Subcategory	Revenue	Quantity Sold	Gross Profit
<a href="#">Accessories</a>	Charger	\$4,022,834.91	105,257	\$1,970,123.91
	Headphones	\$76,186,587.97	228,349	\$24,523,023.97
	Universal Remote Controls	\$49,398,915.65	178,061	\$13,361,292.65
<a href="#">Camcorder</a>	Handheld	\$41,970,570.97	250,167	\$21,393,654.97
	Professional	\$44,053,830.75	12,872	\$8,835,522.75
	Standard	\$68,441,300.52	192,205	\$19,369,667.52

The ENWarm StyleSheet style settings are as follows:

### Heading

- ☐ 14 pt
- ☐ Bold
- ☐ Font color: RGB (75, 75, 75)
- ☐ Left justify

### Page Heading

- ☐ 12 pt
- ☐ Bold
- ☐ Font color: RGB (75, 75, 75)
- ☐ Left justify

### Page Footing

- ☐ 10 pt
- ☐ Font color: RGB (102, 102, 102)

### Report Footing

- ☐ 10 pt

- ❑ Font color: RGB (102, 102, 102)

**Subheading and Subfooting Styling**

Styling for the subheading element is shown in the following image.

Product Subcategory	Revenue	Quantity Sold	Gross Profit
Subhead: Accessories			
Charger	\$4,022,834.91	105,257	\$1,970,123.91
Headphones	\$76,186,587.97	228,349	\$24,523,023.97
Universal Remote Controls	\$49,398,915.65	178,061	\$13,361,292.65
Subhead: Camcorder			
Handheld	\$41,970,570.97	250,167	\$21,393,654.97
Professional	\$44,053,830.75	12,872	\$8,835,522.75
Standard	\$68,441,300.52	192,205	\$19,369,667.52

**Subheading**

- ❑ Background color: RGB (246, 246, 246)
- ❑ Top border: RGB (219, 219, 219)

**Subheading Data**

- ❑ Bold

Styling for the subfooting element is shown in the following image.

Product Subcategory	Revenue	Quantity Sold	Gross Profit
Charger	\$4,022,834.91	105,257	\$1,970,123.91
Headphones	\$76,186,587.97	228,349	\$24,523,023.97
Universal Remote Controls	\$49,398,915.65	178,061	\$13,361,292.65
Subfoot: Accessories			
Handheld	\$41,970,570.97	250,167	\$21,393,654.97
Professional	\$44,053,830.75	12,872	\$8,835,522.75
Standard	\$68,441,300.52	192,205	\$19,369,667.52
Subfoot: Camcorder			

### Subfooting Data

☐ Bold

### Across Styling

Across styling for a report is shown in the following image.

		2009					
		Sale,Year					
		Sale,Quarter	1	2	3	4	TOTAL
Product Category	Product Subcategory						
Accessories	Charger	\$5,181.41	\$4,589.43	\$5,317.37	\$1,951.83	\$17,040.04	
	Headphones	\$74,057.43	\$80,921.68	\$101,399.64	\$40,590.68	\$296,969.43	
	Universal Remote Controls	\$62,536.37	\$54,652.87	\$51,125.55	\$17,227.14	\$185,541.93	
Subtotal: Accessories		\$141,775.21	\$140,163.98	\$157,842.56	\$59,769.65	\$499,551.40	
Media Player	Blu Ray	\$295,079.56	\$277,242.69	\$286,701.67	\$109,739.83	\$968,763.75	
	DVD Players	\$23,668.08	.	.	.	\$23,668.08	
	Streaming	\$5,059.35	\$4,343.45	\$4,855.37	\$3,100.56	\$17,358.73	
Subtotal: Media Player		\$323,806.99	\$281,586.14	\$291,557.04	\$112,840.39	\$1,009,790.56	
Stereo Systems	Home Theater Systems	\$110,372.82	\$111,928.78	\$103,813.67	\$40,813.60	\$366,928.87	
	Receivers	\$57,633.53	\$71,127.68	\$64,601.26	\$30,004.08	\$223,366.55	
	Speaker Kits	\$131,257.54	\$142,409.55	\$114,713.56	\$65,217.46	\$453,598.11	
	iPod Docking Station	\$52,944.39	\$56,255.68	\$46,783.34	\$16,516.95	\$172,500.36	
Subtotal: Stereo Systems		\$352,208.28	\$381,721.69	\$329,911.83	\$152,552.09	\$1,216,393.89	
TOTAL		\$817,790.48	\$803,471.81	\$779,311.43	\$325,162.13	\$2,725,735.85	

**Across Title**

- ☐ Right justify on outer BY field

**Across Data**

- ☐ Center justify

**Row Total Data**

- ☐ Bold

**Subtotal and Column Total Styling**

**Subtotal**

- ☐ Bold

**Column Total**

- ☐ Bold
- ☐ Top Border: RGB (102, 102, 102)

**Active Reports**

The following topic explains the specifications of the ENWarm StyleSheet and how it applies to active reports.

## Pagination, Menu, and Hover Text Styling in WebFOCUS Active Reports

Styling for pagination, menu, and hover text elements in WebFOCUS active reports are shown in the following image.

Product Category ▼	Model ▼	Revenue ▼	Quantity Sold ▼	
Accessories	Audio Technica ATHW5000	\$9,341,397.65	13,9	Sort Ascending
	B00D7MOHDO	\$2,514,622.50	52,6	Sort Descending
	BCG34HRE4KN	\$1,508,212.41	52,6	
	Denon AHD5000	\$9,272,133.77	13,8	Filter ▶
	Grado RS1	\$9,452,243.25	14,2	Calculate ▶
	Logitech 1100	\$9,301,960.89	27,7	Chart ▶
	Logitech 900	\$14,419,020.31	50,3	Rollup ▶
	Niles Audio RCAHT2	\$14,276,128.75	50,0	Pivot (Cross Tab) ▶
	Niles Audio RCATT2	\$11,401,805.70	49,9	Visualize
	Pioneer HDJ1000	\$8,028,218.25	49,7	Hide Column
	Sennheiser HD650	\$13,482,341.62	28,2	Grid Tool
	Sennheiser HD800	\$10,417,165.58	7,7	Chart/Rollup Tool
	Sennheiser SET830S	\$8,166,325.05	50,5	Pivot Tool
	Sony MDRV900HD	\$8,026,762.80	49,9	
Camcorder	Canon FS300	\$14,280,110.10	49,8	Show Records ▶
	Canon HFR11	\$9,462,118.35	14,1	Comments ▶
	Canon XHA1S	\$10,676,938.80	3,2	Send as E-mail
	JVC GCFM2BUS	\$8,546,471.35	49,9	Save Changes
	JVC GYHD200U	\$10,398,220.80	3,1	Export ▶
	JVC GZHD620B	\$8,620,412.37	13,8	Print ▶
				Window ▶
				Restore Original

157 of 157 records, Page 1 of 8 ▶▶▶▶

### Pagination

- ☐ Location: Bottom
- ☐ Background color: None
- ☐ Left justify

### Menu

- ☐ Text color: RGB (#6B6B6B)

- ☐ Background color: RGB (#F8F8F8)
- ☐ Hover text color: RGB (#495263)
- ☐ Hover background color: RGB (#DFDFDF)

**Hover**

- ☐ Background color: RGB (243, 243, 243)

**AR Iconset**

- ☐ Blue

## Usage Notes for ENWarm.sty

- ☐ AHTML does not support borders.
- ☐ Sort objects can contain multiple BY fields. In procedures where the sort object is the first BY field, the ENWarm StyleSheet defines attributes specifically for the first BY field and, therefore, is not supported.
- ☐ Sort Objects are not supported with any reference to a BY column in the StyleSheet. ENWarm styles the first BY subhead to distinguish a break.
- ☐ Horizontal borders are used to distinguish each data row. When borders are present in a report, the following behavior will change:
  - ☐ Blank line between heading and titles.
  - ☐ Blank line between end of data and footing.
- ☐ FML bar will not show the bar line.

**Note:** For scenarios involving these usage notes, use the endeflt StyleSheet as an alternative to the ENWarm StyleSheet.

## Controlling Report Formatting

---

When you format a report, you can control *how* the formatting is applied. You can:

- ☐ **Generate an internal cascading style sheet for HTML reports.** This improves performance and provides more formatting options.
- ☐ **Apply formatting conditionally.** You can make the appearance of a report conditional based on the report values. You can also make its links to other reports and Internet resources conditional. For example, in a monthly order report, you can specify that all unpaid orders be displayed in red and be linked to a recent payment history report of a customer.
- ☐ **Select a scale for specifying measurements,** such as the size of a report top margin. You can select an inch, centimeter, or point as the unit of measure.
- ☐ **Control the display of empty reports.** If a report request returns no records, you can choose to display the report without data, or to display a message stating there is no output.

**In this chapter:**

- ☐ [Generating an Internal Cascading Style Sheet for HTML Reports](#)
  - ☐ [Selecting a Unit of Measurement](#)
  - ☐ [Conditionally Formatting, Displaying, and Linking in a StyleSheet](#)
  - ☐ [Including Summary Lines, Underlines, Skipped Lines, and Page Breaks](#)
  - ☐ [Conditionally Including Summary Lines, Underlines, Skipped Lines, and Page Breaks](#)
  - ☐ [Controlling the Display of Empty Reports](#)
  - ☐ [Formatting a Report Using Only StyleSheet Defaults](#)
-

## Generating an Internal Cascading Style Sheet for HTML Reports

When you create a report in HTML format, code is generated that specifies how the report is formatted. You can set up your report to generate an internal cascading style sheet as part of this HTML code. This will:

- ❑ **Improve performance** by significantly reducing the size of the HTML file, decreasing transmission bandwidth, and displaying large reports more quickly.
- ❑ **Provide more formatting options** for your HTML report. Some WebFOCUS StyleSheet attributes are supported for HTML display format only in reports that generate an internal cascading style sheet.

Internal cascading style sheets enable HTML support for the UNITS, BOTTOMMARGIN, TOPMARGIN, LEFTMARGIN, RIGHTMARGIN, SIZE, POSITION, WRAP, and PAGECOLOR attributes. It also enables you to add and remove underlines from most report components and specify the starting position and size of an image. For details on the UNITS attribute, see [Selecting a Unit of Measurement](#) on page 1141. For more information on all other attributes, see [Laying Out the Report Page](#) on page 1249.

You can generate an internal cascading style sheet and apply an external cascading style sheet in the same report request. If any formatting instructions conflict, the internal cascading style sheet should override the external cascading style sheet.

**Note:** In most cases, you should not specify native WebFOCUS StyleSheet attributes and external CSS classes in the same report or style sheet. For more information, see [Using an External Cascading Style Sheet](#) on page 1211.

### **Syntax:** How to Generate an Internal Cascading Style Sheet

You can have an internal cascading style sheet created as part of the HTML code that is generated for a report in HTML format. To do so:

**Outside of a report request,** use

```
SET HTMLCSS = {ON|OFF}
```

**Within a report request,** use

```
ON TABLE SET HTMLCSS {ON|OFF}
```

where:

[ON](#)

Generates an internal cascading style sheet in the HTML output to control most aspects of the report appearance. ON is the default value.



**OFF**

Turns off the generation of an internal cascading style sheet. Instead, formatting tags are placed in each HTML table cell used to create the report.

**Reference: Requirements for Internal Cascading Style Sheets**

If you wish to display a report formatted via an internal cascading style sheet (CSS), you must have a web browser that supports cascading style sheets. Microsoft Internet Explorer Version 5.0 or higher, support cascading style sheets.

Note that how an internal cascading style sheet formats your report is determined entirely by your web browser support and implementation of cascading style sheets, not by WebFOCUS. Some web browsers may not fully support the latest CSS version, or may implement a CSS feature in a different way.

**Selecting a Unit of Measurement**

You can select the unit of measurement for page margins and column width for HTML reports that generate an internal cascading style sheet, as well as PDF and PostScript reports. In addition, you can also select the unit of measurement for column position in PDF and PS reports. You can select inches, centimeters, or points as the unit of measure.

If you change the unit of measure, all existing measurements are automatically converted to the new scale. For example, if the unit of measure is inches and the top margin for the report is set to 1, and you later change the unit of measure to centimeters, the size of the top margin is automatically converted to 1 centimeter.

You can set the unit of measure using a:

- ☐ **StyleSheet** by using the UNITS attribute.
- ☐ **SET command** by using the UNITS parameter.

**Syntax: How to Set the Unit of Measurement**

To set a unit of measurement:

**In a StyleSheet**, add the following attribute

```
UNITS = units
```

**Outside** of a report request, use

```
SET UNITS = units
```

**Within** a report request, use

`ON TABLE SET UNITS units`

where:

*units*

Is the unit of measure. Values can be:

- ☐ **INCHES**, that specifies the unit of measure as inches. This is the default value.
- ☐ **CM**, that specifies the unit of measure as centimeters.
- ☐ **PTS**, that specifies the unit of measure as points. Points is a common measurement scale for typefaces.

## Conditionally Formatting, Displaying, and Linking in a StyleSheet

You can conditionally format report components, display a graphic, and include links in your report based on the values in your report. Using conditional styling, you can:

- ☐ Draw attention to particular items in the report.
- ☐ Emphasize differences between significant values.
- ☐ Customize the resources to which an end user navigates from different parts of the report.

To conditionally format reports, add the **WHEN** attribute to a StyleSheet declaration. The **WHEN** attribute specifies a condition that is evaluated for each instance of a report component (that is, for each cell of a tabular report column, each element in a graph, or each free-form report page). The StyleSheet declaration is applied to each instance that satisfies the condition, and is ignored by each instance that fails to satisfy the condition.

You can also apply sequential conditional formatting.

**Note:** The variables **TABPAGENO** and **TABLASTPAGE** cannot be used to define styling with conditional styling (**WHEN**).

## Applying Sequential Conditional Formatting

You can apply sequential conditional logic to a report component by creating a series of declarations, each with a different condition. This is the StyleSheet equivalent of a sequence of nested IF-THEN-ELSE statements. When several conditional declarations specify the same report component (for example, the same column) *and* evaluate the same field in the condition, they are processed together as a group. For each instance of the report component (for example, for each cell of a column):

1. The conditional declarations in the "group" are evaluated, in the order in which they are found in the StyleSheet, until one of the conditions is satisfied. That declaration is then applied to that instance of the report component. The other conditional declarations in the "group," and any non-conditional declarations that specify the same report component and the same attributes, are ignored for that instance.
2. If, however, none of the conditional declarations have been satisfied for that instance, then the first unconditional declaration for that report component that specifies the same attribute(s) is applied to that instance.
3. Any unconditional declarations for that report component that specify other attributes (that is, attributes that have not already been applied to the instance in Steps 1 or 2) are now applied to the instance.
4. The entire process is repeated for the next instance of the report component (for example, for the next cell of the column).

### **Syntax:** How to Conditionally Format, Display, or Link in a StyleSheet

```
TYPE=type, [subtype,] attributes, WHEN=field1 operator {field2|value},$
```

or

```
TYPE=type, [subtype,] attributes, WHEN=FORECAST, $
```

where:

*type*

Is the value of the TYPE attribute. You can specify any report component. For details, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

*subtype*

Are any additional attributes, such as COLUMN, ACROSS, or ITEM, that are needed to identify the report component to which you are applying the declaration.

*attributes*

Are the attributes in the StyleSheet declaration that are made conditional by the WHEN attribute. They can include most formatting, graphic images, and hyperlink attributes.

### *field1, field2*

Identifies the report fields that are being compared. Each one can be:

- ☐ The name of a display field or vertical or horizontal sort field in a graph or tabular report. ACROSS values can be used as part of the conditional expressions used to define styling attributes for each cell in the table.
- ☐ A column reference in a graph or tabular report.
- ☐ The name of an embedded field in the heading or footing of a free-form report.

If you wish to use a field that you do not want to display in the report, you can specify the field in the report request, and use the NOPRINT option to prevent the field from being displayed (for example, PRINT *fieldname* NOPRINT).

To apply a prefix operator to a field in a report, you can:

- ☐ Use the same prefix operator in the WHEN attribute. You must refer to the field by name in the WHEN attribute (for example, WHEN=AVE.PRICE GT 300).
- ☐ Refer to the field in the WHEN attribute by column position and omit the prefix operator (for example, WHEN=N3 GT 300). This is not supported for the ST. and CT. prefix operators.
- ☐ You cannot use compound boolean expressions with the WHEN attribute.

The field cannot be a packed (P) numeric field.

### *operator*

Defines how the condition is satisfied. You can use these relational operators:

**EQ** where the condition is satisfied if the values on the left and right are equal. If the values being compared are alphanumeric, their case (uppercase, lowercase, or mixed case) must match.

**NE** where the condition is satisfied if the values on the left and right are not equal.

**LT** where the condition is satisfied if the value on the left is less than the value on the right.

**LE** where the condition is satisfied if the value on the left is less than or equal to the value on the right.

**GT** where the condition is satisfied if the value on the left is greater than the value on the right.

**GE** where the condition is satisfied if the value on the left is greater than or equal to the value on the right.

*value*

Is a constant, such as a number, character string, or date. You must enclose non-numeric constants, such as character strings and dates, in single quotation marks.

Although you cannot use functions or operators here to specify the value, you can define a temporary field (COMPUTE or DEFINE) using functions and operators, use the temporary field in the report, and specify it here instead of a constant.

**FORECAST**

Identifies fields that are generated using the FORECAST command.

**Example: Using Sequential Conditional Formatting**

This example illustrates how to apply sequential conditional formatting to a report. This report uses sequential conditional logic to format each row based on its order total (LINEPRICE).

```
TABLE FILE CENTORD
HEADING
"Order Revenue"
" "
SUM ORDER_DATE LINEPRICE AS 'Order,Total:'
BY HIGHEST 10 ORDER_NUM
ON TABLE SET PAGE-NUM OFF

ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
1. TYPE=DATA, BACKCOLOR=AQUA, STYLE=BOLD+ITALIC,
WHEN=LINEPRICE GT 500000, $
2. TYPE=DATA, BACKCOLOR=YELLOW, STYLE=BOLD,
WHEN=LINEPRICE GT 400000, $
3. TYPE=DATA, BACKCOLOR=ORANGE, STYLE=ITALIC,
WHEN=LINEPRICE GT 100000, $
4. TYPE=DATA, BACKCOLOR=SILVER, FONT='Arial', $
TYPE=HEADING, FONT='Arial', STYLE=BOLD, SIZE=11, $
ENDSTYLE

END
```

Notice that:

1. The first conditional declaration formats any rows whose order total is greater than 500,000.
2. The second conditional declaration formats any rows whose order total is greater than 400,000 and less than or equal to 500,000. This is because rows with an order total greater than 500,000 would have already been formatted by the first conditional declaration.

3. The third conditional declaration formats any rows whose order total is greater than 100,000 and less than or equal to 400,000. This is because rows with an order total greater than 400,000 would have already been formatted by one of the first two conditional declarations.
4. The unconditional declaration following the conditional declarations specifies:
  - ☐ Background color, which is also specified by the conditional declarations. It applies background color (silver) to any rows whose order total is less than or equal to 100,000, since those rows have not already been formatted by the conditional declarations.
  - ☐ Font, which is *not* specified by the conditional declarations. It applies font (Arial) to all data rows.

The output is:

## Order Revenue

Order Number:	Date Of Order:	Order Total:
94710	2001/01/02	\$406,964.24
94680	2001/01/02	\$421,916.60
94670	2001/01/02	\$513,868.76
94550	2001/01/02	\$496,323.64
94530	2001/01/02	\$3,472.41
94520	2001/01/02	\$261,808.72
94490	2000/12/31	\$633,723.06
94460	2001/01/02	\$3,872.39
94430	2001/01/02	\$3,033.38
94410	2001/01/02	\$2,337.28

### *Example:* Applying Basic Conditional Formatting

This example illustrates how to apply conditional formatting to a report. The conditional formatting draws attention to orders that total more than 200,000.

Notice that because a particular column is not specified in the declaration, the formatting is applied to the entire row.

```

TABLE FILE CENTORD
HEADING
"Order Revenue"
" "
SUM ORDER_DATE LINEPRICE AS 'Order,Total:'
BY HIGHEST 10 ORDER_NUM
ON TABLE SET PAGE-NUM OFF

ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
TYPE=DATA, BACKCOLOR=AQUA, STYLE=BOLD, WHEN=LINEPRICE GT 200000, $
TYPE=HEADING, FONT='Arial', STYLE=BOLD, SIZE=11, $
ENDSTYLE

END

```

The output is:

## Order Revenue

Order Number:	Date Of Order:	Order Total:
<b>94710</b>	<b>2001/01/02</b>	<b>\$406,964.24</b>
<b>94680</b>	<b>2001/01/02</b>	<b>\$421,916.60</b>
<b>94670</b>	<b>2001/01/02</b>	<b>\$513,868.76</b>
<b>94550</b>	<b>2001/01/02</b>	<b>\$496,323.64</b>
94530	2001/01/02	\$3,472.41
<b>94520</b>	<b>2001/01/02</b>	<b>\$261,808.72</b>
<b>94490</b>	<b>2000/12/31</b>	<b>\$633,723.06</b>
94460	2001/01/02	\$3,872.39
94430	2001/01/02	\$3,033.38
94410	2001/01/02	\$2,337.28

**Example: Applying Conditional Formatting to a Column**

This example illustrates how you can use conditional formatting to draw attention to columns that are not specified in the condition. The WHEN condition states that the order number for orders exceeding 200,000 should display in boldface with an aqua background.

Notice that the column that is evaluated in the WHEN condition (LINEPRICE) is different from the column that is formatted (ORDER\_NUM); they do not need to be the same.

```
TABLE FILE CENTORD
HEADING
"Order Revenue"
" "
SUM ORDER_DATE LINEPRICE AS 'Order,Total:'
BY HIGHEST 10 ORDER_NUM
ON TABLE SET PAGE-NUM OFF

ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
TYPE=DATA, COLUMN=ORDER_NUM,
BACKCOLOR=AQUA, STYLE=BOLD, WHEN=LINEPRICE GT 200000, $
TYPE=HEADING, FONT='Arial', STYLE=BOLD, SIZE=11, $
ENDSTYLE

END
```

The output is:

**Order Revenue**

<u>Order Number:</u>	<u>Date Of Order:</u>	<u>Order Total:</u>
<b>94710</b>	2001/01/02	\$406,964.24
<b>94680</b>	2001/01/02	\$421,916.60
<b>94670</b>	2001/01/02	\$513,868.76
<b>94550</b>	2001/01/02	\$496,323.64
94530	2001/01/02	\$3,472.41
<b>94520</b>	2001/01/02	\$261,808.72
<b>94490</b>	2000/12/31	\$633,723.06
94460	2001/01/02	\$3,872.39
94430	2001/01/02	\$3,033.38
94410	2001/01/02	\$2,337.28



**Example: Conditionally Styling an ACROSS Value**

The example below demonstrates how an ACROSS value can be referenced using either the ACROSS field name or the ACROSS column designator (A1, A2).

In this example, the ACROSS values are used in conditional styling to set a unique backcolor for all ACROSS columns in the Category Coffee, and additional font styling for the Espresso ACROSS column.

```
SET ACROSSTITLE=SIDE
TABLE FILE GGSales
SUM DOLLARS/I8M AS ''
BY REGION
BY ST
BY CITY
ACROSS CATEGORY
ACROSS PRODUCT
WHERE CATEGORY EQ 'Coffee' OR 'Food';
ON TABLE SET PAGE-NUM NOPAGE
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
SQUEEZE=ON,UNITS=IN,ORIENTATION=PORTRAIT,$
TYPE=REPORT,Font='ARIAL',SIZE=10,BORDER=LIGHT,$
TYPE=ACROSSTITLE,COLOR=WHITE, BACKCOLOR=GREY,$
TYPE=ACROSSVALUE,COLOR=WHITE, BACKCOLOR=GREY,$
TYPE=TITLE,COLOR=WHITE, BACKCOLOR=GREY,$
TYPE=DATA, ACROSSCOLUMN=DOLLARS, BACKCOLOR=THISTLE, WHEN=CATEGORY EQ
'Coffee',$
TYPE=DATA, ACROSSCOLUMN=DOLLARS, STYLE=BOLD+ITALIC, WHEN=A2 EQ 'Espresso', $
ENDSTYLE
END
```

The output is:

Category			Coffee			Food		
Product			Capuccino	Espresso	Latte	Biscotti	Croissant	Scone
Region	State	City						
Midwest	IL	Chicago		<b>\$420,439</b>	\$978,340	\$378,412	\$549,366	\$595,069
	MO	St. Louis		<b>\$419,143</b>	\$966,981	\$368,077	\$613,871	\$481,953
	TX	Houston		<b>\$455,365</b>	\$938,245	\$345,238	\$587,887	\$418,398
Northeast	CT	New Haven	\$158,995	<b>\$279,373</b>	\$926,052	\$589,355	\$551,489	\$283,874
	MA	Boston	\$174,344	<b>\$248,356</b>	\$917,737	\$570,391	\$497,234	\$332,486
	NY	New York	\$208,756	<b>\$322,378</b>	\$928,026	\$642,259	\$622,095	\$290,811
Southeast	FL	Orlando	\$317,027	<b>\$256,539</b>	\$889,887	\$511,597	\$602,076	\$311,836
	GA	Atlanta	\$352,161	<b>\$317,389</b>	\$907,365	\$555,231	\$661,806	\$273,420
	TN	Memphis	\$274,812	<b>\$279,644</b>	\$820,584	\$438,889	\$638,477	\$315,399
West	CA	Los Angeles	\$306,468	<b>\$267,809</b>	\$809,647	\$266,030	\$800,084	\$315,584
		San Francisco	\$279,830	<b>\$338,270</b>	\$935,862	\$269,518	\$824,457	\$292,839
	WA	Seattle	\$309,197	<b>\$301,538</b>	\$924,896	\$328,320	\$801,060	\$304,445

**Example:     Conditionally Formatting a Data Visualization Bar Graph**

This example illustrates how to apply conditional formatting to a data visualization bar graph. This report request incorporates a data visualization bar chart to graphically represent the data in the LINEPRICE column. It uses conditional formatting to draw attention to orders that total more than 200,000. It conditionally applies that formatting both to the data columns (TYPE=DATA) and to the bar graph (GRAPHTYPE=DATA).

Note that data visualization is only supported for HTML reports.

```
TABLE FILE CENTORD
HEADING
"Order Revenue"
" "
SUM ORDER_DATE LINEPRICE AS 'Order,Total:'
BY HIGHEST 10 ORDER_NUM
ON TABLE SET PAGE-NUM OFF

ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
TYPE=DATA, BACKCOLOR=AQUA, STYLE=BOLD, WHEN=LINEPRICE GT 200000, $
GRAPHTYPE=DATA, COLUMN=LINEPRICE, $
GRAPHTYPE=DATA, GRAPHCOLOR=AQUA, WHEN=LINEPRICE GT 200000,$
TYPE=HEADING, FONT='Arial', STYLE=BOLD, SIZE=11, $
ENDSTYLE

END
```

The output is:

Order Revenue		Order
Order Number:	Date Of Order:	Total:
94710	2001/01/02	\$406,964.24
94680	2001/01/02	\$421,916.60
94670	2001/01/02	\$513,868.76
94550	2001/01/02	\$496,323.64
94530	2001/01/02	\$3,472.41
94520	2001/01/02	\$261,808.72
94490	2000/12/31	\$633,723.06
94460	2001/01/02	\$3,872.39
94430	2001/01/02	\$3,033.38
94410	2001/01/02	\$2,337.28

**Example: Applying Conditional Formatting Based on Hidden (NOPRINT) Field Values**

This example illustrates how to apply conditional formatting based on the values of a hidden (NOPRINT) field. This report uses conditional formatting to draw attention to those employees who have resigned.

Notice that the WHEN attribute condition evaluates a field (STATUS) that is hidden in the report. Although the field that is evaluated in the condition must be included in the report request, you can prevent it from displaying in the report by using the NOPRINT option, as shown in the following request.

```
TABLE FILE CENTHR
HEADING
"Employee List for Boston"
" "
"For Pay Levels 5+"
" "
"Resigned Employees Shown in <0>Red Bold"
" "
PRINT LNAME FNAME Payscale STATUS NOPRINT
BY ID_NUM
WHERE PLANT EQ 'BOS' AND Payscale GE 5
ON TABLE SET PAGE-NUM OFF

ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
TYPE=DATA, COLUMN=LNAME,
    COLOR=RED, FONT='Arial', STYLE=BOLD, WHEN=STATUS EQ 'RESIGNED', $
TYPE=DATA, COLUMN=FNAME,
    COLOR=RED, FONT='Arial', STYLE=BOLD, WHEN=STATUS EQ 'RESIGNED', $
TYPE=HEADING, FONT='Arial', STYLE=BOLD, SIZE=11, $
TYPE=HEADING, LINE=5, STYLE=--BOLD, $
TYPE=HEADING, LINE=5, ITEM=2, STYLE=BOLD, COLOR=RED, $
ENDSTYLE

END
```

The output is:

<b>Employee List for Boston</b>		
<b>For Pay Levels 5+</b>		
<b>Resigned Employees Shown in <b>Red Bold</b></b>		
Employee Last <u>ID# Name</u>	First <u>Name</u>	Pay <u>Level</u>
39 GLIOZZO	ANTHONY	5
46 <b>HEBERT</b>	<b>BRYAN</b>	5
70 MIKITKA	MARK	5
79 PALMER	TED	5
85 <b>PHILLIPS</b>	<b>CLAIRE</b>	5
91 ROUSSEAU	DIANE	5
104 WHELEHAN	JAMES	5
129 <b>HAZARD</b>	<b>DAVID</b>	6
142 FLYNN	PAUL	6

**Example:** Applying Conditional Formatting to a Sort Group

This example illustrates how to apply conditional formatting to a sort group. This report uses conditional formatting to draw attention to those employees who have resigned.

Notice that one conditional declaration can apply formatting to all the sort group rows. You can accomplish this by evaluating the sort field (STATUS) in the WHEN attribute condition.

```

TABLE FILE CENTHR
HEADING
"Employee List for Boston"
" "
"For Pay Levels 5+"
" "
PRINT LNAME FNAME Payscale
BY STATUS SKIP-LINE
WHERE PLANT EQ 'BOS' AND Payscale GE 5
ON TABLE SET PAGE-NUM OFF

ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
TYPE=DATA,
COLOR=RED, FONT='Arial', STYLE=BOLD, WHEN=STATUS EQ 'RESIGNED',$
TYPE=HEADING, FONT='Arial', STYLE=BOLD, SIZE=11, $
ENDSTYLE

END

```

The output is:

<b>Employee List for Boston</b>			
<b>For Pay Levels 5+</b>			
<u>Current</u> <u>Status</u>	<u>Last</u> <u>Name</u>	<u>First</u> <u>Name</u>	<u>Pay</u> <u>Level</u>
DECLINED	ROUSSEAU	DIANE	5
EMPLOYED	MIKITKA	MARK	5
	WHELEHAN	JAMES	5
	FLYNN	PAUL	6
<b>RESIGNED</b>	<b>HEBERT</b>	<b>BRYAN</b>	<b>5</b>
	<b>PHILLIPS</b>	<b>CLAIRE</b>	<b>5</b>
	<b>HAZARD</b>	<b>DAVID</b>	<b>6</b>
TERMINAT	GLIOZZO	ANTHONY	5
	PALMER	TED	5

In order to apply the same conditional formatting to only two columns, instead of all the columns, this version of the report request uses two declarations, each specifying a different column (LNAME and FNAME):

```
TABLE FILE CENTHR
HEADING
"Employee List for Boston"
" "
"Pay Levels 5+"
" "
PRINT LNAME FNAME Payscale
BY STATUS SKIP-LINE
WHERE PLANT EQ 'BOS' AND Payscale GE 5
ON TABLE SET PAGE-NUM OFF

ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
TYPE=DATA, COLUMN=LNAME,
    COLOR=RED, FONT='Arial', STYLE=BOLD, WHEN=STATUS EQ 'RESIGNED', $
TYPE=DATA, COLUMN=FNAME,
    COLOR=RED, FONT='Arial', STYLE=BOLD, WHEN=STATUS EQ 'RESIGNED', $
TYPE=HEADING, FONT='Arial', STYLE=BOLD, SIZE=11, $
ENDSTYLE

END
```

The output is:

<b>Employee List for Boston</b>			
<b>For Pay Levels 5+</b>			
<u>Current</u> <u>Status</u>	<u>Last</u> <u>Name</u>	<u>First</u> <u>Name</u>	<u>Pay</u> <u>Level</u>
DECLINED	ROUSSEAU	DIANE	5
EMPLOYED	MIKITKA	MARK	5
	WHELEHAN	JAMES	5
	FLYNN	PAUL	6
RESIGNED	HEBERT	BRYAN	5
	PHILLIPS	CLAIRE	5
	HAZARD	DAVID	6
TERMINAT	GLIOZZO	ANTHONY	5
	PALMER	TED	5

**Example: Applying Conditional Formatting to Forecasted Values**

The following illustrates how you can apply conditional formatting to forecasted values in a report.

```

DEFINE FILE GGSales
SDATE/YYM = DATE;
SYEAR/Y = SDATE;
SMONTH/M = SDATE;
PERIOD/I2 = SMONTH;
END

TABLE FILE GGSales
SUM UNITS DOLLARS
BY CATEGORY BY PERIOD
WHERE SYEAR EQ 97 AND CATEGORY EQ 'Coffee'
ON PERIOD RECAP MOVAVE/D10.1= FORECAST(DOLLARS,1,3,'MOVAVE',3);
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=REPORT, BACKCOLOR=SILVER, WHEN=FORECAST, $
ENDSTYLE
END

```

The output is:

<u>Category</u>	<u>PERIOD</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>	<u>MOVAVE</u>
Coffee	1	61666	801123	801,123.0
	2	54870	682340	741,731.5
	3	61608	765078	749,513.7
	4	57050	691274	712,897.3
	5	59229	720444	725,598.7
	6	58466	742457	718,058.3
	7	60771	747253	736,718.0
	8	54633	655896	715,202.0
	9	57829	730327	711,158.7
	10	57012	724412	703,545.0
	11	51110	620264	691,667.7
	12	58981	762328	702,334.7
	13	0	0	694,975.6
	14	0	0	719,879.4
	15	0	0	705,729.9

## Including Summary Lines, Underlines, Skipped Lines, and Page Breaks

You can include sort-based options such as subtotals and other summary lines, sort headings and footings, underlines, skipped lines, and page breaks, as well as restart page numbering, by using the ON phrase in your report request. The ON phrase specifies an option that can be triggered by the change in a sort field or display field. The sort-based option (summary line, underline, skipped line, or page break) is applied to each sort group.

To make report requests flexible, options may be included that are not needed in every situation. User selections then control the options used for each execution of the request.

By default, if the field referenced in the ON phrase is not present in the request, or if the option is not supported with the type of field specified, the following message is generated and processing terminates:

```
(FOC013) The 'ON FIELDNAME' FIELD IS NOT A SORT FIELD: sortfield
```

You can use the SET ONFIELD =IGNORE command to instruct WebFOCUS to ignore ON phrases that reference absent fields or fields that are not supported by the specified option.

Note that any field used must be present in the Master File for the data source or the following message is generated and execution terminates:

```
(FOC003) THE FIELDNAME IS NOT RECOGNIZED: field
```

### **Syntax:** How to Display Summary Lines, Underlines, Skipped lines, and Page Breaks

```
{BY|ON} sortfield option [[AND] option ...]
```

where:

**BY|ON**

These are functionally identical. The only difference is syntactic (BY enables you to specify the sort-based feature as part of the sort phrase, while ON enables you to specify it separately from the sort phrase). For more information, see the documentation for the *sortfield* option you are using.

*sortfield*

Is the name of a vertical sort (BY) field.



*option*

Is one of the following sort-based features: PAGE-BREAK, PAGE-BREAK REPAGE, RECAP, RECOMPUTE, SKIP-LINE, SUBFOOT, SUBHEAD, SUBTOTAL, SUB-TOTAL, SUMMARIZE, UNDER-LINE.

*AND*

Can be included between two sets of *sortfield* options to enhance readability.

**Syntax: How to Control Processing of ON Phrases**

```
SET ONFIELD = {ALL| IGNORE}
```

```
ON TABLE SET ONFIELD {ALL| IGNORE}
```

where:

ALL

Issues a message and terminates execution when a field referenced in an ON phrase is not present in the request. ALL is the default value.

IGNORE

Ignores ON phrases that reference fields that are not present in the request, as well as ON phrases that include options not supported by the type of field specified.

**Example: Ignoring ON Phrases for Absent Fields**

The following request against the EMPDATA data source has ON phrases for the fields DEPT, DIV, and PIN. PIN is a sort field, but the other sort field must be entered at run time as the amper variable &F1:

```
SET USER = EUSER
TABLE FILE EMPDATA
  SUM SALARY
  BY &F1
  BY PIN
    ON DEPT SKIP-LINE NOSPLIT
    ON &F1 SUBTOTAL
    ON DIV PAGE-BREAK
    ON TABLE SET ONFIELD ALL
END
```

Run the request supplying the value **DEPT** for the variable &F1. The following messages are generated:

```
ERROR AT OR NEAR LINE      8  IN PROCEDURE IGNORE3 FOCEXEC *
(FOC013) THE 'ON FIELDNAME' FIELD IS NOT A SORT FIELD: DIV
BYPASSING TO END OF COMMAND
(FOC009) INCOMPLETE REQUEST STATEMENT
```

Now change the value of the ONFIELD parameter to IGNORE and run the request again, supplying the value **DEPT** for the variable &F1. The partial output is:

DEPT ----	PIN ---	SALARY -----
ACCOUNTING	000000070	\$83,000.00
	000000100	\$32,400.00
	000000300	\$79,000.00
	000000370	\$62,500.00
	000000400	\$26,400.00
*TOTAL ACCOUNTING		\$283,300.00
ADMIN SERVICES	000000170	\$30,800.00
	000000180	\$25,400.00
*TOTAL ADMIN SERVICES		\$56,200.00

## Conditionally Including Summary Lines, Underlines, Skipped Lines, and Page Breaks

You can conditionally include sort-based options such as subtotals and other summary lines, sort headings and footings, underlines, skipped lines, and page breaks, as well as conditionally restart page numbering, by using the WHEN phrase in your report request. The WHEN phrase specifies a condition that is evaluated for each value of a vertical sort (BY) field. The sort-based option (summary line, underline, skipped line, or page break) is applied to each sort group that satisfies the condition, and is ignored by sort groups that do not satisfy the condition.

The WHEN phrase is an extension of the ON *sortfield* and BY *sortfield* phrases. You can specify a WHEN phrase for each *sortfield* phrase. For example:

```
ON ORDER_NUM UNDER-LINE WHEN QUANTITY GT 5
ON COUNTRY PAGE-BREAK WHEN LINEPRICE GT 200000
```

If a *sortfield* phrase includes several sort-related options, you can specify a different WHEN phrase for each option. For example:

```
ON ORDER_NUM SKIP-LINE WHEN QUANTITY GT 5; UNDER-LINE WHEN QUANTITY GT 10
```

## **Syntax:** How to Conditionally Display Summary Lines, Underlines, Skipped lines, and Page Breaks

```
{BY|ON} sortfield [option WHEN condition [:] [AND]]...
```

where:

BY|ON

These are functionally identical. The only difference is syntactic (BY enables you to specify the sort-based feature as part of the sort phrase, while ON enables you to specify it separately from the sort phrase). For more information, see the documentation for the *sortfield* option you are using.

*sortfield*

Is the name of a vertical sort (BY) field.

*option*

Is one of the following sort-based features: PAGE-BREAK, PAGE-BREAK REPAGE, RECAP, RECOMPUTE, SKIP-LINE, SUBFOOT, SUBHEAD, SUBTOTAL, SUB-TOTAL, SUMMARIZE, UNDER-LINE.

If you specify SUBHEAD or SUBFOOT, you must place the WHEN phrase on the line following the text of the heading or footing.

*condition*

Is a logical expression. For more information, see [Using Expressions](#) on page 431.

You must enclose non-numeric constants, such as character strings and dates, in single quotation marks.

If the condition evaluates a numeric detail field, it evaluates the *sum* of the detail field values within each sort group, not the individual detail values. For example, in the request

```
TABLE FILE CENTHR
PRINT ID_NUM SALARY
BY PLANT
ON PLANT UNDER-LINE
WHEN SALARY GT 2000000
END
```

the condition evaluates the sum of the SALARY values within each PLANT value.

If the condition evaluates an alphanumeric field that appears multiple times in a sort group, it evaluates the last value of the field in each sort group.

You can apply a prefix operator to a field in the condition (for example, WHEN AVE.PRICE GT 300) even if the operator and the field are not used in the report. The aggregation is performed for each value of the sort field.

If the BY or ON phrase includes several options, the WHEN condition applies only to the option that immediately precedes it.

;

Is required if WHEN phrases are being included for several options in this BY or ON phrase. In all other situations it is optional and just enhances readability.

AND

Can be included between two sets of *sortfield* options to enhance readability.

**Example:** Using a WHEN Condition for a Sort Option

This example illustrates how to conditionally display a sub footing in a report. This report uses a conditional sort footing to draw attention to orders that total less than 200,000.

```
TABLE FILE CENTORD
HEADING
"Order Revenue"
" "
SUM ORDER_DATE LINEPRICE AS 'Order,Total:'
BY HIGHEST 5 ORDER_NUM

ON ORDER_NUM
SUBFOOT
    "---- Order total is less than $200,000 ----"
    " "
    WHEN LINEPRICE LT 200000

ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

Order Revenue

<u>Order Number:</u>	<u>Date Of Order:</u>	<u>Order Total:</u>
94710	2001/01/02	\$406,964.24
94680	2001/01/02	\$421,916.60
94670	2001/01/02	\$513,868.76
94550	2001/01/02	\$496,323.64
94530	2001/01/02	\$3,472.41
--- Order total is less than \$200,000 ---		

### **Example:** Using WHEN Conditions for Multiple Sort Options

This example illustrates how to apply multiple conditions to a report component. This report uses conditional sort footings to distinguish between orders that total more than 200,000 and less than 200,000.

Notice that one sort phrase (ON ORDER\_NUM) specifies several sort-related options (two different SUBFOOT phrases), and that each option has its own WHEN phrase.

```
TABLE FILE CENTORD
HEADING
"Order Revenue"
" "
SUM ORDER_DATE LINEPRICE AS 'Order,Total:'
BY HIGHEST 5 ORDER_NUM
ON ORDER_NUM
SUBFOOT
  "---- Order total is less than $200,000 ----"
  " "
  WHEN LINEPRICE LT 200000;
SUBFOOT
  "+++ Order total is greater than or equal to $200,000 +++"
  " "
  WHEN LINEPRICE GE 200000;

ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

Order Revenue

<u>Order Number:</u>	<u>Date Of Order:</u>	<u>Order Total:</u>
94710	2001/01/02	\$406,964.24
+++ Order total is greater than or equal to \$200,000 +++		
94680	2001/01/02	\$421,916.60
+++ Order total is greater than or equal to \$200,000 +++		
94670	2001/01/02	\$513,868.76
+++ Order total is greater than or equal to \$200,000 +++		
94550	2001/01/02	\$496,323.64
+++ Order total is greater than or equal to \$200,000 +++		
94530	2001/01/02	\$3,472.41
--- Order total is less than \$200,000 ---		

## Controlling the Display of Empty Reports

If a report request returns no records (for example, because no records satisfy its selection criteria, or because the data source has no records), you can choose to display or print:

- ❑ **An empty report**, that is, the report without data but including column titles, a report heading (if one was specified in the report request), and a page heading (if one was specified).

To do this, set the EMPTYREPORT parameter to ON.

- ❑ **A message** stating that there is no report output.

This is the default. You can return to the default by setting the EMPTYREPORT parameter to OFF.

This applies to tabular reports, not to free-form reports and graphs.

EMPTYREPORT is not supported with TABLEF. When a TABLEF request retrieves no records, it always generates an empty report.

SET EMPTYREPORT = OFF is not supported with DOC format.

If you have created a report that contains a WHERE TOTAL statement and the test yields zero records, an empty report will display.

### *Syntax:* How to Control the Display of an Empty Report

You can control what is displayed (or printed) when a report request returns no records, using the EMPTYREPORT parameter. To issue the SET command:

**Outside** of a report request, use the syntax

```
SET EMPTYREPORT = {ANSI|ON|OFF}
```

**Within** a report request, use the syntax

```
ON TABLE SET EMPTYREPORT {ANSI|ON|OFF}
```

where:

**ANSI**

Produces a single-line report and displays the missing data character or a zero if a COUNT is requested. In each case, &RECORDS will be 0, and &LINES will be 1.

If the SQL Translator is invoked, ANSI automatically replaces OFF as the default setting for EMPTYREPORT.

### ON

Specifies that the report will be displayed without data but with column titles, a report heading (if one was specified in the report request), and a page heading (if one was specified).

### OFF

Specifies that a message will be displayed indicating that there is no report output. OFF is the default value.

### *Example:* Controlling the Display of Empty Reports

The following request does not retrieve any records and sets the EMPTYREPORT parameter to OFF.

```
SET EMPTYREPORT=OFF
TABLE FILE WF_RETAIL_LITE
HEADING
"This is the heading"
SUM COGS_US REVENUE_US COLUMN-TOTAL ROW-TOTAL
BY COUNTRY_NAME
WHERE COUNTRY_NAME EQ 'Louisiana'
FOOTING
"This is the footing"
ON TABLE SET STYLE *
GRID=OFF,$
END
```

The following output is produced.

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

Changing the EMPTYREPORT setting to ON produces the output shown in the following image.

PAGE 1

This is the heading

Customer

<u>Country</u>	<u>Cost of Goods</u>	<u>Revenue</u>	<u>TOTAL</u>
----------------	----------------------	----------------	--------------



Changing the EMPTYREPORT setting to ANSI produces the output shown in the following image.

PAGE 1				
This is the heading				
Customer				
<u>Country</u>	<u>Cost of Goods</u>	<u>Revenue</u>	<u>TOTAL</u>	
.	.	.	.	.
TOTAL	.	.		\$00
This is the footing				

## Formatting a Report Using Only StyleSheet Defaults

You can format a report using only default StyleSheet values. This does not permit you to specify specific formatting, and does not access a StyleSheet.

Each display format, such as HTML or PDF, has its own set of defaults. For example, HTML defaults to a proportional font, while PDF defaults to a monospace font. For information on the default value of a specific StyleSheet attribute, see the documentation for that attribute.

### *Syntax:* How to Format a Report Using Only StyleSheet Defaults

To use only default StyleSheet values to format:

- ❑ **All report requests within a procedure.** Issue the following command at the beginning of the procedure.

```
SET STYLE[SHEET] = ON
```

- ❑ **One report request.** Issue the following command within the request.

```
ON TABLE SET STYLE[SHEET] ON
```

where:

**SHEET**

Can be omitted to make the command shorter, and has no effect on its behavior.



A report consists of several types of components, each of which you can identify in a StyleSheet in order to format it.

Report components are subject to StyleSheet inheritance. For details, see [Creating and Managing a WebFOCUS StyleSheet](#) on page 1117.

For information about identifying components in a graph, see [Creating a Graph](#) on page 1657.

Unless otherwise noted, all StyleSheet references in this chapter refer to WebFOCUS StyleSheets.

**In this chapter:**

- ❑ [Identifying an Entire Report, Column, or Row](#)
  - ❑ [Identifying Tags for SUBTOTAL and GRANDTOTAL Lines](#)
  - ❑ [Identifying Data](#)
  - ❑ [Identifying a Heading, Footing, Title, or FML Free Text](#)
  - ❑ [Identifying a Page Number, Underline, or Skipped Line](#)
- 

## Identifying an Entire Report, Column, or Row

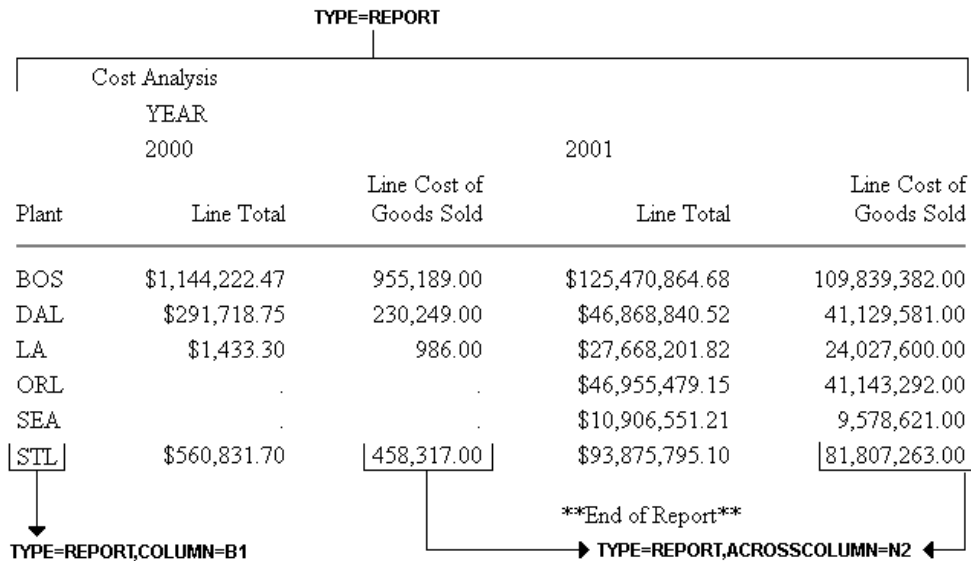
You can apply formatting to an:

- ❑ **Entire report.** For more information, see [How to Identify an Entire Report](#) on page 1168.
- ❑ **Entire column** within a report, both its title and data (including ROW-TOTAL columns). For more information, see [How to Identify an Entire Column](#) on page 1170.
- ❑ **Entire row** within a report, either an FML (Financial Modeling Language) row, or a total or subtotal row, comprising the labeling text and data of the row. For more information, see [How to Identify an Entire Financial Modeling Language \(FML\) Row](#) on page 1173, and [How to Identify an Entire Total or Subtotal Row](#) on page 1174.

You can also identify an entire horizontal sort (ACROSS) title or value row in a StyleSheet, although each of these rows contains only a single kind of information. For details, see [How to Identify a Column Title](#) on page 1192.

The following illustrates where the REPORT component and the COLUMN and ACROSSCOLUMN attributes appear in a report, and which TYPE values you use to identify them. Although in this example the value for COLUMN is B1 and the value for ACROSSCOLUMN is N2, these are not the only values you can use to identify these components.

```
TABLE FILE CENTORD
SUM LINEPRICE LINE_COGS AS 'Line Cost of,Goods Sold'
BY PLANT AS 'Plant'
ACROSS YEAR
WHERE YEAR EQ 2000 or 2001
HEADING
"Cost Analysis"
FOOTING CENTER
"***End of Report***"
ON TABLE SET PAGE-NUM OFF
END
```



**Note:** Since this request simply illustrates where the components appear in a report, it omits a StyleSheet.

**Syntax:**      **How to Identify an Entire Report**

To identify an entire report in a StyleSheet, use this attribute and value:

```
TYPE=REPORT
```

**Example: Identifying an Entire Report**

The following illustrates how to identify and apply formatting to an entire report. The relevant StyleSheet declarations are highlighted in the request.

```
TABLE FILE CENTINV
HEADING
"Excess Stock Report"
SUM QTY_IN_STOCK
BY PRODNAME
WHERE QTY_IN_STOCK GT 10000
FOOTING CENTER
***End of Report***
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF,$
TYPE=REPORT, STYLE=BOLD,$
ENDSTYLE
END
```

The output is:

### Excess Stock Report

<b>Product</b>	<b>Quantity</b>
<b><u>Name:</u></b>	<b><u>In Stock:</u></b>
2 Hd VCR LCD Menu	43068
250 8MM Camcorder 40 X	60073
330DX Digital Camera 1024K P	12707
750SL Digital Camcorder 300 X	10758
AR2 35MM Camera 8 X	11499
AR3 35MM Camera 10 X	12444
Combo Player - 4 Hd VCR + DVD	13527
QX Portable CD Player	22000
ZC Digital PDA - Standard	33000
ZT Digital PDA - Commercial	21000

**\*\*End of Report\*\***

**Syntax:**      **How to Identify an Entire Column**

```
TYPE=REPORT, coltype=column
```

where:

*coltype*

Specifies the type of column. It can be:

- ❑ **COLUMN**, which specifies a sort column (generated by BY), a display column (generated by PRINT, LIST, SUM, or COUNT), a computed column (generated by COMPUTE), or a column of row totals (generated by ROW-TOTAL).
- ❑ **ACROSSCOLUMN**, which specifies every instance of a column that is repeated across a horizontal sort (ACROSS) row. This also applies the formatting to the horizontal sort (ACROSS) values that appear above the column titles.

*column*

Specifies one or more columns. If you are identifying an ACROSSCOLUMN, the only valid identifiers are *Nn* and *Pn*, and these only count ACROSS fields, not display fields.

Options for identifying columns in a StyleSheet are:

Identifier	Description
<i>Nn</i>	Identifies a column by its position in the report. To determine this value, count vertical sort (BY) fields, display fields, and ROW-TOTAL fields, from left to right, including NOPRINT fields. For ACROSSCOLUMN, only ACROSS fields are counted.
<i>Pn</i>	Identifies a column by its position in the report. To determine the value of <i>n</i> , count vertical sort (BY) fields, display fields, and ROW-TOTAL fields from left to right. Do not count NOPRINT fields. For ACROSSCOLUMN, only ACROSS fields are counted.
<i>Cn</i>	Identifies a display column by its position in the report. To determine the value of <i>n</i> , count only display fields from left to right, including NOPRINT fields. Do not count vertical sort (BY) fields or ROW-TOTAL fields.  To select all display fields use C*.

Identifier	Description
<i>Bn</i>	Identifies a vertical sort (BY) column by its position in the report. To determine the value of <i>n</i> , count only vertical sort (BY) fields, including NOPRINTs, from left to right.  To select all BY fields use B*.
<i>field</i>	Identifies a column by its field name.  When a field occurs more than once, use field( <i>n</i> ) to select a particular occurrence or field(*) to select all occurrences of the field.
ROWTOTAL	Identifies a column of row totals generated using ROW-TOTAL. When used with ACROSS and multiple display commands, ROWTOTAL generates multiple total columns. Use ROWTOTAL( <i>n</i> ) to select a particular total column. Use ROWTOTAL( <i>field</i> ) to select the row total column for a particular field.  Use ROWTOTAL(*) to select all row total columns in the report.

**Note:** Within a StyleSheet, all columns must be specified in the same way, either by field name or positional reference.

### **Example:** Identifying an Entire Column

The following illustrates how to identify an entire column, which consists of the column data and the column title, in a report. The relevant StyleSheet declaration is highlighted in the request.

**Note:** To produce the same results you can, alternatively, use the values P1, B1, or the fieldname (PRODNAME) for the COLUMN attribute in the StyleSheet declaration.

```
TABLE FILE CENTINV
HEADING
"Excess Stock Report"
SUM QTY_IN_STOCK
BY PRODNAME
WHERE QTY_IN_STOCK GT 10000
FOOTING CENTER
"***End of Report***"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF,$
TYPE=REPORT, COLUMN=N1, STYLE=ITALIC,$
ENDSTYLE
END
```

The output is:

```

Excess Stock Report

Product                                Quantity
Name:                                In Stock:
2 Hd VCR LCD Menu                      43068
250 8MM Camcorder 40 X                 60073
330DX Digital Camera 1024K P          12707
750SL Digital Camcorder 300X'         10758
AR2 35 MM Camera 8 X                   11499
AR3 35MM Camera 10 X                   12444
Combo Player - 4 Hd VCR + DVD          13527
QX Portable CD Player                  22000
ZC Digital PDA - Standard               33000
ZT Digital PDA - Commercial             21000

**End of Report**

```

### **Example:** Identifying an Entire Horizontal (ACROSS) Column

The following illustrates how to identify a horizontal (ACROSS) column. When you identify and format an ACROSSCOLUMN, all data values, the column title, and any horizontal sort (ACROSS) values associated with the field are formatted for every instance of the column in the report output. The relevant StyleSheet declarations are highlighted in the request.

**Note:** To produce the same results you can alternatively use the values P1 and P2, respectively, for the ACROSSCOLUMN attribute.

```

TABLE FILE CENTORD
SUM LINEPRICE LINE_COGS AS 'Line Cost of, Goods Sold'
BY PLANT AS 'Plant'
ACROSS YEAR
WHERE YEAR EQ 2000 OR 2001
HEADING
"Cost Analysis"
FOOTING CENTER
***End of Report**
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF,$
TYPE=REPORT, ACROSSCOLUMN=N1, STYLE=ITALIC,$
TYPE=REPORT, ACROSSCOLUMN=N2, STYLE=BOLD,$
ENDSTYLE
END

```



The output is:

Cost Analysis				
YEAR				
2000			2001	
Plant	Line Total	Line cost of Goods Sold	Line Total	Line cost of Goods Sold
BOS	\$1,144,222.47	955,189.00	\$125,470,864.68	109,839,382.00
DAL	\$291,718.75	230,249.00	\$46,868,840.52	41,129,581.00
LA	\$1,433.30	986.00	\$27,668,201.82	24,027,600.00
ORL	.	.	\$46,955,479.15	41,143,292.00
SEA	.	.	\$10,906,551.21	9,578,621.00
STL	\$560,831.70	458,317.00	\$93,875,795.10	81,807,263.00

\*\*End of Report\*\*

**Syntax:** How to Identify an Entire Financial Modeling Language (FML) Row

```
TYPE=REPORT, LABEL=label
```

where:

*label*

Is an explicit row label.

**Example:** Identifying an Entire FML Row

The following illustrates how to identify an entire FML row, consisting of the row label and the row data. The relevant StyleSheet declarations are highlighted in the request.

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 AS 'CASH ON HAND' LABEL COH OVER
1020 AS 'DEMAND DEPOSITS' LABEL DD OVER
1030 AS 'TIME DEPOSITS' LABEL TD OVER
BAR OVER
RECAP TOTCASH = R1 + R2 + R3; AS 'TOTAL CASH'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=REPORT, LABEL=COH, STYLE=ITALIC, $
TYPE=REPORT, LABEL=DD, STYLE=ITALIC, $
TYPE=REPORT, LABEL=TD, STYLE=ITALIC, $
ENDSTYLE
END
```

The output is:

	<u>AMOUNT</u>
<i>CASH ON HAND</i>	8,784
<i>DEMAND DEPOSITS</i>	4,494
<i>TIME DEPOSITS</i>	7,961
<hr/>	
TOTAL CASH	21,239

**Syntax:**      **How to Identify an Entire Total or Subtotal Row**

*TYPE=type, [BY=sortcolumn]*

where:

*type*

Identifies a subtotal or total. Select from:

*GRANDTOTAL* which is a grand total (generated by COLUMN-TOTAL, SUBTOTAL, SUB-TOTAL, RECOMPUTE, or SUMMARIZE).

*SUBTOTAL* which is a subtotal (generated by SUBTOTAL, SUB-TOTAL, RECOMPUTE, or SUMMARIZE).

*RECAP* which is a subtotal calculation (generated by ON *sortfield* RECAP or ON *sortfield* COMPUTE).

*BY*

When there are several subtotal commands, each associated with a different vertical sort (BY) column, this enables you to identify which of the subtotal commands you wish to format.

*sortcolumn*

Specifies the vertical sort (BY) column associated with one of the several subtotal in the report commands. Use the field name to identify the sort column.

**Example: Identifying an Entire Total Row**

The following illustrates how to identify an entire COLUMN-TOTAL row in a StyleSheet. The relevant StyleSheet declaration is highlighted in the request.

```
TABLE FILE SALES
SUM RETURNS DAMAGED AND ROW-TOTAL AND COLUMN-TOTAL
BY PROD_CODE
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=GRANDTOTAL, STYLE=BOLD, SIZE=12, $
ENDSTYLE
END
```

The output is:

<u>PROD_CODE</u>	<u>RETURNS</u>	<u>DAMAGED</u>	<u>TOTAL</u>
B10	13	10	23
B12	4	3	7
B17	4	2	6
B20	1	2	3
C13	3	0	3
C17	0	0	0
C7	5	4	9
D12	3	2	5
E1	4	7	11
E2	9	4	13
E3	12	11	23
<b>TOTAL</b>	<b>58</b>	<b>45</b>	<b>103</b>

**Example: Identifying a Row Total**

The following illustrates how to identify a row total. The relevant StyleSheet declaration is highlighted in the request. Note that if you want to format an instance of row-total, you can add a WHEN statement to your StyleSheet. For details, see [Controlling Report Formatting](#) on page 1139.

```
TABLE FILE SALES
SUM RETURNS DAMAGED AND ROW-TOTAL
BY PROD_CODE AS 'PRODUCT, CODE'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=REPORT, COLUMN=ROWTOTAL, STYLE=BOLD, $
ENDSTYLE
END
```

The output is:

PRODUCT			
<u>CODE</u>	<u>RETURNS</u>	<u>DAMAGED</u>	<u>TOTAL</u>
B10	13	10	<b>23</b>
B12	4	3	<b>7</b>
B17	4	2	<b>6</b>
B20	1	2	<b>3</b>
C13	3	0	<b>3</b>
C17	0	0	<b>0</b>
C7	5	4	<b>9</b>
D12	3	2	<b>5</b>
E1	4	7	<b>11</b>
E2	9	4	<b>13</b>
E3	12	11	<b>23</b>

**Identifying Tags for SUBTOTAL and GRANDTOTAL Lines**

The tag is the text that is displayed in the leftmost portion of each SUBTOTAL and GRANDTOTAL row in a report. The tag is used to identify the type of data represented within this row. The text used to generate this tag can be customized by adding an AS name to the SUBTOTAL syntax.

You can define styling for the subtotal and grand total tag separately from the rest of the row. Text attributes available for the tag, including font, color, size, and style, can be used to differentiate and highlight the tags. Additionally, styling can be applied that turns tags into drill-down links.

Styling is supported for text attributes only. Cell or column features, such as borders, background color, or justification are not supported.

This feature is available for PDF, DHTML, PS, HTML, AHTML, PPTX, XLSX, and EXL2K formats.

**Syntax:**      **How to Style Subtotal and Grand Total Tags**

```
TYPE={SUBTOTAL|GRANDTOTAL}, OBJECT=TAG,
  [FONT=font], [SIZE=size], [STYLE=style],
  [COLOR={color|RGB({rgb|#hexcolor})}],
  [drilltype=drillparms], $
```

where:

*font*

Is the name of the font.

*size*

Is the point size of the font.

*style*

Is the font style, for example bold, italic, or bold+italic.

*color*

Is a color name.

*rgb*

Specifies the font color using a mixture of red, green, and blue.

(r g b) is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense. Note that using the three color components in equal intensities results in shades of gray.

#*hexcolor*

Is the hexadecimal value for the color. For example, FF0000 is the hexadecimal value for red. The hexadecimal digits can be in uppercase or lowercase and must be preceded by a pound sign (#).

*drilltype*

Is any valid drill-down attribute, for example, URL= or FOCEXEC=. For more information about drill-down links, see [Linking a Report to Other Resources](#) on page 763.

*drillparms*

Are valid attribute values for the type of drill down.

### **Example:** Styling SUBTOTAL and GRANDTOTAL Tags

The following request against the GGSALES data source generates subtotal and grand total rows. The tags for the subtotal rows are in italics and are white. The tag for the grand total row has a drill-down link to a URL:

```
TABLE FILE GGSALES
SUM UNITS/D8C DOLLARS/D12CM BUDUNIT/D8C BUDDOLLARS/D12CM
BY REGION
BY CATEGORY
ON REGION SUBTOTAL
HEADING
"Gotham Grinds Sales Report"
ON TABLE SET HTMLCSS ON
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET DROPBLNKLINE ALL
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/
ENIADefault_combine.sty,$
TYPE=SUBTOTAL, OBJECT=TAG,STYLE=ITALIC,COLOR=WHITE,$
TYPE=GRANDTOTAL, BACKCOLOR='LIGHT GREY',$
TYPE=GRANDTOTAL, OBJECT=TAG,URL='http://www.informationbuilders.com',$
ENDSTYLE
END
```

The output is:

Gotham Grinds Sales Report					
Region	Category	Unit Sales	Dollar Sales	Budget Units	Budget Dollars
Midwest	Coffee	332,777	\$4,178,513	335,526	\$4,086,032
	Food	341,414	\$4,338,271	339,263	\$4,220,721
	Gifts	230,854	\$2,883,881	232,318	\$2,887,620
<b>*TOTAL Midwest</b>		<b>905,045</b>	<b>\$11,400,665</b>	<b>907,107</b>	<b>\$11,194,373</b>
Northeast	Coffee	335,778	\$4,164,017	335,920	\$4,252,462
	Food	353,368	\$4,379,994	351,431	\$4,453,907
	Gifts	227,529	\$2,848,289	227,008	\$2,870,552
<b>*TOTAL Northeast</b>		<b>916,675</b>	<b>\$11,392,300</b>	<b>914,359</b>	<b>\$11,576,921</b>
Southeast	Coffee	350,948	\$4,415,408	355,693	\$4,431,429
	Food	349,829	\$4,308,731	351,509	\$4,409,288
	Gifts	234,455	\$2,986,240	235,045	\$2,967,254
<b>*TOTAL Southeast</b>		<b>935,232</b>	<b>\$11,710,379</b>	<b>942,247</b>	<b>\$11,807,971</b>
West	Coffee	356,763	\$4,473,517	358,784	\$4,523,963
	Food	340,234	\$4,202,337	335,361	\$4,183,244
	Gifts	235,042	\$2,977,092	236,636	\$2,934,306
<b>*TOTAL West</b>		<b>932,039</b>	<b>\$11,652,946</b>	<b>930,781</b>	<b>\$11,641,513</b>
<b>TOTAL</b>		<b>3,688,991</b>	<b>\$46,156,290</b>	<b>3,694,494</b>	<b>\$46,220,778</b>

## Identifying Data

You can identify and format many categories of data in a report, including:

- ☐ **All of a report data.** For more information, see [How to Identify All Data](#) on page 1180.
- ☐ **Columns of data**, including sort columns and display columns. For more information, see [How to Identify a Column of Data](#) on page 1182.
- ☐ **Sort rows** (that is, ACROSS field values). For more information, see [How to Identify a Row of Horizontal Sort \(ACROSS\) Data](#) on page 1183.
- ☐ **Totals and subtotals.** For more information, see [Identifying Totals and Subtotals](#) on page 1185.

The following illustrates where the DATA and ACROSSVALUE components appear in a report, and which TYPE values you use to identify them.

```
TABLE FILE CENTORD
HEADING CENTER
"UNITS SOLD IN 2002 BY PLANT"
SUM QUANTITY AND ROW-TOTAL AS '2002 TOTAL'
ACROSS QUARTER
BY PLANTLNG AS 'PLANT'
WHERE YEAR EQ 2002
ON TABLE COLUMN-TOTAL AS 'TOTAL UNITS'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF,$
END
```

		UNITS SOLD IN 2002 BY PLANT				
TYPE=ACROSSVALUE		QUARTER				
		Q1	Q2	Q3	Q4	2002 TOTAL
PLANT						
TYPE=DATA	Boston	165,938	202,100	84,571	86,227	538,836
	Dallas	62,693	63,403	35,747	42,094	203,937
	Los Angeles	55,542	30,326	17,188	16,869	119,925
	Orlando	76,261	57,639	32,595	35,895	202,390
	Seattle	16,887	17,496	5,454	5,512	45,349
	St Louis	150,293	109,284	65,161	80,530	405,268
TOTAL UNITS		527,614	480,248	240,716	267,127	1,515,705

**Note:** Since this request simply illustrates where the components appear in a report, it omits a StyleSheet.

**Syntax:**      **How to Identify All Data**

To identify all report data in a StyleSheet (except totals, grand totals, subtotals, and horizontal sort (ACROSS) values, which need to be identified separately) use this attribute and value:

```
TYPE = DATA
```



**Example: Identifying All Data in a Report**

The following illustrates how to identify all of the data in a report. The relevant StyleSheet declaration is highlighted in the request.

```
TABLE FILE CENTORD
HEADING CENTER
"UNITS SOLD IN 2002 BY PLANT"
SUM QUANTITY AND ROW-TOTAL AS '2002 TOTAL'
ACROSS QUARTER
BY PLANTLNG AS 'PLANT'
WHERE YEAR EQ 2002
ON TABLE COLUMN-TOTAL AS 'TOTAL UNITS'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=DATA, STYLE=BOLD, $
ENDSTYLE
END
```

The output is:

UNITS SOLD IN 2002 BY PLANT					
QUARTER					
	Q1	Q2	Q3	Q4	2002 TOTAL
PLANT					
<b>Boston</b>	<b>165,938</b>	<b>202,100</b>	<b>84,571</b>	<b>86,227</b>	<b>538,836</b>
<b>Dallas</b>	<b>62,693</b>	<b>63,403</b>	<b>35,747</b>	<b>42,094</b>	<b>203,937</b>
<b>Los Angeles</b>	<b>55,542</b>	<b>30,326</b>	<b>17,188</b>	<b>16,869</b>	<b>119,925</b>
<b>Orlando</b>	<b>76,261</b>	<b>57,639</b>	<b>32,595</b>	<b>35,895</b>	<b>202,390</b>
<b>Seattle</b>	<b>16,887</b>	<b>17,496</b>	<b>5,454</b>	<b>5,512</b>	<b>45,349</b>
<b>St Louis</b>	<b>150,293</b>	<b>109,284</b>	<b>65,161</b>	<b>80,530</b>	<b>405,268</b>
<b>TOTAL UNITS</b>	<b>527,614</b>	<b>480,248</b>	<b>240,716</b>	<b>267,127</b>	<b>1,515,705</b>

**Syntax:**      **How to Identify a Column of Data**

`TYPE=DATA, COLUMN=column`

where:

*column*

Specifies one or more columns that you wish to format. For a list of values, see [How to Identify an Entire Column](#) on page 1170.

**Example:**      **Identifying a Column of Data**

The following illustrates how to identify a column of data. The relevant StyleSheet declaration is highlighted in the request.

Note that when identifying a column using Nn, NOPRINT columns are counted. Even though the Product Name field is the first column in this report, it is identified with N2 because of the NOPRINT column.

```
TABLE FILE CENTORD
PRINT QUANTITY LINEPRICE LINE_COGS
BY ORDER_NUM NOPRINT
BY PRODNAME
WHERE ORDER_NUM EQ '48045'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF,$
TYPE=DATA, COLUMN=N2, STYLE=ITALIC,$
ENDSTYLE
END
```

The output is:

<u>Product Name:</u>	<u>Quantity:</u>	<u>Line Total:</u>	<u>Line Cost of Goods Sold</u>
110 VHS-C Camcorder 20X	266	\$70,901.90	66,234.00
120 VHS-C Camcorder 40X	266	\$76,511.09	68,894.00
150 8MM Camcorder 20X	121	\$28,473.78	29,040.00
DVD Upgrade Unit for VCR	121	\$18,177.77	16,819.00
QX Portable CD Player	266	\$33,847.80	26,334.00

**Syntax:**      **How to Identify a Row of Horizontal Sort (ACROSS) Data**

```
TYPE=ACROSSVALUE, [ACROSS={fieldname|{N|A}n}]
```

where:

**ACROSS**

If you have a request with multiple ACROSS fields, you can identify each field using the ACROSS identifier. You only need to include the ACROSS identifier if you have multiple ACROSS fields in your request.

*fieldname*

Specifies a horizontal sort row by its field name.

*n*

Specifies a horizontal sort row by its position in the sequence of horizontal sort rows.  
Cannot be combined with a field name specification in the same StyleSheet.

**Example:**      **Identifying a Row of Horizontal Sort (ACROSS) Data**

The following illustrates how to identify a row of horizontal sort data. The relevant StyleSheet declaration is highlighted in the request.

```
TABLE FILE CENTORD
HEADING
"Units Sold"
SUM QUANTITY
BY PRODNAME
ACROSS PLANT AS 'Manufacturing Plant'
WHERE PRODTYPE EQ 'Digital'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=HEADING, SIZE=12, $
TYPE=ACROSSVALUE, ACROSS=PLANT, STYLE=BOLD, $
ENDSTYLE
END
```

The output is:

Units Sold

Product Name:	Manufacturing Plant					
	BOS	DAL	LA	ORL	SEA	STL
330DX Digital Camera 1024K P	3,949	2,767	562	561	2	1,132
650DL Digital Camcorder 150 X	44,930	14,548	7,995	17,972	4,967	31,436
750SL Digital Camcorder 300 X	4,098	2	6	283	642	1,644
Combo Player - 4 Hd VCR + DVD	27,687	15,985	9,634	13,816	2,359	22,097
DVD Upgrade Unit for Cent. VCR	47,112	25,602	16,152	23,429	4,539	32,875
QX Portable CD Player	108,696	35,355	26,215	46,388	8,056	83,564
R5 Micro Digital Tape Recorder	188,384	73,255	41,169	63,120	15,385	142,714
ZC Digital PDA - Standard	45,890	13,154	4,664	10,249	6,439	30,213
ZT Digital PDA - Commercial	181,750	70,111	41,149	61,244	15,039	130,942

**Note:** To produce the same results you can alternatively use the value N1 for the ACROSS attribute in the StyleSheet declaration. For example, TYPE=ACROSSVALUE, ACROSS=N1, STYLE=BOLD, \$.

*Example:*     **Identifying Row Totals (ACROSS-TOTAL) for Horizontal Sort Data**

The following illustrates how to identify a row total (ACROSS-TOTAL) for horizontal sort (ACROSS) data using the ACROSSVALUE component and a numeric column reference (Nn). The relevant StyleSheet declaration is highlighted in the request.

```
TABLE FILE CENTORD
SUM QUANTITY
BY PRODNAME
ACROSS PLANT AS 'Manufacturing Plant'
ACROSS-TOTAL AS 'Plant Totals'
WHERE PRODTYPE EQ 'Digital'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=ACROSSVALUE, COLUMN=N8, STYLE=ITALIC, COLOR='RED', $
ENDSTYLE
END
```

The following image shows the output with the ACROSS-TOTAL value, Plant Totals, styled in red italics.

Product Name:	Manufacturing Plant						<i>Plant Totals</i>
	BOS	DAL	LA	ORL	SEA	STL	
330DX Digital Camera 1024K P	3,949	2,767	562	561	2	1,132	8,973
650DL Digital Camcorder 150 X	44,930	14,548	7,995	17,972	4,967	31,436	121,848
750SL Digital Camcorder 300 X	4,098	2	6	283	642	1,644	6,675
Combo Player - 4 Hd VCR + DVD	27,687	15,985	9,634	13,816	2,359	22,097	91,578
DVD Upgrade Unit for Cent. VCR	47,112	25,602	16,152	23,429	4,539	32,875	149,709
QX Portable CD Player	108,696	35,355	26,215	46,388	8,056	83,564	308,274
R5 Micro Digital Tape Recorder	188,384	73,255	41,169	63,120	15,385	142,714	524,027
ZC Digital PDA - Standard	45,890	13,154	4,664	10,249	6,439	30,213	110,609
ZT Digital PDA - Commercial	181,750	70,111	41,149	61,244	15,039	130,942	500,235

## Identifying Totals and Subtotals

Within a StyleSheet, you can identify the grand totals, subtotals, subtotal calculations (generated by ON *sortfield* RECAP or ON *sortfield* COMPUTE), column totals, and row totals of a report in order to format them. For details on identifying row totals, see [Identifying an Entire Report, Column, or Row](#) on page 1167.

The following example illustrates where these components are in a report, and which TYPE values you use to identify them.

```
TABLE FILE EMPLOYEE
SUM DED_AMT AND GROSS
BY DEPARTMENT BY PAY_DATE
ON DEPARTMENT RECAP DEPT_NET/D8.2M = GROSS-DED_AMT;
WHEN PAY_DATE GT 820101
ON DEPARTMENT SUBTOTAL
END
```

The following figure shows each component:

DEPARTMENT	PAY_DATE	DED_AMT	GROSS	
-----	-----	-----	-----	
MIS	81/11/30	\$1,406.79	\$2,147.75	
	81/12/31	\$1,406.79	\$2,147.75	
*TOTAL MIS		\$2813.58	\$4295.50	← TYPE = SUBTOTAL
** DEPT_NET		\$2,311.98		← TYPE = RECAP
-----				
PRODUCTION	81/11/30	\$141.66	\$833.33	
	82/01/29	\$1,560.09	\$3,705.84	
*TOTAL PRODUCTION		\$1701.75	\$4539.17	← TYPE = SUBTOTAL
** DEPT_NET		\$2,531.14		← TYPE = RECAP
-----				
TOTAL		\$5,578.35	\$10,421.47	← TYPE = GRANDTOTAL

**Note:** Since this request simply illustrates how to identify different types of totals and subtotals, it omits a StyleSheet.

**Syntax:**      **How to Identify a Grand Total, Subtotal, or Subtotal Calculation**

`TYPE=type, [BY=sortfield] [coltype=column]`

where:

*type*

Identifies a subtotal or total. Select from:

**GRANDTOTAL** which is a grand total (generated by COLUMN-TOTAL, SUBTOTAL, SUB-TOTAL, RECOMPUTE, or SUMMARIZE). See *Identifying a Grand Total* on page 1188 for an example.

**SUBTOTAL** which is a subtotal (generated by SUBTOTAL, SUB-TOTAL, RECOMPUTE, or SUMMARIZE). See *Identifying Subtotals* on page 1189 for an example.

**RECAP** which is a subtotal calculation (generated by ON *sortfield* RECAP or ON *sortfield* COMPUTE). See *Identifying a Subtotal Calculation (RECAP/COMPUTE)* on page 1190 for an example.

**BY**

If you have a request with multiple BY fields, and two or more have subtotal commands associated with them, you can identify each field using the BY identifier. This is helpful when you want to format each subtotal differently or when you want to format only one subtotal.

You only need to include the BY identifier if you have multiple BY fields in your request.

*sortfield*

Specifies the BY field associated with several subtotal commands of a report. Use the fieldname for the value (BY=*fieldname*).

*coltype*

Identifies a specific column to apply formatting. When you include the COLUMN or ACROSSCOLUMN identifier in your declaration, only the subtotal *values* receive the formatting, the labeling text will not. Values can be:

**COLUMN** which is a display column (generated by PRINT, LIST, SUM, or COUNT) or a computed column (generated by COMPUTE).

**ACROSSCOLUMN** where every instance of a display or computed column is repeated across a horizontal sort (ACROSS) row.

If there are several columns being totaled or subtotaled by one command, and you do not specify a column in the StyleSheet, the formatting will be applied to the totals or subtotals for *all* of the columns. It will also be applied to the labeling text for the total and subtotal values.

*column*

Specifies the column whose totals or subtotals you wish to format. For a list of values, see [How to Identify an Entire Column](#) on page 1170.

**Example: Identifying a Grand Total**

The following illustrates how to identify a grand total in a report request. In this example, we only want to format the grand total value for the LINE\_COGS field, so the COLUMN attribute is included in the StyleSheet declaration. The grand total in this request is generated by COLUMN-TOTAL. The relevant StyleSheet declaration is highlighted in the request.

**Note:**

- ❑ To style the entire grand total row, remove the COLUMN attribute from the StyleSheet declaration.
- ❑ To produce the same results you can alternatively use the values N5, P5, or C3 for the COLUMN attribute in the StyleSheet declaration.

```
TABLE FILE CENTORD
SUM QUANTITY LINEPRICE LINE_COGS AND COLUMN-TOTAL
BY ORDER_NUM BY PRODNAME
WHERE ORDER_NUM EQ '48053' OR '48798'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF,$
TYPE=GRANDTOTAL, COLUMN=LINE_COGS, STYLE=BOLD, SIZE=11,$
ENDSTYLE
END
```

The output is:

<u>Order Number:</u>	<u>Product Name:</u>	<u>Quantity:</u>	<u>Line Total</u>	<u>Line Cost Of Goods Sold</u>
48053	150 8MM Camcorder 20 X	1	\$338.28	240.00
	Combo Player - 4 Hd VCR + DVD	1	\$415.73	289.00
	DVD Upgrade Unit for Cent. VCR	1	\$210.42	139.00
	R5 Micro Digital Tape Recorder	1	\$88.52	69.00
	ZT Digital PDA - Commercial	1	\$523.75	349.00
48798	150 8MM Camcorder 20 X	1	\$319.89	240.00
	Combo Player - 4 Hd VCR + DVD	1	\$418.26	289.00
	DVD Upgrade Unit for Cent. VCR	1	\$200.63	139.00
	R5 Micro Digital Tape Recorder	1	\$92.09	69.00
	ZT Digital PDA - Commercial	1	\$483.52	349.00
<b>TOTAL</b>		10	\$3,091.09	<b>2,172.00</b>



**Example: Identifying Subtotals**

The following illustrates how to identify subtotals in a report request. In this example, only subtotal values in the QUANTITY and LINE\_COGS fields are formatted, so the COLUMN attribute is included in the StyleSheet declarations.

Also, since there are two SUBTOTAL commands associated with two of the three BY fields (PLANT and ORDER\_NO), the BY attribute is also included in each declaration to ensure the formatting is applied to the correct value. The relevant StyleSheet declarations are highlighted in the request.

**Note:**

- ☐ To style an entire subtotal row, remove the COLUMN and BY attributes from the StyleSheet declaration.
- ☐ To produce the same results you can, alternatively, use the values COLUMN=N6, COLUMN=P6, or COLUMN=C3 for the COLUMN=LINE\_COGS attribute.
- ☐ To produce the same results you can, alternatively, use the values COLUMN=N4, COLUMN=P4, or COLUMN=C1 for the COLUMN=QUANTITY attribute.

```
TABLE FILE CENTORD
SUM QUANTITY LINEPRICE LINE_COGS AS 'Line Cost of, Goods Sold'
BY PLANT
BY ORDER_NUM
BY PRODNAME
ON PLANT SUBTOTAL
ON ORDER_NUM SUBTOTAL
WHERE ORDER_NUM EQ '35774' OR '48041'
WHERE PLANT EQ 'BOS'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF,$
TYPE=SUBTOTAL, BY=PLANT, COLUMN=LINE_COGS, STYLE=BOLD+ITALIC,
COLOR=BLUE,$
TYPE=SUBTOTAL, BY=ORDER_NUM, COLUMN=QUANTITY, STYLE=BOLD, SIZE=11,$
ENDSTYLE
END
```

The output is:

<u>Manufacturing Plant</u>	<u>Order Number</u>	<u>Product Name</u>	<u>Quantity</u>	<u>Line Total</u>	<u>Line Cost of Goods Sold</u>
BOS	35774	110 VHS-C Camcorder 20 X	146	\$38,817.56	36,354.00
		120 VHS-C Camcorder 40 X	146	\$40,426.90	37,814.00
		2 Hd VCR LCD Menu	279	\$39,693.32	35,991.00
		650DL Digital Camcorder 150 X	146	\$116,907.15	103,660.00
*TOTAL ORDER_NUM 35774			<b>717</b>	\$235,844.93	213,819.00
	48041	150 8MM Camcorder 20 X	1	\$307.43	240.00
		Combo Player - 4 Hd VCR + DVD	1	\$398.95	289.00
		DVD Upgrade Unit for Cent. VCR	1	\$194.40	139.00
		R5 Micro Digital Tape Recorder	1	\$93.91	69.00
		ZT Digital PDA - Commercial	1	\$528.38	349.00
*TOTAL ORDER_NUM 48041			<b>5</b>	\$1,523.07	1,086.00
*TOTAL PLANT BOS			722	\$237,368.00	<b>214,905.00</b>
TOTAL			722	\$237,368.00	214,905.00

### **Example:** Identifying a Subtotal Calculation (RECAP/COMPUTE)

The following illustrates how to identify a subtotal calculation created with a RECAP or COMPUTE phrase. In this example, the subtotal calculation is generated with ON PLANT RECAP QTY/F6=QUANTITY. The relevant StyleSheet declaration is highlighted in the request.

**Note:** If there is more than one RECAP or COMPUTE field in your request, you can distinguish them by adding BY=*fieldname* to the StyleSheet declaration.

```
TABLE FILE CENTORD
SUM QUANTITY LINEPRICE LINE_COGS AS 'Line Cost of, Goods Sold'
BY PLANT BY ORDER_NUM
ON PLANT RECAP QTY/F6=QUANTITY;
WHERE PLANT EQ 'BOS'
WHERE ORDER_NUM LT '56098'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=RECAP, STYLE=BOLD+ITALIC, $
ENDSTYLE
END
```

The output is:

<u>Manufacturing Plant</u>	<u>Order Number</u>	<u>Quantity</u>	<u>Line Total</u>	<u>Line Cost of Goods Sold</u>
BOS	28004	6	\$2,023.62	1,395.00
	28011	597	\$1,25,556.08	125,306.00
	28024	597	\$1,44,012.51	125,306.00
	28035	5	\$1,541.26	986.00
	28036	542	\$1,29,859.27	109,331.00
	28037	675	\$1,66,729.48	135,574.00
	28040	5	\$1,659.77	1,086.00
	28041	5	\$1,642.86	1,086.00
	28042	5	\$1,724.17	1,086.00
	28043	4	\$2,020.11	1,347.00
	28045	1,142	\$260,105.24	227,651.00

**\*\* QTY 655661**

## Identifying a Heading, Footing, Title, or FML Free Text

A report data is framed by headings, footings, and titles. These provide context for the data. You can identify and format many categories of headings, footings, and titles in a report, including:

- ☐ Report, page, and sort headings.
- ☐ Report, page, and sort footings.
- ☐ Column titles.
- ☐ Horizontal sort (ACROSS) titles and values.
- ☐ Free text in Financial Modeling Language (FML) reports.

## Identifying a Column or Row Title

Within a StyleSheet you can identify column titles and horizontal sort (ACROSS) values of a report in order to format them. The following example illustrates where column titles and horizontal sort values are in a report, and which TYPE values you use to identify them.

```
TABLE FILE EMPLOYEE
SUM GROSS AND DED_AMT
ACROSS DEPARTMENT BY PAY_DATE
END
```

PAGE 1

DEPARTMENT		TYPE = ACROSSTITLE		TYPE = ACROSSVALUE	
MIS		PRODUCTION		TYPE = TITLE	
PAY_DATE	GROSS	DED_AMT	GROSS	DED_AMT	
81/11/30	\$2,147.75	\$1,406.79	\$833.33	\$141.66	
81/12/31	\$2,147.75	\$1,406.79	\$833.33	\$141.66	
82/01/29	\$3,247.75	\$1,740.89	\$3,705.84	\$1,560.09	
82/02/26	\$3,247.75	\$1,740.89	\$4,959.84	\$2,061.69	
82/03/31	\$3,247.75	\$1,740.89	\$4,959.84	\$2,061.69	
82/04/30	\$5,890.84	\$3,386.73	\$4,959.84	\$2,061.69	
82/05/28	\$6,649.50	\$3,954.35	\$7,048.84	\$3,483.88	
82/06/30	\$7,460.00	\$4,117.03	\$7,048.84	\$3,483.88	
82/07/30	\$7,460.00	\$4,117.03	\$7,048.84	\$3,483.88	
82/08/31	\$9,000.00	\$4,575.72	\$9,523.84	\$4,911.12	
PAGE	1				

**Note:** Since this request simply illustrates how to identify column titles and horizontal sort values in a report, it omits a StyleSheet.

**Syntax:** How to Identify a Column Title

TYPE=TITLE, [COLUMN=column]

where:

COLUMN

Is used to specify one or more column titles. If you omit this attribute and value, the formatting will be applied to all of the report column titles.

column

Specifies the column whose title you wish to format. For column values, see [How to Identify an Entire Column](#) on page 1170.

**Syntax:** How to Identify a Horizontal Sort Title or Value

TYPE={ACROSSTITLE|ACROSSVALUE}, [ACROSS=column]

where:

ACROSSTITLE

Specifies a horizontal sort (ACROSS) title.

**ACROSSVALUE**

Specifies a horizontal sort (ACROSS) value.

Although horizontal sort values are not technically titles, they often function as titles that categorize the column titles appearing beneath them.

**ACROSS**

Is used to specify titles or values for a specific horizontal sort field. If you omit this attribute and value, the formatting will be applied to the titles or values of the horizontal sort fields of all reports.

*column*

Specifies the horizontal sort (ACROSS) field whose title or values you wish to format. For values you can assign to this attribute, see [Identifying a Row of Horizontal Sort \(ACROSS\) Data](#) on page 1183.

**Example: Identifying Column Titles and Horizontal Sort (ACROSS) Values**

This example illustrates how to identify vertical sort titles, horizontal sort titles, and horizontal sort values. The vertical sort titles (TYPE=TITLE) are Manufacturing Plant, Quantity Sold and Product Cost, the horizontal sort title (TYPE=ACROSSTITLE) is Year, and the horizontal sort values (TYPE=ACROSSVALUE) are 2001, 2002, and TOTAL.

The following also demonstrates how to assign drill-down values to the individual ACROSS values of 2001 and 2002, and not the ROW-TOTAL value of TOTAL. The StyleSheet declarations in this request are shown in bold.

```
TABLE FILE CENTORD
SUM QUANTITY AS 'Quantity,Sold' LINE_COGS/I9 AS 'Product,Cost'
BY PLANT
ACROSS YEAR
WHERE YEAR EQ '2001' OR '2002'
HEADING
"Plant Production Cost Analysis"
ON TABLE ROW-TOTAL AS 'TOTAL'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF,$
TYPE=TITLE, STYLE=BOLD, $
TYPE=ACROSSTITLE, STYLE=BOLD, $
TYPE=ACROSSVALUE, STYLE=BOLD+ITALIC, COLOR=BLUE, FOCEXEC=DETAILS, $
TYPE=ACROSSVALUE, COLUMN=N4, STYLE=BOLD, COLOR=RED, $
TYPE=ACROSSVALUE, COLUMN=ROWTOTAL(1), COLOR='BLACK',FOCEXEC=NONE, $
ENDSTYLE
END
```

The following image shows the report output.

#### Plant Production Cost Analysis

Manufacturing Plant	YEAR				TOTAL	
	<u>2001</u>		<u>2002</u>			
	Quantity Sold	Product Cost	Quantity Sold	Product Cost	Quantity Sold	Product Cost
BOS	491,080	109839382	538,836	120518527	1,029,916	230357909
DAL	185,785	41129581	203,937	45147907	389,722	86277488
LA	109,326	24027600	119,925	26356066	229,251	50383666
ORL	184,519	41143292	202,390	45127226	386,909	86270518
SEA	41,331	9578621	45,349	10510177	86,680	20088798
STL	369,456	81807263	405,268	89734521	774,724	171541784

#### **Syntax:** How to Identify Free Text in an FML Report

`TYPE=FREE TEXT, LABEL={Rn | label}`

where:

*n*

Is an implicit row label. To determine the value of *n*, count the number of rows up to and including the desired row.

*label*

Is an explicit row label.

**Example: Identifying Free Text in an FML Report**

The following illustrates how to identify free text in an FML report. In this example, the free text are the rows "CASH ACCOUNTS" and "OTHER CURRENT ASSETS". The relevant StyleSheet declarations are displayed in bold.

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
" --- CASH ACCOUNTS ---" LABEL CA          OVER
1010 AS 'CASH ON HAND'                     OVER
1020 AS 'DEMAND DEPOSITS'                   OVER
1030 AS 'TIME DEPOSITS'                     OVER
" "                                         OVER
" --- OTHER CURRENT ASSETS ---" LABEL OCA   OVER
1100 AS 'ACCOUNTS RECEIVABLE'               OVER
1200 AS 'INVENTORY'                         OVER
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=FREETEXT, LABEL=CA, STYLE=BOLD, SIZE=12, $
TYPE=FREETEXT, LABEL=OCA, STYLE=BOLD, SIZE=12, $
ENDSTYLE
END
```

The output is:

	<u>AMOUNT</u>
<b>--- CASH ACCOUNTS ---</b>	
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
<b>--- OTHER CURRENT ASSETS ---</b>	
ACCOUNTS RECEIVABLE	18,829
INVENTORY	27,307

**Identifying a Heading or Footing**

Within a StyleSheet you can identify a report headings and footings, and the individual lines, text strings, and fields within them, in order to format them.

A TABLE request can have more than one page heading or footing. For each heading or footing, a WHEN clause against the data being retrieved can determine whether the heading or footing displays on the report output. The CONDITION StyleSheet attribute enables you to identify a specific WHEN clause so that you can style each heading or footing separately.

The following code and output examples illustrate where a report heading (TABHEADING), a page heading (HEADING), a sort heading (SUBHEAD), a sort footing (SUBFOOT), and a report footing (TABFOOTING) are in a report, and which TYPE values you use to identify them.

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL HIRE_DATE
BY LAST_NAME
BY FIRST_NAME
ON TABLE SUBHEAD
"CONFIDENTIAL INFORMATION"
"SWIFTY INFORMATION GROUP - EMPLOYEE LIST BY DEPARTMENT"
HEADING CENTER
"</1>EMPLOYEE LIST FOR DEPARTMENT: <DEPARTMENT"
ON LAST_NAME SUBHEAD
"ID: <EMP_ID"
ON LAST_NAME SUBFOOT
"*** REVIEW SALARY FOR <FIRST_NAME <LAST_NAME"
FOOTING
"CONFIDENTIAL INFORMATION"
ON TABLE SUBFOOT
"</1>***END OF REPORT***"
ON TABLE SET ONLINE-FMT STANDARD
END
```

The following output goes with the previous code example:

```
PAGE          1

CONFIDENTIAL INFORMATION
SWIFTY INFORMATION GROUP - EMPLOYEE LIST BY DEPARTMENT
EMPLOYEE LIST FOR DEPARTMENT: PRODUCTION
LAST_NAME      FIRST_NAME      CURR_SAL  HIRE_DATE
-----
ID: 119329144  JOHN              $29,700.00  82/08/01
** REVIEW SALARY FOR JOHN BANNING
ID: 326179357  ROSEMARIE         $21,780.00  82/04/01
** REVIEW SALARY FOR ROSEMARIE BLACKWOOD
CONFIDENTIAL INFORMATION
***END OF REPORT***
```

TYPE = TABHEADING

TYPE = HEADING

TYPE = SUBHEAD

TYPE = SUBFOOT

TYPE = SUBHEAD

TYPE = FOOTING

TYPE = TABFOOTING



**Note:** Since this request simply illustrates how to identify different types of headings and footings, it omits a StyleSheet.

**Syntax:**      **How to Identify a Heading or Footing**

*TYPE=headfoot, [BY=sortcolumn]*

where:

*headfoot*

Identifies a heading or footing. Select from:

- ☐ **TABHEADING**, which is a report heading. This appears once at the beginning of the report and is generated by ON TABLE SUBHEAD.
- ☐ **TABFOOTING**, which is a report footing. This appears once at the end of the report and is generated by ON TABLE SUBFOOT.
- ☐ **HEADING**, which is a page heading. This appears at the top of every report page and is generated by HEADING.
- ☐ **FOOTING**, which is a page footing. This appears at the bottom of every report page and is generated by FOOTING.
- ☐ **SUBHEAD**, which is a sort heading. This appears at the beginning of a vertical (BY) sort group (generated by ON *sortfield* SUBHEAD).
- ☐ **SUBFOOT**, which is a sort footing. This appears at the end of a vertical (BY) sort group (generated by ON *sortfield* SUBFOOT).

*BY*

When there are several sort headings or sort footings, each associated with a different vertical sort (BY) column, this enables you to identify which sort heading or sort footing you wish to format.

If there are several sort headings or sort footings associated with different vertical sort (BY) columns, and you omit this attribute and value, the formatting will be applied to all of the sort headings or footings.

*sortcolumn*

Specifies the vertical sort (BY) column associated with one of the report sort headings or sort footings.

**Example: Identifying a Page Heading and a Report Footing**

The following illustrates how to identify a page heading, which appears at the top of every report page, and a report footing, which appears only on the last page of the report. The relevant StyleSheet declarations are highlighted in the request.

```
TABLE FILE CENTORD
HEADING
"Sales Quantity and Amount by Plant"
SUM QUANTITY LINEPRICE
BY PLANT
ON TABLE SUBFOOT
" "
"***End of Report***"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=HEADING, FONT=TIMES, SIZE=12, STYLE=BOLD,$
TYPE=TABFOOTING, JUSTIFY=CENTER, STYLE=BOLD, SIZE=11,$
ENDSTYLE
END
```

The output is:

## Sales Quantity and Amount by Plant

<u>Manufacturing Plant</u>	<u>Quantity:</u>	<u>Line Total</u>
BOS	1,033,818	\$262,433,960.63
DAL	390,844	\$97,751,846.65
LA	229,256	\$57,507,080.82
ORL	386,909	\$97,616,855.94
SEA	86,680	\$22,742,743.88
STL	776,743	\$195,970,536.09

**\*\*\*End of Report\*\*\***

**Syntax:**      **How to Identify an Individual Line in a Heading or Footing**

*TYPE=type, LINE=line\_#*

where:

*type*

Identifies a type of heading or footing. Select from HEADING, FOOTING, TABHEADING, TABFOOTING, SUBHEAD, or SUBFOOT. For details, see [Identifying a Heading or Footing](#) on page 1195.

*line\_#*

Identifies a line by its position in the heading or footing.

**Example:**      **Identifying an Individual Line in a Heading**

The following example illustrates how to format individual lines in a heading. Heading line 1 (Sales Quantity Analysis) is formatted in bold, point-size 11. Heading line 2 (\*\*Confidential\*\*) is formatted in bold and red. The relevant StyleSheet declarations are highlighted in the request.

```
TABLE FILE CENTORD
HEADING
"Sales Quantity Analysis"
"***Confidential***"
" "
SUM QUANTITY
ACROSS YEAR
BY PLANT
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=HEADING, LINE=1, SIZE=11, STYLE=BOLD,$
TYPE=HEADING, LINE=2, COLOR=RED, STYLE=BOLD,$
TYPE=HEADING, JUSTIFY=CENTER,$
ENDSTYLE
END
```

The output is:

Sales Quantity Analysis			
**Confidential**			
	YEAR		
	2000	2001	2002
Manufacturing Plant			
BOS	3,902	491,080	538,836
DAL	1,122	185,785	203,937
LA	5	109,326	119,925
ORL	.	184,519	202,390
SEA	.	41,331	45,349
STL	2,019	369,456	405,268

**Syntax:**      **How to Identify a Text String in a Heading or Footing**

`TYPE=type, [LINE=line_#], [OBJECT=TEXT], ITEM=item_#`

where:

*type*

Identifies a type of heading or footing. Select from HEADING, FOOTING, TABHEADING, TABFOOTING, SUBHEAD, or SUBFOOT. For details, see [Identifying a Heading or Footing](#) on page 1195.

*line\_#*

Identifies a line by its position in the heading or footing. You only need to include the LINE attribute if you have a multi-line heading or footing.

`TEXT`

Formats only text strings and Dialogue Manager variables (also known as &variables). It is not necessary to use OBJECT=TEXT in your declaration unless you are styling both text strings and embedded fields in the same heading or footing.

*item\_#*

Identifies an item by its position in a line.

If you need to apply formatting to several parts of a continuous text string that displays on one line, you can break the header or footer into multiple parts using spot markers. Place the spot marker after the text string you wish to specify. The `<+0>` spot marker will not add any additional spaces to your heading or footing. When using spot markers, text is divided as follows:

```
TABLE FILE GGSales
SUM UNITS DOLLARS BY CATEGORY BY PRODUCT
HEADING
    "First Quarter <+0>Sales <+0>Report"
```

```
ON TABLE SET STYLE *
TYPE=HEADING, OBJECT=TEXT, ITEM=1, FONT=ARIAL, $
TYPE=HEADING, OBJECT=TEXT, ITEM=2, SIZE=14, $
TYPE=HEADING, OBJECT=TEXT, ITME=3, STYLE=BOLD, $
ENDSTYLE
END
```

**Note:** When a closing spot marker is immediately followed by an opening spot marker (`><`), a single space text item will be placed between the two spot markers (`> <`). This must be considered when applying formatting.

The position value also depends on whether you are using the `OBJECT` attribute or not. If you are using:

- ☐ `OBJECT=TEXT`, count only text strings from left to right.
- ☐ No `OBJECT`, count text strings and embedded field values from left to right.

**Example:**     **Identifying a Text String in a Heading Using Spot Markers**

The following illustrates how to apply different formats to text strings in a heading using spot markers. The spot markers used in this example are <+0> since they do not add any spaces. The relevant StyleSheet declarations are highlighted in the request.

```
TABLE FILE CENTORD
HEADING
"Third Quarter,<+0>2002:<+0> Sales Quantity Analysis"
SUM QUANTITY BY PLANT
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=HEADING, OBJECT=TEXT, ITEM=1, STYLE=BOLD+UNDERLINE, SIZE=12, $
TYPE=HEADING, OBJECT=TEXT, ITEM=2, COLOR=BLUE, SIZE=12,
STYLE=BOLD+UNDERLINE, $
TYPE=HEADING, OBJECT=TEXT, ITEM=3, STYLE=ITALIC,$
ENDSTYLE
END
```

The output is:

<b><u>Third Quarter,2002:</u> Sales Quantity Analysis</b>	
<u>Manufacturing Plant</u>	<u>Quantity</u>
BOS	1,033,818
DAL	390,844
LA	229,256
ORL	386,909
SEA	86,680
STL	776,743

**Syntax:**     **How to Identify an Embedded Field in a Heading or Footing**

```
TYPE=type, [LINE=line_#], OBJECT=FIELD, [ITEM=item #]
```

where:

*type*

Identifies a type of heading or footing. Select from HEADING, FOOTING, TABHEADING, TABFOOTING, SUBHEAD, or SUBFOOT. For details, see [Identifying a Heading or Footing](#) on page 1195.

*line\_#*

Identifies a line by its position in the heading or footing. You only need to include the LINE attribute if you have a multi-line heading or footing.

*item\_#*

Identifies an item by its position in a line.

If you have more than one embedded field in a heading or footing, you must specify the field you wish to format by giving the item number. Count items from left to right. Do not include text fields in the count. You do not need to specify the item number if there is only one embedded field in the heading or footing.

**Note:** BORDER options are not supported on specific ITEMS in a HEADING, FOOTING, SUBHEAD, SUBFOOT.

**Example: Identifying Embedded Fields in a Heading**

The following illustrates how to format an embedded field in a heading. Notice that the item number is not specified in the StyleSheet declaration since there is only one embedded field in the heading. The relevant StyleSheet declaration is highlighted in the request.

```
TABLE FILE CENTORD
HEADING
"Sales For <YEAR By Plant"
SUM QUANTITY BY PLANT
WHERE YEAR EQ 2000
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=HEADING, OBJECT=TEXT, COLOR=BLUE,$
TYPE=HEADING, OBJECT=FIELD, COLOR=RED, STYLE=BOLD,$
ENDSTYLE
END
```

The output is:

Sales For 2000 By Plant	
<u>Manufacturing Plant</u>	<u>Quantity</u>
BOS	3,902
DAL	1,122
LA	5
STL	2,019

**Syntax:**      How to Identify a Specific Heading or Footing Based on a WHEN Clause

```
TYPE = {HEADING|FOOTING}, CONDITION = n, ... , $
```

where:

*n*

Is the number of the WHEN condition in the heading or footing from top to bottom. If not specified, formatting applies to all headings and footings.

**Example:**      Styling Multiple Headings With WHEN

The following request displays a page for each employee with salary and job code information for that employee. The first WHEN condition applies if the employee is female. The second WHEN condition applies if the employee is male. The third WHEN condition applies if the department is MIS. The fourth WHEN condition applies if the department is PRODUCTION. The StyleSheet declarations include styling elements for the second and third conditions:



```

DEFINE FILE EMPLOYEE
GENDER/A1 = DECODE FIRST_NAME(ALFRED 'M' RICHARD 'M' JOHN 'M'
    ANTHONY 'M' ROGER 'M' MARY 'F' DIANE 'F' JOAN 'F' ROSEMARIE 'F'
    BARBARA 'F');
MIXEDNAME/A15 = LCWORD(15, LAST_NAME, MIXEDNAME);
NAME/A16 = MIXEDNAME||',';
END

TABLE FILE EMPLOYEE
PRINT LAST_NAME NOPRINT GENDER NOPRINT NAME NOPRINT
HEADING
"Dear Ms. <NAME>"
    WHEN GENDER EQ 'F';
HEADING
"Dear Mr. <NAME>"
    WHEN GENDER EQ 'M';
HEADING
" "
HEADING
"This is to inform you that your current salary is "
"<CURR_SAL and <CURR_JOBCODE>is your job code."
" "
"Sincerely,"
HEADING
"Barbara Cross "
    WHEN DEPARTMENT EQ 'MIS';
HEADING
"John Banning "
    WHEN DEPARTMENT EQ 'PRODUCTION' ;
WHERE LAST_NAME NE 'BANNING' OR 'CROSS'
BY EMP_ID NOPRINT PAGE-BREAK
ON TABLE SET PAGE NOPAGE
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=HEADING, CONDITION=2, STYLE=ITALIC,$
TYPE=HEADING, CONDITION=3, STYLE=BOLD,$
ENDSTYLE
END

```

In the StyleSheet for the request, heading lines displayed because of the first condition are in italics and heading lines displayed because of the third condition are in boldface.

The first page of output is for a male employee, so the greeting line is in italics:

```

Dear Mr. Stevens,

This is to inform you that your current salary is
$11,000.00 and that A07 is your job code.

Sincerely,
John Banning

```

The second page of output is for an employee in the MIS department, so the signature line is in boldface:

Dear Ms. Smith,

This is to inform you that your current salary is  
\$13,200.00 and that B14 is your job code.

Sincerely,  
**Barbara Cross**

## Identifying a Page Number, Underline, or Skipped Line

In a report you can identify and format page numbers, underlines, and skipped lines using the PAGENUM, SKIPLINE, and UNDERLINE attributes.

Note that although you can insert skipped lines and underlines in an HTML report, formatting is not supported.

The following code and output examples illustrate where the PAGENUM, UNDERLINE, and SKIPLINE components appear in a report, and which TYPE values you use to identify them.

```
SET ONLINE-FMT=PDF
TABLE FILE CENTORD
HEADING
"Sales By Plant"
SUM QUANTITY
BY PLANT BY YEAR
WHERE PLANT EQ 'BOS' OR 'DAL'
ON YEAR UNDER-LINE
ON PLANT SKIP-LINE
END
```

The following output goes with the previous code example.

<b>TYPE=PAGENUM</b>	→	PAGE	1
		Sales By Plant	
		<u>Manufacturing Plant</u>	<u>YEAR</u> <u>Quantity:</u>
		BOS	2000      3,902
<b>TYPE=UNDERLINE</b>	→		2001      491,080
	→		2002      538,836
<b>TYPE=SKIPLINE</b>	→	DAL	2000      1,122
			2001      185,785
			2002      203,937

**Note:** Since this request simply illustrates where the components appear in a report, it omits a StyleSheet.

### **Syntax:**      How to Identify a Page Number, Underline, or Skipped Line

*TYPE=type*

where:

*type*

Identifies the report component. Select from:

**PAGENUM** which identifies page numbers.

**SKIPLINE** which denotes skipped lines generated by ON *field* SKIP-LINE. This is not supported for reports in HTML format.

**UNDERLINE** which identifies underlines generated by ON *field* UNDER-LINE, or by BAR in a Financial Modeling Language (FML) report. This is not supported for reports in HTML format.

**Example: Identifying Underlines and Page Numbers**

The following illustrates how to identify underlines and page numbers in a report request. The relevant StyleSheet declarations appear in boldface in the request.

Note that this report is formatted in PDF since formatting is not supported for underlines in an HTML report.

```
SET ONLINE-FMT=PDF
TABLE FILE CENTORD
HEADING
"Sales By Plant"
SUM QUANTITY
BY PLANT BY YEAR
WHERE PLANT EQ 'BOS' OR 'DAL' OR 'LA'
ON PLANT UNDER-LINE SKIP-LINE
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=HEADING, OBJECT=TEXT, COLOR=BLUE, FONT=ARIAL, $
TYPE=PAGENUM, STYLE=ITALIC, SIZE=8, $
TYPE=UNDERLINE, COLOR=RED, $
ENDSTYLE
END
```

The output is:

*PAGE* 1

### *Sales By Plant*

<u>Manufacturing Plant</u>	<u>YEAR</u>	<u>Quantity:</u>
BOS	2000	3,902
	2001	491,080
	2002	538,836
<hr/>		
DAL	2000	1,122
	2001	185,785
	2002	203,937
<hr/>		
LA	2000	5
	2001	109,326
	2002	119,925
<hr/>		

**Example: Identifying Skipped Lines**

The following illustrates how to identify skipped lines in a report. The relevant StyleSheet declaration is highlighted in the request.

```
SET ONLINE-FMT=PDF
TABLE FILE CENTINV
HEADING
"Low Stock Report"
" "
SUM QTY_IN_STOCK
WHERE QTY_IN_STOCK LT 5000
BY PRODNAME
ON PRODNAME SKIP-LINE
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=SKIPLINE, BACKCOLOR=SILVER, $
ENDSTYLE
END
```

The output is:

### Low Stock Report

<u>Product Name:</u>	<u>Quantity In Stock:</u>
110 VHS-C Camcorder 20 X	4000
120 VHS-C Camcorder 40 X	2300
340SX Digital Camera 65K P	990
650DL Digital Camcorder 150 X	2972
DVD Upgrade Unit for Cent. VCR	199
R5 Micro Digital Tape Recorder	1990





# Chapter 20

## Using an External Cascading Style Sheet

---

Cascading style sheets (CSS) provide a standard way of formatting HTML documents. To format WebFOCUS HTML report output using an external CSS, simply link the CSS to the report (using the WebFOCUS StyleSheet CSSURL attribute) and, optionally, apply the CSS classes to specific report components (using the CLASS attribute).

### In this chapter:

- ☐ [What Is a Cascading Style Sheet?](#)
  - ☐ [Why Use an External Cascading Style Sheet?](#)
  - ☐ [Formatting a Report With an External Cascading Style Sheet](#)
  - ☐ [Working With an External Cascading Style Sheet](#)
  - ☐ [Applying External Cascading Style Sheet Formatting](#)
  - ☐ [Combining an External CSS With Other Formatting Methods](#)
  - ☐ [Linking to an External Cascading Style Sheet](#)
  - ☐ [Inheritance and External Cascading Style Sheets](#)
  - ☐ [Using External Cascading Style Sheets With Non-HTML Reports](#)
  - ☐ [Requirements for Using an External Cascading Style Sheet](#)
  - ☐ [FAQ About Using External Cascading Style Sheets](#)
  - ☐ [Troubleshooting External Cascading Style Sheets](#)
- 

### What Is a Cascading Style Sheet?

Cascading style sheets (CSS) are an extension to HTML that allow you to specify formatting for an HTML document. You can use two kinds of CSS with WebFOCUS:

- ☐ **An internal cascading style sheet**, which is stored internally in the HTML document that it formats. For information about generating and using an internal CSS for a WebFOCUS report, see [Generating an Internal Cascading Style Sheet for HTML Reports](#) on page 1140.

- ❑ **An external cascading style sheet**, which is stored in a separate file that can be shared by multiple documents. The external CSS file can reside on any web server accessible to the browser. You specify its location using the CSSURL WebFOCUS StyleSheet attribute, the CSSURL SET parameter, or (in special cases) the LINK element.

You can define classes in a cascading style sheet, and format a report component by assigning one of these CSS classes to it. Classes are described in [What Are Cascading Style Sheet Rules and Classes?](#) on page 1212.

Cascading style sheets are called *cascading* because several different style sheets can be active for a single document at the same time. For example, one style sheet may be associated with the document itself, another style sheet may be linked to the first one, and yet another may be associated with the web browser on which the document is being displayed. When multiple style sheets are in effect, they are applied to the document in a pre-determined sequence set by the browser: their formatting instructions can be thought of as cascading from one style sheet to the next.

The benefits of using an external cascading style sheet to format a report are described in [Why Use an External Cascading Style Sheet?](#) on page 1213.

You will find external cascading style sheets relevant if you:

- ❑ **Develop reports**, since you now have an improved way of formatting those reports.
- ❑ **Are responsible for presentation guidelines** for web documents, since you will now be able to apply your existing cascading style sheets to reports.

For information about *internal* cascading style sheets, see [Generating an Internal Cascading Style Sheet for HTML Reports](#) on page 1140.

**Need more information about CSS?** This WebFOCUS documentation assumes that you have a working knowledge of cascading style sheets. Teaching about CSS is beyond the scope of this documentation, but many sources of information are available to you. A useful place to begin online is the World Wide Web Consortium's cascading style sheets home page (<http://www.w3.org/Style/CSS/>).

## What Are Cascading Style Sheet Rules and Classes?

A cascading style sheet (CSS) defines formatting in statements called *rules*. For example, this is a simple rule that makes the background color of the body of an HTML page yellow:

```
BODY {background: yellow}
```

Each rule has a *selector* (BODY in this example) and a *declaration* (background: yellow). A declaration has a *property* (background) and a *value* assigned to the property (yellow).



A declaration defines formatting, and a selector determines to what the formatting will be applied. A selector can be any HTML element. A selector can also be a *class*. You can define a class simply by creating a rule for it. By creating rules for classes of an element, you can define different formatting for the same element.

For example, you may wish to display text in a different color depending on whether it is in a sort column, aggregate column, or detail column in a report. To accomplish this you could create three classes of the BODY element, `sortColumn`, `aggregateColumn`, and `detailColumn`:

```
BODY.sortColumn {color: blue}
BODY.aggregateColumn {color: green}
BODY.detailColumn {color: black}
```

You can also define a generic class, that is, a class that is not limited to a single element. For example:

```
.pageFootings {font-weight: bolder}
```

You can use a generic class to specify formatting for any WebFOCUS report component.

Using external cascading style sheet rules and classes to format a report is described in [Formatting a Report With an External Cascading Style Sheet](#) on page 1214.

## Why Use an External Cascading Style Sheet?

If you already use WebFOCUS StyleSheets to format reports, you can realize these additional benefits by combining them with external cascading style sheets:

- ❑ **Increased formatting options.** Almost any formatting that you can specify in a cascading style sheet you can apply to a report. This enables you to take advantage of formatting options that are unavailable via native WebFOCUS StyleSheet attributes. For example, you can use browser-based measurements so that the person who views the report can control the size of fonts, margins, and other elements whose size has been specified in the CSS in terms of *em*, a unit of relative measurement. You can also use CSS to control line height and letter spacing, and in general can use CSS to exercise more control over positioning items in a report.
- ❑ **Improved performance.** Cascading style sheets enable WebFOCUS to generate more concise HTML output. This reduces the bandwidth used by the network to return a report to a browser, and displays the report faster.
- ❑ **Reduced effort.** Enterprises that already use cascading style sheets can now also apply them to WebFOCUS reports, avoiding duplication of effort to specify and maintain formatting instructions.

- ❑ **Easier standards conformance.** You can ensure that reports conform to your enterprise's formatting guidelines, because now formatting instructions for all your Web documents can be specified in one set of cascading style sheets (instead of replicating some of them in WebFOCUS StyleSheets).

### Formatting a Report With an External Cascading Style Sheet

There are just three items required to format a report with an external cascading style sheet (CSS):

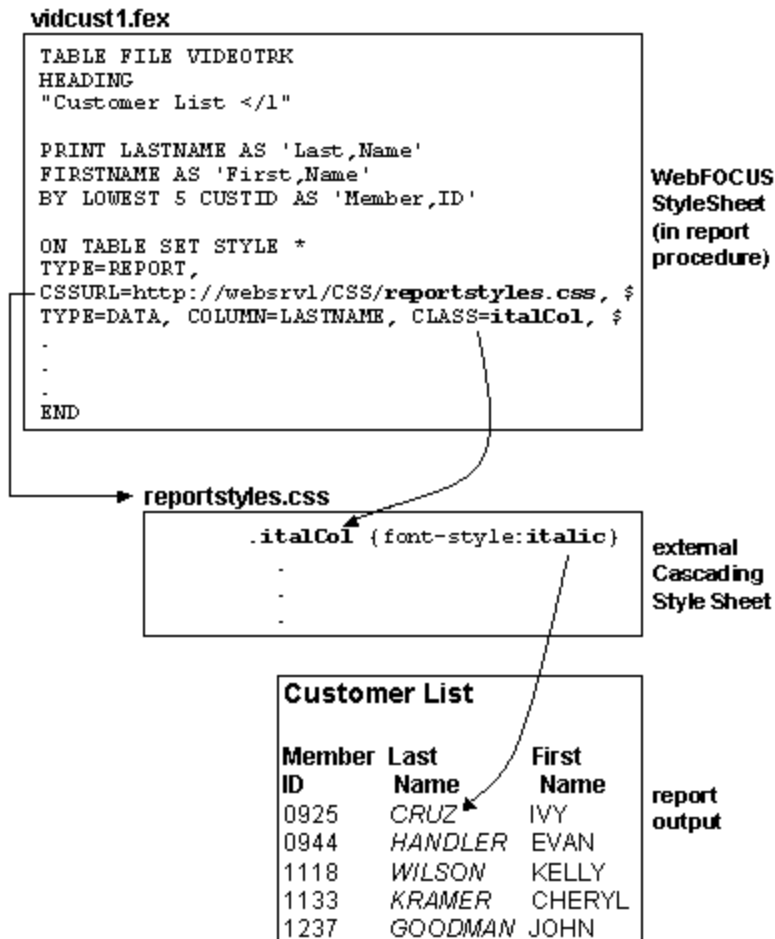
- ❑ **An external cascading style sheet** that specifies the formatting to be applied to the report. For more information, see [Working With an External Cascading Style Sheet](#) on page 1220.
- ❑ **A WebFOCUS StyleSheet** in which you apply external CSS formatting to the components of your report. However, you do not need a WebFOCUS StyleSheet when you apply formatting to the *entire* report. For more information, see [Applying External Cascading Style Sheet Formatting](#) on page 1224.

(Although you can also use a WebFOCUS StyleSheet to specify additional formatting outside of the external CSS, this is subject to restrictions. For more information, see [Combining an External CSS With Other Formatting Methods](#) on page 1226.)

- ❑ **A link to the external cascading style sheet** from the report. For more information, see [Linking to an External Cascading Style Sheet](#) on page 1228.

To find out how to use these three items to format a report, see [How to Format a Report Using an External Cascading Style Sheet](#) on page 1216.

For an example that demonstrates how these items work together, see [Linking to the ReportStyles External Cascading Style Sheet](#) on page 1217 and the following diagram:



### **Procedure: How to Format a Report Using an External Cascading Style Sheet**

To format a report using an external cascading style sheet (CSS):

1. **Specify the report formatting in the CSS.** Determine which cascading style sheet you will use, which of its rules specify default formatting for your report, and which of its classes are suitable for you to apply to the report components. Alternatively, if you are creating a new CSS, or extending an existing one, define new rules to specify formatting for your report. To specify formatting for:

- ❑ **A report component**, you can use a rule for any generic class (that is, any class not declared for an element). For an example, see [A CSS Rule for the ColumnTitle Class](#) on page 1217.

Graphs differ from other types of reports: to specify formatting for the page in which the graph appears, or for the graph heading or footing, you can use a rule for the BODY element, but not a rule for a class. You cannot format other graph components.

- ❑ **The entire report**, use a rule for the BODY or TD elements (not a rule for a class of these elements), and skip Step 2. This is an effective way of specifying the default formatting of a report, and generates more efficient report output than does applying a CSS class to the entire report. For an example, see [A CSS Rule for the TD Element](#) on page 1217.

Graphs differ from other types of reports: you can use a rule for the BODY element, but not one for TD. A rule for BODY will format the page in which the graph appears, and its heading and footing, but not the graph itself.

For more information, see [Working With an External Cascading Style Sheet](#) on page 1220.

2. **Assign classes to report components.** In a WebFOCUS StyleSheet, assign a cascading style sheet class to each report component that you want to format. Specify the class using the CLASS attribute. You can assign each component a different class, and you can assign the same class to multiple components.

For an example, see [Applying a CSS Class to ACROSS Values in a Report](#) on page 1217.

For more information, see [Applying External Cascading Style Sheet Formatting](#) on page 1224.

3. **Link to the CSS.** Link to the external cascading style sheet by assigning the CSS file URL to either the CSSURL WebFOCUS StyleSheet attribute or to the CSSURL SET parameter. For instructions, see [Using the CSSURL Attribute and Parameter](#) on page 1228.

There is one exception: if you embed the report output in an existing HTML page using the -HTMLFORM command, include a LINK element in that HTML page instead of setting CSSURL.

For an example, see [Linking to the ReportStyles External Cascading Style Sheet](#) on page 1217. For more information, see [Linking to an External Cascading Style Sheet](#) on page 1228.

**Problems?** If you encounter problems, see [Troubleshooting External Cascading Style Sheets](#) on page 1245.

**Reference:** **A CSS Rule for the ColumnTitle Class**

This cascading style sheet (CSS) rule declares the ColumnTitle generic class (that is, a class not tied to an element):

```
.ColumnTitle {font-family:helvetica; font-weight:bold; color:blue;}
```

**Reference:** **A CSS Rule for the TD Element**

This cascading style sheet (CSS) rule for the TD element specifies the element's font family:

```
TD {font-family:helvetica}
```

Because this rule is for the TD element, its formatting is applied to an entire report, not just a component of the report.

For a more comprehensive example of using a rule for the TD element to provide general report formatting, see [Linking to the ReportStyles External Cascading Style Sheet](#) on page 1217.

**Reference:** **Applying a CSS Class to ACROSS Values in a Report**

This WebFOCUS StyleSheet declaration formats ACROSS values by applying the formatting specified for the ColumnTitle class:

```
TYPE=AcrossValue, CLASS=ColumnTitle, $
```

**Reference:** **Linking to the ReportStyles External Cascading Style Sheet**

This WebFOCUS StyleSheet declaration links to the ReportStyles external cascading style sheet:

```
TYPE=REPORT, CSSURL=http://webserv1/css/reportstyles.css
```

You could accomplish the same thing using a SET command:

```
SET CSSURL=http://webserv1/css/reportstyles.css
```

Alternatively, if you want to embed your report output in an existing HTML page using - HTMLFORM, you would specify the link by coding the LINK element in the HTML page in which the report will be embedded, instead of setting CSSURL:

```
<HEAD>
<TITLE>Accounts Receivable Report</TITLE>
<LINK REL="STYLESHEET" HREF="http://srv3/css/reports.css"
TYPE="text/css">
</HEAD>
```

### **Example:** Formatting a Report Using an External CSS

This report displays the products currently offered by Gotham Grinds, and is formatted using an external cascading style sheet (CSS). The report is formatted so that:

- ☐ Its default font family is Arial.
- ☐ The report heading overrides the default with a font family of Times New Roman. The heading is also in a larger font and center justified.
- ☐ All column titles are in a bolder font and have a light-blue background.
- ☐ When a product unit price is less than \$27, the report displays the product row in green italics.

The report request and inline WebFOCUS StyleSheet are shown in the following procedure, curprods.fex. The external cascading style sheet, named report01.css, follows the procedure.

#### **curprods.fex**

```
TABLE FILE GGPRODS
HEADING
"</1 Current Products</1"
PRINT PRODUCT_DESCRIPTION UNIT_PRICE
BY PRODUCT_ID
ON TABLE SET PAGE-NUM OFF

1. ON TABLE SET STYLE *
2. TYPE=REPORT, CSSURL=http://websrv2/css/report01.css, $
3. TYPE=HEADING, CLASS=headText, $
4. TYPE=TITLE, CLASS=reportTitles, $
5. TYPE=DATA, CLASS=lowCost, WHEN=N3 LT 27, $
6. ENDSTYLE
END
```

**report01.css**

```

7. BODY {font-family:Arial, sans-serif}
8. TABLE {border:0}
8. TD {border:0}
9. .reportTitles {font-weight:bolder; background:lightblue;}
10. .lowCost {color:green; font-style:italic;}
11. .headText {font-family:Times New Roman, serif; font-size:larger;
            text-align:center}

```

1. Begin the inline WebFOCUS StyleSheet.
2. Link to the external cascading style sheet, report01.css.
3. Format the report heading using the cascading style sheet rule for the headText class.
4. Format the report column titles using the CSS rule for the reportTitles class.
5. For each report row for which the product unit cost is less than \$27, format that row using the CSS rule for the lowCost class.
6. End the inline WebFOCUS StyleSheet.
7. This CSS rule for the BODY element specifies the font family Arial and, if Arial is unavailable, the generic font family sans serif.

Because this is a rule for BODY, it is applied to the entire report: all text in the report will default to Arial. You can override this for a particular report component by applying a rule for a generic class to that component, as is done in this procedure with the rule for the headText class (see line 11).

8. These CSS rules for the TABLE and TD elements remove the report default grid.
9. This CSS rule for the generic class reportTitles specifies a bolder relative font weight and a light blue background color.

The WebFOCUS StyleSheet applies this to the report column titles (see line 4).

10. This CSS rule for the generic class lowCost specifies the text color green and the font style italic.

The WebFOCUS StyleSheet applies this rule conditionally to report rows for which the product unit cost is less than \$27 (see line 5).

11. The CSS rule for the generic class headText specifies the font family Times New Roman and, if Times New Roman is unavailable, the generic font family serif. It also specifies a larger relative font size and center justification.

The WebFOCUS StyleSheet applies this rule to the report heading. It overrides the default font family specified in the rule for the BODY element (see line 7).

The procedure displays this report:

### Current Products

Product Code	Product	Unit Price
B141	Hazelnut	58.00
B142	French Roast	81.00
B144	Kona	76.00
<i>F101</i>	<i>Scone</i>	<i>13.00</i>
<i>F102</i>	<i>Biscotti</i>	<i>17.00</i>
F103	Croissant	28.00
<i>G100</i>	<i>Mug</i>	<i>26.00</i>
G104	Thermos	96.00
G110	Coffee Grinder	125.00
G121	Coffee Pot	140.00

## Working With an External Cascading Style Sheet

When you work with an external cascading style sheet (CSS) to specify report formatting, you need to know about:

- ☐ Choosing an existing or new external CSS. For more information, see [Choosing an External Cascading Style Sheet](#) on page 1221.
- ☐ Where an external CSS can reside. For more information, see [External Cascading Style Sheet Location](#) on page 1221.
- ☐ How you can apply multiple cascading style sheets to one report. For more information, see [Using Several External Cascading Style Sheets](#) on page 1221.
- ☐ Editing an external CSS. For more information, see [Editing an External Cascading Style Sheet](#) on page 1221.
- ☐ Using CSS rules and classes to specify report formatting. For more information, see [Choosing a Cascading Style Sheet Rule](#) on page 1222.
- ☐ Suggestions for naming cascading style sheet classes. For more information, see [Naming a Cascading Style Sheet Class](#) on page 1223.



- ❑ Combining other formatting methods, such as WebFOCUS StyleSheets or TABLE language instructions, with an external CSS. For more information, see [Combining an External CSS With Other Formatting Methods](#) on page 1226.

## Choosing an External Cascading Style Sheet

To format a report using an external cascading style sheet (CSS), you can choose to:

- ❑ **Apply an existing CSS with no changes.** The external cascading style sheet can be one that you use for other documents, and can contain all kinds of rules, not only rules that format reports. For example, the CSS could include rules to format other elements in the webpages used by your WebFOCUS applications, as well as rules for other kinds of webpages. This enables you to use one cascading style sheet to format all of your web documents.
- ❑ **Edit an existing CSS** to add or modify rules. For example, you might edit a cascading style sheet to add new generic classes to format report components.
- ❑ **Create a new CSS.** You can create a new cascading style sheet to format your reports. See the recommendations in [Naming a Cascading Style Sheet Class](#) on page 1223 about naming classes.

To create an external cascading style sheet, use a text editor or a third-party web development tool.

## External Cascading Style Sheet Location

An external cascading style sheet (CSS) can reside on any web server platform. However, if CSSURL (the StyleSheet attribute or the SET parameter) specifies a relative URL, the cascading style sheet must reside on the web server used by WebFOCUS.

## Using Several External Cascading Style Sheets

Although each report procedure can link to only one external cascading style sheet (CSS), you can use several cascading style sheets to format a report by linking to one CSS that then imports several others. For information about importing one CSS into another, see your third-party CSS documentation.

## Editing an External Cascading Style Sheet

You can edit an external cascading style sheet (CSS) using a text editor or a third-party web development tool.

If the formatting of a report is specified entirely using a cascading style sheet, and you edit that CSS, the next time someone displays the report it will reflect the changes to the CSS without the report having to be rerun.

However, if the report does not reflect the changes, it may be because the web browser is continuing to use the old version of the CSS that it had stored in cache. The person displaying the report may need to reload the CSS file from the web server by clicking the Refresh button of the browser in Microsoft Internet Explorer to ensure that the browser uses the most current version of the CSS to format the report.

Choosing a Cascading Style Sheet Rule

You can format different parts of a report using different types of rules.

To format:	Use a rule for:
The entire report	BODY or TD
Any report component	A generic class (that is, one declared without an element)

To choose between using a rule for BODY or for TD, note that a rule for:

- ☐ **BODY** will specify default formatting for the entire webpage in which the report appears, including the report itself. (Note that this relies upon CSS inheritance which, like all CSS behavior, is implemented by the web browser of each user and is browser-dependent.)  
  
Graphs differ from other types of reports: a rule for BODY will format the page in which the graph appears, and its heading and footing, but not the graph itself.
- ☐ **TD** will specify default formatting only for the report, and for any other table cells that you may have on the page. TD is the table data (that is, table cell) element. WebFOCUS generates most HTML report output as an HTML table, placing each report item in a separate cell. This enables a rule for TD to format the entire report.  
  
Graphs differ from other types of reports: to specify default formatting for a graph, use a rule for BODY, not for TD. See the previous note regarding formatting graphs using a rule for BODY.

When you use a rule for a class to format a report component, you must assign the class to the component in a WebFOCUS StyleSheet using the CLASS attribute, as described in [How to Use the CLASS Attribute to Apply CSS Formatting](#) on page 1225.

If you wish to apply several CSS properties to a single report component, we recommend that you declare them in a single class. This generates more efficient output than does declaring one property per class.

The owner of each cascading style sheet should consider making available a list of all the classes in that CSS that can be used to format reports, so that everyone who develops reports knows from which classes they may choose.

For an example of a rule for:

- ❑ A generic class to format a report component, see [A CSS Rule for the ColumnTitle Class](#) on page 1217.
- ❑ The TD element to format an entire report, see [A CSS Rule for the TD Element](#) on page 1217.

## Naming a Cascading Style Sheet Class

When you provide a name for a new class, note that class names are case-sensitive (although some web browsers may not enforce case sensitivity).

When you create a new class, we recommend naming it after the function, not the appearance, of the report component to which you will be applying it. This ensures that the name remains meaningful even if you later change the appearance of the report component. For example, if you want all report titles to be red, the class you declare to format titles might be named Title, but not Red.

## Applying External Cascading Style Sheet Formatting

You can apply external cascading style sheet (CSS) formatting to:

- ❑ **A report component** (for example, to make a column italic). Assign a cascading style sheet class to the report component using the WebFOCUS StyleSheet CLASS attribute. For information about the CLASS attribute, see [How to Use the CLASS Attribute to Apply CSS Formatting](#) on page 1225. For information about specifying different types of report components, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167..

When formatting a tabular or free-form report, you can format any report component by assigning a CSS class.

When formatting a graph report, you can format the graph heading and footing, and can specify the background color and background image of the page in which the graph appears, in a rule for the BODY element. (Note that when formatting a graph heading or footing, you cannot format individual lines, strings, and field values. If you wish to center a heading or footing, it is recommended that you do so using the CENTER option of the HEADING or FOOTING command, not in a style sheet.)

When working with the WebFOCUS StyleSheet CLASS attribute, you must edit the WebFOCUS StyleSheet using the WebFOCUS text editor.

- ❑ **An entire report** (for example, to make the entire report italic). You specify the formatting in the external CSS in a rule for the BODY or TD elements (for graphs, specify the formatting in a rule for the BODY element only. This will format the page in which the graph appears, and the graph heading and footing, but not the graph itself). You must also link the report to the CSS. You do not need a rule for a class of an element, and you do not need a WebFOCUS StyleSheet declaration. For an example, see [A CSS Rule for the TD Element](#) on page 1217.

We recommend that when you use an external cascading style sheet to format a report, you do not also use a WebFOCUS StyleSheet to specify the report formatting, unless you also generate an internal cascading style sheet. For more information, see [Combining an External CSS With Other Formatting Methods](#) on page 1226.

**Syntax:**      **How to Use the CLASS Attribute to Apply CSS Formatting**

To apply an external cascading style sheet (CSS) class to a report component, use the following syntax in a WebFOCUS StyleSheet declaration

```
TYPE = type, [subtype,] CLASS = classname, [when,] [link,] $
```

where:

*type*

Identifies the report component to which you are applying the class formatting. For tabular and free-form reports, it can be any component, as described in [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167. You cannot specify a component of a graph report: to format a graph's heading and footing, and the background color and background image of the page in which the graph appears, use a rule for the BODY element without a WebFOCUS StyleSheet declaration.

Each report component can be formatted by one class. If you specify several classes for a report component:

1. The classes that are in declarations with conditional formatting are evaluated first. For each cell in the report component, the first class whose condition is satisfied by the cell row is assigned to the cell.
2. If none of the conditions is satisfied, or if there are no conditional declarations, the class in the first unconditional declaration is assigned to the report component. All following declarations for that component are ignored.

*subtype*

Is an optional attribute and value needed to completely specify some kinds of report components. For example, COLUMN and a column identifier are needed to specify a particular report column.

*classname*

Is the name of the cascading style sheet class whose formatting you are applying to the report component. You can assign the same class to multiple report components.

Class names can be up to 511 characters and are case-sensitive: you must use the same case found in the class rule in the external cascading style sheet. (Note, however, that some web browsers may not enforce case sensitivity.)

*when*

Is an optional WHEN attribute and value. Supply this if you want to apply the formatting conditionally. For more information, see [Formatting Reports: An Overview](#) on page 1107.

### *link*

Is an optional FOCEXEC, URL, or JAVASCRIPT attribute and value. Supply this if you want to link the report component to another resource, such as a report to which the user can drill down. For more information, see [Linking a Report to Other Resources](#) on page 763.

For an example, see [Applying a CSS Class to ACROSS Values in a Report](#) on page 1217.

## Combining an External CSS With Other Formatting Methods

When you use an external cascading style sheet (CSS) to format a report, you can use other formatting methods at the same time. Some of these other methods are subject to restrictions. The other methods that you can use with an external CSS are:

- ❑ **WebFOCUS StyleSheets.** An effective way of combining an external CSS with a WebFOCUS StyleSheet is to link to an external CSS to provide default formatting, and use a WebFOCUS StyleSheet to override those defaults for individual report components. Note that if you combine a WebFOCUS StyleSheet and an external cascading style sheet, you should generate an internal cascading style sheet to avoid reducing the performance benefits associated with an external CSS.

Do not attempt to format the same property of the same report component using both an external CSS class (through the CLASS attribute) and a WebFOCUS StyleSheet attribute, since the two formatting instructions could conflict with each other.

For complete instructions about using an external CSS with a WebFOCUS StyleSheet, see [Combining an External CSS With a WebFOCUS StyleSheet](#) on page 1227.

- ❑ **TABLE language instructions.** You can use TABLE language (or GRAPH language) formatting instructions, such as HEADING CENTER, PAGE-BREAK, and spot markers (for example, </3). However, you should not apply both a TABLE (or GRAPH) language instruction, and an external cascading style sheet rule, to perform the same formatting on the same report component, because they might conflict with each other.

For example, you should not specify *both* of the following for the same report:

- ❑ HEADING CENTER in the report request.
- ❑ Text-align in an external CSS, applied to the report page heading.

Both of these will attempt to align the report page heading.

## Combining an External CSS With a WebFOCUS StyleSheet

When you use an external cascading style sheet (CSS) to format a report, you can use a WebFOCUS StyleSheet at the same time. You may do this with or without generating an internal cascading style sheet.

**An effective way of doing this** is to link to an external CSS to provide default formatting, and use a WebFOCUS StyleSheet to override those defaults for individual report components. The cascading style sheet BODY or TD rule will provide the default formatting for the report. If you wish, you can override the defaults for individual report components via native WebFOCUS StyleSheet attributes. This enables you to conform to your organization's formatting standards as they are implemented in a CSS, while allowing you to customize those standards for WebFOCUS reports using WebFOCUS StyleSheet attributes. For information about using a BODY or TD rule for default formatting, see [Choosing a Cascading Style Sheet Rule](#) on page 1222. For an example, see [Inheritance and External Cascading Style Sheets](#) on page 1231.

**Performance considerations.** Note that, unless you generate an internal cascading style sheet from the WebFOCUS StyleSheet, combining an external CSS and a WebFOCUS StyleSheet may reduce the performance benefits associated with the external CSS. This is because a report that uses *both* an external CSS *and* native WebFOCUS StyleSheet attributes generates more HTML code than the same report using an external CSS alone, although it still generates less code than if the report had used native WebFOCUS StyleSheet attributes alone. (Reducing the amount of generated HTML code can reduce network load and browser display time.) For information about generating an internal cascading style sheet, see [Generating an Internal Cascading Style Sheet for HTML Reports](#) on page 1140.

**You cannot double-format.** You should not attempt to format the same property of the same report component using both an external CSS class (via the CLASS attribute) and a WebFOCUS StyleSheet attribute, since the class and the StyleSheet attribute could conflict with each other.

For example, you should not include the following declarations in the same StyleSheet because they would both try to assign a color to the Country column:

```
TYPE=Data, COLUMN=Country, COLOR=Orange, $
TYPE=Data, CLASS=TextColor, $
```

You can specify classes and WebFOCUS StyleSheet attributes that format different properties of the same report component, and that format different report components. For example, the following declarations are acceptable in the same StyleSheet:

1. `TYPE=Heading, COLOR=Green, $`
1. `TYPE=Heading, CLASS=HeadingFontSize, $`
2. `TYPE=Data, Column=Country, BACKCOLOR=Yellow, $`
2. `TYPE=Data, Column=Car, CLASS=DataBackgroundColor, $`
3. `TYPE=Data, Column=Model, FOCEXEC=NewSales(CarGroup=Car), $`

1. These two declarations are compatible because they format different properties (color and font size).
2. These two declarations are compatible because they format different report components (the Country column and the Car column).
3. This declaration will be compatible with all CSS classes, since it does not format a report component, but instead defines a hyperlink.

## Linking to an External Cascading Style Sheet

To format a report using an external cascading style sheet (CSS), you must link the cascading style sheet to the report in one of the following ways:

- ☐ **For most report procedures**, assign the CSS file's URL to the CSSURL attribute or parameter. For more information, see [Using the CSSURL Attribute and Parameter](#) on page 1228.
- ☐ **For report output that you are embedding** in an existing HTML page using the -HTMLFORM command, include a LINK element in the existing HTML page to point to the external CSS file. For example, if you are embedding a report from an output (HOLD) file that was generated in HTMTABLE format, use LINK. For more information, see your third-party CSS documentation. For an example, see [Linking to the ReportStyles External Cascading Style Sheet](#) on page 1217.

## Using the CSSURL Attribute and Parameter

You can link an external cascading style sheet (CSS) to a report using the CSSURL WebFOCUS StyleSheet attribute or the CSSURL SET parameter. To choose between them, consider the advantages of:

- ☐ **An attribute.** Using CSSURL as a StyleSheet attribute enables you to specify:
  - ☐ **A longer URL**, since the maximum URL length is 255 characters in the attribute, compared with 69 characters in the parameter.



- ❑ **All formatting information in one place**, since you can specify the link to the external CSS *and* the references to CSS classes within the WebFOCUS StyleSheet. This makes it easier for you to maintain your formatting logic.
- ❑ **A SET parameter.** Using CSSURL as a SET parameter enables you to quickly redirect a link (from one CSS to another) for many reports at once. You do this by putting the SET CSSURL command in its own procedure, and merging that into report procedures using a - INCLUDE Dialogue Manager statement in each report procedure.

If you specify CSSURL in several ways, the specification with the most local scope takes precedence. The order of precedence, from highest (1) to lowest (3), is:

1. `TYPE=REPORT, CSSURL = url`
2. `ON TABLE SET CSSURL url`
3. `SET CSSURL = url`

For more information about the CSS attribute, see [How to Use the CSSURL Attribute to Link to an External CSS](#) on page 1229. For more information about the CSS parameter, see [How to Use the CSSURL Parameter to Link to an External CSS](#) on page 1231.

### **Syntax:**

#### **How to Use the CSSURL Attribute to Link to an External CSS**

To link an external cascading style sheet (CSS) to report using a WebFOCUS StyleSheet attribute, use the following syntax

```
[TYPE=REPORT,] CSSURL=css_url, $
```

where:

#### **TYPE=REPORT**

Specifies that this attribute is being applied to the entire report. If it is omitted, the StyleSheet declaration defaults to it.

#### ***css\_url***

Is the URL of the external cascading style sheet. If the external CSS resides on a web server platform that is case-sensitive, you must specify it using the correct case.

The URL can be up to 255 characters. If your external cascading style sheet URL exceeds this limit, you can shorten the URL by defining an alias (also known as a virtual directory) on the web server to represent part of the path.

You can specify an absolute or relative URL. If it is relative, the external CSS must reside on the web server used by WebFOCUS.

### Example: Linking to an External Cascading Style Sheet Using the CSSURL Attribute

This report displays the products currently offered by Gotham Grinds. It is formatted using an external cascading style sheet (CSS), and links to the CSS using the CSSURL attribute in the WebFOCUS StyleSheet:

```
TABLE FILE GGPRODS
HEADING
"</1 Current Products</1"
PRINT PRODUCT_DESCRIPTION UNIT_PRICE
BY PRODUCT_ID
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE=REPORT, CSSURL=http://websrv2/css/report01.css, $
TYPE=HEADING, CLASS=headText, $
TYPE=TITLE, CLASS=reportTitles, $
TYPE=DATA, CLASS=lowCost, WHEN=N3 LT 27, $
ENDSTYLE

END
```

The request produces this report:

#### Current Products

Product Code	Product	Unit Price
B141	Hazelnut	58.00
B142	French Roast	81.00
B144	Kona	76.00
<i>F101</i>	<i>Scone</i>	<i>13.00</i>
<i>F102</i>	<i>Biscotti</i>	<i>17.00</i>
F103	Croissant	28.00
<i>G100</i>	<i>Mug</i>	<i>26.00</i>
G104	Thermos	96.00
G110	Coffee Grinder	125.00
G121	Coffee Pot	140.00

**Syntax:**      **How to Use the CSSURL Parameter to Link to an External CSS**

To link an external cascading style sheet (CSS) to a report using a SET parameter, issue the following SET command in a procedure

```
SET CSSURL = css_url
```

or the following ON TABLE SET command in a report request

```
ON TABLE SET CSSURL css_url
```

where:

*css\_url*

Is the URL of the external cascading style sheet. If the external CSS resides on a web server platform that is case-sensitive, you must specify it using the correct case.

The URL can be up to:

- ❑ 69 characters long in a SET command.
- ❑ 57 characters long in an ON TABLE SET command.

If your external cascading style sheet URL exceeds this limit, you can shorten the URL by defining an alias (also known as a virtual directory) on the web server to represent part of the path.

You can specify an absolute or relative URL. If it is relative, the external CSS must reside on the web server used by WebFOCUS.

For an example, see [Linking to the ReportStyles External Cascading Style Sheet](#) on page 1217.

If you specify CSSURL multiple times, the last value specified using ON TABLE SET overrides all the other values within that report request. If CSSURL is not specified within a report request, the last value specified using SET overrides all the others.

For general information about using SET commands, see *Customizing Your Environment* in the *Developing Reporting Applications* manual.

## Inheritance and External Cascading Style Sheets

In a report that is formatted using an external cascading style sheet (CSS), a report component inherits formatting from the TD element and from all elements that TD nests within, such as BODY. (Note that inheritance, like all CSS behavior, is implemented by the web browser of each user and is browser-dependent.)

This differs from a report that is formatted using a WebFOCUS StyleSheet, in which a report component inherits formatting from a higher-level component. When you format a report using external cascading style sheet classes, a class assigned to a report component *does not* inherit formatting from a class that has been assigned to a higher-level component.

### **Example:** A Report Column Inheriting Formatting From the TD Element

This report displays a list of the vendors that supply products to Gotham Grinds. Its formatting instructions specify that:

- ❑ The entire report has an orange default background color. This is specified in a rule for the TD element.
- ❑ The report data is displayed in an italic Arial font. The report data inherits the orange background color from the rule for TD.
- ❑ The report PRODUCT\_ID data has a yellow background color, overriding the default specified in the rule for TD.

If the formatting of the report had been specified in a WebFOCUS StyleSheet instead of in an external CSS, PRODUCT\_ID would inherit the italic Arial font from its parent report component (that is, from the report data). Instead, because its formatting is specified in an external CSS, PRODUCT\_ID inherits formatting from the rule for the TD element, not from a higher-level report component, and so it does not inherit the italic Arial font.

The report request and inline WebFOCUS StyleSheet are shown in the following procedure, prodvend.fex. The external cascading style sheet, named report02.css, follows the procedure.

#### **prodvend.fex**

```
TABLE FILE GGPRODS
PRINT PRODUCT_DESCRIPTION VENDOR_NAME
BY PRODUCT_ID
ON TABLE SET PAGE-NUM OFF

ON TABLE SET STYLE *
1. TYPE=REPORT, CSSURL = http://websrv2/css/report02.css, $
2. TYPE=DATA, CLASS=Data, $
3. TYPE=DATA, COLUMN=PRODUCT_ID, CLASS=Sort, $
   ENDSTYLE

END
```

**report02.css**

```

4. TD      {background:orange; border:0}
5. TABLE  {border:0}
6. .Data   {font-style:italic; font-family:Arial}
7. .Sort    {background:yellow}

```

1. Set CSSURL to link to the external cascading style sheet report01.css.
2. Format the report data using the CSS rule for the Data class.
3. Format the report PRODUCT\_ID data using the CSS rule for the Sort class. (This overrides the declaration for report data in general in line 2.)
4. This CSS rule for the TD element specifies an orange background. Because it is a rule for TD, it is applied to the entire report. You can override this for a particular report component by applying a rule for a generic class to that component, as is done in this procedure with the rule for the Sort class (see line 7).
5. These CSS rules for the TD and TABLE elements remove the default grid for the report.
6. This CSS rule for the generic class Data specifies an Arial font family and an italic font style. The WebFOCUS StyleSheet applies this to the report data (see line 2).  
  
This rule inherits background color from the rule for the TD element (line 4).
7. This CSS rule for the generic class Sort specifies a yellow background. The WebFOCUS StyleSheet applies this rule to data for PRODUCT\_ID (see line 3).  
  
This rule overrides the default background color specified in line 4.

The procedure displays this report:

Product Code	Product	Vendor Name
B141	Hazelnut	Coffee Connection
B142	French Roast	European Specialities,
B144	Kona	Evelina Imports, Ltd
F101	Scone	Ridgewood Bakeries
F102	Biscotti	Delancey Bakeries
F103	Croissant	West Side Bakers
G100	Mug	NY Ceramic Supply
G104	Thermos	ThermoTech, Inc
G110	Coffee Grinder	Appliance Craft
G121	Coffee Pot	Appliance Craft

## Using External Cascading Style Sheets With Non-HTML Reports

You can use an external cascading style sheet (CSS) to format a report that is generated as HTML, but not one that is generated as a different output type, such as PDF. If you have a report that you will sometimes generate as HTML and sometimes generate as a different output type, and you wish to gain the benefits of cascading style sheets, we recommend that you use this technique:

- ☐ **Shared formatting.** Specify formatting that is shared by all output types in WebFOCUS StyleSheet macros. (For example, setting a font style to italic is something that can be applied to both HTML and PDF report output, so you would specify it in a macro.) Define two versions of each macro:
  - ☐ One version for HTML output. This version specifies formatting using a cascading style sheet class.
  - ☐ One version for non-HTML output. This version specifies formatting using native WebFOCUS StyleSheet attributes.
- ☐ **Unique formatting.** Specify formatting that is applicable only to HTML output, or only to non-HTML output, in standard WebFOCUS StyleSheet declarations. Place each of these declarations in the WebFOCUS StyleSheet section that contains macro definitions for that type of output. (For example, turning a grid on or off is applicable to HTML output, but not to Excel 2000, so you would place it with the macro definitions for HTML.)

❑ **Branch** between the HTML and non-HTML declarations using Dialogue Manager.

You can see the basic code for this technique in [How to Use an External CSS With Multiple Output Types](#) on page 1236.

**Syntax: How to Use an External CSS With Multiple Output Types**

If you have a report that you will sometimes generate as HTML and sometimes as other types of output, and you wish to gain the benefits of cascading style sheets (CSS), we recommend that you use this technique:

```

1.  -DEFAULTS &FORMAT='output_type';
2.  SET ONLINE-FMT = &FORMAT
    TABLE FILE datasource
        report_logic

    ON TABLE SET STYLE *
3.  TYPE=REPORT, CSSURL = CascadingStyleSheetURL, $
4.  -IF &FORMAT NE 'HTML' GOTO NONHTML;
5.  DEFMACRO=macro1, CLASS=class1, $
    DEFMACRO=macro2, CLASS=class2, $
    .
    .
    .
6.  TYPE=component3, CLASS=class3, $
    .
    .
    .
7.  -GOTO SHARED
8.  -NONHTML
9.  DEFMACRO=macro1, attribute1=value1, $
    DEFMACRO=macro2, attribute2=value2, $
    .
    .
    .
10. TYPE=component4, attribute4=value4, $
    .
    .
    .
11. -SHARED
12. TYPE=component1, MACRO=macro1, $
    TYPE=component2, MACRO=macro2, $
    .
    .
    .
    ENDSTYLE
    END

```

1. Assign the type of report output (for example, HTML, PDF, PS, or EXL2K) to the Dialogue Manager variable &FORMAT. You will use this variable to toggle the WebFOCUS StyleSheet between formatting for HTML output and formatting for non-HTML output, and also to provide a value for SET ONLINE-FMT.

You can use forms and other presentation logic to enable the application user to select the type of report output.

2. Set the report output type to the value of &FORMAT. In this procedure, SET ONLINE-FMT sets the display type for the report. Alternatively, you could use ON TABLE HOLD to save



the report as a file and set its file type.

3. Set CSSURL to link to the external cascading style sheet to be used for formatting the report HTML output. When the report generates non-HTML output, this command will be ignored.
4. Branch to the WebFOCUS StyleSheet declarations for the current type of report output (which is indicated by &FORMAT).
5. Define the HTML version of the WebFOCUS StyleSheet macros. These macros specify formatting that is shared by all output types.

This HTML version of the macros is implemented using external cascading style sheet classes.

6. If there is any formatting that is applicable only to HTML output, specify it here, using external cascading style sheet classes.
7. Branch to the WebFOCUS StyleSheet declarations that apply the macros to the report components.
8. This label marks the beginning of the macro definitions and unique formatting declarations for non-HTML report output.
9. Define the non-HTML version of the WebFOCUS StyleSheet macros. These macros specify formatting that is shared by all output types.

This non-HTML version of the macros is implemented using native WebFOCUS StyleSheet attributes.

10. If there is any formatting that is applicable only to non-HTML output, specify it here using native WebFOCUS StyleSheet attributes.
11. This label marks the beginning of the declarations that apply macros to the report.
12. These are the macros that were defined earlier and are being applied to the report.

**Example:**    **Using an External CSS With PDF and HTML Output**

This report procedure (videort.fex) can generate both HTML and PDF output. When it generates HTML output, it uses an external cascading style sheet (reports.css) to format the report. When it generates PDF output, it uses an inline WebFOCUS StyleSheet. In both cases, the report provides a light blue background for the LASTNAME column and makes all column titles bold.

The procedure as shown is set to generate HTML output.

**videort.fex**

```
1.  -DEFAULTS &FORMAT='HTML';
2.  SET CSSURL = http://websrv2/css/reports.css
3.  SET ONLINE-FMT = &FORMAT
    TABLE FILE VIDEOTRK
    PRINT LASTNAME AS 'Last Name' FIRSTNAME AS 'First Name'
    BY LOWEST 5 CUSTID AS 'Cust ID'
    ON TABLE SET PAGE-NUM OFF
    ON TABLE SET STYLE *
4.  -IF &FORMAT NE 'HTML' GOTO NONHTML;
5.  DEFMACRO=boldTitles, CLASS=bold, $
    DEFMACRO=blueColumn, CLASS=blueBack, $
6.  -GOTO SHARED
7.  -NONHTML
8.  DEFMACRO=boldTitles, STYLE=bold, $
    DEFMACRO=blueColumn, BACKCOLOR=light blue, $
9.  -SHARED
10. TYPE=DATA, COLUMN=LastName, MACRO=blueColumn, $
    TYPE=TITLE, MACRO=boldTitles, $
    ENDSTYLE
    END
```

**reports.css**

```
11. .bold {font-weight: bolder}
12. .blueBack {background: lightblue}
13. TABLE {border:0}
    TD     {border:0}
```

1. Assign a default value to &FORMAT to toggle the WebFOCUS StyleSheet between formatting for HTML output and formatting for PDF output. It is currently set to HTML output.
2. Set CSSURL to link to the external cascading style sheet reports.css to format the HTML output of the report.
3. Set the display type of the report to the value of &FORMAT.
4. Branch to the WebFOCUS StyleSheet declarations for the current type of report output (HTML).
5. Define the HTML version of the WebFOCUS StyleSheet macros, which are implemented using external cascading style sheet classes.

6. Branch to the WebFOCUS StyleSheet declarations that apply the macros to the components for the report.
7. This label marks the beginning of the macro definitions for PDF report output.
8. These declarations define the PDF version of the WebFOCUS StyleSheet macros, which are implemented using native WebFOCUS StyleSheet attributes. These macro definitions will be ignored because &FORMAT is set to HTML.
9. This label marks the beginning of the declarations that apply macros to the report.
10. These are the macros that were defined earlier and are being applied to the report.
11. This cascading style sheet declaration makes text bolder than it had been.
12. This cascading style sheet declaration makes a background light blue.
13. These CSS rules for the TABLE and TD elements remove the default grid for the report.

The procedure displays this report:

<b>Cust ID</b>	<b>Last Name</b>	<b>First Name</b>
0925	CRUZ	IVY
0944	HANDLER	EVAN
1118	WILSON	KELLY
1133	KRAMER	CHERYL
1237	GOODMAN	JOHN

## Requirements for Using an External Cascading Style Sheet

When you use an external cascading style sheet (CSS) to format a report, be aware of the following requirements:

- ☐ **Generate HTML report output.** You can use an external cascading style sheet to format any report that you generate as HTML, whether you save the report output in a file or send it directly to a web browser. You cannot use an external CSS for a report generated in a different format, such as PDF or Excel.

If you wish to use an external CSS with a report that you will sometimes generate as HTML and sometimes as a different format, such as PDF, see [Using External Cascading Style Sheets With Non-HTML Reports](#) on page 1234.

- ☐ If you are not generating an internal cascading style sheet, do not specify external CSS classes (CLASS=) and native WebFOCUS StyleSheet attributes in the same WebFOCUS StyleSheet (other than the exceptions noted in the next paragraph). Doing so could create formatting conflicts.

**Exceptions.** Even when specifying external CSS classes, you should use native WebFOCUS StyleSheet attributes to:

- ❑ Create hyperlinks (using the FOCEXEC, JAVASCRIPT, and URL attributes). However, if you wish to format a hyperlink, you should do so using the cascading style sheet.
- ❑ Make a WebFOCUS StyleSheet declaration conditional (using the WHEN attribute).
- ❑ Embed an image (using the IMAGE attribute). However, if you wish to format the image (for example, to position it), you should do so using the cascading style sheet.

For more information, see [Combining an External CSS With Other Formatting Methods](#) on page 1226.

- ❑ **Do not specify the same formatting using TABLE/GRAPH and CSS.** You can use TABLE language (or GRAPH language) formatting instructions, such as HEADING CENTER, PAGE-BREAK, and spot markers (for example, </3). However, you should not apply both a TABLE (or GRAPH) language instruction, and an external cascading style sheet rule, to perform the same formatting on the same report component. For more information, see [Combining an External CSS With Other Formatting Methods](#) on page 1226.
- ❑ **SET STYLEMODE.** If you wish to use cascading style sheets to format a report in the usual way, you can set STYLEMODE to FULL (the default) or PAGED. If you set it to FIXED and link to an external cascading style sheet, the report will inherit formatting from the BODY and PRE elements, but you will not be able to format the report using classes and the TD element.
- ❑ **Use a cascading style sheet-enabled web browser.** Each user who wishes to display a report formatted using a cascading style sheet must have a web browser that supports CSS. All versions of Microsoft Internet Explorer that are certified for use with WebFOCUS support cascading style sheets.

Note that how a cascading style sheet rule formats your report is determined entirely by the support of your web browser and implementation of cascading style sheets, not by WebFOCUS. Some web browsers may not fully support the latest CSS version, or may implement a CSS feature in different ways.

- ❑ **Do not override the cascading style sheet specified for the report.** If a browser has been customized to ignore cascading style sheets or to employ the personal cascading style sheet of the user, and the user wishes to view reports as they were intended to be seen (with the specified cascading style sheet), the user must reset his or her browser to accept the cascading style sheet of each document.

For instructions about checking or changing a browser setting, see the browser Help. For information about how conflicts between CSS rules are resolved (for example, between a rule specified in a CSS document and a rule specified in the reader web browser CSS), see your third-party CSS documentation.

**Reference: Usage Notes for External Cascading Style Sheets With SET HTMLCSS ON**

- ☐ Styling only to the CLASS, referenced from the external cascading style sheet, is honored when internal and external styling is applied to multiple subtypes in an element. The internal styling applied to the element is ignored. With HTMLCSS OFF, only the internal styling applied to each subtype is honored.
- ☐ In a report with no borders specified in the external cascading style sheet, borders in the internal styling are respected and only grids are shown when HTMLCSS is OFF.
- ☐ Border style specified in the external cascading style sheet overrides that specified in the internal style sheet. With HTMLCSS OFF, both grids and the border style specified in the external cascading style sheet are displayed.
- ☐ Border weights are consistent in all elements of the report. With HTMLCSS OFF, different border weights are seen in different elements of the report.
- ☐ When the heading of the report contains multiple lines, the border outlines the entire heading. With HTMLCSS OFF, the border outlines each line in the heading.

## FAQ About Using External Cascading Style Sheets

This topic answers the most frequently asked questions (FAQ) about using external cascading style sheets (CSS) to format reports.

**Does it answer your question?** We invite you to send us *any* questions that you would like answered. Each question will get a response, and will also be considered for inclusion in a future release of FAQ. (We also invite your comments on anything in this document.)

### **How do I specify a report default formatting using CSS?**

You can specify default formatting for an entire report in an external cascading style sheet rule for the BODY or TD element. For more information, see [Choosing an External Cascading Style Sheet](#) on page 1221.

**Do I always need to use the CLASS attribute?**

No. You need the CLASS attribute in a WebFOCUS StyleSheet if you specify formatting for an individual report component. (You use CLASS to assign a rule for a generic class to the report component.) When you specify formatting for the entire report, you do so in a rule for the BODY or TD element, not a rule for a class, so you omit the CLASS attribute.

If you place a reference to a CSS class in your stylesheet, it will be applied to the <A> tag as well as the <TD> tag. For example, if you have the class ".class1" in your external CSS, WebFOCUS would generate the following HTML for a value with a drilldown:

```
<TD CLASS='class1'>
<A class='class1' HREF="...">ENGLAND</A>
</TD>
```

For example, if you want red hyperlinks without underlines, issue:

```
SET CSSURL=http://myserver/mycss.css
TABLE FILE CAR
SUM SALES BY COUNTRY
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET STYLE *
type=data, column=country, focexec=fex1, class=class1, $
END
```

where mycss.css contains:

```
.class1 { color:red; text-decoration:none }
```

The output is:



For more information, see [Applying External Cascading Style Sheet Formatting](#) on page 1224.

### **Can I use a cascading style sheet and a WebFOCUS StyleSheet together?**

When you link to an external cascading style sheet, you can also specify native WebFOCUS StyleSheet attributes in a WebFOCUS StyleSheet. However, if you do not generate an internal cascading style sheet, you should not specify CSS classes (CLASS=) and native WebFOCUS StyleSheet attributes in the same WebFOCUS StyleSheet (except to specify a condition for conditional formatting, to specify a link to another resource, and to embed an image). For more information, see [Combining an External CSS With Other Formatting Methods](#) on page 1226. For information about internal cascading style sheets, see [Generating an Internal Cascading Style Sheet for HTML Reports](#) on page 1140.

**Which version of CSS does WebFOCUS support?**

Support for different versions of cascading style sheets (such as CSS2) is determined entirely by your web browser support and implementation of cascading style sheets, not by WebFOCUS. Note that some web browsers may not fully support the latest CSS version, or may implement a CSS feature in different ways. For more information, see [Requirements for Using an External Cascading Style Sheet](#) on page 1239, and [Troubleshooting External Cascading Style Sheets](#) on page 1245.

**Can I use CSS to format reports generated as PDF, PostScript, or Excel 2000?**

No, you can only use external cascading style sheets to format reports that are generated as HTML.

**Which types of reports can I format using an external cascading style sheet?**

You can format all types of reports using an external CSS:

- ☐ **Tabular reports**, including regular (column-oriented) reports and Financial Modeling Language (FML, also known as extended matrix or row-oriented) reports.
- ☐ **Graphs**. Note that, while you can format a graph heading and footing, and the background color and background image of the page in which the graph appears, the graph itself is generated using Java and so cannot be formatted using CSS.
- ☐ **Free-form reports**. Most people choose to generate free-form reports using output types other than HTML, making CSS a rarely-used option for formatting free form.



## Troubleshooting External Cascading Style Sheets

This topic will help you solve some common problems encountered when formatting reports with external cascading style sheets (CSS).

**Which problems have you needed to troubleshoot?** If you have troubleshooting suggestions that you think others will find helpful, we invite you to send them to us so that we can consider including them in a future release.

**Symptom: The report does not reflect recent changes to the cascading style sheet.**

- ☐ **Reason:** When you run a report that references an external cascading style sheet, your web browser stores the CSS file in its memory or disk cache. When you later edit the CSS and run the report again, your browser may continue to use the earlier version of the CSS file that it had stored.

**Solution:** Click your browser Refresh button (Microsoft Internet Explorer) to reload the CSS file from the web server. This ensures that your web browser will use the most current version of the cascading style sheet to format the report.

**Symptom: The report is not using any of the cascading style sheet formatting.**

- ☐ **Reason 1:** You may have specified an incorrect URL when you attempted to link to the external cascading style sheet.

**Solution 1:** Check the URL that specifies the link (in the CSSURL attribute or in the SET CSSURL command, or if the report procedure uses -HTMLFORM, in the LINK element) and correct it, if necessary.

- ☐ **Reason 2:** Your web browser may not support cascading style sheets.

**Solution 2:** All versions of Microsoft Internet Explorer that are certified for use with WebFOCUS support cascading style sheets. Check to be sure that your browser is certified. If it is not, install an appropriate version of Internet Explorer or Communicator.

- ☐ **Reason 3:** Your web browser may be set to ignore cascading style sheets.

**Solution 3:** Reset your browser to accept a document cascading style sheet. For instructions about checking or changing a browser setting, see the browser Help.

- ☐ **Reason 4:** Some web browsers, if they do not support a single property specified in a rule, ignore the entire rule. If this is true of your web browser, and all of your report formatting is specified in a single rule (for example, a rule for TD or BODY), but the browser does not support one of the properties specified for the rule, none of the formatting will be applied to the report.

**Solution 4:** Remove the unsupported property, or upgrade your browser to a version that supports the property.

- ❑ **Reason 5:** Some web browsers implement CSS inheritance rules for nested elements in ways that do not conform to the CSS standard. If you are using such a browser, and for example, you specify all formatting in a rule for the BODY element, your browser may not apply the rule to other elements nested within BODY.

**Solution 5:** Specify the report formatting in a rule for a different element (for example, if the browser does not correctly implement inheritance from BODY, use a rule for TD), or else upgrade your browser to a version that correctly supports inheritance.

- ❑ **Reason 6:** The CSS file was not found on the server path when the report that references the .css file was run.

**Solution 6:** Make sure the directory with the CSS file is on the server search path.

**Symptom: The report reflects some, but not all, of the CSS formatting.**

- ❑ **Reason 1:** How a cascading style sheet rule formats your report is determined entirely by your web browser support and implementation of cascading style sheets, not by WebFOCUS. You may be experiencing this symptom because your browser does not support the level of cascading style sheets that you are using, leaving some CSS features unimplemented.

**Solution 1:** Upgrade your browser to a version that supports all the CSS features used to format the report, or edit the cascading style sheet to remove features that are unsupported by some of the browsers that will be used to display the report.

- ❑ **Reason 2:** Your web browser may be set to use your personal cascading style sheet, and some of the rules you had specified there may override rules specified in the cascading style sheet assigned to the report. For information about how conflicts between rules in different cascading style sheets are resolved, see your third-party CSS documentation.

**Solution 2:** Reset your browser to accept the cascading style sheet for each document, or edit the rules in the two cascading style sheets so that they no longer conflict.

- ❑ **Reason 3:** Some web browsers, if they do not support a property specified in a rule, ignore the entire rule. If this is true of your web browser, and the browser does not support one of the properties specified in the rule for one of the classes assigned to the report, none of the report components to which that rule has been assigned will be formatted.

**Solution 3:** Remove the unsupported property, or upgrade your browser to a version that supports the property.

- ❑ **Reason 4:** Each report component can be assigned only one cascading style sheet class. If you have specified more than one class, only the first one specified is assigned to the component; the others are ignored.

If a class has not yet been assigned to a report cell, and you specify conditional formatting for it, only the first class whose condition is satisfied by that row is assigned to the cell. The others are ignored.

**Solution 4:** Do not assign more than one CSS class to each report component. If you need to apply multiple attributes, bundle them into a single class.

- ❑ **Reason 5:** Some web browsers implement CSS inheritance rules for nested elements in ways that do not conform to the CSS standard. If you are using such a browser, and for example, you specify some formatting in a rule for the BODY element, your browser may not apply the rule to other elements nested within BODY.

**Solution 5:** Specify the report formatting in a rule for a different element (for example, if the browser does not correctly implement inheritance from BODY, use a rule for TD), or else upgrade your browser to a version that correctly supports inheritance.

- ❑ **Reason 6:** External cascading style sheets can be subject to certain restrictions when used with other formatting methods. For example, if a WebFOCUS StyleSheet report does not generate an internal cascading style sheet, but it references external CSS classes and also specifies native WebFOCUS StyleSheet attributes, there may be a formatting conflict.

**Solution 6:** The solution depends on the kind of formatting conflict. In the example above, the solution is to generate an internal cascading style sheet. For a complete description of which formatting methods are compatible with an external CSS, and how to avoid formatting conflicts, see [Combining an External CSS With Other Formatting Methods](#) on page 1226.

**Symptom: A report distributed with ReportCaster does not have the CSS styling, but it does have CSS styling when run interactively.**

- ❑ **Reason:** The mail server to which ReportCaster is distributing reports may not support externally referenced CSS files. For example, Gmail strips the CSS from HTML email, you must use an inline CSS for Gmail. For information, see:

<http://www.ajaxapp.com/2009/02/19/gmail-strips-css-of-html-email-you-must-use-inline-css-for-gmail/>

**Solution:** Issue the WebFOCUS command SET HTMLCSS=ON in your procedure, or issue the command ON TABLE SET HTMLCSS ON in your request. This creates reports with an inline CSS.





# Chapter 21

## Laying Out the Report Page

---

You can customize page layout using StyleSheet attributes. The page layout attributes available to you, like all other attributes, depend on the display format. For instance, some attributes support print-oriented display formats such as PDF, and they do not apply to HTML reports displayed in a browser. For details on display formats, see [Choosing a Display Format](#) on page 565 and [Saving and Reusing Your Report Output](#) on page 471.

A report page has default layout characteristics. However, you can change any of them to customize the layout. You control a report's appearance, including column arrangement on a page, page numbering, page breaks, use of grids and images, and much more.

You can use SET parameters in place of most StyleSheet attributes to define page layout characteristics. For details on SET, see the *Developing Reporting Applications* manual.

### In this chapter:

- ☐ [Selecting Page Size, Orientation, and Color](#)
  - ☐ [Setting Page Margins](#)
  - ☐ [Positioning a Report Component](#)
  - ☐ [Arranging Columns on a Page](#)
  - ☐ [Suppressing Column Display](#)
  - ☐ [Inserting a Page Break](#)
  - ☐ [Inserting Page Numbers](#)
  - ☐ [Adding Grids and Borders](#)
  - ☐ [Defining Borders Around Boxes With PPTX and PDF Formats](#)
  - ☐ [Displaying Superscripts On Data, Heading, and Footing Lines](#)
  - ☐ [Adding Underlines and Skipped Lines](#)
  - ☐ [Removing Blank Lines From a Report](#)
  - ☐ [Adding an Image to a Report](#)
  - ☐ [Associating Bar Graphs With Report Data](#)
  - ☐ [Working With Mailing Labels and Multi-Pane Pages](#)
- 

### Selecting Page Size, Orientation, and Color

You can select the page size, page orientation (portrait or landscape), and page color for your report. The default page size is letter (8.5 x 11 inches), but you can select from many other sizes, including legal and envelopes.

Reference: Page Size, Orientation, and Color Attributes

Attribute	Description	Applies to
PAGESIZE	Sets page size.	In the Development environment: <input type="checkbox"/> PDF <input type="checkbox"/> PS <input type="checkbox"/> PPT <input type="checkbox"/> PPTX
ORIENTATION	Sets page orientation.	In the Development environment: <input type="checkbox"/> PDF <input type="checkbox"/> PS <input type="checkbox"/> EXL2K <input type="checkbox"/> PPT <input type="checkbox"/> PPTX
PAGECOLOR	Sets page color.	HTML report with internal cascading style sheet

Syntax: How to Set Page Size

This syntax applies to a PDF, PS, PPT, or PPTX report.

[TYPE=REPORT,] PAGESIZE={size|LETTER}, \$

where:

TYPE=REPORT

Applies the page size to the entire report. Not required, as it is the default value.

size

Is the page size. If printing a report, the value should match the size of the paper. Otherwise, the report may be cropped or printed with extra blank space.

Valid values are:

Value	Description
SCREEN	Sets the page size so that the report fills the screen. Each print line is infinitely wide. The report width determines the page width. The number of lines per page depends on how many lines fit on the screen. This number varies, depending on the font and screen resolution.
LETTER	8.5 x 11 inches. LETTER is the default value.
LEGAL	8.5 x 14 inches.
TABLOID	11 x 17 inches.
LEDGER	17 x 11 inches.
WIDESCREEN	13.333 x 7.5 inches.
C	17 x 22 inches.
D	22 x 34 inches.
E	34 x 44 inches.
STATEMENT	5.5 x 8.5 inches.
EXECUTIVE	7.5 x 10.5 inches.
FOLIO	8.5 x 13 inches.
10x14	10 x 14 inches.
A3	297 x 420 millimeters.
A4	210 x 297 millimeters.
A5	148 x 210 millimeters.

Value	Description
B4	250 x 354 millimeters.
B5	182 x 257 millimeters.
QUARTO	215 x 275 millimeters.
ENVELOPE-9	3.875 x 8.875 inches.
ENVELOPE-10	4.125 x 9.5 inches.
ENVELOPE-11	4.5 x 10.375 inches.
ENVELOPE-12	4.5 x 11 inches.
ENVELOPE-14	5 x 11.5 inches.
ENVELOPE-MONARCH	3.875 x 7.5 inches.
ENVELOPE-PERSONAL	3.625 x 6.5 inches.
ENVELOPE-DL	110 x 220 millimeters.
ENVELOPE-C3	324 x 458 millimeters.
ENVELOPE-C4	229 x 324 millimeters.
ENVELOPE-C5	162 x 229 millimeters.
ENVELOPE-C6	114 x 162 millimeters.
ENVELOPE-C65	114 x 229 millimeters.
ENVELOPE-B4	250 x 353 millimeters.
ENVELOPE-B5	176 x 250 millimeters.
ENVELOPE-B6	176 x 125 millimeters.



Value	Description
ENVELOPE-ITALY	110 x 230 millimeters.
US-STANDARD-FANFOLD	14.875 x 11 inches.
GERMAN-STANDARD-FANFOLD	8.5 x 12 inches.
GERMAN-LEGAL-FANFOLD	8.5 x 13 inches.

**Syntax:**      **How to Set Page Orientation**

This syntax applies to a PDF, PS, or EXL2K report.

[TYPE=REPORT, ] ORIENTATION={PORTRAIT|LANDSCAPE}, \$

where:

TYPE=REPORT

Applies the page orientation to the entire report. Not required, as it is the default.

PORTRAIT

Displays the report across the narrower dimension of a vertical page, producing a page that is longer than it is wide. PORTRAIT is the default value.

LANDSCAPE

Displays the report across the wider dimension of a horizontal page, producing a page that is wider than it is long.

**Example: Setting Page Orientation**

This request sets the page orientation of a PDF report to landscape.

```
SET ONLINE-FMT = PDF
TABLE FILE CENTQA
SUM CNT.PROBNUM AS 'Total Number, of Problems'
SUM CNT.PROBNUM AS 'Problems From, Each Plant' BY PLANT
SUM CNT.PROBNUM AS 'Problem by Product' BY PLANT BY PRODNAME
ON PLANT PAGE-BREAK
HEADING CENTER
"QA Report for Company, Plant, and Product"
" "
ON TABLE COLUMN-TOTAL
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, ORIENTATION=LANDSCAPE, $
ENDSTYLE
END
```

**Syntax: How to Set Page Color**

This syntax applies to an HTML report with internal cascading style sheet.

```
[TYPE=REPORT,] ... PAGECOLOR=color, ... , $
```

where:

```
TYPE=REPORT
```

The TYPE specification is optional with this feature. If omitted, TYPE defaults to REPORT.

*color*

Is a supported color. For a list of values, see [Formatting Report Data](#) on page 1611.

**Example: Setting Page Color**

This request sets the page color of an HTML report with internal cascading style sheet to silver.

```
SET HTMLCSS = ON
TABLE FILE CENTORD
ON TABLE SUBHEAD
"SELECTED PRODUCT INVENTORY"
SUM QTY_IN_STOCK/D12 BY PROD_NUM BY SNAME BY STATE
WHERE PROD_NUM EQ '1004'
WHERE SNAME EQ 'eMart'
WHERE STATE EQ 'CA'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, PAGECOLOR=SILVER, GRID-OFF, $
ENDSTYLE
END
```

The output is:

SELECTED PRODUCT INVENTORY			
<u>Product</u>	<u>Store</u>	<u>Quantity</u>	
<u>Number#:</u>	<u>Name:</u>	<u>State:</u>	<u>In Stock:</u>
1004	eMart	CA	689,088

Setting Page Margins

You can set the page margins for your report. This includes the top, bottom, left, and right margins. You can also change the default unit of measurement (inches) to either centimeters or points. The unit of measurement applies to page margins, column width, and column position.

*Reference:* **Page Margin Attributes**

Attribute	Description	Applies to
UNITS	Sets unit of measurement.  Used when specifying margin size or other page characteristics. If you change the current unit of measurement, the new value is applied to all instances in which unit of measurement is used.	PDF  PS  HTML report with internal cascading style sheet
TOPMARGIN BOTTOMMARGIN LEFTMARGIN RIGHTMARGIN	Sets size of top, bottom, left, and right margin.	PDF  PS  HTML report with internal cascading style sheet

*Syntax:* **How to Set the Unit of Measurement**

This syntax applies to a PDF, PS, or HTML report with internal cascading style sheet.

In a StyleSheet, add the following attribute

`UNITS = units`

Outside of a report request, use

`SET UNITS = units`

Within a report request, use

`ON TABLE SET UNITS units`

where:

*units*

Is the unit of measure. Values can be:

- ☐ **INCHES**, which specifies the unit of measure as inches. This is the default value.
- ☐ **CM**, which specifies the unit of measure as centimeters.
- ☐ **PTS**, which specifies the unit of measure as points. Points is a common measurement scale for typefaces.

### **Syntax:**      **How to Set Margin Size**

This syntax applies to a PDF, PS, or HTML report with internal cascading style sheet.

```
[TYPE=REPORT,] [TOPMARGIN={value|.25},] [BOTTOMMARGIN={value|.25},]  
[LEFTMARGIN={value|.25},] [RIGHTMARGIN={value|.25},] $
```

where:

`TYPE=REPORT`

Applies the margin size to the entire report. Not required, as it is the default.

`TOPMARGIN`

Sets the top boundary of the report content.

`BOTTOMMARGIN`

Sets the bottom boundary of the report content.

`LEFTMARGIN`

Sets the left boundary of the report content.

`RIGHTMARGIN`

Sets the right boundary of the report content.

*value*

Is the size of the specified margin. The report content displays inside the margin. If printing a report, specify a value compatible with the printer's print area. For example, if the print area has 0.25 inch margins all around, set the margins to 0.25 inches, or larger.

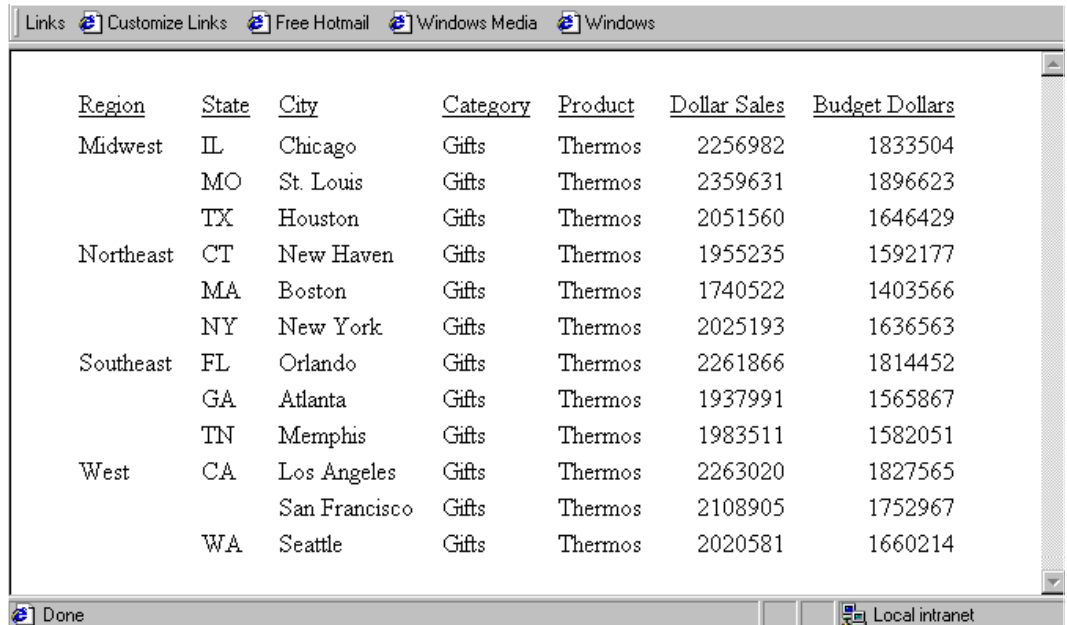
The default value for all margins is 0.25 inches.

### **Example:** Setting the Left Margin

This request sets the left margin of an HTML report with internal cascading style sheet to one inch.

```
SET HTMLCSS = ON
TABLE FILE GGSALES
SUM CATEGORY PRODUCT DOLLARS BUDDOLLARS
BY REGION BY ST BY CITY
WHERE DOLLARS GT BUDDOLLARS
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
LEFTMARGIN = 1, $
ENDSTYLE
END
```

The output is:



<u>Region</u>	<u>State</u>	<u>City</u>	<u>Category</u>	<u>Product</u>	<u>Dollar Sales</u>	<u>Budget Dollars</u>
Midwest	IL	Chicago	Gifts	Thermos	2256982	1833504
	MO	St. Louis	Gifts	Thermos	2359631	1896623
	TX	Houston	Gifts	Thermos	2051560	1646429
Northeast	CT	New Haven	Gifts	Thermos	1955235	1592177
	MA	Boston	Gifts	Thermos	1740522	1403566
	NY	New York	Gifts	Thermos	2025193	1636563
Southeast	FL	Orlando	Gifts	Thermos	2261866	1814452
	GA	Atlanta	Gifts	Thermos	1937991	1565867
	TN	Memphis	Gifts	Thermos	1983511	1582051
West	CA	Los Angeles	Gifts	Thermos	2263020	1827565
		San Francisco	Gifts	Thermos	2108905	1752967
	WA	Seattle	Gifts	Thermos	2020581	1660214

## Positioning a Report Component

A StyleSheet enables you to specify an absolute or relative starting position for a column, heading, or footing, or element in a heading or footing. You can also add blank space around a report component.

This topic addresses column positioning using the StyleSheet attribute POSITION. For details on the column positioning command IN, see [Positioning a Column](#) on page 1283.

For details on positioning a heading or footing, or an element in a heading or footing, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.

**Reference:**    **Positioning Attributes**

Attribute	Description	Applies to
POSITION	Sets absolute or relative starting position of a column.  An absolute position is the distance from the left margin of the printed paper.  A relative position is the distance from the default position. After the first column, the default position is the end of the preceding column.	PDF  PS
TOPGAP BOTTOMGAP	Adds blank space to the top or bottom of a report line.	PDF  PS
LEFTGAP RIGHTGAP	Adds blank space to the left or right of a report column.	PDF  PS

**Syntax:**    **How to Specify the Starting Position of a Column**

This syntax applies to a PDF or PS report.

```
TYPE=REPORT, COLUMN=identifier, POSITION={+|-}position, $
```

where:

*identifier*

Selects a single column and collectively positions the column title, data, and totals if applicable. For valid values, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

+

Starts the column at the specified distance to the right of the default starting position.

By default, text items and alphanumeric fields are left-justified in a column, and numeric fields are right-justified in a column.

-

Starts the column at the specified distance to the left of the default starting position.

It is possible to create a report in which columns overlap. If this occurs, simply adjust the values.

*position*

Is the desired distance, in the unit of measurement specified with the UNITS attribute.

### **Example:** Specifying an Absolute Starting Position for a Column

The following illustrates how to position a column in a printed report. It is specified in the request that the PRODUCT\_DESCRIPTION field display three inches from the left margin of the PDF report.

```
SET ONLINE-FMT = PDF
TABLE FILE GGORDER
"PRODUCTS ORDERED ON 08/01/96"
SUM QUANTITY BY PRODUCT_DESCRIPTION
WHERE ORDER_DATE EQ '080196'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, COLUMN=PRODUCT_DESCRIPTION, POSITION=3, $
ENDSTYLE
END
```

The output is:

PRODUCTS ORDERED ON 08/01/96

<u>Product</u>	<u>Ordered Units</u>
Biscotti	6140
Coffee Grinder	2493
Coffee Pot	3100
Croissant	7465
French Roast	12965
Hazelnut	4186
Kona	2591
Mug	5332
Scone	5949
Thermos	2362



**Example:**    **Specifying a Relative Starting Position for a Column**

This request positions the column title and data for the QUANTITY field two inches from the default position, in this case, two inches from the end of the preceding column.

```
SET ONLINE-FMT = PDF
TABLE FILE GGORDER
"PRODUCTS ORDERED ON 08/01/96"
SUM QUANTITY BY PRODUCT_DESCRIPTION
WHERE ORDER_DATE EQ '080196'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, COLUMN=PRODUCT_DESCRIPTION, POSITION=3, $
TYPE=REPORT, COLUMN=QUANTITY, POSITION=+2, $
ENDSTYLE
END
```

QUANTITY, titled Ordered Units in the report, is relatively positioned to Product:

PRODUCTS ORDERED ON 08/01/96		
	<u>Product</u>	<u>Ordered Units</u>
	Biscotti	6140
	Coffee Grinder	2493
	Coffee Pot	3100
	Croissant	7465
	French Roast	12965
	Hazelnut	4186
	Kona	2591
	Mug	5332
	Scone	5949
	Thermos	2362

**Syntax:**    **How to Add Blank Space Around a Report Component**

This syntax applies to a PDF or PS report.

```
TYPE=REPORT, {TOPGAP|BOTTOMGAP}=gap, $
TYPE=type, [COLUMN=identifier,|ACROSSCOLUMN=acrosscolumn,]
{LEFTGAP|RIGHTGAP}=gap, $

TYPE=type, [COLUMN=identifier,|ACROSSCOLUMN=acrosscolumn,]
{LEFTGAP|RIGHTGAP}=gap, $
```

where:

TOPGAP

Indicates how much space to add above the report line.

### BOTTOMGAP

Indicates how much space to add below the report line.

### *gap*

Is the amount of blank space, in the unit of measurement specified with the UNITS attribute.

In the absence of grids or background color, the default value is 0. For RIGHTGAP, the default value is proportional to the size of the text font.

In the presence of grids or background color, the default value increases to provide space between the grid and the text or to extend the color beyond the text.

The gaps must be the same within a single column or row. That is, you cannot specify different left or right gaps for individual cells in the same column, or different top and bottom gaps for individual cells in the same row.

### *type*

Identifies the report component. For valid values, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

### *identifier*

Selects one or more columns using the COLUMN attribute described in [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

### *acrosscolumn*

Selects the same column under every occurrence of an ACROSS sort field using the ACROSSCOLUMN attribute described in [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

### LEFTGAP

Indicates how much space to add to the left of a report column.

### RIGHTGAP

Indicates how much space to add to the right of a report column.

**Note:** For TOPGAP, BOTTOMGAP, LEFTGAP, and RIGHT GAP, you must specify a value of at least 0.013889 (the decimal size of a point in inches). If you specify a value less than this, WebFOCUS will round down to the nearest point, which is zero.

### **Example:** Adding Blank Space Above Data Values

This request generates one-tenth of an inch of blank space above every data value in a PDF report.

```

SET ONLINE-FMT = PDF
SET PAGE-NUM = OFF
TABLE FILE GGORDER
"PRODUCTS ORDERED ON 08/01/96"
" "
SUM QUANTITY BY PRODUCT_DESCRIPTION
WHERE ORDER_DATE EQ '080196'
ON TABLE SET STYLE *
TYPE=DATA, TOPGAP = 0.1, $
ENDSTYLE
END

```

The data is spaced for readability:

PRODUCTS ORDERED ON 08/01/96

<u>Product</u>	<u>Ordered Units</u>
Biscotti	6140
Coffee Grinder	2493
Coffee Pot	3100
Croissant	7465
French Roast	12965
Hazelnut	4186
Kona	2591
Mug	5332
Scone	5949
Thermos	2362

**Example:**    **Adding Blank Space to the Left of a Column**

The following illustrates how to add blank space to the left of a report component. In this example, 1.5 inches of blank space are inserted to the left of the Product Category column.

```
SET ONLINE-FMT=PDF
TABLE FILE CENTORD
HEADING CENTER
"Summary Report for Digital Products"
" "
SUM      LINE_COGS/D12      AS 'Cost of Goods Sold'
BY       PRODTYPE           AS 'Product Type'
BY       PRODCAT            AS 'Product Category'
WHERE PRODTYPE EQ 'Digital';
ON TABLE COLUMN-TOTAL/D12
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, COLUMN=PRODCAT, LEFTGAP=1.5, $
ENDSTYLE
END
```

The output is:

Summary Report for Digital Products		
<u>Product Type</u>	<u>Product Category</u>	<u>Cost of Goods Sold</u>
Digital	CD Players	30,519,126
	Camcorders	91,518,330
	Cameras	1,785,627
	DVD	47,275,593
	Digital Tape Recorders	36,157,863
	PDA Devices	202,123,656
TOTAL		409,380,195

**Arranging Columns on a Page**

How easily a user locates data depends on the arrangement of columns on a page. You have many design options. Using StyleSheet attributes or commands you can:

- ☐ Determine column width.
- ☐ Control the number of spaces between columns.
- ☐ Change the order of vertical sort (BY) columns.
- ☐ Stack columns to reduce report width, or to easily compare values in a report by creating a matrix.
- ☐ Specify the absolute or relative starting position for a column.

**Reference: Column Arrangement Features**

Feature	Description	Applies to
<code>SQUEEZE</code>	Sets column width.	HTML PDF PS
<code>SET SPACES</code>	Sets number of spaces between columns.	HTML
<code>SEQUENCE</code>	Sets column order.	PDF PS HTML EXL2K ( <b>Note:</b> Does not work with XLSX and EXL2K FORMULA)
<code>FOLD-LINE</code>	Reduces report width by stacking columns.	PDF PS
<code>OVER</code>	Stacks columns by placing them over one another.	HTML PDF PS
<code>IN {n +n}</code>	Sets absolute or relative starting position of a column.	HTML PDF PS

**Determining Column Width**

The value of the `SQUEEZE` attribute in a StyleSheet determines column width in a report. You can use a `SET` parameter instead of a StyleSheet to set the value of `SQUEEZE`. If there are conflicting StyleSheet and `SET` values, the StyleSheet overrides the `SET`. For details on `SET`, see the *Developing Reporting Applications* manual.

When SQUEEZE is set to ON (the default), StyleSheet column width is ignored. Column width is determined using your browser's default settings.

When using SQUEEZE it may affect the way headings, footings, and column titles display in your report. For details, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.

### **Syntax:**    **How to Determine Column Width (HTML)**

This syntax applies to an HTML report. For the syntax for a PDF or PS report, see [How to Determine Column Width \(PDF or PS\)](#) on page 1268.

The sizing of tables and column width within a standard HTML report is done by browser processing, including the browser default settings and the size of the browser window or iframe. The columns are sized to fit the largest data value or column title, whichever is greater, and trailing spaces are automatically removed.

In standard HTML reports, the data is presented in a single table, so the column widths are fixed for all data rows.

In HFREEZE HTML reports, the data is presented with no wrapping in three tables containing the heading, data rows, and footing, to ensure the alignment of the column titles with the data rows.

```
[TYPE=REPORT,] SQUEEZE={ON|OFF}, $
```

where:

[TYPE=REPORT](#)

Applies the column width to the entire report. Not required, as it is the default.

[ON](#)

Determines column width based on the longest data value or column title, whichever is greater. ON is the default value.

For HTML reports, the web browser shrinks the column width to the shortest column title or field value.

[OFF](#)

Determines column width based on the longest data value or column title, whichever is greater. Blank spaces pad the column width up to the length of the column title or field format, whichever is greater.

**Note:**

- ❑ With output format HTML, the browser has control over some of the styling elements and WebFOCUS cannot override them. That is why SQUEEZE=n is not supported syntax for output format HTML.
- ❑ SQUEEZE is not supported for columns created with the OVER phrase.

**Example: Using Default Column Width (HTML)**

This request uses SQUEEZE=ON (the default) for an HTML report. Column width is based on the wider of the data value or column title.

```
SET PAGE-NUM = OFF
TABLE FILE GGSALES
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, FONT=COURIER, $
ENDSTYLE
END
```

For Category, Unit Sales, and Dollar Sales, the column title is wider than the corresponding data values. For Product, the wider data values determine column width. The HTML report is:

<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

**Example: Using Column Width Based on Field Format (HTML)**

This request sets SQUEEZE to OFF for an HTML report. Column width is based on the longest data value or column title, whichever is greater.

```
SET PAGE-NUM = OFF
TABLE FILE GGSales
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, SQUEEZE=OFF, FONT=COURIER, $
ENDSTYLE
END
```

Blank spaces pad the column width up to the length of the field format for Category (A11) and Product (A16). The HTML report is:

<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

**Syntax:**      **How to Determine Column Width (PDF or PS)**

This syntax applies to a PDF or PS report. For the syntax for an HTML report, see [How to Determine Column Width \(HTML\)](#) on page 1266.

```
[TYPE=REPORT,] COLUMN=identifier, SQUEEZE={ON|OFF|width}, $
```

where:

```
TYPE=REPORT
```

Applies the column width to the entire report. Not required, as it is the default.

*identifier*

Selects a column using the COLUMN attribute described in [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167. If you omit a column identifier, the value for SQUEEZE applies to all columns in a report. You can also use SET SQUEEZE to set the width of all columns.



ON

Determines column width based on the widest data value or column title, whichever is greater.

OFF

Determines column width based on the longest data value or column title, whichever is greater. Blank spaces pad the column width up to the length of the column title or field format, whichever is greater. OFF is the default value.

*width*

Is a measurement for the column width, specified with the UNITS attribute.

If the widest data value exceeds the specified measurement:

And the field is...	The following displays...
Alphanumeric	As much of the value as will fit in the specified width, followed by an exclamation mark (!) to indicate truncation.
Numeric	Asterisks (*) in place of the field value.

**Note:** SQUEEZE is not supported for columns created with the OVER phrase.

**Example: Determining Column Width (PDF)**

This request uses SQUEEZE=2.5 to increase the default column width of the PRODUCT field in a PDF report. Note that this feature is used primarily for printed reports. Depending on your screen resolution, the column width may look different than how it will print.

```
SET ONLINE-FMT = PDF
TABLE FILE GGSales
SUM UNITS
BY PRODUCT
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, COLUMN=PRODUCT, SQUEEZE=2.5, $
ENDSTYLE
END
```

The PDF report is:

<u>Product</u>	<u>Unit Sales</u>
Biscotti	421377
Capuccino	189217
Coffee Grinder	186534
Coffee Pot	190695
Croissant	630054
Espresso	308986
Latte	878063
Mug	360570
Scone	333414
Thermos	190081

**Controlling Column Spacing**

By default, report columns are separated by one or two spaces, depending on the output width. The SET SPACES or ON TABLE SET SPACES parameter controls the number of spaces between columns in a report.

In a horizontal sort (ACROSS) phrase, the SPACES parameter determines the distance between horizontal sort sets. Within a set, the distance between columns is always one space and cannot be changed.

This feature applies to an HTML report. It requires you to set the STYLEMODE parameter to FIXED.

**Syntax:**      **How to Control Column Spacing**

This syntax applies to an HTML report.

**For all report requests in a procedure**

```
SET SPACES = {n|AUTO}
```

**For one report request**

```
ON TABLE SET SPACES {n|AUTO}
```

where:

*n*

Is an integer between 1 and 8, indicating the number of spaces between report columns.

AUTO

Automatically separates report columns with one or two spaces. AUTO is the default value.

**Example:**      **Controlling Column Spacing Between Horizontal (ACROSS) Fields**

This request uses ACROSS with ON TABLE SET SPACES. The ON TABLE SET STYLEMODE FIXED parameter is required for HTML.

```
TABLE FILE CENTORD
SUM QUANTITY LINEPRICE ACROSS ORDER_NUM BY PLANT AS 'Plant'
WHERE ORDER_NUM EQ '28003' OR '28004'
ON TABLE SET SPACES 7
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLEMODE FIXED
END
```

The ACROSS set consists of the fields titled Quantity and Line Total. The distance between each set is seven spaces:

Plant	Order Number:			
	28003		28004	
	Quantity:	Line Total	Quantity:	Line Total
BOS	.	.	6	\$2,023.62
LÅ	5	\$1,688.47	.	.

**Changing Column Order**

You can change the order in which vertical sort (BY) columns are displayed in a report. This feature does not apply to horizontal sort (ACROSS) rows or stacked (OVER) columns.

**Syntax:**      **How to Change Column Order**

This syntax applies to PDF, PS, HTML, XLSX, and EXL2K reports. XLSX FORMULA and EXL2K FORMULA formats are not supported.

```
[TYPE=REPORT,] COLUMN=identifier, SEQUENCE=sequence, $
```

where:

**TYPE=REPORT**

Applies the column order to the entire report. Not required, as it is the default.

*identifier*

Selects a column using the COLUMN attribute described in [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

*sequence*

Is a number that represents the order of the selected column.

Numbers need not be in sequential order or in increments of one. The order of the columns is from lowest to highest. NOPRINT columns are not included.

**Example:**      **Changing Column Order**

This request rearranges the order in which columns normally appear in the report, that is, with SNAME first, PRODCAT second, and LINEPRICE third.

```
SET ONLINE-FMT = PDF
TABLE FILE CENTORD
SUM LINEPRICE AS 'Sales'
BY SNAME BY PRODCAT AS 'Product'
WHERE SNAME EQ 'eMart' OR 'City Video'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, COLUMN=SNAME, SEQUENCE=3, $
TYPE=REPORT, COLUMN=PRODCAT, SEQUENCE=2, $
TYPE=REPORT, COLUMN=LINEPRICE, SEQUENCE=1, $
ENDSTYLE
END
```

LINEPRICE (Sales) is now the first column, PRODCAT (Product) is the second column (as it was by default), and SNAME (Store Name) is the third column. The PDF report is:

<u>Sales</u>	<u>Product</u>	<u>Store Name:</u>
\$469,771.73	CD Players	City Video
\$5,488,218.35	Camcorders	
\$359,190.46	Cameras	
\$1,023,484.18	DVD	
\$1,151,410.41	Digital Tape Recorders	
\$6,444,433.57	PDA Devices	
\$362,542.59	VCRs	
\$20,371,041.94	CD Players	eMart
\$142,957,872.94	Camcorders	
\$2,428,149.11	Cameras	
\$18,215,609.08	DVD	
\$14,123,119.89	Digital Tape Recorders	
\$82,376,713.78	PDA Devices	
\$10,015,249.62	VCRs	

## Stacking Columns

You can stack columns in a report to reduce report width, or to easily compare values in a report by creating a matrix. To stack columns, you can use:

### ☐ FOLD-LINE:

- ☐ Reduces the use of space for a vertical sort (BY) column that changes infrequently. You can include up to 16 FOLD-LINE phrases in a request.
- ☐ Is available for PDF, PS, and DHTML reports, but not cell-based formats, such as HTML, XLSX, and EXL2K.
- ☐ Cell based styling features including BACKCOLOR and BORDERS are not supported.
- ☐ Alternating back colors are not supported for stacked columns.

### ☐ OVER:

- ☐ Stacks columns one over another, which increases readability, especially when you are sorting your report horizontally with ACROSS. OVER is useful when you are creating financial reports. For complete details, see [Creating Financial Reports With Financial Modeling Language \(FML\)](#) on page 1729.
- ☐ Alternating back colors are not supported for stacked columns.

The difference between FOLD-LINE and OVER is that FOLD-LINE begins the second line (not the second column) just underneath the first line, but slightly indented. OVER literally stacks the values of one column directly over another. You can use FOLD-LINE and OVER in the same request.

### **Syntax:** How to Stack Columns With FOLD-LINE

```
display_command fieldname ... FOLD-LINE fieldname ...
```

or

```
{ON|BY} fieldname FOLD-LINE
```

where:

*display\_command*

Is a display command. There is no offset when a line is folded after a display field.

*fieldname*

Is a display field or sort field placed on a separate line when the value of the ON or BY field changes. When folded on a sort field, a line is offset by two spaces from the preceding line.

ON|BY

Is a vertical sort phrase. The terms are synonymous.

### **Example:** Stacking Columns With FOLD-LINE

The following illustrates how to use FOLD-LINE to decrease the width of your report. In this example, columns are stacked when the value of the sort field CATEGORY changes.

```
TABLE FILE GGSALES
SUM UNITS BUDUNITS
BY CATEGORY
ON CATEGORY FOLD-LINE
ON TABLE SET ONLINE-FMT PDF
ON TABLE SET PAGE-NUM OFF
END
```

The report is:

<u>Category</u>	<u>Unit Sales</u>	<u>Budget Units</u>
Coffee	1376266	1385923
Food	1384845	1377564
Gifts	927880	931007

Without FOLD-LINE, the report looks like this:

<u>Category</u>	<u>Unit Sales</u>	<u>Budget Units</u>
Coffee	1376266	1385923
Food	1384845	1377564
Gifts	927880	931007

**Syntax:**      **How to Stack Columns With OVER**

*display\_command fieldname1 OVER fieldname2 OVER fieldname3 ...*

where:

*display\_command*

Is a display command.

*fieldname1, fieldname2, fieldname3*

Is a display field or calculated value. A text field is not valid.

**Example:**      **Stacking Columns With OVER**

This request contains an ACROSS phrase in an HTML report to sort horizontally by department. It uses two OVER phrases to stack columns.

```
TABLE FILE EMPLOYEE
SUM GROSS OVER DED_AMT OVER
COMPUTE NET/D8.2M = GROSS - DED_AMT;
ACROSS DEPARTMENT
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF,$
ENDSTYLE
END
```

With the use of OVER, the columns GROSS, DED\_AMT, and NET are stacked for readability:

	DEPARTMENT	
	MIS	PRODUCTION
GROSS	\$50,499.09	\$50,922.38
DED_AMT	\$28,187.11	\$23,391.24
NET	\$22,311.98	\$27,531.14

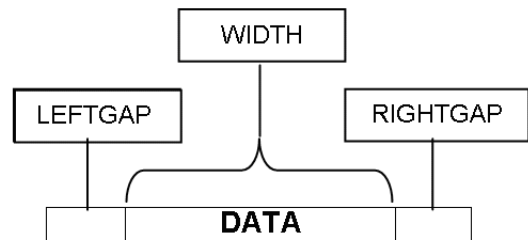
Without the use of OVER, the HTML report looks like this:

DEPARTMENT					
MIS			PRODUCTION		
GROSS	DED_AMT	NET	GROSS	DED_AMT	NET
\$50,499.09	\$28,187.11	\$22,311.98	\$50,922.38	\$23,391.24	\$27,531.14

Alignment of Fields in Reports Using OVER in PDF Report Output

When columns are placed on report output, they are separated by gaps. You can control the size of the gaps between columns with the LEFTGAP and RIGHTGAP StyleSheet attributes.

By default, the gaps between columns are placed outside of the boundaries reserved for the fields on the report output. Therefore, the width or squeeze value defined for a field defines the size of the text area for the data value. It does not count the width of the gaps between columns. The bounding box used to define borders and background color is determined based on the data width plus the left gap plus the right gap.

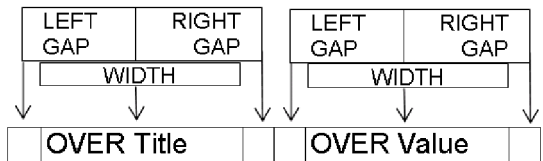




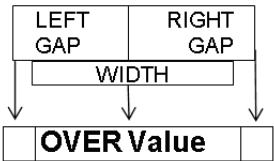
Gaps external to the column boundaries must be accounted for when you try to align fields in reports that use the OVER phrase.

This feature is designed to support the development of multi-row reports using blank AS names (column titles). Unless otherwise noted, these features work with non-blank titles, but they have not been designed to support alignment with non-blank column titles.

By default, column titles are placed to the left of the field values in a report using OVER. The OVER Title and the OVER Value each are measured by the combination of three parameters, LEFTGAP, WIDTH, and RIGHTGAP:



With OVER and blank AS names, each data value becomes a data cell that can be used to construct rows and columns within the data lines of the report. In order to align data values on a lower line with the columns above them, you must calculate widths for the lower level columns that take into account the widths of the data above them plus the widths of all of the left gaps and right gaps in between.



It can be complex to calculate how to size each column when aligning data and headings in reports using OVER. Each calculation of the column size must additionally account for the external left and right gap, and these gaps are cumulative as the number of columns on a given row increases.

Using the GAPINTERNAL=ON StyleSheet attribute, you can have the gaps placed within the column boundaries for PDF report output. This feature makes it much easier to align fields and headings in reports that use the OVER phrase to create multiple lines.

**Note:** OVER is now supported with SQUEEZE.

**Syntax:**      **How to Control GAP Placement on Reports**

```
TYPE=REPORT, GAPINTERNAL={OFF|ON}
```

where:

OFF

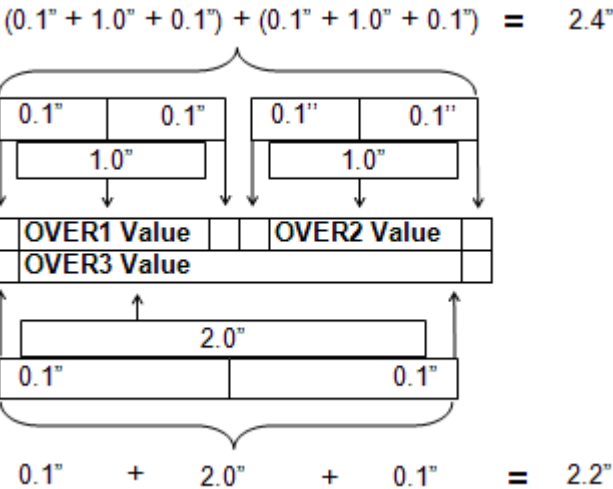
Places the left and right gaps outside the defined field width. OFF is the default value.

ON

Places the left and right gaps internal to the defined field width.

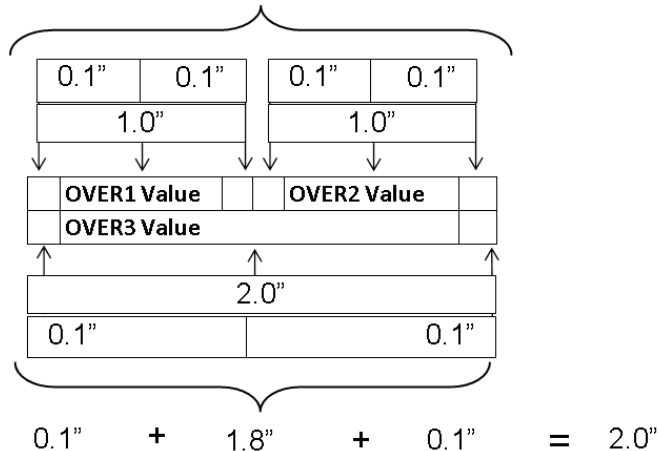
**Example:**      **Comparing External Gaps With Internal Gaps**

With GAPINTERNAL=OFF, you must account for the accumulation of left and right gaps as well as the field widths when defining widths of stacked columns.



With GAPINTERNAL=ON, the defined WIDTH represents the entire space used by the given data cell or column. This takes the cumulative effect out as the OVER values proceed across a row.

$$(0.1'' + 0.8'' + 0.1'') + (0.1'' + 0.8'' + 0.1'') = 2.0''$$



### Example: Using GAPINTERNAL in a Report

The following request against the GGSALES data source places the PRODUCT field over the UNITS and DOLLARS fields and sets GAPINTERNAL to OFF:

```
SET LAYOUTGRID=ON
TABLE FILE GGSALES
"Product<+0>"
"Units<+0>Dollars"
SUM
PRODUCT AS ''
OVER
UNITS/D8C AS '' DOLLARS/D12.2CM AS ''
BY PRODUCT NOPRINT
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, SQUEEZE=ON, FONT=ARIAL, SIZE=8, LEFTMARGIN=1, TOPMARGIN=1,
LEFTGAP=.1, RIGHTGAP=.1, GAPINTERNAL=OFF, $
TYPE=REPORT, BORDER=ON, $
TYPE=HEADING, BORDERALL=ON, $
TYPE=HEADING, LINE=1, ITEM=1, POSITION = PRODUCT, $
TYPE=HEADING, LINE=2, ITEM=1, POSITION = UNITS, $
TYPE=HEADING, LINE=2, ITEM=2, POSITION = DOLLARS, $
TYPE=REPORT, COLUMN=PRODUCT(2), SQUEEZE=2, $
TYPE=REPORT, COLUMN=UNITS, SQUEEZE=1, $
TYPE=REPORT, COLUMN=DOLLARS, SQUEEZE=1, $
ENDSTYLE
END
```

The widths specified for UNITS and DOLLARS are one inch each, while the PRODUCT field is specified to be two inches. With GAPINTERNAL=OFF, the LAYOUTGRID shows that the widths used to place the columns are greater than the widths specified in the request. The additional space presented by the external leftgap and rightgap accounts for this effect:

PAGE 1		
Product		
Units	Dollars	
Biscotti		
421,377	\$5,263,317.00	
Capuccino		
189,217	\$2,381,590.00	
Coffee Grinder		
186,534	\$2,337,567.00	
Coffee Pot		
190,695	\$2,449,585.00	
Croissant		
630,054	\$7,749,902.00	
Espresso		
308,986	\$3,906,243.00	
Latte		
878,063	\$10,943,622.00	
Mug		
360,570	\$4,522,521.00	
Scone		
333,414	\$4,216,114.00	
Thermos		
190,081	\$2,385,829.00	

The heading borders are aligned on the right of the report because of the SQUEEZE=ON attribute in the StyleSheet. Extra space was added to the report to align the headings. If you change the StyleSheet declaration for the PRODUCTS field to JUSTIFY=RIGHT, you can see that the extra space prevents the product value from aligning with the dollar value:

PAGE 1		
Product		
Units		Dollars
Biscotti		
421,377		\$5,263,317.00
Capuccino		
189,217		\$2,381,590.00
Coffee Grinder		
186,534		\$2,337,567.00
Coffee Pot		
190,695		\$2,449,585.00
Croissant		
630,054		\$7,749,902.00
Espresso		
308,986		\$3,906,243.00
Latte		
878,063		\$10,943,622.00
Mug		
360,570		\$4,522,521.00
Scone		
333,414		\$4,216,114.00
Thermos		
190,081		\$2,385,829.00

Changing the StyleSheet declaration to GAPINTERNAL=ON causes the specified widths to be used because the gaps are internal and are included in the specified values:

PAGE 1	
Product	
Units	Dollars
Biscotti	
421,377	\$5,263,317.00
Capuccino	
189,217	\$2,381,590.00
Coffee Grinder	
186,534	\$2,337,567.00
Coffee Pot	
190,695	\$2,449,585.00
Croissant	
630,054	\$7,749,902.00
Espresso	
308,986	\$3,906,243.00
Latte	
878,063	\$10,943,622.00
Mug	
360,570	\$4,522,521.00
Scone	
333,414	\$4,216,114.00
Thermos	
190,081	\$2,385,829.00

The following report output demonstrates that the values align properly even if the PRODUCT values are defined with JUSTIFY=RIGHT:

PAGE 1	
Product	
Units	Dollars
	Biscotti
421,377	\$5,263,317.00
	Capuccino
189,217	\$2,381,590.00
	Coffee Grinder
186,534	\$2,337,567.00
	Coffee Pot
190,695	\$2,449,585.00
	Croissant
630,054	\$7,749,902.00
	Espresso
306,986	\$3,906,243.00
	Latte
878,063	\$10,943,622.00
	Mug
360,570	\$4,522,521.00
	Scone
333,414	\$4,216,114.00
	Thermos
190,081	\$2,385,829.00

## Positioning a Column

You can specify the absolute or relative starting position for a column in a report. The relative starting position is the number of characters to the right of the last column.

When using this feature with an HTML report, set the STYLEMODE parameter to FIXED.

**Syntax:**      **How to Position a Column**

*field* IN {*n*|+*n*}

where:

*field*

Is the column that is positioned.

*n*

Is a number indicating the absolute position of the column.

When used with ACROSS, *n* specifies the starting position of the ACROSS set.

When used with FOLD-LINE or OVER, *n* applies to the line on which the referenced field occurs.

+*n*

Is a number indicating the relative position of the column. The value of *n* is the number of characters to the right of the last column.

**Example:**      **Positioning Columns**

This request specifies absolute positioning for the three columns in the report. The ON TABLE SET STYLEMODE FIXED parameter is required for HTML.

```
TABLE FILE CENTQA
SUM CNT.PROBNUM IN 1 AS 'Total #,Problems'
SUM CNT.PROBNUM IN 45 AS '# Problems,by Product'
BY PLANT NOPRINT BY PRODNAME IN 15
WHERE PLANT EQ 'ORL'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLEMODE FIXED
END
```



The columns are spaced for readability:

Total # Problems -----	Product Name: -----	# Problems by Product -----
95	120 VHS-C Camcorder 40 X	7
	2 Hd VCR LCD Menu	6
	250 8MM Camcorder 40 X	10
	650DL Digital Camcorder 150 X	6
	AR2 35MM Camera 8 X	17
	AR3 35MM Camera 10 X	9
	Combo Player - 4 Hd VCR + DVD	12
	DVD Upgrade Unit for Cent. VCR	9
	QX Portable CD Player	9
	R5 Micro Digital Tape Recorder	3
	ZC Digital PDA - Standard	7

**Example: Positioning Horizontal Sort (ACROSS) Columns**

This request uses the IN phrase with the horizontal sort field PLANT to specify the column starting position. It also uses relative positioning to add extra spaces between the PROBNUM columns. The ON TABLE SET STYLEMODE FIXED parameter is required for HTML reports.

```
TABLE FILE CENTQA
SUM PROBNUM IN +8
ACROSS PLANT IN 35
BY PROBLEM_CATEGORY
WHERE PLANT EQ 'BOS' OR 'ORL'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLEMODE FIXED
END
```

The ACROSS set starts in column 35, and there are eight extra spaces between the data columns in the ACROSS:

Problem Category	Plant		
	BOS		ORL
-----			
Incorrect Labeling	39921		19877
Mechanical Failure	19564		13792
Missing Component	38285		18709
Physical Damage	65497		16175
Power Failure	41472		14277
Remote Failure	59655		8635

### *Example:* Positioning Stacked (OVER) Columns

The following request uses OVER to stack columns and IN to position them.

```
TABLE FILE EMPLOYEE
SUM GROSS IN 40
OVER DED_AMT IN 40
BY DEPARTMENT BY LAST_NAME IN 20
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLEMODE FIXED
END
```

In the report, GROSS and DED\_AMT are stacked, starting in column 40. LAST\_NAME starts in column 20.

DEPARTMENT	LAST_NAME		
-----	-----		
MIS	BLACKWOOD	GROSS	\$9,075.00
		DED_AMT	\$6,307.00
	CROSS	GROSS	\$22,013.75
		DED_AMT	\$15,377.40
	GREENSPAN	GROSS	\$2,970.84
		DED_AMT	\$505.04
	JONES	GROSS	\$6,099.50
		DED_AMT	\$2,866.18
	MCCOY	GROSS	\$1,540.00
		DED_AMT	\$458.69
PRODUCTION	SMITH	GROSS	\$8,800.00
		DED_AMT	\$2,672.80
	BANNING	GROSS	\$2,475.00
		DED_AMT	\$1,427.24
	IRVING	GROSS	\$17,094.00
		DED_AMT	\$11,949.44
	MCKNIGHT	GROSS	\$9,130.00
		DED_AMT	\$3,593.92
	ROMANS	GROSS	\$7,040.00
		DED_AMT	\$3,507.88
	SMITH	GROSS	\$6,183.36
		DED_AMT	\$1,104.96
	STEVENS	GROSS	\$9,000.02
		DED_AMT	\$1,807.80

## Suppressing Column Display

A report request may include a field to create a certain result. For example, it may name a sort field by which to arrange data. However, you may not want to display the title or values of that field if they appear elsewhere in the report. The phrase NOPRINT (synonym SUP-PRINT) suppresses column display.

Reference: Column Suppression Commands

Command	Description	Applies to
<code>NOPRINT</code> or <code>SUP-PRINT</code>	Suppresses column display.	HTML PDF PS

Syntax: How to Suppress Column Display

`display_command fieldname {NOPRINT|SUP-PRINT}`

or

`{ON|BY} fieldname {NOPRINT|SUP-PRINT}`

where:

`display_command`

Is a display command.

`fieldname`

Is a display field or sort field. The field values are used but not displayed. A HOLD file will not contain the values of a suppressed BY field.

For a calculated value with NOPRINT, repeat AND COMPUTE before the next calculated value if applicable.

`NOPRINT|SUP-PRINT`

Suppresses column display. The terms are synonymous.

`ON|BY`

Is a vertical sort phrase. The terms are synonymous.

Example: Suppressing the Display of a Sort Field

This request sorts data by city. Since the page heading contains the name of the city, the sort field occurrence is suppressed.

```

TABLE FILE SALES
HEADING
"Page <TABPAGENO"
"SALES REPORT FOR <CITY"
PRINT UNIT_SOLD AND DELIVER_AMT
BY CITY PAGE-BREAK NOPRINT
BY PROD_CODE
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END

```

The page heading identifies the city to which the data applies:

Page 1

SALES REPORT FOR NEW YORK

<u>PROD CODE</u>	<u>UNIT SOLD</u>	<u>DELIVER AMT</u>
B10	30	30
B17	20	40
B20	15	30
C17	12	10
D12	20	30
E1	30	25
E3	35	25

Page 2

SALES REPORT FOR NEWARK

<u>PROD CODE</u>	<u>UNIT SOLD</u>	<u>DELIVER AMT</u>
B10	13	30
B12	29	30

Without NOPRINT, the report would unnecessarily repeat the city:

Page 1

SALES REPORT FOR NEW YORK

<u>CITY</u>	<u>PROD CODE</u>	<u>UNIT SOLD</u>	<u>DELIVER AMT</u>
NEW YORK	B10	30	30
	B17	20	40
	B20	15	30
	C17	12	10
	D12	20	30
	E1	30	25
	E3	35	25

Page 2

SALES REPORT FOR NEWARK

<u>CITY</u>	<u>PROD CODE</u>	<u>UNIT SOLD</u>	<u>DELIVER AMT</u>
NEWARK	B10	13	30
	B12	29	30

**Example: Suppressing Display of a Sort Field With Subtotal**

This request generates a subtotal for each value of the sort field CATEGORY but suppresses the display of the sort field occurrence.

```
TABLE FILE GGSales
SUM UNITS BY CATEGORY
BY PRODUCT
ON CATEGORY SUB-TOTAL SUP-PRINT PAGE-BREAK
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The default subtotal line identifies each category (for example, \*TOTAL Coffee):

<u>Product</u>	<u>Unit Sales</u>
Capuccino	189217
Espresso	308986
Latte	878063

\*TOTAL Coffee 1376266

<u>Product</u>	<u>Unit Sales</u>
Biscotti	421377
Croissant	630054
Scone	333414

\*TOTAL Food 1384845

Without SUP-PRINT, the report would unnecessarily repeat the category:

<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>
Coffee	Capuccino	189217
	Espresso	308986
	Latte	878063
*TOTAL Coffee		1376266

<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>
Food	Biscotti	421377
	Croissant	630054
	Scone	333414
*TOTAL Food		1384845



**Example:   Sorting Alphabetically**

This request sorts last names alphabetically but avoids duplication of data by suppressing the sort field occurrence of LAST\_NAME.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME
BY LAST_NAME NOPRINT
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

Last names are arranged alphabetically:

```
LAST NAME
BANNING
BLACKWOOD
CROSS
GREENSPAN
IRVING
JONES
MCCOY
MCKNIGHT
ROMANS
SMTH
SMTH
STEVENS
```

**Inserting a Page Break**

Use the PAGE-BREAK command to generate a new page each time the value of a specified vertical sort (BY) field changes. This helps to prevent related information from being presented over multiple pages. When you use a page break, column titles and any page headings appear at the top of each new page. When the request has a PAGE-BREAK, the GRANDTOTAL is on a page by itself.

PAGE-BREAK does not apply when report output is stored in a HOLD, SAVE, or SAVB file.

In an HTML report, PAGE-BREAK creates a new section of the report, with column titles and an incremented page number, on the same webpage. It does not, by itself, create a new webpage. To create multiple webpages in an HTML report:

- ☐ Burst the report, with each page corresponding to a different sort field value. For details, see *Bursting Reports Into Multiple HTML Files* on page 965.
- ☐ Use SET STYLEMODE=PAGED in conjunction with PAGE-BREAK. This is useful when you are distributing a report via ReportCaster.
- ☐ Use SET WEBVIEWER=ON, which accesses the WebFOCUS Viewer, in conjunction with PAGE-BREAK. For details, see the *Developing Reporting Applications* manual.

**Reference: Page Break Commands**

Command	Description	Applies to
PAGE-BREAK	Generates new page.	HTML PDF PS
NOSPLIT	Prevents undesirable page break.	PDF PS
SET LINES	Synchronizes report page with browser page.	HTML

**Reference: Working With Multi-Table HTML Reports**

- ☐ You can control where a report breaks using SET LINES or PAGE-BREAK in the request.
- ☐ ON *sortfield* PAGE-BREAK or BY *sortfield* PAGE-BREAK overrides a SET LINES command and breaks a report into multiple HTML tables whenever the sort field value changes.
- ☐ Column titles are generated for every PAGE-BREAK or according to the SET LINES parameter.
- ☐ When a report is broken into multiple HTML tables, the browser displays each table according to its own algorithm. Set SQUEEZE to OFF and/or WRAP to OFF to ensure that HTML tables are aligned consistently across pages.

**Syntax:**      **How to Insert a Page Break**

```
{ON|BY} fieldname PAGE-BREAK [REPAGE] [WHEN expression;
```

where:

**ON|BY**

Is a vertical sort phrase. The terms are synonymous.

*fieldname*

Is the sort field on which the page break occurs. Specify the lowest level sort field at which the page break occurs. A page break occurs automatically whenever a higher level sort field changes.

**REPAGE**

Resets the page number to 1 at each page break or, if combined with WHEN, whenever the WHEN criteria are met.

**WHEN *expression***

Specifies a conditional page break in the printing of a report as determined by a logical expression. See [Controlling Report Formatting](#) on page 1139 for details.

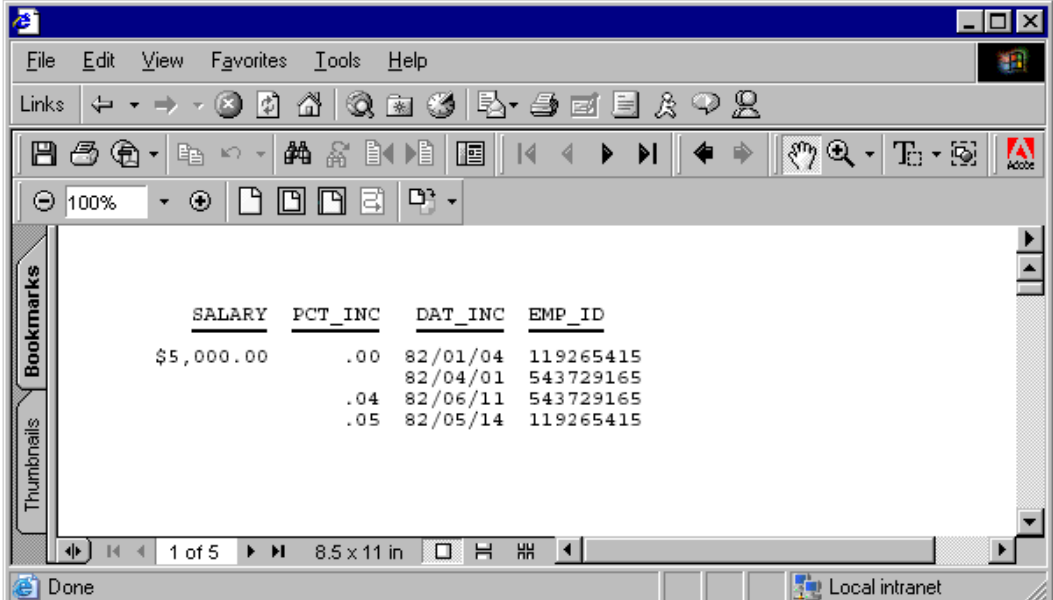
**Example:**      **Inserting a Page Break**

This request generates a new page whenever the value of the sort field SALARY changes.

```
TABLE FILE EMPLOYEE
PRINT EMP_ID
BY SALARY IN-GROUPS-OF 5000
BY PCT_INC BY DAT_INC
ON SALARY PAGE-BREAK
ON TABLE SET ONLINE-FMT PDF
ON TABLE SET PAGE-NUM OFF
END
```

## Inserting a Page Break

The first two pages of the report are displayed to illustrate where the page breaks occur:

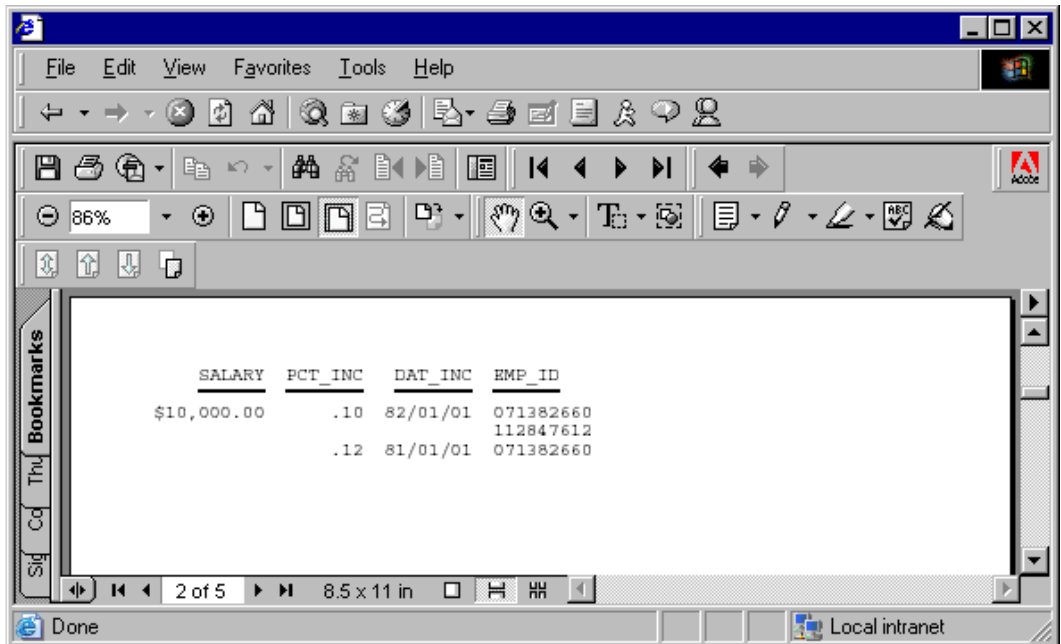


The screenshot shows a web browser window with a report displayed. The report contains a table with the following data:

<u>SALARY</u>	<u>PCT_INC</u>	<u>DAT_INC</u>	<u>EMP_ID</u>
\$5,000.00	.00	82/01/04	119265415
		82/04/01	543729165
	.04	82/06/11	543729165
	.05	82/05/14	119265415

The browser window includes a menu bar (File, Edit, View, Favorites, Tools, Help), a Links toolbar, a main toolbar with various icons, a status bar at the bottom showing "1 of 5" and "8.5 x 11 in", and a taskbar at the very bottom with "Done" and "Local intranet" buttons.

The second page is:



### Example: Displaying a Multiple-Table HTML Report

In this request, each page is returned to the browser as a separate HTML table. SQUEEZE is set to OFF for consistent alignment of tables across pages.

```
SET STYLEMODE = PAGED
SET LINES = 12
TABLE FILE CENTORD
HEADING
"SALES OVER $200,000"
PRINT LINEPRICE AS 'Sales'
BY SNAME BY ORDER_NUM
WHERE LINEPRICE GT 200000
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, SQUEEZE=OFF, $
ENDSTYLE
END
```

Two pages of the report follow, showing consistent alignment:

PAGE 10

SALES OVER \$200,000

<u>Store Name:</u>	<u>Order Number:</u>	<u>Sales</u>
Audio Expert	59079	\$241,754.82
	59487	\$299,272.98
	59515	\$241,570.90
	74080	\$200,724.66
	74549	\$326,165.31
	74557	\$267,960.68

PAGE 11

SALES OVER \$200,000

<u>Store Name:</u>	<u>Order Number:</u>	<u>Sales</u>
Audio Expert	74610	\$222,234.35
	76810	\$286,025.60
	77790	\$504,891.18
City Video	54691	\$256,552.45
	74691	\$250,675.75
Consumer Merchandise	37395	\$283,885.51

The same two pages illustrate inconsistent alignment with SQUEEZE set to ON:

PAGE 10

SALES OVER \$200,000

<u>Store Name:</u>	<u>Order Number:</u>	<u>Sales</u>
Audio Expert	59079	\$241,754.82
	59487	\$299,272.98
	59515	\$241,570.90
	74080	\$200,724.66
	74549	\$326,165.31
	74557	\$267,960.68

PAGE 11

SALES OVER \$200,000

<u>Store Name:</u>	<u>Order Number:</u>	<u>Sales</u>
Audio Expert	74610	\$222,234.35
	76810	\$286,025.60
	77790	\$504,891.18
City Video	54691	\$256,552.45
	74691	\$250,675.75
Consumer Merchandise	37395	\$283,885.51

### Preventing an Undesirable Split

A page break may occur in the middle of information logically grouped by a sort field, causing one or more group-related lines to appear by themselves on the next page or in the next window. Use the NOSPLIT option to avoid this kind of break. When the value of the sort field changes, the total number of lines related to the new value appear on a new page, including sort headings, sort footings, and subtotals if applicable.

This feature applies to a PDF or PS report.

If you use NOSPLIT with PAGE-BREAK, the PAGE-BREAK must apply to a higher-level sort field. Otherwise, NOSPLIT is ignored. NOSPLIT is also ignored when report output is stored in a HOLD, SAVE, or SAVB file. NOSPLIT is not compatible with the TABLEF command and generates an error message.

### **Syntax:**      **How to Prevent an Undesirable Split**

This syntax applies to a PDF or PS report. Use only one NOSPLIT per report request.

```
{ON|BY} fieldname NOSPLIT
```

where:

*ON|BY*

Is a vertical sort phrase. The terms are synonymous.

*fieldname*

Is the name of the sort field for which sort groups are kept together on the same page.

### **Example:**      **Preventing an Undesirable Split**

This request uses NOSPLIT to keep related information on the same page:

```
SET ONLINE-FMT = PDF
TABLE FILE EMPLOYEE
PRINT DED_CODE AND DED_AMT
BY PAY_DATE BY LAST_NAME
ON LAST_NAME NOSPLIT
END
```



When the value of LAST\_NAME changes from STEVENS to CROSS, the lines related to CROSS do not fit on the current page. With NOSPLIT, they appear on the next page:

	HLTH	\$50.87
	LIFE	\$30.52
	SAVE	\$122.10
	STAT	\$85.47
SMITH	CITY	\$1.43
	FED	\$121.55
	FICA	\$100.10
	HLTH	\$22.75
	LIFE	\$13.65
	SAVE	\$54.60
	STAT	\$20.02
	CITY	\$.75
	FED	\$64.10
	FICA	\$52.79
	STAT	\$10.56
STEVENS	CITY	\$.92
	FED	\$77.92
	FICA	\$64.17
	HLTH	\$8.68
	LIFE	\$5.21
	SAVE	\$20.83
	STAT	\$12.83



PAGE 2

<u>PAY_DATE</u>	<u>LAST_NAME</u>	<u>DED_CODE</u>	<u>DED_AMT</u>
82/02/26	CROSS	CITY	\$7.52
		FED	\$638.96
		FICA	\$526.20
		HLTH	\$32.22
		LIFE	\$19.33
	IRVING	SAVE	\$77.32
		STAT	\$105.24
		CITY	\$6.10
		FED	\$518.92
		FICA	\$427.35
	MCKNIGHT	HLTH	\$50.87
		LIFE	\$30.52
		SAVE	\$122.10
		STAT	\$85.47
		CITY	\$2.51
		FED	\$213.18
		FICA	\$175.56
		HLTH	\$18.81
		LIFE	\$11.29
		SAVE	\$45.14

Without NOSPLIT, the information for CROSS falls on the first and second pages:

	LIFE	\$30.52
	SAVE	\$122.10
	STAT	\$85.47
SMITH	CITY	\$1.43
	FED	\$121.55
	FICA	\$100.10
	HLTH	\$22.75
	LIFE	\$13.65
	SAVE	\$54.60
	STAT	\$20.02
	CITY	\$ .75
	FED	\$64.10
	FICA	\$52.79
	STAT	\$10.56
STEVENS	CITY	\$ .92
	FED	\$77.92
	FICA	\$64.17
	HLTH	\$8.68
	LIFE	\$5.21
	SAVE	\$20.83
	STAT	\$12.83
82/02/26 CROSS	CITY	\$7.52
	FED	\$638.96
	FICA	\$526.20



PAGE	2		
<u>PAY_DATE</u>	<u>LAST_NAME</u>	<u>DED_CODE</u>	<u>DED_AMT</u>
82/02/26	CROSS	HLTH	\$32.22
		LIFE	\$19.33
		SAVE	\$77.32
		STAT	\$105.24
	IRVING	CITY	\$6.10
		FED	\$518.92
		FICA	\$427.35
		HLTH	\$50.87
		LIFE	\$30.52
		SAVE	\$122.10
		STAT	\$85.47
	MCKNIGHT	CITY	\$2.51
		FED	\$213.18
		FICA	\$175.56
		HLTH	\$18.81
		LIFE	\$11.29
		SAVE	\$45.14
		STAT	\$35.11
	SMITH	CITY	\$1.43
		FED	\$121.55
		FICA	\$100.10

## Inserting Page Numbers

By default, the first two lines of a report page are reserved. The first line contains the page number in the top-left corner, and the second line is blank.

**Note:** The features in this section are not supported for Compound Reports.

You can:

- ☐ Change the position of the default page number with the system variable TABPAGENO, which contains the current page number.
- ☐ Insert the total page count using the TABLASTPAGE system variable.
- ☐ Assign any number to the first page using the FOCFIRSTPAGE parameter.
- ☐ Suppress the display of the default page number.

**Note:** The variables TABPAGENO and TABLASTPAGE cannot be used to define styling with conditional styling (WHEN).

If you enable Section 508 accessibility, a default page number is not included in the HTML table.

**Reference: Page Number Commands**

Command	Description	Applies to
<code>&lt;BYLASTPAGE</code>	Used with REPAGE. Inserts the total page count within the sort group that has the REPAGE option.	HTML PDF PS PPTX
<code>REPAGE</code>	Resets page number to one.	HTML PDF PS PPTX
<code>&lt;TABPAGENO</code>	Inserts the current page number. TABPAGENO suppresses the default page number, and the top two lines of a page are blank.	HTML PDF PS PPTX
<code>&lt;TABLASTPAGE</code>	Inserts the total page count in the report.	HTML PDF PS PPTX
<code>SET FOCFIRSTPAGE</code>	Assigns the designated page number to the first page.	HTML PDF PS PPTX

Command	Description	Applies to
SET PAGE-NUM	Controls page number display.	HTML PDF PS

**Syntax:**      **How to Insert the Current Page Number**

To add the current page number, add the following to your request.

```
<TABPAGENO
```

**Example:**      **Inserting the Current Page Number in a Sort Footing**

This request generates a new page whenever the value of the sort field REGION changes. It uses TABPAGENO to insert a page number in the sort footing.

```
TABLE FILE GGSales
SUM BUDDOLLARS
BY REGION BY ST BY CITY
ON REGION PAGE-BREAK SUBFOOT
"Sales Quota for <REGION Cities"
"Page <TABPAGENO"
ON TABLE SET ONLINE-FMT PDF
END
```

The first page of output is:

<u>Region</u>	<u>State</u>	<u>City</u>	<u>Budget Dollars</u>
Midwest	IL	Chicago	3866856
	MO	St. Louis	3646838
	TX	Houston	3680679
Sales Quota for Midwest Cities			
Page	1		

**Inserting the Total Page Count**

You can use the <TABLASTPAGE system variable to insert the total page count into your report. For example, if you wanted to add a footing in your report that said "Page 1 of 5", you could use the <TABLASTPAGE system variable in conjunction with the <TABPAGENO system variable to do so.

**Syntax: How to Insert the Total Page Count**

To insert the total number of pages, add the following to your request:

```
<TABLASTPAGE
```

**Reference: Usage Notes for TABLASTPAGE**

- ❑ TABLASTPAGE does not adjust for changes in FOCFIRSTPAGE or for the REPAGE command. For example, if the report has 10 pages and the user uses FOCFIRSTPAGE to set the first page number to 3 rather than 1, the value of TABLASTPAGE will still be 10.
- ❑ TABLASTPAGE is supported only for a single report, not compound reports. A separate page count is generated for each report in a compound report.
- ❑ TABLASTPAGE is supported only for styled reports such as HTML, PDF, and PS. it is not supported for EXL2K, WP, DOC, or HTML with STYLE=OFF and STYLEMODE=FIXED.
- ❑ TABLASTPAGE causes a second pass through the report results, first to calculate the last page then to print it with TABPAGENO (even when SQUEEZE=OFF).
- ❑ TABLASTPAGE does not support the system (external) sort.
- ❑ GRAPH FILE does not support TABLASTPAGE.
- ❑ TABLEF is not supported with TABLASTPAGE.

**Example: Inserting the Current Page Number and the Total Page Count**

The following illustrates how to add the current page number and the total page count to a report. The relevant syntax is highlighted in the request.

```
TABLE FILE EMPLOYEE
PRINT EMP_ID AS 'Employee ID'
BY SALARY IN-GROUPS-OF 5000 AS 'Salary'
BY PCT_INC AS 'Percent,Increase'
BY DAT_INC AS 'Date of,Increase'
ON SALARY PAGE-BREAK
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=TITLE, STYLE=BOLD, SIZE=11, $
ENDSTYLE
FOOTING
"Page <TABPAGENO of <TABLASTPAGE"
END
```

The first two pages of output are:

<b><u>Salary</u></b>	<b><u>Percent Increase</u></b>	<b><u>Date of Increase</u></b>	<b><u>Employee ID</u></b>
\$5,000.00	.00	82/01/04	119265415
		82/04/01	543729165
	.04	82/06/11	543729165
	.05	82/05/14	119265415

Page 1 of 5

<b><u>Salary</u></b>	<b><u>Percent Increase</u></b>	<b><u>Date of Increase</u></b>	<b><u>Employee ID</u></b>
\$10,000.00	.10	82/01/01	071382660
			112847612
	.12	81/01/01	071382660

Page 2 of 5

## Displaying the Total Page Count Within a Sort Group

The <BYLASTPAGE variable used in a heading or footing displays the number of pages of output within each sort group when a report uses the REPAGE option to reset the page numbers for each sort group. This variable can only be used with styled output formats.

If the REPAGE option is not used in the report, the total number of pages in the report (<TABLASTPAGE variable) is used for <BYLASTPAGE.

### **Syntax:** How to Display the Total Number of Pages Within Each Sort Group

The request must have the following syntax and hold the output in a styled output format:

```
BY sortfield REPAGE
```

The heading or footing can use the following syntax to display “Page x of y”



```
{HEADING|FOOTING}
"Page <TABPAGENO of <BYLASTPAGE"
```

where:

*sortfield*

Is the sort field that has the REPAGE option. A PAGE-BREAK is required on the same sort field or a lower level sort field. PAGE-BREAK starts a new page for each sort break.

REPAGE resets the page number to 1 for each sort break.

<TABPAGENO

Is the current page number.

<BYLASTPAGE

Is the last page number before the repage.

### **Example: Paginating Within a Sort Group**

The following request against the GGSALES data source sorts by product, region, category, and city. It resets the pagination each time the product changes. The heading prints the current page number and the total within each product group.

Note that by default, the TABPAGENO and BYLASTPAGE variables have format I5, which leaves a lot of blank space before the page numbers. Therefore, you can use spot markers or COMPUTE commands to move the page numbers to the left.

In the following example, a COMPUTE command creates a field named X that has the value of TABPAGENO but stores it as an I2 field, and the spot marker in the heading moves the BYLASTPAGE page number four spaces to the left. The heading command must come after the COMPUTE command or the field named X will not be recognized:

```
TABLE FILE GGSALES
SUM UNITS
COMPUTE X/I2 = TABPAGENO;
BY PRODUCT NOPRINT REPAGE
BY REGION PAGE-BREAK
BY CATEGORY
BY CITY
HEADING CENTER
"<PRODUCT : Page <X of <-4> <BYLASTPAGE "
ON TABLE PCHOLD FORMAT PDF
END
```

The following partial output shows that the page number resets to 1 when the product changes and that the BYLASTPAGE variable displays the total number of pages for each product:

Biscotti : Page 1 of 4			
<u>Region</u>	<u>Category</u>	<u>City</u>	<u>Unit Sales</u>
Midwest	Food	Chicago	29413
		Houston	27504
		St. Louis	29188

Biscotti : Page 2 of 4			
<u>Region</u>	<u>Category</u>	<u>City</u>	<u>Unit Sales</u>
Northeast	Food	Boston	47064
		New Haven	46214
		New York	51964

Biscotti : Page 3 of 4			
<u>Region</u>	<u>Category</u>	<u>City</u>	<u>Unit Sales</u>
Southeast	Food	Atlanta	43639
		Memphis	35349
		Orlando	40606

Biscotti : Page 4 of 4			
<u>Region</u>	<u>Category</u>	<u>City</u>	<u>Unit Sales</u>
West	Food	Los Angeles	20773
		San Francisco	22987
		Seattle	26676

Capuccino : Page 1 of 3			
<u>Region</u>	<u>Category</u>	<u>City</u>	<u>Unit Sales</u>
Northeast	Coffee	Boston	15358
		New Haven	12386
		New York	17041

## Assigning Any Page Number to the First Page

You can assign a page number to the first page of a report using the FOCFIRSTPAGE parameter. This feature is useful when a report is printed and assembled as part of another one.

You can also control the page numbering of multiple reports in the same procedure using the FOCFIRSTPAGE parameter with the &FOCNEXTPAGE variable.

If TABPAGENO is used, FOCFIRSTPAGE is ignored.

### **Syntax:** How to Assign a Page Number to the First Page

#### **For all report requests in a procedure**

```
SET FOCFIRSTPAGE = {n|1|&FOCNEXTPAGE}
```

#### **For one report request**

```
ON TABLE SET FOCFIRSTPAGE {n|1|&FOCNEXTPAGE}
```

where:

*n*

Is an integer between 1 and 999999, which is the number assigned to the first page of the report.

1

Assigns the number 1 to the first page. 1 is the default value.

&FOCNEXTPAGE

Is a variable whose value is one more than the last page number of the previous report in a multiple request. The value is calculated at run time.

### **Example:** Assigning a Page Number to the First Page

This request assigns the number 3 to the first page of the report.

```
SET FOCFIRSTPAGE = 3
TABLE FILE CENTORD
HEADING
"Sales By Store"
SUM LINEPRICE AS 'Sales'
BY SNAME
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The report is:

PAGE 3

Sales By Store

<u>Store Name:</u>	<u>Sales</u>
AV VideoTown	\$63,993,429.34
Audio Expert	\$247,093,518.58
City Video	\$15,299,051.29
Consumer Merchandise	\$30,080,503.15
TV City	\$83,041,182.81
Web Sales	\$4,027,582.48
eMart	\$290,487,756.36

**Example: Controlling Page Numbers in Consecutive Reports**

This procedure contains two report requests. The second request sets FOCFIRSTPAGE to the value of &FOCNEXTPAGE.

```
SET FOCFIRSTPAGE = 3
TABLE FILE CENTORD
HEADING
"Sales By Store"
SUM LINEPRICE AS 'Sales'
BY SNAME
WHERE SNAME EQ 'eMart'
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
-RUN
```

```
SET FOCFIRSTPAGE = &FOCNEXTPAGE
TABLE FILE CENTORD
HEADING
"Sales By Product"
SUM LINEPRICE AS 'Sales'
BY PRODCAT AS 'Product'
WHERE PRODCAT EQ 'VCRs'
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The first page of the second report is numbered 4, which is one more than the last page of the previous report:

PAGE 3

Sales By Store

<u>Store Name:</u>	<u>Sales</u>
eMart	\$290,487,756.36

PAGE 4

Sales By Product

<u>Product</u>	<u>Sales</u>
VCRs	\$23,801,009.34

## Controlling the Display of Page Numbers

By default, the first two lines of a report page are reserved. The first line displays the page number in the top-left corner, and the second line is blank. To suppress the default display, use the PAGE-NUM parameter.

### **Syntax:** How to Control the Display of Page Numbers

#### **For all report requests in a procedure**

```
SET PAGE[-NUM] = num_display
```

#### **For one report request**

```
ON TABLE SET PAGE[-NUM] num_display
```

where:

**-NUM**

Is optional. PAGE and PAGE-NUM are synonymous.

*num\_display*

Is one of the following:

**ON** displays page numbers in the top-left corner, followed by a reserved blank line. ON is the default value.

**OFF** suppresses default page numbers.

You can use the system variable TABPAGENO.

**NOPAGE** suppresses default page numbers and makes the top two lines of a page available for your use.

You can use the system variable TABPAGENO.

**TOP** or **NOLEAD** removes the line at the top of each page reserved for the page number, and the blank line after it. The first line of a report contains the report or page heading if specified, or column titles if there is no heading.

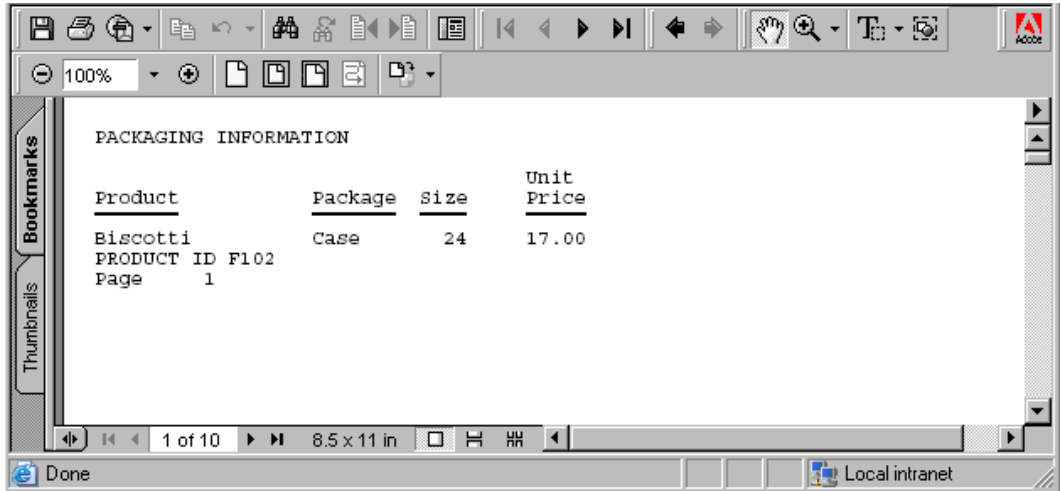
You can use the system variable TABPAGENO to show page numbers elsewhere in the report.

### ***Example:*** Suppressing Default Page Numbers

This request uses SET PAGE-NUM = NOPAGE to suppress default page numbers. It uses the top line of the first page of the report for the report heading.

```
SET PAGE-NUM = NOPAGE
TABLE FILE GGPRODS
ON TABLE SUBHEAD
"PACKAGING INFORMATION"
" "
PRINT PACKAGE_TYPE AND SIZE AND UNIT_PRICE
BY PRODUCT_DESCRIPTION
ON PRODUCT_DESCRIPTION PAGE-BREAK SUBFOOT
"PRODUCT ID <PRODUCT_ID"
"Page <TABPAGENO "
ON TABLE SET ONLINE-FMT PDF
END
```

TABPAGEÑO inserts the page number in the sort footing. The first page of the report is:



## Setting the Number of Data Rows For Each Page in an AHTML Report Request

You can use the `LINES-PER-PAGE` StyleSheet attribute in an AHTML report request to set the number of data rows to display on each page in the report output.

### **Syntax:** How to Set the Number of Data Rows For Each Page in an AHTML Report Request

To control the number of data rows to display on each page in the report output, use the following StyleSheet syntax:

```
TYPE=REPORT, LINES-PER-PAGE={n|UNLIMITED},$
```

where:

*n*

Specifies the number of data rows to display on each page. The default value is 57 rows.

UNLIMITED

Specifies that you want to show all the results on one page.

#### **Note:**

- ❑ In an AHTML report request, using the following SET command, you will see the same number of data rows as the `LINES-PER-PAGE` StyleSheet option:

```
ON TABLE SET LINES {n|UNLIMITED}
```

- ❑ In an HTML report request, using either the LINES-PER-PAGE StyleSheet option or the SET LINES command, you will see the number of lines, as opposed to the number of data rows.

### ***Example:*** Setting the Number of Data Rows For Each Report Page

The following example uses the default WebFOCUS StyleSheet and displays 20 data rows on each page of the report output.

```
TABLE FILE GGSALES
HEADING
"Sales Report"
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT BY DATE NOPRINT
WHERE DATE GE 19960101 AND DATE LE 19960401
ON TABLE PCHOLD FORMAT AHTML
ON TABLE SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, GRID=OFF, LINES-PER-PAGE = 20, $
ENDSTYLE
END
```



The following image shows the output for the first page.

Sales Report			
Category ▼	Product ▼	Unit Sales ▼	Dollar Sales ▼
Coffee	Capuccino	5507	71728
		5219	67429
		6343	87290
		4170	58692
	Espresso	14499	182900
		14015	169023
		14871	184474
		11613	156396
	Latte	31469	401873
		38725	490645
Food	Biscotti	42717	520948
		36374	468137
		18993	236698
		18816	241975
	Croissant	15602	197254
		17601	218437
		22537	274926
		29691	361053
		30954	388868
		27412	344197

40 of 40 records, Page 1 of 2 ▶▶▶

The following image shows the output for the second page.

Sales Report

Category ▼	Product ▼	Unit Sales ▼	Dollar Sales ▼
Food	Scone	15717	205213
		10904	146866
		15249	190702
		11663	142739
Gifts	Coffee Grinder	7954	100055
		8398	98255
		8027	106492
		7971	106131
	Coffee Pot	7507	98807
		8588	112339
		8531	107154
		7849	101550
	Mug	17199	217796
		16172	204380
		16328	204106
		15037	194585
	Thermos	8603	110445
		7102	90235
		7193	88375
		9612	113911

◀◀|◀ 40 of 40 records, Page 2 of 2

## Adding Grids and Borders

By default, an HTML report contains horizontal and vertical grid lines. You can remove the grid lines or adjust their use on a horizontal (BY) sort field. Grid characteristics apply to an entire HTML report, not to individual components of a report.

You can emphasize headings, footings, and column titles in a report by adding borders and grid lines around them.

**Borders:** In a PDF, HTML, DHTML, XLSX, EXL2K, PPTX, PPT, or PS report, you can use BORDER attributes in a StyleSheet to specify the weight, style, and color of border lines. If you wish, you can specify formatting variations for the top, bottom, left, and right borders.

For an example, see [Inserting and Formatting a Border](#) on page 1323.

The BORDERALL StyleSheet attribute supports a heading or footing grid with borders around each individual heading or footing cell in PDF, DHTML, and PPTX report output. Using this attribute along with BORDER attributes for individual objects in a heading or footing enables you to create borders around individual items.

Currently, with SQUEEZE=ON, the right margin border for subheadings and subfootings is defined based on the maximum width of all heading, footing, subheading, and subfooting lines. The length of subheading and subfooting lines is tied to the lengths of the page heading and page footing, not to the size of the data columns in the body of the report. You can use the ALIGN-BORDERS=BODY attribute in a StyleSheet to align the subheadings and subfootings with the report body on PDF report output instead of the other heading elements.

**Grids:** In an HTML report, you can use the GRID attribute in a StyleSheet to turn grid lines on and off for the entire report. When used in conjunction with internal cascading style sheets, GRID produces a thin grid line rather than a thick double line (the HTML default). In PDF reports you can use the HGRID and VGRID attributes to add horizontal or vertical grid lines and adjust their density.

**Note:** The SET GRID parameter, which applies to graphs, is not the same as the GRID StyleSheet attribute.

**Reference:** Grid Display Attributes

Attribute	Description	Applies to
GRID	Controls grid display.	HTML PDF PS DHTML PPTX PPT

Attribute	Description	Applies to
HGRID	Controls horizontal grid display and grid line density.	PDF PS DHTML PPTX PPT
VGRID	Control vertical grid display and grid line density.	PDF PS DHTML PPTX PPT

**Note:** When viewing PDF reports with the Adobe Reader, GRID lines may appear thinner than specified if the view is not set to 100%. For example, if you view the document at the 50% setting, some GRID lines may be thinner than others.

**Syntax:**      **How to Control Grid Display in HTML Reports**

[TYPE=REPORT,] GRID= *option*, \$

where:

TYPE=REPORT

Applies the grid to the entire report. Not required, as it is the default.

*option*

Is one of the following:

- [ON](#) applies a grid to a report. Does not apply grid lines to cells underneath a BY field value until the value changes. Column titles are not underlined. ON is the default value.
- [OFF](#) disables the default grid. Column titles are underlined. You can include blank lines and underlines. You cannot wrap cell data. With this setting, a report may be harder to read.
- [FILL](#) applies grid lines to all cells of a report. Column titles are not underlined.

**Syntax:**      **How to Add and Format Borders**

To request a uniform border, use this syntax:

```
TYPE=type, BORDER=option, [BORDER-STYLE=line_style,]
  [BORDER-COLOR={color|RGB(r g b)},] $
```

To specify different characteristics for the top, bottom, left, and/or right borders, use this syntax:

```
TYPE=type, BORDER-position=option,
  [BORDER[-position]-STYLE=line_style,]
  [BORDER[-position]-COLOR={color|RGB(r g b)},] $
```

where:

*type*

Identifies the report component to which borders are applied. See [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167 for valid values.

*option*

Can be one of the following values:

**ON** turns borders on. ON generates the same line as MEDIUM.

**Note:** The MEDIUM line setting ensures consistency with lines created with GRID attributes.

**OFF** turns borders off. OFF is the default value.

**LIGHT** specifies a thin line.

**MEDIUM** identifies a medium line. ON sets the line to MEDIUM.

**HEAVY** identifies a thick line.

*width* specifies the line width in points, where 72 pts=1 inch. Note that this option is not supported with Excel 2003, which does not have an option for specifying a number to precisely set the border width (thickness) in points.

**Tip:** Line width specified in points is displayed differently in HTML and PDF output. For uniform appearance, regardless of display format, use LIGHT, MEDIUM, or HEAVY.

*position*

Specifies which border line to format. Valid values are: TOP, BOTTOM, LEFT, RIGHT.

You can specify a position qualifier for any of the BORDER attributes. This enables you to format line width, line style, and line color individually, for any side of the border.

*line\_style*

Sets the style of the border line. WebFOCUS StyleSheets support all of the standard cascading style sheet line styles. Several 3-dimensional styles are available only in HTML, as noted by asterisks. Valid values are:

Style	Description
NONE	No border is drawn.
SOLID	Solid line.
DOTTED	Dotted line.
DASHED	Dashed line.
DOUBLE	Double line.
GROOVE*	3D groove. (Not supported with Excel 2003, which has no option for specifying this type of border.)
RIDGE*	3D ridge. (Not supported with Excel 2003, which has no option for specifying this type of border.)
INSET*	3D inset.
OUTSET*	3D outset.

*color*

Is one of the preset color values. The default value is BLACK.

If the display or output device does not support colors, it substitutes shades of gray. For a complete list of available color values, see [Formatting Report Data](#) on page 1611.

RGB

Specifies the font color using a mixture of red, green, and blue.

*( r g b )*

Is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense. Using the three color components in equal intensities results in shades of gray.

**Note:** Format EXL2K does not support the GRID=ON parameter.

**Example:** **Inserting and Formatting a Border**

This request generates an HTML report with a heavy red dotted line around the entire report heading.

```
TABLE FILE GGSales
SUM  BUdUNITS UNITS BUdDOLLARS DOLLARS
BY  CATEGORY
ON  TABLE SUBHEAD
    "</1 Sales Report"
    "***CONFIDENTIAL***"
    "December 2002 </1"
ON  TABLE SET PAGE-NUM OFF
ON  TABLE SET ONLINE-FMT HTML
ON  TABLE SET HTMLCSS ON
ON  TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
TYPE=TABLEHEADING, STYLE=BOLD, JUSTIFY=CENTER, BORDER=HEAVY,
BORDER-COLOR=RED, BORDER-STYLE=DOTTED, $
ENDSTYLE
END
```

The output is:

<div style="border: 2px dotted red; padding: 10px; text-align: center;"> <p><b>Sales Report</b>  <b>**CONFIDENTIAL**</b>  <b>December 2002</b></p> </div>				
---	--	--	--	--

<u>Category</u>	<u>Budget Units</u>	<u>Unit Sales</u>	<u>Budget Dollars</u>	<u>Dollar Sales</u>
Coffee	1385923	1376266	17293886	17231455
Food	1377564	1384845	17267160	17229333
Gifts	931007	927880	11659732	11695502

**Tip:** You can use the same BORDER syntax to generate this output in a PDF or PS report.

**Example: Displaying the Default Grid on an HTML Report**

This request uses the default setting GRID=ON.

```
TABLE FILE GGSales
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT
ON TABLE SET PAGE-NUM OFF
END
```

The cells underneath the sort field CATEGORY do not have grid lines until the value changes (for example, from Coffee to Food):

Category	Product	Unit Sales	Dollar Sales
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

**Example: Applying Grid Lines to All Cells of an HTML Report**

This request uses GRID=FILL to apply grid lines to all cells, including those underneath the sort field CATEGORY.

```
TABLE FILE GGSales
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=FILL, $
ENDSTYLE
END
```



All cells have grid lines:

Category	Product	Unit Sales	Dollar Sales
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

**Example: Removing a Grid From an HTML Report**

This request uses GRID=OFF to remove the default grid from a report.

```
TABLE FILE GGSales
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

Column titles are underlined:

<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

**Reference: Adding Borders to Excel Report Output**

Adding borders to Excel report output also may add blank rows.

The presence of any BORDER syntax (even BORDERS=OFF), whether it is in the inline procedure or in the referenced StyleSheet, places the procedure in BORDER mode where the styling and spacing is adjusted and may result in extra blank rows.

**Example: Adding Borders to FORMAT XLSX Report Output**

The following request turns borders on and generates FORMAT XLSX report output.

```
TABLE FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
HEADING
"This is the heading"
ON TABLE PCHOLD FORMAT XLSX
ON TABLE NOTOTAL
ON TABLE SET STYLE *
TYPE=REPORT,BORDER=ON,$
ENDSTYLE
END
```

The output is shown in the following image. A blank row has been added after the heading.

	A	B	C
1	This is the heading		
2			
3	Product Category	Customer Business Region	Cost of Goods
4	Accessories	EMEA	\$143,987.00
5		North America	\$171,306.00
6		Oceania	\$1,861.00
7		South America	\$25,723.00
8	Camcorder	EMEA	\$187,000.00
9		North America	\$242,967.00
10		Oceania	\$552.00
11		South America	\$22,686.00
12	Computers	EMEA	\$47,616.00
13		North America	\$53,329.00
14		Oceania	\$491.00
15		South America	\$7,845.00
16	Media Player	EMEA	\$328,401.00
17		North America	\$386,681.00
18		Oceania	\$4,328.00
19		South America	\$60,183.00
20	Stereo Systems	EMEA	\$361,756.00
21		North America	\$412,036.00
22		Oceania	\$1,427.00
23		South America	\$81,823.00
24	Televisions	EMEA	\$100,443.00
25		North America	\$101,212.00
26		Oceania	\$365.00
27		South America	\$25,800.00
28	Video Production	EMEA	\$78,722.00
29		North America	\$89,489.00
30		Oceania	\$589.00
31		South America	\$11,740.00

The following version of the request has no border attributes.

```
TABLE FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
HEADING
"This is the heading"
ON TABLE PCHOLD FORMAT XLSX
ON TABLE NOTOTAL
END
```

The output is shown in the following image. There is no blank row between the report heading and report body.

	A	B	C
1	This is the heading		
2	Product Category	Customer Business Region	Cost of Goods
3	Accessories	EMEA	\$143,987.00
4		North America	\$171,306.00
5		Oceania	\$1,861.00
6		South America	\$25,723.00
7	Camcorder	EMEA	\$187,000.00
8		North America	\$242,967.00
9		Oceania	\$552.00
10		South America	\$22,686.00
11	Computers	EMEA	\$47,616.00
12		North America	\$53,329.00
13		Oceania	\$491.00
14		South America	\$7,845.00
15	Media Player	EMEA	\$328,401.00
16		North America	\$386,681.00
17		Oceania	\$4,328.00
18		South America	\$60,183.00
19	Stereo Systems	EMEA	\$361,756.00
20		North America	\$412,036.00
21		Oceania	\$1,427.00
22		South America	\$81,823.00
23	Televisions	EMEA	\$100,443.00
24		North America	\$101,212.00
25		Oceania	\$365.00
26		South America	\$25,800.00
27	Video Production	EMEA	\$78,722.00
28		North America	\$89,489.00
29		Oceania	\$589.00
30		South America	\$11,740.00

**Syntax:**      **How to Insert Inner and Outer Borders Within Headings or Footings**

BORDERALL is the quickest way to add borders to the entire heading grid. This feature is supported in PDF, DHTML, and PPTX formats. Individual borders can be removed by explicitly turning the border off in individual items using BORDER, BORDER-LEFT, BORDER-RIGHT, BORDER-TOP, and BORDER-BOTTOM. For a given item that is bordered by BORDERALL, BORDER-LEFT=OFF presents the item with no left border, but the defined border style is retained for top, bottom, and right borders.

Three levels of borders for headings and footings are supported:

1. Individual cell borders.

BORDER-LEFT, BORDER-RIGHT, BORDER-TOP, and BORDER-BOTTOM can be used to set the individual components of the external border of the heading or a selected item or cell.

2. All outer borders.

BORDER= is used to set the external borders within a heading or footing.

3. All outer and internal borders.

BORDERALL is used to apply border characteristics to both the internal and external borders of the selected heading or footing.

**Note:** BORDERALL applies to the entire heading or footing element. It cannot be used for individual lines or items within a heading or footing element.

To turn on all external and internal borders (a border grid):

```
TYPE=headfoot, BORDERALL=option, [BORDER-STYLE=line_style,] [BORDER-COLOR={color|RGB(r g b)},] $
```

where:

*headfoot*

Is the type of heading or footing. Valid values are TABHEADING, TABFOOTING, HEADING, FOOTING, SUBHEAD, and SUBFOOT.

**Note:** BORDERALL applies to the entire heading or footing element. It cannot be used for individual lines or items within a heading or footing element.

*option*

Can be one of the following values:

- ☐ **ON** turns borders on. ON generates the same line as MEDIUM.

**Note:** The MEDIUM line setting ensures consistency with lines created with GRID attributes.

- ☐ **OFF** turns borders off. OFF is the default value.
- ☐ **LIGHT** specifies a thin line.
- ☐ **MEDIUM** identifies a medium line. ON sets the line to MEDIUM.
- ☐ **HEAVY** identifies a thick line.
- ☐ Entering a numeric value specifies the line width in points, where 72 pts=1 inch. Note that this option is not supported with Excel 2003, which does not have an option for specifying a number to precisely set the border width (thickness) in points.

**Tip:** Line width specified in points is displayed differently in HTML and PDF output. For uniform appearance, regardless of display format, use LIGHT, MEDIUM, or HEAVY.

To request a uniform border, use this syntax:

```
TYPE=headfoot, BORDER=option
```

To specify different characteristics for the top, bottom, left, and/or right borders, use this syntax:

```
TYPE=headfoot, BORDER-position=option,  
[BORDER[-position]-STYLE=line_style,]  
[BORDER[-position]-COLOR={color|RGB(r g b)},] $
```

where:

*headfoot*

Identifies the heading, footing, subheading, or subfooting to which borders are applied.

*position*

Specifies which border line to format. Valid values are: TOP, BOTTOM, LEFT, RIGHT.

You can specify a position qualifier for any of the BORDER attributes. This enables you to format line width, line style, and line color individually, for any side of the border.

*option*

Can be one of the following values:

- ☐ **ON** turns borders on. ON generates the same line as MEDIUM.

**Note:** The MEDIUM line setting ensures consistency with lines created with GRID attributes.

- ☐ **OFF** turns borders off. OFF is the default value.
- ☐ **LIGHT** specifies a thin line.

- ☐ **MEDIUM** identifies a medium line. ON sets the line to MEDIUM.
  - ☐ **HEAVY** identifies a thick line.
  - ☐ Entering a numeric value specifies the line width in points, where 72 pts=1 inch. Note that this option is not supported with Excel 2003, which does not have an option for specifying a number to precisely set the border width (thickness) in points.
- Tip:** Line width specified in points is displayed differently in HTML and PDF output. For uniform appearance, regardless of display format, use LIGHT, MEDIUM, or HEAVY.

*line\_style*

Sets the style of the border line. WebFOCUS StyleSheets support all of the standard cascading style sheet line styles. Several three-dimensional styles are available only in HTML, as noted by asterisks. Valid values are:

Style	Description
NONE	No border is drawn.
SOLID	Solid line.
DOTTED	Dotted line.
DASHED	Dashed line.
DOUBLE	Double line.
GROOVE*	3D groove. (Not supported with Excel 2003, which has no option for specifying this type of border.)
RIDGE*	3D ridge. (Not supported with Excel 2003, which has no option for specifying this type of border.)
INSET*	3D inset.
OUTSET*	3D outset.

**Note:** All line types supported for PDF, DHTML, and PPTX can be used for individual internal borders with HEADALIGN=BODY.



*color*

Is one of the preset color values. The default value is BLACK.

If the display or output device does not support colors, it substitutes shades of gray. For a complete list of available color values, see [Color Values in a Report](#) on page 1615.

*RGB*

Specifies the font color using a mixture of red, green, and blue.

*( r g b )*

Is the desired intensity of red, green, and blue, respectively, separated by spaces. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense. Using the three color components in equal intensities results in shades of gray.

### **Example: Controlling Borders Within Heading and Footing Elements in PDF Report Output**

The following request against the EMPLOYEE data source has a page heading, a subheading, a subfooting, and a report footing:

```
TABLE FILE EMPLOYEE
HEADING
" Department Report Page <TABPAGENO "
PRINT LAST_NAME AS ''
FIRST_NAME AS ''
CURR_SAL AS ''
CURR_JOBCODE AS ''
BY DEPARTMENT AS ''
WHERE CURR_SAL NE 0.0
ON TABLE PCHOLD FORMAT PDF
ON DEPARTMENT SUBFOOT
" "
"Subtotal:<ST.CURR_SAL"
" "
ON DEPARTMENT SUBHEAD
"Department <+0>Last Name <+0>First Name <+0>Salary<+0>Jobcode <+0>"
ON TABLE SUBFOOT
"Grand Total:<ST.CURR_SAL"
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE=REPORT, FONT=ARIAL, BORDER=ON, SQUEEZE=ON, $
TYPE=REPORT, COLUMN=CURR_JOBCODE,SQUEEZE=.75,$
TYPE = SUBHEAD, HEADALIGN=BODY, BORDERALL=ON,$
TYPE = SUBFOOT, HEADALIGN=BODY,$
TYPE = SUBFOOT, LINE=2, ITEM=1, COLSPAN=3, JUSTIFY=RIGHT,$
TYPE = SUBFOOT, LINE=2, ITEM=2, JUSTIFY=RIGHT,$
TYPE = TABFOOTING, HEADALIGN=BODY,$
TYPE = TABFOOTING, ITEM=1, COLSPAN=3, JUSTIFY=RIGHT,$
TYPE = TABFOOTING, ITEM=2, JUSTIFY=RIGHT,$
ENDSTYLE
END
```

The REPORT component has BORDER=ON, so the page heading has an external border.

The subheading has BORDERALL=ON and HEADALIGN=BODY, so the subheading grid aligns with the body grid, and each item within the subhead is presented as fully bordered individual cells.

The StyleSheet aligns the subfooting elements with the body of the report, and has the salary subtotal on the second line aligned and justified with the CURR\_SAL column.

The table footing has a border around the entire footing because the REPORT component specifies BORDER=ON. The grand total is aligned and justified with the CURR\_SAL column on the report.

The output is:

Department Report Page 1				
Department	Last Name	First Name	Salary	Jobcode
MIS	SMITH	MARY	\$13,200.00	B14
	JONES	DIANE	\$18,480.00	B03
	MCCOY	JOHN	\$18,480.00	B02
	BLACKWOOD	ROSEMARIE	\$21,780.00	B04
	GREENSPAN	MARY	\$9,000.00	A07
	CROSS	BARBARA	\$27,062.00	A17
Subtotal:			\$108,002.00	
Department	Last Name	First Name	Salary	Jobcode
PRODUCTION	STEVENS	ALFRED	\$11,000.00	A07
	SMITH	RICHARD	\$9,500.00	A01
	BANNING	JOHN	\$29,700.00	A17
	IRVING	JOAN	\$26,862.00	A15
	ROMANS	ANTHONY	\$21,120.00	B04
	MCKNIGHT	ROGER	\$16,100.00	B02
Subtotal:			\$114,282.00	
Grand Total:			\$222,284.00	

### **Syntax:** How to Align Subheading and Subfooting Margins With the Report Body

Currently, with SQUEEZE=ON, the right margin border for subheadings and subfootings is defined based on the maximum width of all heading, footing, subheading, and subfooting lines. The length of subheading and subfooting lines is tied to the lengths of the page heading and page footing, not to the size of the data columns in the body of the report.

You can use the ALIGN-BORDERS=BODY attribute in a StyleSheet to align the subheadings and subfootings with the report body on PDF report output instead of the other heading elements.

You can align subheading and subfooting margins with the report body either by adding the ALIGN-BORDERS=BODY attribute to the StyleSheet declaration for the REPORT component, or placing it in its own declaration without a TYPE attribute.

```
[TYPE=REPORT, ] ALIGN-BORDERS={OFF|BODY} , $
```

where:

OFF

Does not align the right margin of subheadings and subfootings with the report body.

BODY

Specifies that the width of subheading and subfooting lines is independent of heading, footing, tabheading, and tabfooting lines, and that the right border of the report body will be aligned by either extending subheading and subfooting lines (if they are narrower than the data columns) or extending the data columns (if the data columns are narrower than the maximum width of subheadings and subfootings).

***Reference:* Considerations for Aligning Subheading and Subfooting Margins With the Report Body**

Without the ALIGN-BORDERS=BODY attribute, the width of the subheading and subfooting lines is determined by the largest width of all of the headings and footings (report, page, subheadings, and subfootings).

The following image illustrates report output without the ALIGN-BORDERS=BODY attribute.

Report Heading

TYPE=REPORT, ALIGN-BORDERS=OFF, BORDER=ON, \$

Page Heading

Information Builders

The Standard for Enterprise Business Intelligence

			Product Sales		
			Coffee / Capuccino	Coffee / Espresso	Coffee / Latte
Region	State	City			
Subheading Region Midwest					
Midwest	IL	Chicago	.	\$420,439	\$978,340
	MO	St. Louis	.	\$419,143	\$986,981
	TX	Houston	.	\$455,385	\$938,245
*TOTAL Midwest			\$0	\$1,294,947	\$2,883,566
Subheading Region Northeast					
Northeast	CT	New Haven	\$158,995	\$279,373	\$926,052
	MA	Boston	\$174,344	\$248,358	\$917,737
	NY	New York	\$208,756	\$322,378	\$928,028
*TOTAL Northeast			\$542,095	\$850,107	\$2,771,815
Subheading Region Southeast					
Southeast	FL	Orlando	\$317,027	\$256,539	\$889,887
	GA	Atlanta	\$352,161	\$317,389	\$907,385
	TN	Memphis	\$274,812	\$279,644	\$820,584
*TOTAL Southeast			\$944,000	\$853,572	\$2,617,836
Subheading Region West					
West	CA	Los Angeles	\$306,468	\$267,809	\$809,647
		San Francisco	\$279,830	\$338,270	\$935,862
	WA	Seattle	\$309,197	\$301,538	\$924,896
*TOTAL West			\$895,495	\$907,617	\$2,670,405
TOTAL			\$2,381,590	\$3,906,243	\$10,943,622

Page Footing

Page 1


Report Footing

When the body lines are wider than the subheading and subfooting lines, the border and backcolor of the subheading and subfooting lines are expanded to match the width of the data lines, as shown on the following report output.

Report Heading

TYPE=REPORT, ALIGN=BORDERS=BODY, BORDER=ON, \$

Page Heading



			Product Sales		
			Coffee / Capuccino	Coffee / Espresso	Coffee / Latte
Region	State	City			
Subheading Region Midwest			- Body is wider than Subheading		
Midwest	IL	Chicago	.	\$420,439	\$978,340
	MO	St. Louis	.	\$419,143	\$966,981
	TX	Houston	.	\$455,365	\$938,245
*TOTAL Midwest			\$0	\$1,294,947	\$2,883,566
Subheading Region Northeast			- Body is wider than Subheading		
Northeast	CT	New Haven	\$158,995	\$279,373	\$926,052
	MA	Boston	\$174,344	\$248,356	\$917,737
	NY	New York	\$208,756	\$322,378	\$928,026
*TOTAL Northeast			\$542,095	\$850,107	\$2,771,815
Subheading Region Southeast			- Body is wider than Subheading		
Southeast	FL	Orlando	\$317,027	\$256,539	\$889,887
	GA	Atlanta	\$352,161	\$317,389	\$907,365
	TN	Memphis	\$274,812	\$279,644	\$820,584
*TOTAL Southeast			\$944,000	\$853,572	\$2,617,836
Subheading Region West			- Body is wider than Subheading		
West	CA	Los Angeles	\$306,468	\$267,809	\$809,647
		San Francisco	\$279,830	\$338,270	\$935,862
	WA	Seattle	\$309,197	\$301,538	\$924,896
*TOTAL West			\$895,495	\$907,617	\$2,670,405
TOTAL			\$2,381,590	\$3,906,243	\$10,943,622

Page Footing

Page 1


Report Footing

If the subheading and subfooting lines are longer than the body lines, an additional filler cell is added to each data line to allow the defined borders and bgcolor to fill the width defined by the subheading and subfooting lines, as shown on the following report output.

Report Heading

TYPE=REPORT, ALIGN-BORDERS=BODY, BORDER=ON, \$

Page Heading


**Information  
Builders**

The Standard for  
Enterprise  
Business  
Intelligence

Product Sales

Coffee /  
Capuccino

Coffee /  
Espresso

Coffee /  
Latte

Region State City

Subheading Region Midwest - This Subheading is wider than Body

Midwest	IL	Chicago	.	\$420,439	\$978,340	
	MO	St. Louis	.	\$419,143	\$966,981	
	TX	Houston	.	\$455,365	\$938,245	

*TOTAL Midwest			\$0	\$1,294,947	\$2,883,566	
----------------	--	--	-----	-------------	-------------	--

Subheading Region Northeast - This Subheading is wider than Body

Northeast	CT	New Haven	\$158,995	\$279,373	\$926,052	
	MA	Boston	\$174,344	\$248,356	\$917,737	
	NY	New York	\$208,756	\$322,378	\$928,026	

*TOTAL Northeast			\$542,095	\$850,107	\$2,771,815	
------------------	--	--	-----------	-----------	-------------	--

Subheading Region Southeast - This Subheading is wider than Body

Southeast	FL	Orlando	\$317,027	\$256,539	\$889,887	
	GA	Atlanta	\$352,161	\$317,389	\$907,365	
	TN	Memphis	\$274,812	\$279,644	\$820,584	

*TOTAL Southeast			\$944,000	\$853,572	\$2,617,836	
------------------	--	--	-----------	-----------	-------------	--

Subheading Region West - This Subheading is wider than Body

West	CA	Los Angeles	\$306,468	\$267,809	\$809,647	
		San Francisco	\$279,830	\$338,270	\$935,862	
	WA	Seattle	\$309,197	\$301,538	\$924,896	

*TOTAL West			\$895,495	\$907,617	\$2,670,405	
-------------	--	--	-----------	-----------	-------------	--

TOTAL			\$2,381,590	\$3,906,243	\$10,943,622	
-------	--	--	-------------	-------------	--------------	--

Page Footing

Page 1

Report Footing

ALIGN-BORDERS=BODY has been designed to work on:

- ☐ Single panel reports (reports that do not panel horizontally).
- ☐ Paneled reports where HEADPANEL has been turned on for all of the subheadings and subfootings defined in the report.

Setting HEADPANEL ON causes the headings and footings from the first page of a Paneled report to replicate on the subsequent panels. If HEADPANEL is not used, content can be placed in the Paneled headings by explicitly positioning items within the headings using the StyleSheet attribute POSITION. In these situations, ALIGN-BORDERS=BODY is ignored.

Therefore, if HEADPANEL is turned on at the REPORT level and not explicitly turned off for any of the individual subheadings or subfootings, or if it is explicitly turned on for all subheadings and subfootings, ALIGN-BORDERS=BODY will align the borders of all subheadings and subfootings to the data. Otherwise, the borders will continue to exhibit the default behavior of aligning with the page headings and footings.

**Example: Aligning Subheading and Subfooting Margins in a Single Panel PDF Report**

The following request against the GGSALES data source has a report heading, report footing, page heading, page footing, and a subheading for each region. The margins of the subheadings and subfootings are not aligned (ALIGN-BORDERS=OFF , \$):

```

DEFINE FILE GGSALES
SHOWCATPROD/A30 = CATEGORY || ( ' / ' | PRODUCT);
END
TABLE FILE GGSALES
SUM
    DOLLARS/I8M AS ''
BY REGION
BY ST
BY CITY
ACROSS SHOWCATPROD AS 'Product Sales'

ON REGION SUBHEAD
" "
"Subheading <+0>Region <REGION<+0> "
" "
ON REGION SUBTOTAL AS '*TOTAL'
ON TABLE SUBHEAD
"Report Heading"
" "
"TYPE=REPORT, ALIGN-BORDERS=OFF, BORDER=ON, $"
HEADING
"Page Heading "
" "
" "
" "
FOOTING
" "
"Page Footing<+0>Page <TABPAGE NO "
ON TABLE SUBFOOT
" "
"Report Footing"

```



```

WHERE CATEGORY EQ 'Coffee';
ON TABLE SET PAGE-NUM OFF
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
    TYPE=REPORT,
    FONT='ARIAL',
    SIZE=9,
    LEFTMARGIN=.75,
    RIGHTMARGIN=.5,
    TOPMARGIN=.1,
    BOTTOMMARGIN=.1,
    ALIGN-BORDERS=OFF,
    BORDER=ON,
    SQUEEZE=ON,$
$
TYPE=TITLE,
    STYLE=BOLD,
$
TYPE=TABHEADING,
    SIZE=12,
    STYLE=BOLD,
$
TYPE=TABHEADING,
    LINE=3,
    JUSTIFY=CENTER,
$
TYPE=TABFOOTING,
    SIZE=12,
    STYLE=BOLD,
$
TYPE=HEADING,
    SIZE=12,
    STYLE=BOLD,
$
TYPE=HEADING,
    IMAGE=smplogol.gif,
    POSITION=(+4.6000000 +0.03000000),
    JUSTIFY=RIGHT,
$
TYPE=FOOTING,
    SIZE=12,
    STYLE=BOLD,
$
TYPE=FOOTING,
    LINE=2,
    ITEM=2,
    OBJECT=TEXT,
    POSITION=6.3,
    SIZE=12,
    STYLE=BOLD,

```


```
$
TYPE=SUBHEAD,
    SIZE=10,
    STYLE=BOLD,
$
TYPE=SUBHEAD,
    LINE=2,
    ITEM=3,
    OBJECT=TEXT,
    POSITION=2.5,
$
TYPE=SUBFOOT,
    SIZE=10,
    STYLE=BOLD,
$
TYPE=SUBTOTAL,
    BACKCOLOR=RGB(210 210 210),
$
TYPE=ACROSSVALUE,
    SIZE=9,
    WRAP=ON,
$
TYPE=ACROSSTITLE,
    STYLE=BOLD,
$
TYPE=GRANDTOTAL,
    BACKCOLOR=RGB(210 210 210),
    STYLE=BOLD,
$
ENDSTYLE
END
```

The output shows that the subheading margins align with the heading, not with the report body.

Report Heading

TYPE=REPORT, ALIGN-BORDERS=OFF, BORDER=ON, \$

Page Heading



			Product Sales		
			Coffee / Capuccino	Coffee / Espresso	Coffee / Latte
Region	State	City			
Subheading Region Midwest					
Midwest	IL	Chicago	.	\$420,439	\$978,340
	MO	St. Louis	.	\$419,143	\$966,981
	TX	Houston	.	\$455,365	\$938,245
*TOTAL Midwest			\$0	\$1,294,947	\$2,883,568
Subheading Region Northeast					
Northeast	CT	New Haven	\$158,995	\$279,373	\$926,052
	MA	Boston	\$174,344	\$248,356	\$917,737
	NY	New York	\$208,756	\$322,378	\$928,026
*TOTAL Northeast			\$542,095	\$850,107	\$2,771,815
Subheading Region Southeast					
Southeast	FL	Orlando	\$317,027	\$256,539	\$889,887
	GA	Atlanta	\$352,161	\$317,389	\$907,385
	TN	Memphis	\$274,812	\$279,644	\$820,584
*TOTAL Southeast			\$944,000	\$853,572	\$2,617,836
Subheading Region West					
West	CA	Los Angeles	\$306,468	\$267,809	\$809,647
		San Francisco	\$279,830	\$338,270	\$935,862
	WA	Seattle	\$309,197	\$301,538	\$924,896
*TOTAL West			\$895,495	\$907,617	\$2,670,405
TOTAL			\$2,381,590	\$3,906,243	\$10,943,622

Page Footing

Page 1


Report Footing

Now change the ALIGN-BORDERS attribute to ALIGN-BORDERS=BODY and rerun the request. The subheadings now align with the report body, as shown in the following image.

Report Heading

TYPE=REPORT, ALIGN=BORDERS=BODY, BORDER=ON, S

Page Heading



The Standard for  
 Information  
 Builders

			Product Sales	
			Column 1 Calculator	Column 2 Expenses
Region	State	City	Column 3 Labs	
Subheading Region Midwest				
Midwest	IL	Chicago	\$120,420	\$175,540
	MO	St. Louis	\$470,543	\$556,881
	IN	Indianapolis	\$463,987	\$634,247
TOTAL Midwest			\$1,054,950	\$1,366,668
Subheading Region Northeast				
Northeast	CT	New Haven	\$753,805	\$270,875
	MA	Boston	\$774,544	\$243,850
	NY	New York	\$209,555	\$121,870
TOTAL Northeast			\$1,737,904	\$636,595
Subheading Region Southeast				
Southeast	FL	Orlando	\$517,627	\$156,620
	GA	Atlanta	\$252,569	\$171,350
	TX	Memphis	\$274,812	\$270,841
TOTAL Southeast			\$1,045,008	\$598,811
Subheading Region West				
West	CA	Los Angeles	\$205,455	\$287,850
		San Francisco	\$270,620	\$180,270
	WA	Seattle	\$209,557	\$101,500
TOTAL West			\$685,632	\$569,620
TOTAL			\$3,386,184	\$3,066,244

Page Footing

Page 1

Report Footing

**Example: Aligning Subheading and Subfooting Margins in a Multi-Panel Report**

The following request has HEADPANEL=ON for all headings and footings. It also has the ALIGN-BORDERS=BODY attribute:

```

SET BYPANEL=ON
DEFINE FILE GGSales
SHOWCATPROD/A30 = CATEGORY || ( ' / ' | PRODUCT);
END
TABLE FILE GGSales
SUM
    DOLLARS/I8M AS ''
BY REGION
BY ST
BY CITY
ACROSS SHOWCATPROD AS 'Product Sales'

ON REGION SUBHEAD
" "
"Subheading <+0>Region <REGION<+0> "
" "
ON REGION SUBTOTAL AS '*TOTAL'
ON TABLE SUBHEAD
"Report Heading"
" "
"TYPE=REPORT, ALIGN-BORDERS=BODY, HEADPANEL=ON, BORDER=ON, $"
HEADING
"Page Heading "
" "
" "
" "
FOOTING
" "
"Page Footing<+0>Page <TABPAGENO "
ON TABLE SUBFOOT
" "
"Report Footing"

```

```
WHERE CATEGORY NE 'Coffee';
ON TABLE SET PAGE-NUM OFF
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
    UNITS=IN,
    SQUEEZE=ON,
    ORIENTATION=PORTRAIT,
$
TYPE=REPORT,
    FONT='ARIAL',
    SIZE=9,
    LEFTMARGIN=.75,
    RIGHTMARGIN=.5,
    TOPMARGIN=.1,
    BOTTOMMARGIN=.1,
    HEADPANEL=ON,
    ALIGN-BORDERS=BODY,
    BORDER=ON,
$
TYPE=TITLE,
    STYLE=BOLD,
$
TYPE=TABHEADING,
    SIZE=12,
    STYLE=BOLD,
$
TYPE=TABHEADING,
    LINE=3,
    JUSTIFY=CENTER,
$
TYPE=TABFOOTING,
    SIZE=12,
    STYLE=BOLD,
$
TYPE=HEADING,
    SIZE=12,
    STYLE=BOLD,
$
TYPE=HEADING,
    IMAGE=smplogol.gif,
    POSITION=(+4.6000000 +0.03000000),
    JUSTIFY=RIGHT,
```

```

$
TYPE=FOOTING,
    SIZE=12,
    STYLE=BOLD,
$
TYPE=FOOTING,
    LINE=2,
    ITEM=2,
    OBJECT=TEXT,
    POSITION=6.3,
    SIZE=12,
    STYLE=BOLD,
$
TYPE=SUBHEAD,
    SIZE=10,
    STYLE=BOLD,
$
TYPE=SUBHEAD,
    LINE=2,
    ITEM=3,
    OBJECT=TEXT,
    POSITION=2.5,
$
TYPE=SUBFOOT,
    SIZE=10,
    STYLE=BOLD,
$
TYPE=SUBTOTAL,
    BACKCOLOR=RGB(210 210 210),
$
TYPE=ACROSSVALUE,
    SIZE=9,
    WRAP=ON,
$
TYPE=ACROSSTITLE,
    STYLE=BOLD,
$
TYPE=GRANDTOTAL,
    BACKCOLOR=RGB(210 210 210),
    STYLE=BOLD,
$
ENDSTYLE
END

```

The output shows that the subheadings are aligned with the data on each panel.

Report Heading

TYPE=REPORT, ALIGN=BORDERS=BODY, HEADPANEL=ON, BORDER=ON, \$

Page Heading

Information Builders

The Standard for Training in Business Intelligence

			Product Sales								
			Food / Biscotti	Food / Croissant	Food / Scone	Gifts / Coffee Grinder	Gifts / Coffee Pot				
Region	State	City									
Subheading Region Midwest											
Midwest	IL	Chicago	\$378,412	\$549,385	\$895,989	\$233,292	\$254,828				
	MO	St. Louis	\$268,077	\$612,871	\$481,353	\$181,570	\$190,183				
	TX	Houston	\$345,238	\$687,887	\$418,398	\$204,292	\$254,897				
TOTAL Midwest			\$1,091,727	\$1,751,124	\$1,495,420	\$619,154	\$699,878				
Subheading Region Northeast											
Northeast	CT	New Haven	\$689,385	\$951,489	\$283,874	\$169,908	\$208,209				
	MA	Boston	\$670,351	\$497,234	\$232,486	\$177,940	\$184,119				
	NY	New York	\$642,288	\$622,095	\$290,811	\$161,382	\$199,482				
TOTAL Northeast			\$1,802,025	\$1,670,818	\$907,171	\$609,230	\$690,780				
Subheading Region Southeast											
Southeast	FL	Orlando	\$811,897	\$602,076	\$211,836	\$217,294	\$212,087				
	GA	Atlanta	\$666,231	\$661,806	\$273,420	\$217,254	\$223,252				
	TN	Memphis	\$438,889	\$638,477	\$316,389	\$171,319	\$200,694				
TOTAL Southeast			\$1,505,717	\$1,502,359	\$900,655	\$605,777	\$645,303				
Subheading Region West											
West	CA	Los Angeles	\$269,630	\$800,084	\$219,884	\$214,697	\$202,495				
	WA	San Francisco	\$389,818	\$824,487	\$230,839	\$187,123	\$187,845				
	WA	Seattle	\$328,320	\$801,080	\$204,445	\$207,768	\$213,494				
TOTAL West			\$888,688	\$2,425,651	\$912,668	\$609,436	\$603,834				
TOTAL			\$6,288,917	\$17,448,982	\$4,216,114	\$2,387,687	\$2,448,686				
Page Footing											
Page 1.1											
Report Footing											

Report Heading

TYPE=REPORT, ALIGN=BORDERS=BODY, HEADPANEL=ON, BORDER=ON, \$

Page Heading

Information Builders

The Standard for Training in Business Intelligence

			Product Sales											
			Gifts / Mug	Gifts / Thermos										
Region	State	City												
Subheading Region Midwest														
Midwest	IL	Chicago	\$376,754	\$187,901										
	MO	St. Louis	\$242,652	\$195,886										
	TX	Houston	\$368,337	\$194,319										
TOTAL Midwest			\$1,088,943	\$677,905										
Subheading Region Northeast														
Northeast	CT	New Haven	\$352,997	\$221,827										
	MA	Boston	\$402,944	\$223,435										
	NY	New York	\$345,300	\$179,835										
TOTAL Northeast			\$1,444,211	\$664,098										
Subheading Region Southeast														
Southeast	FL	Orlando	\$408,488	\$195,525										
	GA	Atlanta	\$355,447	\$227,452										
	TN	Memphis	\$337,750	\$209,449										
TOTAL Southeast			\$1,102,705	\$632,427										
Subheading Region West														
West	CA	Los Angeles	\$381,528	\$207,812										
	WA	San Francisco	\$375,388	\$165,115										
	WA	Seattle	\$427,339	\$198,640										
TOTAL West			\$1,186,694	\$671,388										
TOTAL			\$4,655,651	\$2,966,809										

Page Footing

Page 1.2

Report Footing

Syntax: How to Add and Adjust Grid Lines (PDF or PS)

This syntax applies to a PDF or PS report.

TYPE=type, {HGRID|VGRID}={ON|OFF|HEAVY}, \$

where:

type

Identifies the report component to which grid lines are applied. See [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167 for valid values.

HGRID

Specifies horizontal grid lines.

VGRID

Specifies vertical grid lines.

ON

Applies light grid lines.

OFF

Suppresses grid lines. OFF is the default value.



**HEAVY**

Applies heavy grid lines.

**Example: Applying Grid Lines to Report Data (PDF)**

This request applies light, horizontal grid lines to report data.

```
SET ONLINE-FMT = PDF
TABLE FILE GGDEMOG
HEADING
"State Statistics"
" "
SUM HH AS 'Number of,Households' AVGHHSZ98 AS 'Avg.,Size'
MEDHHI98 AS 'Avg.,Income'
BY ST
WHERE ST EQ 'CA' OR 'FL' OR 'NY'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=DATA, HGRID=ON, $
ENDSTYLE
END
```

In the PDF report, the lines make it easier to distinguish the data by state:

**State Statistics**

State	Number of Households	Avg. Size	Avg. Income
CA	11478067	3	44925
FL	5968392	2	34264
NY	6799434	3	42023

**Defining Borders Around Boxes With PPTX and PDF Formats**

In PPTX and PDF formats, the bgcolor of a box may be defined independently of the border color.

**Borders Without Backcolor**

When the border color is defined and there is no bgcolor, only the border or outline of the box is displayed in the border color specified, as shown in the example below.

```
TABLE FILE GGSales
BY REGION NOPRINT
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
TYPE=REPORT, OBJECT=BOX, POSITION=(1 1), DIMENSION=(2 1), BORDER-
COLOR=GREEN,$
ENDSTYLE
END
```

The output is shown in the following image.



### Backcolor Without Borders

In PPTX format, when a backcolor is defined and there is no border (BORDER-STYLE=NONE), the box retains the color defined for the backcolor.

```
TABLE FILE GGSales
BY REGION NOPRINT
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
TYPE=REPORT, OBJECT=BOX, POSITION=(1 1), DIMENSION=(2 1), BACKCOLOR=GREEN,
BORDER-STYLE=NONE, $
ENDSTYLE
END
```

The output is shown in the following image.



In PDF format, a gray outline appears around the backcolor, as shown in the following image.



### Border Styles Supported

Border styles, except 3D border styles such as ridged, groove, inset, and outset, are supported in PPTX and PDF formats.

```
TABLE FILE GGSales
BY REGION NOPRINT
ON TABLE PCHOLD FORMAT PPTX
ON TABLE SET STYLE *
TYPE=REPORT, OBJECT=BOX, POSITION=(1 1), DIMENSION=(2 1), BORDER-
COLOR=GREEN, BORDER-STYLE=DASHED, $
ENDSTYLE
END
```

The output is shown in the following image.



**Note:** When OBJECT=BOX and BORDER-STYLE=DOUBLE is used with FORMAT PPTX and FORMAT PDF in StyleSheet syntax, a solid border, instead of a double border, is generated.

## Displaying Superscripts On Data, Heading, and Footing Lines

Superscript characters are supported as a text style in text objects using HTML markup tags. The superscript markup tag is now supported in data columns, headings, and footings in HTML, PDF, and PS output formats. Superscript values can be defined within the data, added to virtual fields, or added to text strings displayed in headings and footings.

In order to activate the translation of the HTML markup tags, in the StyleSheet set MARKUP=ON for any report component that will display superscripts. Without this attribute, the markup tags will be treated as text, not tags.

### *Syntax:* How to Display Superscripts on Report Data, Heading, and Footing Lines

If the tags are not within the data itself, create a field that contains the text to be used as a superscript. Also, turn markup tags on for the components that will display superscripts:

- ☐ In a DEFINE or COMPUTE command, define a field that contains the text to be displayed as a superscript.

For a DEFINE FILE command, the syntax is:

```
DEFINE FILE ...
field/An = <sup>text</sup>;
END
```

For a COMPUTE command or a DEFINE in a Master File, the syntax is:

```
{COMPUTE|DEFINE} field/An = <sup>text</sup>;
```

where:

*n*

Is the length of the string defining the superscript, including the text to be used as the superscript and the opening and closing markup tags (<sup> and </sup>).

*text*

Is the text to be used as the superscript.

- ❑ In the StyleSheet, set MARKUP=ON for any report component that will display superscripts:

```
TYPE=component, MARKUP=ON ... , $
```

where:

*component*

Is one of the following report components: DATA, HEADING, FOOTING, SUBHEAD, SUBFOOT, TABHEADING, TABFOOTING.

### **Example:** Displaying Superscripts in Data and Footing Lines in PDF Output

The following request against the GGSALES data source defines two fields that will display as superscripts. SUP1 and SUP2 consist of the numbers 1 and 2, respectively. SUPCOPY consists of a copyright symbol. Note that the difference is the syntax as defined for a text value as opposed to a HEX value.

The COMPUTE command compares sales dollars to budgeted dollars. If the value calculated is less than a minimum defined, the superscript SUP1 is concatenated after the category name. If the value is greater, SUP2 is concatenated.

The superscript SUPCOPY is used to display the copyright symbol in the footing of the report.

The footing concatenates the superscript fields in front of their explanations.

In the StyleSheet, every component that will display a superscript has the attribute MARKUP=ON.

```

DEFINE FILE GGSales
SUP1/A12= '<SUP>1</SUP>';
SUP2/A15= '<SUP>2</SUP>';
SUPCOPY/A20= '<SUP>' || HEXBYT(169,'A2') || '</SUP>';
END
TABLE FILE GGSales
SUM
COMPUTE PROFIT/D12CM=DOLLARS-BUDDOLLARS; NOPRINT
COMPUTE SHOWCAT/A100=IF PROFIT LE -50000 THEN CATEGORY || SUP1
        ELSE IF PROFIT GT 50000 THEN CATEGORY || SUP2
        ELSE CATEGORY; AS Category
BUDDOLLARS/D12CM
DOLLARS/D12CM
BY REGION
BY CATEGORY NOPRINT
HEADING
"Analysis of Budgeted and Actual Sales"
FOOTING
" "
"<SUP1 Dollar sales $50,000 less than budgeted amount."
"<SUP2 Dollar sales $50,000 greater than budgeted amount."
" "
"Copyright<SUPCOPY 2012, by Information Builders, Inc "
ON TABLE SET HTMLCSS ON
ON TABLE SET SQUEEZE ON
ON TABLE SET PAGE-NUM OFF
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/
ENIADefault_combine.sty,$
TYPE=DATA,MARKUP=ON,$
TYPE=DATA,COLUMN=N5, COLOR=RED, WHEN=PROFIT LT -50000,$
TYPE=DATA,COLUMN=N6, COLOR=GREEN, WHEN=PROFIT GT 50000,$
TYPE=HEADING, JUSTIFY=LEFT,$
TYPE=FOOTING, MARKUP=ON, JUSTIFY=LEFT,$
TYPE=FOOTING, LINE=2,JUSTIFY=LEFT, COLOR=RED,$
TYPE=FOOTING, LINE=3,JUSTIFY=LEFT, COLOR=GREEN,$
ENDSTYLE
END

```

The output is:

## Analysis of Budgeted and Actual Sales

Region	Category	Budget Dollars	Dollar Sales
Midwest	Coffee <sup>2</sup>	\$4,086,032	\$4,178,513
	Food <sup>2</sup>	\$4,220,721	\$4,338,271
	Gifts	\$2,887,620	\$2,883,881
Northeast	Coffee <sup>1</sup>	\$4,252,462	\$4,164,017
	Food <sup>1</sup>	\$4,453,907	\$4,379,994
	Gifts	\$2,870,552	\$2,848,289
Southeast	Coffee	\$4,431,429	\$4,415,408
	Food <sup>1</sup>	\$4,409,288	\$4,308,731
	Gifts	\$2,967,254	\$2,986,240
West	Coffee <sup>1</sup>	\$4,523,963	\$4,473,517
	Food	\$4,183,244	\$4,202,337
	Gifts	\$2,934,306	\$2,977,092

<sup>1</sup> Dollar sales \$50,000 less than budgeted amount.

<sup>2</sup> Dollar sales \$50,000 greater than budgeted amount.

Copyright ©2012, by Information Builders, Inc

## Adding Underlines and Skipped Lines

You can make a detailed tabular report easier to read by separating sections with blank lines or underlines.

You cannot add blank lines or underlines to an HTML report that displays a grid. You can add blank lines or underlines if you set the GRID attribute to OFF.

When inserting blank lines, the setting of the LINES parameter should be at least one less than the setting of the PAPER parameter to allow room for blanks after the display of data on a page.

A Financial Modeling Language (FML) report with columns of numbers includes, by default, an underline before a RECAP calculation for readability. In these types of reports, you can change the default underline from light to heavy (or single to double in a PDF report).

**Reference: Section Separation Features**

Feature	Description	Applies to
<code>SKIP-LINE*</code>	Adds a blank line.	HTML (requires GRID=OFF) DHTML PDF PS XLSX EXL2K
<code>TYPE=SKIPLINE</code>	Formats a blank line.	DHTML PDF PS
<code>UNDER-LINE*</code>	Underlines a sort group.	HTML (requires GRID=OFF) DHTML PDF PS
<code>TYPE=UNDERLINE</code>	Formats an underline.	DHTML PDF PS

Feature	Description	Applies to
<code>STYLE={+ -}UNDERLINE*</code>	Adds an underline to a report component, or removes an underline from a report component other than a column title.	HTML DHTML PDF PS XLSX EXL2K
<code>STYLE={+ -} EXTUNDERLINE*</code>	Extends the underline to or removes the underline from the entire report column in a styled report.	DHTML PDF PS PPT PPTX
<code>BAR AS '{- =}'*</code>	Selects a single or double underline in an FML report.  For HTML, selects a light or heavy underline in an FML report.	HTML DHTML PDF PS XLSX EXL2K

\* Not supported with border.

**Syntax:**      **How to Add a Blank Line**

Use only one SKIP-LINE per report request.



*display\_command fieldname SKIP-LINE*

or

{ON|BY} *fieldname* SKIP-LINE [WHEN *expression*;

where:

*display\_command*

Is a display command.

*fieldname*

Is the display or sort field after which a blank line is inserted.

SKIP-LINE used with a display field adds a blank line after every displayed line, in effect, double-spacing a report. Double-spacing is helpful when a report is reviewed, making it easy for the reader to write comments next to individual lines.

SKIP-LINE used with a sort field adds a blank line before every change in the value of that field. This is one of the only ON conditions that does not have to refer solely to a sort (BY) field.

ON|BY

Is a vertical sort phrase. The terms are synonymous.

WHEN *expression*

Specifies conditional blank lines in the display of a report as determined by a logical expression. See [Using Expressions](#) on page 431 for details on expressions.

### **Example: Adding a Blank Line Between Sort Groups**

This request inserts a blank line before every change in value of the sort field EMP\_ID.

```
DEFINE FILE EMPLOYEE
INCREASE/D8.2M = .05*CURR_SAL;
CURR_SAL/D8.2M=CURR_SAL;
NEWSAL/D8.2M=CURR_SAL + INCREASE;
END

TABLE FILE EMPLOYEE
PRINT CURR_SAL OVER INCREASE OVER NEWSAL
BY LOWEST 4 EMP_ID BY LAST_NAME BY FIRST_NAME
ON EMP_ID SKIP-LINE
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT PDF
END
```

The data for each employee stands out and is easy to read:

<u>EMP_ID</u>	<u>LAST_NAME</u>	<u>FIRST_NAME</u>		
071382660	STEVENS	ALFRED	CURR_SAL	\$11,000.00
			INCREASE	\$550.00
			NEWSAL	\$11,550.00
112847612	SMITH	MARY	CURR_SAL	\$13,200.00
			INCREASE	\$660.00
			NEWSAL	\$13,860.00
117593129	JONES	DIANE	CURR_SAL	\$18,480.00
			INCREASE	\$924.00
			NEWSAL	\$19,404.00
119265415	SMITH	RICHARD	CURR_SAL	\$9,500.00
			INCREASE	\$475.00
			NEWSAL	\$9,975.00

### **Syntax:** How to Format a Blank Line

`TYPE=SKIPLINE, attribute=value, $`

where:

*attribute*

Is a valid StyleSheet attribute.

*value*

Is the value of the attribute.

**Note:** This option is supported for PDF, PS, and HTML reports (when used in conjunction with internal cascading style sheets).

**Example: Adding Color to Blank Lines**

In this request, blank lines are formatted to display as silver in the output. The relevant StyleSheet declaration is highlighted in the request.

```
SET ONLINE-FMT=PDF
TABLE FILE CENTINV
HEADING
"Low Stock Report"
" "
SUM QTY_IN_STOCK
WHERE QTY_IN_STOCK LT 5000
BY PRODNAME
ON PRODNAME SKIP-LINE
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=SKIPLINE, BACKCOLOR=SILVER, $
ENDSTYLE
END
```

The report is:

**Low Stock Report**

<u>Product Name:</u>	<u>Quantity In Stock:</u>
110 VHS-C Camcorder 20 X	4000
120 VHS-C Camcorder 40 X	2300
340SX Digital Camera 65K P	990
650DL Digital Camcorder 150 X	2972
DVD Upgrade Unit for Cent. VCR	199
R5 Micro Digital Tape Recorder	1990

**Syntax:**      **How to Underline a Sort Group**

```
{ON|BY} fieldname UNDER-LINE [WHEN expression;
```

where:

**ON|BY**

Is a vertical sort phrase. The terms are synonymous.

*fieldname*

Is the sort field to which the underline applies. UNDER-LINE adds an underline when the value of the sort field changes. An underline automatically displays after options such as RECAP or SUB-TOTAL but displays before page breaks.

**WHEN *expression***

Specifies conditional underlines in the display of a report as determined by a logical expression. See [Using Expressions](#) on page 431 for details on expressions.

**Example:**      **Underlining a Sort Group**

This request adds an underline when the value of the sort field BANK\_NAME changes. It sets the GRID attribute to OFF, as required by an HTML report.

```
TABLE FILE EMPLOYEE
PRINT EMP_ID AND BANK_ACCT AND LAST_NAME
BY BANK_NAME
ON BANK_NAME UNDER-LINE
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The data for each bank stands out and is easy to read:

<u>BANK NAME</u>	<u>EMP ID</u>	<u>BANK ACCT</u>	<u>LAST NAME</u>
	071382660		STEVENS
	112847612		SMITH
	119265415		SMITH
	126724188		ROMANS
	219984371		MCCOY
	543729165		GREENSPAN
ASSOCIATED	123764317	819000702	IRVING
	326179357	122850108	BLACKWOOD
	451123478	136500120	MCKNIGHT
BANK ASSOCIATION	818692173	163800144	CROSS
BEST BANK	119329144	160633	BANNING
STATE	117593129	40950036	JONES

**Syntax:** **How to Format an Underline**

`TYPE=UNDERLINE ... COLOR={color|RGB} (r g b), $`

where:

**UNDERLINE**

Denotes underlines generated by ON *fieldname* UNDER-LINE.

**COLOR**

Specifies the color of the underline. If the display or output device does not support colors, it substitutes shades of gray. The default value is black.

*color*

Is one of the supported color values. For a list of supported values, see [Color Values in a Report](#) on page 1615.

## RGB

Specifies the text color using a mixture of red, green, and blue.

(*r g b*)

Is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense.

Note that using the three-color components in equal intensities results in shades of gray.

**Note:** This option is supported for PDF, PS, and HTML reports (when used in conjunction with internal cascading style sheets).

## **Example:** Formatting a Sort Group Underline

This request uses UNDERLINE to change the default color of an underline from black to red.

```
SET ONLINE-FMT = PDF
TABLE FILE GGSales
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT
HEADING
"Sales Report"
" "
ON CATEGORY UNDER-LINE
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=UNDERLINE, COLOR=RED, $
ENDSTYLE
END
```

The result is an eye-catching separation between sort group values. The online PDF report is:

### Sales Report

<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

**Syntax:**      **How to Add or Remove a Report Component Underline**

`TYPE=type, [ subtype, ] STYLE=[+|-]UNDERLINE, $`

where:

*type*

Is the report component. For valid values, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

*subtype*

Are additional attributes, such as COLUMN, ACROSS, or ITEM, needed to identify the report component. For valid values, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

`+`

Adds an underline to the inherited text style or specifies a combination of text styles (for example, STYLE=BOLD+UNDERLINE). This is the default value.

`-`

Removes an underline from an inherited text style.

**Syntax:**      **How to Remove an Underline From a Column Title**

This syntax applies to an HTML report with internal cascading style sheet.

`TYPE=TITLE, [ COLUMN=column, ] STYLE=-UNDERLINE, $`

where:

`COLUMN=column`

Specifies a column. For valid values, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

**Example: Adding Column Underlines and Removing Column Title Underlines**

This request adds underlines to the values of the column CATEGORY and removes the default underlines from the column titles in an HTML report with internal cascading style sheet.

```
SET HTMLCSS = ON
TABLE FILE MOVIES
PRINT TITLE DIRECTOR
BY CATEGORY
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=TITLE, STYLE=-UNDERLINE, $
TYPE=REPORT, COLUMN=CATEGORY, STYLE=UNDERLINE, $
ENDSTYLE
END
```

The partial report is:

CATEGORY	TITLE	DIRECTOR
<u>ACTION</u>	JAWS	SPIELBERG S.
	ROBOCOP	VERHOVEN P.
	TOTAL RECALL	VERHOVEN P.
	TOP GUN	SCOTT T.
	RAMBO III	MCDONALD P.
<u>CHILDREN</u>	SMURFS, THE	
	SHAGGY DOG, THE	BARTON C.
	SCOOBY-DOO-A DOG IN THE RUFF	
	ALICE IN WONDERLAND	GEROMINI
	SESAME STREET-BEDTIME STORIES AND SONGS	
	ROMPER ROOM-ASK MISS MOLLY	
	SLEEPING BEAUTY	DISNEY W.
	BAMBI	DISNEY W.
<u>CLASSIC</u>	EAST OF EDEN	KAZAN E.
	CITIZEN KANE	WELLES O.
	CYRANO DE BERGERAC	GORDON M.
	MARTY	MANN D.
	MALTESE FALCON, THE	HUSTON J.
	GONE WITH THE WIND	FLEMING V.



**Syntax:**      **How to Extend an Underline to the Entire Report Column**

By default, underlines for column titles on a report extend only from the beginning to the end of the column title text. You can extend the underline to the entire report column in styled report output using the EXTUNDERLINE option in your WebFOCUS StyleSheet. EXTUNDERLINE is an option of the STYLE attribute for the TITLE report component. It is supported for formats DHTML, PDF, PS, PPT, and PPTX.

```
TYPE = TITLE, [COLUMN = colspec,] STYLE = [+|-]EXTUNDERLINE , $
```

where:

*colspec*

Is any valid column specification.

+EXTUNDERLINE

Adds the EXTUNDERLINE option to the inherited text style or specifies a combination of text styles (for example, STYLE=BOLD+UNDERLINE).

-EXTUNDERLINE

Removes the EXTUNDERLINE option from the inherited text style.

**Reference:**      **Usage Notes for the EXTUNDERLINE Attribute**

- ☐ HTML format is not supported because the browser calculates the column width and renders the report.
- ☐ GRID=ON and EXTUNDERLINE are mutually exclusive since the GRID line spans the width of the column. GRID overrides any styling specified for the column title underline.

**Example:**      **Extending an Underline to the Entire Report Column**

The following request against the GGSALES data source sums dollar sales by city and by date:

```
DEFINE FILE GGSALES
YEAR/YY = DATE;
MONTH/M = DATE;
END
TABLE FILE GGSALES
SUM DOLLARS AS 'Sales'
BY DATE
BY CITY
WHERE YEAR EQ 1997
WHERE MONTH FROM 01 TO 05
WHERE CITY EQ 'Seattle' OR 'San Francisco' OR 'Los Angeles'
ON TABLE SET PAGE NOPAGE
ON TABLE PCHOLD FORMAT DHTML
END
```

The output shows that only the column titles are underlined:

<u>Date</u>	<u>City</u>	<u>Sales</u>
1997/01/01	Los Angeles	159935
	San Francisco	162712
	Seattle	159804
1997/02/01	Los Angeles	148915
	San Francisco	143987
	Seattle	132505
1997/03/01	Los Angeles	165902
	San Francisco	145129
	Seattle	165847
1997/04/01	Los Angeles	146106
	San Francisco	158799
	Seattle	144169
1997/05/01	Los Angeles	178336
	San Francisco	144534
	Seattle	190208

To underline entire columns, generate the output in a format that can be styled and use the EXTUNDERLINE option in the STYLE attribute for the TITLE component. For example, the following request creates DHTML output in which the column titles are in boldface and left justified, and the underline is extended to the entire report column:

```

DEFINE FILE GGSales
YEAR/YY = DATE;
MONTH/M = DATE;
END
TABLE FILE GGSales
SUM DOLLARS AS 'Sales'
BY DATE
BY CITY
WHERE YEAR EQ 1997
WHERE MONTH FROM 01 TO 05
WHERE CITY EQ 'Seattle' OR 'San Francisco' OR 'Los Angeles'
ON TABLE SET PAGE NOPAGE
ON TABLE PCHOLD FORMAT DHTML
ON TABLE SET STYLE *
TYPE=TITLE, STYLE= BOLD +EXTUNDERLINE, JUSTIFY=LEFT, $
ENDSTYLE
END

```

The output is:

<b>Date</b>	<b>City</b>	<b>Sales</b>
1997/01/01	Los Angeles	159935
	San Francisco	162712
	Seattle	159804
1997/02/01	Los Angeles	148915
	San Francisco	143987
	Seattle	132505
1997/03/01	Los Angeles	165902
	San Francisco	145129
	Seattle	165847
1997/04/01	Los Angeles	146106
	San Francisco	158799
	Seattle	144169
1997/05/01	Los Angeles	178336
	San Francisco	144534
	Seattle	190208

The following version of the request makes the EXTUNDERLINE and JUSTIFY=LEFT options the default for the TITLE component, then makes the Date column title bold and removes the extended underline from that column:

```

DEFINE FILE GGSales
YEAR/YY = DATE;
MONTH/M = DATE;
END
TABLE FILE GGSales
SUM DOLLARS AS 'Sales'
BY DATE
BY CITY
WHERE YEAR EQ 1997
WHERE MONTH FROM 01 TO 05
WHERE CITY EQ 'Seattle' OR 'San Francisco' OR 'Los Angeles'
ON TABLE SET PAGE NOPAGE
ON TABLE PCHOLD FORMAT DHTML
ON TABLE SET STYLE *
TYPE=TITLE,STYLE= EXTUNDERLINE, JUSTIFY=LEFT ,$
TYPE=TITLE,COLUMN= DATE, STYLE= -EXTUNDERLINE +BOLD ,$
ENDSTYLE
END

```

The output is:

<u>Date</u>	<u>City</u>	<u>Sales</u>
1997/01/01	Los Angeles	159935
	San Francisco	162712
	Seattle	159804
1997/02/01	Los Angeles	148915
	San Francisco	143987
	Seattle	132505
1997/03/01	Los Angeles	165902
	San Francisco	145129
	Seattle	165847
1997/04/01	Los Angeles	146106
	San Francisco	158799
	Seattle	144169
1997/05/01	Los Angeles	178336
	San Francisco	144534
	Seattle	190208

**Syntax:** **How to Change Density of an Underline in a Financial Modeling Language (FML) Report**

This syntax applies to an HTML report.

```
BAR [AS '{_|=}' ] OVER
```

where:

=

Generates a light underline. Enclose the hyphen in single quotation marks. This is the default value.

=

Generates a heavy underline. Enclose the equal sign in single quotation marks.

**Example:** Changing the Default Underline in a Financial Modeling Language (FML) Report (HTML)

This request changes the default light underline to a heavy underline in an FML report.

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 AS 'CASH ON HAND'          OVER
1020 AS 'DEMAND DEPOSITS'       OVER
1030 AS 'TIME DEPOSITS'         OVER
BAR AS '='                      OVER
RECAP TOTCASH = R1 + R2 + R3;
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

A heavy underline separates total cash from the detail data, making it stand out:

	<u>AMOUNT</u>
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
	<hr/>
TOTCASH	21,239

**Example:** Changing the Default Underline in a Financial Modeling Language (FML) Report (PDF)

This request changes the default single underline in a PDF report to a double underline.

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 AS 'CASH ON HAND'          OVER
1020 AS 'DEMAND DEPOSITS'       OVER
1030 AS 'TIME DEPOSITS'         OVER
BAR AS '='                      OVER
RECAP TOTCASH = R1 + R2 + R3;
ON TABLE SET ONLINE-FMT PDF
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

	<u>AMOUNT</u>
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	<u>7,961</u>
TOTCASH	<u>21,239</u>

## Removing Blank Lines From a Report

The DROPBLNKLINE parameter controls whether blank lines display in a WebFOCUS report. With the options provided, you can affect blank lines that are automatically generated in different locations within a report. You can choose to drop the blank lines around subtotals, subheadings and subfootings, as well as certain data lines that may be blank and appear as blank lines on the report output. Additionally, when using borders, you can select to remove blank lines inserted around the headings and footings. You can eliminate these blank lines from the report output using the SET DROPBLNKLINE options.

**Syntax:**      **How to Control Automatic Blank Lines on Report Output**

SET DROPBLNKLINE={OFF|ON|BODY|HEADING|ALL}

or

ON TABLE SET DROPBLNKLINE {OFF|ON|BODY|HEADING|ALL}

where:

OFF

Inserts system-generated blank lines as well as empty data lines. OFF is the default value.

ON|BODY

Removes system-generated blank lines within the body of the report (for example, before and after subheads). In addition, certain data lines that may be blank and appear as blank lines on the report output will be removed from the output. BODY is a synonym for ON.

HEADING

Removes the blank lines between headings and titles and between the report body and the footing. Works in positioned formats (PDF, PS, DHTML, PPT, and PPTX) when a request has a border or bgcolor StyleSheet attribute anywhere in the report.

**ALL**

Provides both the ON and HEADING behaviors.

**Reference: Usage Notes for SET DROPBLNKLINE=HEADING**

- ☐ In the positioned report formats (PDF, PS, DHTML, PPT, and PPTX) with borders or bgcolor, the system automatically generates a blank line below the heading and above the footing. This is done by design to make bordered lines work together. Generally, the rule is that each line is responsible for the border setting for its top and left border. Therefore, the bottom border of the heading is set by the top border of the row beneath it. To ensure that the bottom of the heading border is complete and does not interfere with the top of the column titles border, a blank filler line is automatically inserted. This filler line contains the defined bottom border of the heading as its top border. The same is true between the bottom of the data and the top of the footing.

DROPBLNKLINE=HEADING removes the filler blank line by defining the height of the filler line to zero. This causes the bottom border of the heading to become the top border of the column titles. When bgcolor is used without borders, this works well to close any blank gaps in color. However, WebFOCUS processing will not remediate between line styles, so using different border styles between different report elements may create some contention between the border styling definitions. To ensure that you have consistent border line styling between different report elements, use a single line style between the elements that present together in the report.

- ☐ DROPBLNKLINE=HEADING is not supported with:
  - ☐ Different border styles between the heading and the column titles or the data and the footing.
  - ☐ Reports that use the ACROSS sort phrase.
- ☐ Usage Considerations:
  - ☐ In some reports, FOOTING BOTTOM requires the space added by the system-generated blank line between the data and the footing in order to present the correct distance between the sections. In these instances, the top of the FOOTING BOTTOM may slightly overlap the bottom of the data grid. You can resolve this by adding a blank line to the top of your footing.

- ❑ Applying borders for the entire report (TYPE=REPORT) is recommended to avoid certain known issues that arise when bordering report elements individually. In some reports that define bgcolor and borders on only select elements, the bgcolor applied to the heading is presenting with a different width than the bgcolor applied to the column titles. This difference causes a ragged right edge to present between the headings and the titles. Additionally, if you can define the color of the border (BORDER-COLOR) for elements with bgcolor to match the bgcolor, the borders will blend into the bgcolor and not be visible.

### ***Example:*** Comparing DROPBLNKLINE Parameter Settings

The following request against the GGSALES data source has a heading, a footing, and a subtotal. Initially, DROPBLNKLINE is set to OFF.

```
TABLE FILE GGSALES
HEADING CENTER
"Gotham Grinds Sales By Region"
FOOTING CENTER
"Generated on: &DATETMDYY"
SUM DOLLARS UNITS
BY REGION SUBTOTAL
BY CATEGORY
BY PRODUCT
WHERE REGION EQ 'Northeast' OR 'West'
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET DROPBLNKLINE OFF
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
SQUEEZE = ON,
FONT = ARIAL,
TYPE=HEADING, BORDER=LIGHT,
$
ENDSTYLE
END
```



The output has a blank line below the heading, above the footing, and above and below the subtotal lines and the grand total line.

Gotham Grinds Sales By Region				
Region	Category	Product	Dollar Sales	Unit Sales
Northeast	Coffee	Capuccino	542095	44785
		Espresso	850107	68127
		Latte	2771815	222866
	Food	Biscotti	1802005	145242
		Croissant	1670818	137394
		Scone	907171	70732
	Gifts	Coffee Grinder	509200	40977
		Coffee Pot	590780	46185
		Mug	1144211	91497
		Thermos	604098	48870
*TOTAL REGION Northeast			11392300	916675
West	Coffee	Capuccino	895495	71168
		Espresso	907617	71675
		Latte	2670405	213920
	Food	Biscotti	863868	70436
		Croissant	2425601	197022
		Scone	912868	72776
	Gifts	Coffee Grinder	603436	48081
		Coffee Pot	613624	47432
		Mug	1188664	93881
		Thermos	571368	45648
*TOTAL REGION West			11652946	932039
TOTAL			23045246	1848714

Generated on: JAN 10, 2012

Changing the DROPBLNKLINE setting to HEADING produces the following output. The blank line below the heading and the blank line above the footing have been removed. The blank lines above and below the subtotal and grand total lines are still inserted.

Gotham Grinds Sales By Region				
Region	Category	Product	Dollar Sales	Unit Sales
Northeast	Coffee	Capuccino	542095	44785
		Espresso	850107	68127
		Latte	2771815	222866
	Food	Biscotti	1802005	145242
		Croissant	1670818	137394
		Scone	907171	70732
	Gifts	Coffee Grinder	509200	40977
		Coffee Pot	590780	46185
		Mug	1144211	91497
		Thermos	604098	48870
*TOTAL REGION Northeast			11392300	916675
West	Coffee	Capuccino	895495	71168
		Espresso	907617	71675
		Latte	2670405	213920
	Food	Biscotti	863868	70436
		Croissant	2425601	197022
		Scone	912868	72776
	Gifts	Coffee Grinder	603436	48081
		Coffee Pot	613624	47432
		Mug	1188664	93881
		Thermos	571368	45648
*TOTAL REGION West			11652946	932039
TOTAL			23045246	1848714
Generated on: JAN 10, 2012				

Changing the DROPBLNKLINE setting to ON (or BODY) produces the following output in which the blank lines above and below the subtotal and grand total lines have been removed, but the blank lines below the heading and above the footing are still inserted.

Gotham Grinds Sales By Region				
Region	Category	Product	Dollar Sales	Unit Sales
Northeast	Coffee	Capuccino	542095	44785
		Espresso	850107	68127
		Latte	2771815	222866
	Food	Biscotti	1802005	145242
		Croissant	1670818	137394
		Scone	907171	70732
	Gifts	Coffee Grinder	509200	40977
		Coffee Pot	590780	46185
		Mug	1144211	91497
		Thermos	604098	48870
	*TOTAL REGION	Northeast		11392300
West	Coffee	Capuccino	895495	71168
		Espresso	907617	71675
		Latte	2670405	213920
	Food	Biscotti	863868	70436
		Croissant	2425601	197022
		Scone	912868	72776
	Gifts	Coffee Grinder	603436	48081
		Coffee Pot	613624	47432
		Mug	1188664	93881
		Thermos	571368	45648
	*TOTAL REGION	West		11652946
TOTAL			23045246	1848714

Generated on: JAN 10, 2012

Changing the DROPBLNKLINE setting to ALL produces the following output in which the blank lines around the subtotal and grandtotal lines as well as the blank lines below the heading and above the footing have been removed.

Gotham Grinds Sales By Region				
Region	Category	Product	Dollar Sales	Unit Sales
Northeast	Coffee	Capuccino	542095	44785
		Espresso	850107	68127
		Latte	2771815	222866
	Food	Biscotti	1802005	145242
		Croissant	1670818	137394
		Scone	907171	70732
	Gifts	Coffee Grinder	509200	40977
		Coffee Pot	590780	46185
		Mug	1144211	91497
		Thermos	604098	48870
	*TOTAL REGION	Northeast		11392300
West	Coffee	Capuccino	895495	71168
		Espresso	907617	71675
		Latte	2670405	213920
	Food	Biscotti	863868	70436
		Croissant	2425601	197022
		Scone	912868	72776
	Gifts	Coffee Grinder	603436	48081
		Coffee Pot	613624	47432
		Mug	1188664	93881
		Thermos	571368	45648
	*TOTAL REGION	West		11652946
TOTAL			23045246	1848714
Generated on: JAN 10, 2012				

Generated on: JAN 10, 2012

## Adding an Image to a Report

With a StyleSheet you can add and position an image in a report. An image, such as a logo, gives corporate identity to a report, or provides visual appeal. You can add more than one image by creating multiple declarations.

You can also add an image as background to a report. A background image is tiled or repeated, covering the entire area on which the report displays. An image attached to an entire report, or an image in a heading or footing, can appear with a background image.

Images must exist in a file format your browser supports, such as GIF (Graphic Interchange Format) or JPEG (Joint Photographic Experts Group, .jpg extension).

### Image support with WebFOCUS standard reporting formats

- ☐ GIF and JPG images are supported in DHTML, HTML, PDF, PS, PPTX, XLSX, and PPT standard report formats. JPEG images are only supported with HTML standard report format. For other report formats, you can change the extension of the image name from .jpeg to .jpg, and the image will be displayed in the report output.
- ☐ PNG images are supported with DHTML, HTML, PPTX, and PDF standard report formats, while WebFOCUS generated SVG charts will display when inserted in HTML, PDF, and PS report formats.
- ☐ SVG images are supported only with HTML reports.
- ☐ Images are not supported in EXL2K standard report format.

### Image support in Compound Report syntax

- ☐ GIF and JPG images are supported in DHTML, PDF, PPTX, and PPT Compound document syntax. JPEG images are not supported with any reporting format, but the images will work in these compound formats if the extension is changed from .jpeg to .jpg.
- ☐ PNG images are supported with DHTML, PPT, PPTX, and PDF Compound documents.
- ☐ SVG images are not supported with any WebFOCUS reporting format in Compound documents, while WebFOCUS generated SVG charts are supported only with PDF Compound documents.
- ☐ Images are not supported in EXL2K Compound documents.

**Note:** Images in report components in Compound documents are supported as described under *Image support with WebFOCUS standard reporting formats*. This section, *Image support in Compound Report syntax*, refers to images inserted in the PAGELAYOUT sections of the Compound syntax.

For PDF, HTML, and DHTML output against data sources that support the Binary Large Object (BLOB) data type (Microsoft SQL Server, DB2, Oracle, Informix, and PostgreSQL, using its BYTEA data type), an image can be stored in a BLOB field in the data source.

The image must reside on the WebFOCUS Reporting Server in a directory named on EDAPATH or APPPATH. If the file is not on the search path, supply the full path name.

**Note:** For JPEG files, currently only the .jpg extension is supported. The .jpeg extension is not supported.

**Reference: Browser and Device Support for Images in HTML Documents**

Support for presenting images and graphs in HTML and DHTML formatted standard reports and compound documents is provided using an image embedding facility that allows 64-bit images to be encoded within a generated .htm file.

The SET HTMLEMBEDIMG command is designed to ensure that all WebFOCUS reports containing images can be accessed from any browser or device. By default, it is set to ON and embeds images within an .htm file.

SET HTMLEMBEDIMG= {[ON](#) | [OFF](#) | [AUTO](#)}

where:

[ON](#)

Encodes images within the .htm file. ON is the default value, and overrides the HTMLARCHIVE settings.

[OFF](#)

Does not embed the image. If HTMLARCHIVE is set to ON, .mht files are generated.

[AUTO](#)

Determines which encoding algorithm to use, based on the browser of the client machine that submits the report request. Where the browser is identified as an Internet Explorer browser, or the browser is unknown (such as reports distributed by ReportCaster), WebFOCUS will continue to generate Web Archive files (.mht). For all other browsers, WebFOCUS will encode the image into an HTML file (.htm).

**Usage Notes for HTMLEMBEDIMG**

- ❑ When HTMLEMBEDIMG is set to ON (the default setting) and HTMLARCHIVE is set to ON, the HTMLEMBEDIMG ON setting overrides the HTMLARCHIVE ON setting, and an .htm file, in which the image is embedded, is generated.
- ❑ When HTMLEMBEDIMG is set to ON (the default setting) and HTMLARCHIVE is set to OFF, an .htm file is generated and the image is embedded.
- ❑ Setting HTMLEMBEDIMG to OFF and enabling HTMLARCHIVE generates an .mht file, which contain the encoded image.
- ❑ When HTMLEMBEDIMG and HTMLARCHIVE are set to OFF, an .html file is generated, but the image is not embedded.

- ❑ The encoding algorithm that uses 64-bit encoding supported for images less than 32K in size is supported by Internet Explorer 8. For Internet Explorer 8, Information Builders recommends continuing to use the .mht format generated by HTMLARCHIVE. In Internet Explorer 9 and higher and browsers other than Internet Explorer, the new algorithm is supported for images of any size. See the browser vendor information to confirm 64-bit encoding support.

**Reference: Image Attributes**

Attribute	Description
<a href="#">IMAGE</a>	Adds an image.
<a href="#">IMAGEALIGN</a>	Positions an image. This applies only to HTML reports.
<a href="#">POSITION</a>	Positions an image.
<a href="#">IMAGEBREAK</a>	Controls generation of a line break after an image. This applies only to HTML reports without internal cascading style sheets.
<a href="#">SIZE</a>	Sizes an image.
<a href="#">ALT</a>	Supplies a description of an image for compliance with Section accessibility (Workforce Investment Act of 1998). ALT only applies to HTML reports.
<a href="#">PRESERVERATIO</a>	ON specifies that the aspect ratio (ratio of height to width) of the image should be preserved when it is scaled to the specified SIZE. This avoids distorting the appearance of the image. The image is scaled to the largest size possible within the bounds specified by SIZE for which the aspect ratio can be maintained. Supported for images in PDF and PostScript report output.
<a href="#">BACKGROUNDIMAGE</a>	Adds a background image.

## **Syntax:**      **How to Add an Image to an HTML Report**

This syntax applies to an HTML report. For details on adding an image to a PDF, PS, or HTML report with an internal CSS, see [How to Add an Image to a PDF, PS, or HTML Report With an Internal Cascading Style Sheet](#) on page 1390.

```
TYPE={REPORT|heading}, IMAGE={url|(column)} [,IMAGEALIGN=position]  
[,IMAGEBREAK={ON|OFF}] [,ALT='description'], $
```

where:

### [REPORT](#)

Embeds an image in the body of a report. REPORT is the default value.

**Note:** The IMAGE=(column) option is not supported with TYPE=REPORT.

### *heading*

Embeds an image in a heading or footing. Valid values are TABHEADING, TABFOOTING, HEADING, FOOTING, SUBHEAD, and SUBFOOT.

### *url*

Is the URL for the image file. The image must exist in a separate file in a format that your browser supports, such as GIF or JPEG (.jpg). The file can be on your local web server, or on any server accessible from your network. For details, see [Specifying a URL](#) on page 1381.

### *column*

Is an alphanumeric field in a request (for example, a display field or a BY field) whose value is a URL that points to an image file. Specify a value using the COLUMN attribute described in [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167. Enclose *column* in parentheses.

This option enables you to add different images to a heading or footing, depending on the value of the field.

### IMAGEALIGN = *position*

Is the position of the image.

**Note:** IMAGEALIGN is not supported with HTMLCSS=ON. With HTMLCSS=ON, you can position images within a heading or footing by using the POSITION attribute to specify a position relative to the upper-left corner of the heading or footing. For more information about the POSITION attribute, see [How to Add an Image to a PDF, PS, or HTML Report With an Internal Cascading Style Sheet](#) on page 1390.



Valid values are:

**TOP** where the top right corner of the image aligns with heading or footing text. If the image is attached to the entire report, it appears on top of the report.

**MIDDLE** where the image appears in the middle of the heading or footing text. If the image is attached to the entire report, it appears in the middle of the report.

**BOTTOM** where the bottom right corner of the image aligns with heading or footing text. If the image is attached to the entire report, it appears at the bottom of the report.

**LEFT** where the image appears to the left of heading or footing text. If the image is attached to the entire report, it appears to the left of the report.

**RIGHT** where the image appears to the right of heading or footing text. If the image is attached to the entire report, it appears to the right of the report.

#### **IMAGEBREAK**

Controls generation of a line break after the image. Valid values are:

**ON** which generates a line break after the image so that an element following it (such as, report heading text) appears on the next line.

**OFF** which suppresses a line break after the image so that an element following it is on the same line. OFF is the default value.

#### *description*

Is a textual description of an image for compliance with Section 508 accessibility. Enclose the description in single quotation marks.

### **Reference: Specifying a URL**

The following guidelines are the same for `IMAGE=url` and `IMAGE=(column)` syntax. In the latter case, they apply to a URL stored in a data source field.

Specify a URL by:

- ☐ Supplying an absolute or relative address that points to an image file, for example:

```
TYPE=TABHEADING, IMAGE=http://www.ibi.com/images/logo_wf3.gif,$
TYPE=TABHEADING, IMAGE=/ibi_apps/ibi_html/ggdemo/gotham.gif,$
```

- ☐ Using the SET BASEURL parameter to establish a URL that is logically prefixed to all relative URLs in the request. With this feature, you can add an image by specifying just its file name in the IMAGE attribute. For example:

```
SET BASEURL=http://host:port/
.
.
.
TYPE=REPORT, IMAGE=gotham.gif,$
```

The following apply:

- ☐ A base URL must end with a slash (/).
- ☐ An absolute URL (which begins with http://) overrides a base URL.
- ☐ A URL is case-sensitive when referring to a UNIX server.
- ☐ If the name of the image file does not contain an extension, .GIF is used.

### ***Example:*** Adding a GIF Image to an HTML Report Heading

This request adds the Gotham Grinds logo to a report heading. The logo is in a separate image file identified by a relative URL in the IMAGE attribute.

```
TABLE FILE GGORDER
ON TABLE SUBHEAD
"PRODUCTS ORDERED ON 08/01/96"
SUM QUANTITY AS 'Ordered Units' BY PRODUCT
WHERE PRODUCT EQ 'Coffee Grinder' OR 'Coffee Pot'
WHERE ORDER_DATE EQ '08/01/96'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=TABLEHEADING, IMAGE=/_IBI_APPS/_IBI_HTML/GGDEMO/GOTHAM.GIF, IMAGEBREAK=ON,
$
ENDSTYLE
END
```

IMAGEBREAK, set to ON, generates a line break between the logo and the heading text:

	
PRODUCTS ORDERED ON 08/01/96	
Product	Ordered Units
Coffee Grinder	2493
Coffee Pot	3100

**Example:** Creating a Report Heading With an Embedded JPEG Image

```
TABLE FILE EMPLOYEE
ON TABLE SUBHEAD
"Employee Salary Information and Courses"
" "
" "
" "
" "
" "
" "
" "
" "
" "
" "
PRINT CURR_SAL BY COURSE_NAME
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=TABHEADING, IMAGE=C:\IBI\APPS\IMAGES\Pencils.jpg,
POSITION=(.5 .5), SIZE=(.5 .5), $
ENDSTYLE
END
```

**Note:** The image used in this request is not distributed with WebFOCUS.

The output is:

EMPLOYEE SALARY INFORMATION AND SKILLS



<u>COURSE NAME</u>	<u>CURR SAL</u>
ADVANCED TECHNIQUES	\$18,480.00
BASIC REPORT PREP DP MGRS	\$27,062.00
BASIC REPORT PREP FOR PROG	\$13,200.00
	\$18,480.00
BASIC REPORT PREP NON-PROG	\$21,780.00
BASIC RPT NON-DP MGRS	\$9,500.00
DECISION SUPPORT WORKSHOP	\$21,780.00
FILE DESC & MAINT NON-PROG	\$21,780.00
FILE DESCRPT & MAINT	\$11,000.00
	\$13,200.00
	\$18,480.00
	\$16,100.00
FOCUS INTERNALS	\$18,480.00
HOST LANGUAGE INTERFACE	\$27,062.00
TIMESHARING WORKSHOP	\$21,780.00
WHAT'S NEW IN FOCUS	\$21,780.00

**Example: Using a File Name in a Data Source Field in an HTML Report**

The following illustrates how to embed an image in a SUBHEAD, and use a different image for each value of the BY field on which the SUBHEAD occurs.

```

DEFINE FILE CAR
FLAG/A12=
DECODE COUNTRY ( 'ENGLAND' 'uk' 'ITALY' 'italy'
'FRANCE' 'france' 'JAPAN' 'japan' );
END

TABLE FILE CAR
PRINT FLAG NOPRINT AND MODEL AS '' BY COUNTRY NOPRINT AS '' BY CAR AS ''
WHERE COUNTRY EQ 'ENGLAND' OR 'FRANCE' OR 'ITALY' OR 'JAPAN'
ON COUNTRY SUBHEAD
"                                <+0>Cars produced in <ST.COUNTRY"
HEADING CENTER
"Car Manufacturer Report"
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=SUBHEAD, IMAGE=(FLAG), $
TYPE=REPORT, GRID=OFF, $
TYPE=HEADING, SIZE=12, STYLE=BOLD, $
TYPE=SUBHEAD, STYLE=BOLD, $
ENDSTYLE
END

```

The output is:

## Car Manufacturer Report



### Cars produced in ENGLAND

JAGUAR	V12XKE AUTO
	XJ12L AUTO
JENSEN	INTERCEPTOR III
TRIUMPH	TR7



### Cars produced in FRANCE

PEUGEOT	504 4 DOOR
---------	------------



### Cars produced in ITALY

ALFA ROMEO	2000 4 DOOR BERLINA
	2000 GT VELOCE
	2000 SPIDER VELOCE
MASERATI	DORA 2 DOOR



### Cars produced in JAPAN

DATSUN	B210 2 DOOR AUTO
TOYOTA	COROLLA 4 DOOR DIX AUTO

**Example: Supplying an Image Description Using the ALT Attribute**

This request adds the Information Builders logo to a report footing. It uses the WebFOCUS StyleSheet ALT attribute to add descriptive text (Information Builders logo) that identifies the image.

```
TABLE FILE GGORDER
SUM QUANTITY AS 'Ordered Units'
BY PRODUCT
ON TABLE SUBHEAD
"PRODUCTS ORDERED"
FOOTING
" "
ON TABLE SET ACCESSHTML 508
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF,$
TYPE=FOOTING, IMAGE=/ibi_html/iblogo.gif, ALT='Information Builders logo',$
ENDSTYLE
END
```

**Note:** If the request is located in the WebFOCUS repository and the WebFOCUS Client *Upload Images to be Embedded in Reports* Applications setting is selected, you need to either:

- ☐ Include the -MRNOEDIT command at the beginning of the line that references the IMAGE not located in the WebFOCUS repository.
- ☐ Include the fully qualified path in the IMAGE parameter.

The WebFOCUS Client *Upload Images to be Embedded in Reports* Applications setting specifies whether to upload Repository images to the Reporting Server for embedding in reports and HTML pages. The default is to upload Repository images. For more information on the *Upload Images to be Embedded in Reports* Applications setting, see the *Security and Administration* manual.

The -MRNOEDIT command instructs the WebFOCUS Client to not process the line of code. For more information on the -MRNOEDIT command, see the *Business Intelligence Portal* manual.

When you run the request, the image displays below the report data, as shown in the following image.

### PRODUCTS ORDERED

<u>Product</u>	<u>Ordered Units</u>
Biscotti	136045
Coffee Grinder	58703
Coffee Pot	61982
Croissant	204776
French Roast	285689
Hazelnut	100427
Kona	61498
Mug	117186
Scone	108359
Thermos	61778



When you hover the mouse over the image, the descriptive text displays in a box if your browser image loader is turned off or if the browser does not display images.

WebFOCUS generates the following HTML code for the image:

```
<IMG SRC="/ibi_html/ibilogo.gif"  
ALT="Information Builders logo">
```

### **Syntax:** How to Add a Background Image

This syntax applies to an HTML report.

```
[TYPE=REPORT,] BACKIMAGE=url, $
```

where:

**TYPE=REPORT**

Applies the image to the entire report. Not required, as it is the default.



*url*

Is the URL of a GIF or JPEG file (.jpg). Specify a file on your local web server, or on a server accessible from your network.

The URL can be an absolute or relative address. See [Image Attributes](#) on page 1379.

When specifying a GIF file, you can omit the file extension.

### **Example: Adding a Background Image**

This request adds a background image to a report. The image file CALM\_BKG.GIF resides in the relative address shown.

```
TABLE FILE GGSales
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, STYLE=BOLD, GRID=OFF, $
TYPE=REPORT, BACKIMAGE=/IBI_APPS/IBI_HTML/TEMPLATE/CALM_BKG.GIF, $
ENDSTYLE
END
```

The background is tiled across the report area:

<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

**Syntax:**      **How to Add an Image to a PDF, PS, or HTML Report With an Internal Cascading Style Sheet**

This syntax applies to a PDF, PS, or HTML report with an internal cascading style sheet. The image can be in a separate file.

A report with an Internal cascading style sheet is an HTML page with an HTML cascading style sheet (CSS) stored between the style tags within the HTML document.

```
TYPE={REPORT|heading}, IMAGE={url|file|(column)} [ ,BY=byfield]
[ ,POSITION=( [+|-]x [+|-]y ) ] [ ,SIZE=(w h) ] , $
```

where:

REPORT

Embeds an image in the body of a report. The image appears in the background of the report. REPORT is the default value.

*heading*

Embeds an image in a heading or footing. Valid values are TABHEADING, TABFOOTING, FOOTING, HEADING, SUBHEAD, and SUBFOOT.

Provide sufficient blank space in the heading or footing so that the image does not overlap the heading or footing text. Also, you may want to place heading or footing text to the right of the image using spot markers or the POSITION attribute in the StyleSheet.

*url*

HTML report with internal cascading style sheet:

Is the absolute or relative address for the image file. The image must exist in a separate file in a format that your browser supports, such as GIF or JPEG (.jpg). The file can be on your local web server, or on any server accessible from your network. For details, see [Specifying a URL](#) on page 1381.

*file*

PDF or PS report:

Is the name of the image file. It must reside on the WebFOCUS Reporting Server in a directory named on EDAPATH or APPPATH. If the file is not on the search path, supply the full path name.

When specifying a GIF file, you can omit the file extension.

*column*

Is an alphanumeric field in the data source that contains the name of an image file. Use the COLUMN attribute described in *Identifying a Report Component in a WebFOCUS StyleSheet* on page 1167. Enclose *column* in parentheses.

The field containing the file name or image must be a display field or BY field referenced in the request.

Note that the value of the field is interpreted exactly as if it were typed as the URL of the image in the StyleSheet. If you omit the suffix, .GIF is supplied by default. SET BASEURL can be useful for supplying the base URL of the images. If you do that, the value of the field does not have to include the complete URL.

This syntax is useful, for example, if you want to embed an image in a SUBHEAD, and you want a different image for each value of the BY field on which the SUBHEAD occurs.

*byfield*

Is the sort field that generated the subhead or subfoot.

## POSITION

Is the starting position of the image.

## +|-

Measures the horizontal or vertical distance from the upper-left corner of the report component in which the image is embedded.

*x*

Is the horizontal starting position of the image from the upper-left corner of the physical report page, expressed in the unit of measurement specified by the UNITS parameter.

Enclose the x and y values in parentheses. Do not include a comma between them.

*y*

Is the vertical starting position of the image from the upper-left corner of the physical report page, expressed in the unit of measurement specified by the UNITS parameter.

## SIZE

Is the size of the image. By default, an image is added at its original size.

*w*

Is the width of the image, expressed in the unit of measurement specified by the UNITS parameter.

Enclose the *w* and *h* values in parentheses. Do not include a comma between them.

*h*

Is the height of the image, expressed in the unit of measurement specified by the UNITS parameter.

### **Example:** Adding a GIF Image to an HTML Report With Internal Cascading Style Sheet

A URL locates the image file GOTHAM.GIF on a server named WEBSRVR1. The TYPE attribute adds the image to the report heading. POSITION places the image one-quarter inch horizontally and one-tenth inch vertically from the upper-left corner of the report page. The image is one inch wide and one inch high as specified by SIZE.

```
SET HTMLCSS = ON
TABLE FILE GGSales
SUM UNITS BY PRODUCT
ON TABLE SUBHEAD
"REPORT ON UNITS SOLD"
" "
" "
" "
" "
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=TABLEHEADING, IMAGE=HTTP://WEBSRVR1/IBI_APPS/IBI_HTML/GGDEMO/GOTHAM.GIF,
POSITION=(.25 .10), SIZE=(1 1), $
ENDSTYLE
END
```

The company logo is positioned and sized in the report heading:

#### REPORT ON UNITS SOLD



<u>Product</u>	<u>Unit Sales</u>
Biscotti	421377
Capuccino	189217
Coffee Grinder	186534
Coffee Pot	190695
Croissant	630054
Espresso	308986
Latte	878063
Mug	360570
Scone	333414
Thermos	190081

#### **Example:** Adding a GIF Image to a PDF Report

The image file for this example is GOTHAM.GIF. The POSITION attribute places the image one-quarter inch horizontally and one-quarter vertically from the upper-left corner of the report page. The image is one-half inch wide and one-half inch high as specified by SIZE.

```

SET ONLINE-FMT = PDF
TABLE FILE GGSales
SUM UNITS BY PRODUCT
ON TABLE SUBHEAD
"Report on Units Sold"
" "
" "
" "
" "
" "
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=TABLEHEADING, IMAGE=GOTHAM.GIF, POSITION=(.25 .25), SIZE=(.5 .5), $
ENDSTYLE
END

```

The report is:

Report on Units Sold



<u>Product</u>	<u>Unit Sales</u>
Biscotti	421377
Capuccino	189217
Coffee Grinder	186534
Coffee Pot	190695
Croissant	630054
Espresso	308986
Latte	878063
Mug	360570
Scone	333414
Thermos	190081

### *Example:* Adding a PNG Image to a PDF Report

The image file for this sample is `lbi_logo.png`. The `POSITION` attribute places the image to the upper-left corner of the report page. The image is one inch wide and half an inch high, as specified by `SIZE`.

```
SET HTMLCSS = ON
TABLE FILE GGSales
SUM UNITS BY PRODUCT
ON TABLE SUBHEAD
"REPORT ON UNITS SOLD"
" "
" "
" "
" "
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=TABLEHEADING, IMAGE=lbi_logo.png, POSITION=(0 .30), SIZE=(1 0.5), $
ENDSTYLE
END
```

The report is:

## REPORT ON UNITS SOLD



Product	Unit Sales
Biscotti	421377
Capuccino	189217
Coffee Grinder	186534
Coffee Pot	190695
Croissant	630054
Espresso	308986
Latte	878063
Mug	360570
Scone	333414
Thermos	190081

### **Syntax:** How to Add an Image From a BLOB Field to a PDF, DHTML, or HTML Report

For PDF, HTML, and DHTML output against data sources that support the Binary Large Object (BLOB) data type (Microsoft SQL Server, DB2, Oracle, Informix, and PostgreSQL using its BYTEA data type), an image can be stored in a BLOB field in the data source.

WebFOCUS StyleSheets used to produce report output in PDF, HTML, or DHTML format can access a BLOB field as an image source when an instance of the BLOB field contains an exact binary copy of a GIF or JPG image. HTML and DHTML reports also support PNG images. Images of different formats (GIF, JPG, PNG) can be mixed within the same BLOB field. WebFOCUS can determine the format from the header of the image. The image can be inserted in report columns, headings, footings, subheadings, and subfootings.

The BLOB field must be referenced in a PRINT or LIST command in the request (aggregation is not supported). Reports containing BLOB images are supported as components in Coordinated Compound Reports.

With the following SET commands, BLOB images will work for both HTML and DHTML in all browsers:

- ☐ SET HTMLMBEDIMG=AUTO.
- ☐ SET HTMLARCHIVE=ON (required to support Internet Explorer with images larger than 32K).
- ☐ SET BASEURL="" (required to make embedded images work as it overrides the default setting sent from the WebFOCUS Client).

- ❑ SET HTMLCSS=ON (required for image positioning in subheads in HTML reports). Setting HTMLCSS=ON creates an HTML report with an Internal cascading style sheet. A report with an Internal cascading style sheet is an HTML page with an HTML cascading style sheet (CSS) stored between the style tags within the HTML document.

```
TYPE={REPORT|heading}, IMAGE={url|file|(column)} [,BY=byfield]  
[,POSITION=( [+|-]x [+|-]y )] [,SIZE=(w h)] [,PRESERVERATIO={ON|OFF}] , $
```

```
TYPE=DATA, COLUMN=imagefield, IMAGE=(imagefield), SIZE=(wh)  
[,PRESERVERATIO={ON|OFF}] , $
```

where:

### REPORT

Embeds an image in the body of a report. The image appears in the background of the report. REPORT is the default value (not supported for images stored in BLOB fields, which are supported for PDF output).

### *heading*

Embeds an image in a heading or footing. Valid values are FOOTING, HEADING, SUBHEAD, and SUBFOOT.

If the image is to be embedded in a heading, subheading, footing, or subfooting rather than a column, the StyleSheet declaration is responsible for placing the image in the heading, subheading, footing, or subfooting. To make the BLOB image accessible to the StyleSheet, the BLOB field must be referenced in the PRINT or LIST command with the NOPRINT option. Do not reference the BLOB field name in the heading or footing itself.

Provide sufficient blank space in the heading or footing so that the image does not overlap the heading or footing text. Also, you may want to place heading or footing text to the right of the image using spot markers or the POSITION attribute in the StyleSheet.

### *file*

Is the name of the image file. It must reside on the WebFOCUS Reporting Server in a directory named on EDAPATH or APPPATH. If the file is not on the search path, supply the full path name.

When specifying a GIF file, you can omit the file extension.



*column*

Is a BLOB field in the data source that contains an exact binary copy of a GIF or JPG image. HTML and DHTML formats also support images in PNG format. Images of different formats (GIF, JPG, PNG) can be mixed within the same BLOB field. WebFOCUS can determine the format from the header of the image. The image can be inserted in report columns, headings, footings, subheadings and subfootings. Use the COLUMN attribute described in [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167. Enclose *column* in parentheses.

The field containing the file name or image must be a display field or BY field referenced in the request.

*byfield*

Is the sort field that generated the subhead or subfoot.

*imagefield*

Is any valid column reference for the BLOB field that contains the image. Note that the BLOB field must be referenced in a PRINT or LIST command in the request.

If omitted, the default size is 1 inch by 1 inch. The width of the column and the spacing between the lines is automatically adjusted to accommodate the image.

## POSITION

Is the starting position of the image.

## +|-

Measures the horizontal or vertical distance from the upper-left corner of the report component in which the image is embedded.

*x*

Is the horizontal starting position of the image from the upper-left corner of the physical report page, expressed in the unit of measurement specified by the UNITS parameter.

Enclose the x and y values in parentheses. Do not include a comma between them.

*y*

Is the vertical starting position of the image from the upper-left corner of the physical report page, expressed in the unit of measurement specified by the UNITS parameter.

## SIZE

Is the size of the image. By default, an image is added at its original size. Note that images stored in BLOB fields are supported only for PDF, HTML, and DHTML output.

*w*

Is the width of the image, expressed in the unit of measurement specified by the UNITS parameter.

Enclose the *w* and *h* values in parentheses. Do not include a comma between them.

*h*

Is the height of the image, expressed in the unit of measurement specified by the UNITS parameter.

If SIZE is omitted, the original dimensions of the image are used (any GIF, JPG, or PNG image has an original, unscaled size based on the dimensions of its bitmap).

[PRESERVERATIO={ON|OFF}]

Not supported for images in PNG format. PRESERVERATIO=ON specifies that the aspect ratio (ratio of height to width) of the image should be preserved when it is scaled to the specified SIZE. This avoids distorting the appearance of the image. The image is scaled to the largest size possible within the bounds specified by SIZE for which the aspect ratio can be maintained. Supported for PDF and PS output. OFF does not maintain the aspect ratio. OFF is the default value.

The actual size of an image stored in a BLOB field may vary from image to image, and scaling the images to a designated size allows them to better fit into a columnar report.

**Note:** Images stored in a BLOB field are supported only for PDF, HTML, and DHTML output.

### ***Example:*** Inserting an Image From a BLOB Field Into a Report Column

The Microsoft SQL Server data source named *retaildetail* contains product information for a sports clothing and shoe retailer. The Microsoft SQL Server data source named *retailimage* has the same product ID field as *retaildetail* and has an image of each product stored in a field named *prodimage* whose data type is BLOB.

The following Master File describes the Microsoft SQL Server data source named `retaildetail`.

```
FILENAME=RETAILDETAIL, SUFFIX=SQLMSS , $
SEGMENT=SEG01, SEGTYPE=S0, $
  FIELDNAME=FOCLIST, ALIAS=FOCLIST, USAGE=I5, ACTUAL=I4, $
  FIELDNAME=PRODUCTID, ALIAS=ProductId, USAGE=A5, ACTUAL=A5,
    MISSING=ON, $
  FIELDNAME=DEPARTMENT, ALIAS=Department, USAGE=A10, ACTUAL=A10,
    MISSING=ON, $
  FIELDNAME=CATEGORY, ALIAS=Category, USAGE=A30, ACTUAL=A30,
    MISSING=ON, $
  FIELDNAME=SPORTS, ALIAS=Sports, USAGE=A30, ACTUAL=A30,
    MISSING=ON, $
  FIELDNAME=GENDER, ALIAS=Gender, USAGE=A10, ACTUAL=A10,
    MISSING=ON, $
  FIELDNAME=BRAND, ALIAS=Brand, USAGE=A25, ACTUAL=A25,
    MISSING=ON, $
  FIELDNAME=STYLE, ALIAS=Style, USAGE=A25, ACTUAL=A25,
    MISSING=ON, $
  FIELDNAME=COLOR, ALIAS=Color, USAGE=A25, ACTUAL=A25,
    MISSING=ON, $
  FIELDNAME=NAME, ALIAS=Name, USAGE=A80, ACTUAL=A80,
    MISSING=ON, $
  FIELDNAME=DESCRIPTION, ALIAS=Description, USAGE=A1000, ACTUAL=A1000,
    MISSING=ON, $
  FIELDNAME=PRICE, ALIAS=Price, USAGE=D7.2, ACTUAL=D8,
    MISSING=ON, $
```

The following Master File describes the Microsoft SQL Server data source named `retailimage`, which has the same product ID field as `retaildetail` and has an image of each product stored in a field named *prodimage* whose data type is BLOB.

```
FILENAME=RETAILIMAGE, SUFFIX=SQLMSS , $
SEGMENT=RETAILIMAGE, SEGTYPE=S0, $
  FIELDNAME=PRODUCTID, ALIAS=PRODUCTID, USAGE=A5, ACTUAL=A5, $
  FIELDNAME=PRODIMAGE, ALIAS=F02BLOB50000, USAGE=BLOB, ACTUAL=BLOB,
    MISSING=ON, $
```

The following request joins the two data sources and prints product names and prices with the corresponding image. The output is generated in DHTML format.











```
-* Rel 7705 DHTML and HTML supports including Image stored in
-* BLOB field in report column, heading, footing, subhead, or
-* subfoot
-* Rel 769 supports PDF format
JOIN PRODUCTID IN RETAILDETAIL TO PRODUCTID IN RETAILIMAGE
TABLE FILE RETAILDETAIL
HEADING CENTER
"Product List"
" "
PRINT NAME/A20 PRICE PRODIMAGE AS 'PICTURE'
BY PRODUCTID NOPRINT
BY NAME NOPRINT
ON NAME UNDER-LINE
ON TABLE SET PAGE NOPAGE
_*****
-* Lines between asterisk lines required for BLOB image support
-* for HTML and DHTML formats.
ON TABLE SET HTMLRENDERING AUTO
-* Required to support IE8 with images larger than 32K
ON TABLE SET HTMLARCHIVE ON
-*Required for image positioning in subheads in HTML reports
ON TABLE SET HTMLCSS ON
_*****
ON TABLE PCHOLD FORMAT DHTML
ON TABLE SET STYLE *
TYPE=REPORT,COLOR=BLUE,Font=ARIAL, GRID=OFF,$
TYPE=HEADING, SIZE = 18, COLOR=RED,$
TYPE=DATA,COLUMN=PRODIMAGE,IMAGE=(PRODIMAGE),SIZE=(1 1),$
ENDSTYLE
END
```

The image is placed in the report column using the following StyleSheet declaration, which names the image field, and establishes the size and position in the column for the image.

```
TYPE=DATA,COLUMN=PRODIMAGE,IMAGE=(PRODIMAGE),SIZE=(1 1),$
```

The partial output shows that DHTML format preserves the specified spacing.

## Product List


<u>NAME</u>	<u>PRICE</u>	<u>PICTURE</u>
		
Miziz 9 Spike Classi	94.00	
		
Kine Cal Ripken Mid	49.00	
		
Miziz 9 Spike Classi	89.00	
		
Adomas Excelsior Low	49.00	
		
3N3 Prometal Hi Base	84.00	
		
3N3 Prometal Hi Base	84.00	
		
Adomas Spinner 7 Lo	54.00	

The following request generates the output in HTML format.

```
-* Rel 7705 DHTML and HTML supports including Image stored in
-* BLOB field in report column, heading, footing, subhead, or
-* subfoot
-* Rel 769 supports PDF format
JOIN PRODUCTID IN RETAILDETAIL TO PRODUCTID IN RETAILIMAGE
TABLE FILE RETAILDETAIL
HEADING CENTER
"Product List"
" "
PRINT NAME/A20 PRICE PRODIMAGE AS 'PICTURE'
BY PRODUCTID NOPRINT
BY NAME NOPRINT
ON NAME UNDER-LINE
ON TABLE SET PAGE NOPAGE
_*****
-* Lines between asterisk lines required for BLOB image support
-* for HTML and DHTML formats.
ON TABLE SET HTMLRENDERING AUTO
-* Required to support IE8 with images larger than 32K
ON TABLE SET HTMLARCHIVE ON
-*Required for image positioning in subheads in HTML reports
ON TABLE SET HTMLCSS ON
_*****
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET STYLE *
TYPE=REPORT,COLOR=BLUE, GRID=OFF, FONT=ARIAL,$
TYPE=HEADING, SIZE = 18, COLOR=RED,$
TYPE=DATA,COLUMN=PRODIMAGE,IMAGE=(PRODIMAGE),SIZE=(1 1),$
ENDSTYLE
END
```

The partial output shows that the spacing is different because the browser removes blank spaces for HTML report output.

## Product List

<u>NAME</u>	<u>PRICE</u>	<u>PICTURE</u>
Mizia 9 Spike Classi	94.00	
Kine Cal Ripken Mid	49.00	
Mizia 9 Spike Classi	89.00	
Adomas Excelsior Low	49.00	
3N3 Prometal Hi Base	84.00	
3N3 Prometal Hi Base	84.00	








The following request generates the report output in PDF format.

```
-* Rel 7705 DHTML and HTML supports including Image stored in
-* BLOB field in report column, heading, footing, subhead, or
-* subfoot
-* Rel 769 supports PDF format
JOIN PRODUCTID IN RETAILDETAIL TO PRODUCTID IN RETAILIMAGE
TABLE FILE RETAILDETAIL
HEADING CENTER
"Product List"
" "
PRINT NAME/A20 PRICE PRODIMAGE AS 'PICTURE'
BY PRODUCTID NOPRINT
BY NAME NOPRINT
ON NAME UNDER-LINE
ON TABLE SET PAGE NOPAGE
_*****
-* Lines between asterisk lines required for BLOB image support
-* for HTML and DHTML formats.
ON TABLE SET HTMLRENDERING AUTO
-* Required to support IE8 with images larger than 32K
ON TABLE SET HTMLARCHIVE ON
-*Required for image positioning in subheads in HTML reports
ON TABLE SET HTMLCSS ON
_*****
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT,COLOR=BLUE, GRID=OFF,$
TYPE=HEADING, SIZE = 18, FONT = ARIAL, COLOR=RED,$
TYPE=DATA,COLUMN=PRODIMAGE,IMAGE=(PRODIMAGE),SIZE=(1 1),$
ENDSTYLE
END
```



The PDF partial output preserves specified spacing providing results similar to DHTML output.

## Product List

<u>NAME</u>	<u>PRICE</u>	<u>PICTURE</u>
Mizia 9 Spike Classi	94.00	
Kine Cal Ripken Mid	49.00	
Mizia 9 Spike Classi	89.00	
Adomas Excelsior Low	49.00	
3N3 Prometal Hi Base	84.00	
3N3 Prometal Hi Base	84.00	
Adomas Spinner 7 Lo	54.00	

**Example: Inserting an Image From a BLOB Field Into a Subheading**

The Microsoft SQL Server data source named retaildetail contains product information for a sports clothing and shoe retailer. The Microsoft SQL Server data source named retailimage has the same product ID field as retaildetail and has an image of each product stored in a field named *prodimage* whose data type is BLOB.

The following request joins the two data sources and prints product images in a subheading. The output is generated in DHTML format. It can also be generated in HTML or PDF format.

```

-* Rel 7705 DHTML and HTML supports including Image stored in
-* BLOB field in HTML report column, heading, footing, subhead,
-* or subfoot
-* Rel 769 supports PDF format
-*SET BASEURL='' - Required for embedded images to work. Overrides default
-* setting from WF Client.
SET BASEURL=''
JOIN PRODUCTID IN RETAILDETAIL TO PRODUCTID IN RETAILIMAGE
TABLE FILE RETAILDETAIL
HEADING CENTER
"Product Catalog"
" "

PRINT NAME NOPRINT PRODIMAGE NOPRINT
BY PRODUCTID NOPRINT
ON PRODUCTID SUBHEAD
" "

" ID: <10<PRODUCTID "
" Name: <10<NAME "
" Price: <7<PRICE "
" Image: "
" "
" "
" "
" "
" "

ON TABLE SET PAGE NOPAGE

_*****
-* Lines between asterisk lines required for BLOB image support
-* for HTML and DHTML formats.
ON TABLE SET HTMLEMBEDIMG AUTO
-* Required to support IE8 with images larger than 32K
ON TABLE SET HTMLARCHIVE ON
-*Required for image positioning in subheads in HTML reports
ON TABLE SET HTMLCSS ON
_*****
ON TABLE PCHOLD FORMAT DHTML
ON TABLE SET STYLE *
TYPE=REPORT,COLOR=BLUE,FONT = ARIAL,$
TYPE=HEADING, COLOR = RED, SIZE = 16, JUSTIFY=CENTER,$
TYPE=SUBHEAD,BY=PRODUCTID,IMAGE=(PRODIMAGE),SIZE=(1 1), POSITION=(+2 +1),$
ENDSTYLE
END

```

The partial output is.

### Product Catalog

ID: 101  
Name: Mizia 9 Spike Classic Mid G4 Metal Baseball Spike Mens  
Price: 94.00  
Image:



ID: 102  
Name: Kine Cal Ripken Mid Molded Baseball Cleat Mens  
Price: 49.00  
Image:



ID: 103  
Name: Mizia 9 Spike Classic Low G4 Metal Baseball Spike Mens  
Price: 89.00  
Image:



ID: 104  
Name: Adomas Excelsior Low Metal Baseball Cleat Mens  
Price: 49.00  
Image:



**Example: Sizing an Image From a BLOB Field**

The Microsoft SQL Server data source named *retaildetail* contains product information for a sports clothing and shoe retailer. The Microsoft SQL Server data source named *retailimage* has the same product ID field as *retaildetail* and has an image of each product stored in a field named *prodimage* whose data type is BLOB.

The following request joins the two data sources and displays the same image on three columns of output using different sizes and different PRESERVERATIO settings. Note that PRESERVERATIO=ON is not supported with images in PNG format.

The output is generated in DHTML format. It can also be generated in HTML or PDF format.

```

-* Rel 7705 DHTML and HTML supports including Image stored in
-* BLOB field in report column, heading, footing, subhead, or
-* subfoot
-* Rel 769 supports PDF format
JOIN PRODUCTID IN RETAILDETAIL TO PRODUCTID IN RETAILIMAGE
TABLE FILE RETAILDETAIL
PRINT PRODIMAGE AS ' ' PRODIMAGE AS ' ' PRODIMAGE AS ' '
BY STYLE NOPRINT
WHERE NAME CONTAINS 'Pant' OR 'Tank'
ON STYLE UNDER-LINE
ON TABLE SET PAGE NOPAGE
_*****
-* Lines between asterisk lines required for BLOB image support
-* for HTML and DHTML formats.
ON TABLE SET HTMLEMBEDIMG AUTO
-* Required to support IE8 with images larger than 32K
ON TABLE SET HTMLARCHIVE ON
-*Required for image positioning in subheads in HTML reports
ON TABLE SET HTMLCSS ON
_*****
ON TABLE PCHOLD FORMAT DHTML
ON TABLE SET STYLE *
TYPE=REPORT,COLOR=BLUE,FONT = ARIAL,$
TYPE=DATA,COLUMN=P1,IMAGE=(PRODIMAGE),SIZE=(.75 .75),$
TYPE=DATA,COLUMN=P2,IMAGE=(PRODIMAGE),SIZE=(.75 1),PRESERVERATIO=ON,$
TYPE=DATA,COLUMN=P3,IMAGE=(PRODIMAGE),SIZE=(.75 1),PRESERVERATIO=OFF,$
ENDSTYLE
END

```

Note that PRESERVERATIO=OFF is specified for the second column to preserve the image height and width ratio for that column even though the styling SIZE height specifies a different value than the first column image styling. In addition, PRESERVERATIO=OFF is specified for the third column, so for that column the image height to width ratio is not preserved and is rendered as specified by the styling SIZE height and width values specified in the request (FEX).

The partial output follows.



**Example: Inserting an Image From a BLOB Field in a Summary Report**

In order to insert an image from a BLOB field in a report that displays summary data, you must include two display commands in the request, a SUM command for the summary information and a PRINT or LIST command for displaying the image and any other detail data.

The Microsoft SQL Server data source named retaildetail contains product information for a sports clothing and shoe retailer. The Microsoft SQL Server data source named retailimage has the same product ID field as retaildetail and has an image of each product stored in a field named *prodimage* whose data type is BLOB.

The following request joins the two data sources. It contains two display commands, a SUM command and a PRINT command. The SUM command aggregates the total price for each category and displays this category name and total price in a subheading. The PRINT command displays the image for each item in the category along with its individual product number and price in a subfooting.

The output is generated in DHTML format. It can also be generated in HTML or PDF format.

```
-* Rel 7705 DHTML and HTML supports including images stored in
-* BLOB field in report column, heading, footing, subhead, or
-* subfoot
-* Rel 769 supports PDF format
SET PRINTPLUS=ON
JOIN PRODUCTID IN RETAILDETAIL TO PRODUCTID IN RETAILIMAGE
TABLE FILE RETAILDETAIL
HEADING CENTER
"Product Price Summary"
" "
SUM PRICE NOPRINT
BY CATEGORY NOPRINT
ON CATEGORY SUBHEAD
" Category: <CATEGORY "
" Total Price: <PRICE "
" "
```

```

PRINT PRICE NOPRINT PRODIMAGE NOPRINT
BY CATEGORY NOPRINT
BY PRODUCTID NOPRINT
ON PRODUCTID SUBFOOT
" "
" "
" "
" "
" "
" "
" Product #: <PRODUCTID "
" Name: <NAME "
" Price: <FST.PRICE "
ON TABLE SET PAGE NOPAGE
_*****
-* Lines between asterisk lines required for BLOB image support
-* for HTML and DHTML formats.
ON TABLE SET HTML EMBED IMG AUTO
-* Required to support IE8 with images larger than 32K
ON TABLE SET HTML ARCHIVE ON
-* Required for image positioning in subheads in HTML reports]
ON TABLE SET HTML CSS ON
_*****
ON TABLE PCHOLD FORMAT DHTML
ON TABLE SET STYLE *
TYPE=REPORT,COLOR=BLUE, FONT=ARIAL,$
TYPE=HEADING, COLOR=RED, SIZE=14, STYLE=BOLD, JUSTIFY=CENTER,$
TYPE=SUBHEAD, COLOR=RED, SIZE=12, STYLE=BOLD, JUSTIFY=CENTER,$
TYPE=SUBFOOT,BY=PRODUCTID,IMAGE=(PRODIMAGE),SIZE=(1 1), POSITION=(0 0),$
TYPE=SUBFOOT,BY=PRODUCTID,OBJECT=FIELD, ITEM=1, WRAP=5,$
ENDSTYLE
END

```

The output for the first category is:

### Product Price Summary

Category: Crib Shoes / Soft Bottoms  
Total Price: 106.00



Product #: 206  
Name: StonyBay Kids Mate Crib Shoe  
Price: 30.00



Product #: 207  
Name: StonyBay Kids Mate Crib Shoe  
Price: 30.00



Product #: 208  
Name: D2D Classic Dazzlin' Crib Shoe  
Price: 24.00



Product #: 209  
Name: ShoeTech KJ574 Crib  
Price: 22.00



**Reference: File Size and Compression Considerations For Images in BLOB Fields**

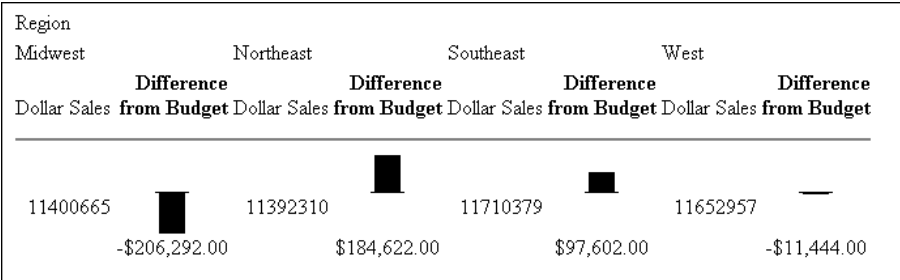
The actual size of an image stored in the BLOB field may vary from image to image, and scaling the images to a designated size allows them to better fit into a columnar report.

Files that contain many images can be large. Scaling the images to a smaller size using the SIZE attribute does not decrease the size of the file. Note also that using SET FILECOMPRESS=ON will not reduce the size of images in a PDF file, since images are already saved in compressed form.

**Associating Bar Graphs With Report Data**

To make PDF, HTML, DHTML, PPTX, PPT, and PS reports more powerful, you can insert visual representations of selected data directly into the report output. These visual representations are in the form of vertical or horizontal bar graphs that make relationships and trends among data more obvious. You can add the following:













- ☐ **Vertical Bar Graph.** You can apply a vertical bar graph to report columns associated with an ACROSS sort field. The report output displays a vertical bar graph in a new row above the associated data values, as shown in the following image.



Bar graphs that emanate above the zero line represent positive values, while bar graphs that emanate below the zero line represent negative values.

To see how each of these types of reports is generated, see the example following [How to Associate Data Visualization Bar Graphs With Report Columns](#) on page 1418.

- ❑ **Horizontal Bar Graph.** You can apply a horizontal bar graph to report columns. The report output displays a horizontal bar graph in a new column to the right of the associated data values, as shown in the following image.

<u>City</u>	<u>Budget Dollars</u>	<u>Dollar Sales</u>	<u>DIFFERENCE</u>	
Atlanta	\$4,247,597.00	\$4,100,107.00	\$147,490.00	
Boston	\$3,818,397.00	\$3,707,986.00	\$110,411.00	
Chicago	\$3,866,856.00	\$3,924,401.00	-\$57,545.00	
Houston	\$3,680,679.00	\$3,714,978.00	-\$34,299.00	
Los Angeles	\$3,669,484.00	\$3,772,014.00	-\$102,530.00	
Memphis	\$3,689,979.00	\$3,687,057.00	\$2,922.00	
New Haven	\$3,832,202.00	\$3,782,049.00	\$50,153.00	
New York	\$3,926,333.00	\$3,902,275.00	\$24,058.00	
Orlando	\$3,870,405.00	\$3,923,215.00	-\$52,810.00	
San Francisco	\$3,916,863.00	\$3,870,258.00	\$46,605.00	
Seattle	\$4,055,166.00	\$4,010,685.00	\$44,481.00	
St. Louis	\$3,646,838.00	\$3,761,286.00	-\$114,448.00	

Bar graphs that emanate to the right of the zero line represent positive values, while bar graphs that emanate to the left of the zero line represent negative values.

The length of each vertical or horizontal bar graph is proportional to the magnitude of its associated data value. The shortest bar graph is displayed for the value with the minimum magnitude, the longest bar graph for the value with the maximum magnitude, and bar graphs of varying length are displayed for each value within the minimum-maximum magnitude range. Notice in the figure above that a value of 147,490.00 produces a longer horizontal bar graph than a value of 50,153.00. Therefore, a complete row of vertical bar graphs or a complete column of horizontal bar graphs forms a bar chart.

You can only apply data visualization bar graphs to numeric report columns (integer, decimal, floating point single-precision, floating point double-precision, and packed). Bar graphs applied to alphanumeric, date, or text field formats are ignored. For details about assigning field formats, see the *Describing Data With WebFOCUS Language* manual.

You apply data visualization bar graphs to columns by adding a declaration to your WebFOCUS StyleSheet that begins with the GRAPHTYPE attribute. This attribute adds either a vertical or horizontal bar graph to the specified data.

**Note:** Data visualization bar graphs are not supported in a request that includes the OVER option.

**Reference: Formatting Options for Data Visualization Bar Graphs**

You can specify optional formatting attributes for data visualization bar graphs in the GRAPHTYPE declaration, for example, graph color, length, and width. The following table lists the formatting attributes and a description of each:

Formatting Attribute	Description
GRAPHBASE	Sets whether the scale should start from zero or the minimum value.
GRAPHCOLOR	Sets the color of the bar graphs.
GRAPHCOLORNEG	Sets a color for the bar graphs that represent negative values.
GRAPHLENGTH	<p>Sets the length of the longest bar graph. The value for GRAPHLENGTH determines the length in measurement units (inches, centimeters, and so on) of the longest bar graph in a vertical or horizontal bar graph.</p> <p>The length value is expressed in the current units, which is set using the UNITS StyleSheet attribute. The GRAPHLENGTH value is then converted into pixels.</p>
GRAPHSCALE	Specifies the relative bar graph scaling for multiple report columns under a common ACROSS sort field in which you have applied data visualization graphics. GRAPHSCALE is a report-level setting (TYPE=REPORT).
GRAPHWIDTH	Sets the width of the bar graphs. The width value is expressed in the current units. See GRAPHLENGTH, above, for more information about units.

## Syntax: How to Incorporate Data Visualization Formatting Attributes

```
TYPE=REPORT, [GRAPHSCALE={UNIFORM|DISTINCT}]
TYPE=DATA, GRAPHTYPE=DATA, [{COLUMN|ACROSSCOLUMN|FIELD}=identifier],
[GRAPHBASE={ZERO|MINIMUM},]
[GRAPHCOLOR={color|RGB({r g b|#hexcolor}),]
[GRAPHNEGCOLOR={color|RGB({r g b|#hexcolor}),]
[GRAPHLENGTH=lengthvalue,]
[GRAPHWIDTH=widthvalue,] $
```

**Note:** TYPE=DATA, GRAPHTYPE=DATA is the equivalent of GRAPHTYPE=DATA.

where:

### GRAPHBASE

Specifies whether the scale should start at zero (the default) or the minimum value. GRAPHBASE=MINIMUM makes it easier to compare values that are not close to zero. If negative values are present, GRAPHBASE=MINIMUM is ignored and will not be enabled.

GRAPHBASE=MINIMUM is visually indicated in the graph with a break in the bar (the small piece of the bar before the break symbolizes the compressed space between 0 and the minimum value).

**Note:** The minimum value is identified by a double bar of equal heights. For all values greater than the minimum, the top bar grows proportionally taller. If the minimum value is actually zero, having this double bar may make it look as if the minimum value is not zero. Therefore, you should use the default (GRAPHBASE=ZERO) in your procedure if you expect to have zero values in the column.

### GRAPHCOLOR

Specifies the color of the bar graphs. Black is the default color, if you omit this attribute from the declaration.

#### *color*

Is one of the supported color values. In addition to the supported named colors, HEX or RGB color values are valid options. For a list of supported values, see [Color Values in a Report](#) on page 1615.

#### RGB(*r g b*)

Specifies the font color using a mixture of red, green, and blue. (*r g b*) is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense. Note that using the three color components in equal intensities results in shades of gray.

*RGB(*#hexcolor*)*

Is the hexadecimal value for the color. For example, FF0000 is the hexadecimal value for red. The hexadecimal digits can be in upper or lowercase, and must be preceded by a pound sign (#).

*GRAPHNEGCOLOR*

Defines a color for the bar graphs that represent negative values.

*GRAPHLENGTH*

Specifies the length of the longest bar graph. The default length is 60 pixels for a vertical bar graph and 80 pixels for a horizontal bar graph.

*lengthvalue*

Sets the value used to display the vertical or horizontal bar graph for the maximum data value in the associated report column. This value must be a positive number.

This value is initially expressed in the current units (using the UNITS attribute). This value is then converted into the corresponding number of pixels.

*GRAPHSCALE*

Specifies the relative bar graph scaling for multiple report columns under a common ACROSS sort field in which you have applied data visualization graphics. GRAPHSCALE is a report-level setting (TYPE=REPORT).

*UNIFORM*

Scales each vertical bar graph based on the minimum and maximum values of the entire set of values compiled from each ACROSS column in which you have applied data visualization graphics

*DISTINCT*

Scales each vertical bar graph based on the distinct minimum and maximum values for each ACROSS column in which you have applied data visualization graphics.

*GRAPHWIDTH*

Specifies the width of the bar graphs in a report.

*widthvalue*

Sets the value used to display the width of the bar graphs in a report. This value must be a positive number.

This value is initially expressed in the current units (defined by the UNITS attribute). This value is then converted into the corresponding number of pixels.

**Syntax:**      **How to Associate Data Visualization Bar Graphs With Report Columns**

To add data visualization graphics to report output, add the following declaration to your WebFOCUS StyleSheet.

```
GRAPHTYPE=DATA, {COLUMN|ACROSSCOLUMN}=identifier, $
```

where:

**GRAPHTYPE=DATA**

Generates vertical or horizontal bar graphs for the data component of a report. Currently, you can only specify DATA as the report component.

**COLUMN**

Displays a horizontal bar graph to the right of the specified report column.

**ACROSSCOLUMN**

Displays a vertical bar graph above every occurrence of the data value associated with an ACROSS sort field.

*identifier*

Is any valid identifier. For details, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

You can define WHEN conditions and bar graph features associated with those conditions using StyleSheet syntax. For details, see [Formatting Report Data](#) on page 1611.












**Example: Generating Data Visualization Bar Graphs in a Report**

The following illustrates how to generate a bar graph for data in your report. Since this report is sorted with the BY field CITY, horizontal bar graphs display in the output. You can change this to vertical bars by changing the sort field to ACROSS CITY and the StyleSheet declaration to GRAPHTYPE=DATA, ACROSSCOLUMN=DIFFERENCE, \$. Bar graphs that represent positive values display in the color blue. Bar graphs that represent negative values display in the color red.

```
DEFINE FILE GGSales
DIFFERENCE/D7M=BUDDOLLARS-DOLLARS;
END

TABLE FILE GGSales
BY CITY
SUM BUDDOLLARS/D7M DOLLARS/D7M DIFFERENCE
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
GRID=OFF, $
GRAPHTYPE=DATA, COLUMN=N4, GRAPHCOLOR=BLUE, GRAPHNEGCOLOR=RED, $
ENDSTYLE
END
```

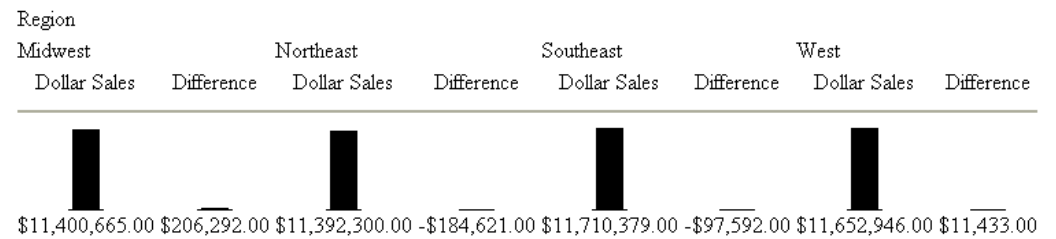
The output is:

<u>City</u>	<u>Budget Dollars</u>	<u>Dollar Sales</u>	<u>DIFFERENCE</u>	
Atlanta	\$4,247,587	\$4,100,107	\$147,480	
Boston	\$3,818,397	\$3,707,986	\$110,411	
Chicago	\$3,866,856	\$3,924,401	-\$57,545	
Houston	\$3,680,679	\$3,714,978	-\$34,299	
Los Angeles	\$3,669,484	\$3,772,003	-\$102,519	
Memphis	\$3,689,979	\$3,687,057	\$2,922	
New Haven	\$3,832,202	\$3,782,049	\$50,153	
New York	\$3,926,322	\$3,902,265	\$24,057	
Orlando	\$3,870,405	\$3,923,215	-\$52,810	
San Francisco	\$3,916,863	\$3,870,258	\$46,605	
Seattle	\$4,055,166	\$4,010,685	\$44,481	
St. Louis	\$3,646,838	\$3,761,286	-\$114,448	

Controlling Bar Graph Scaling in Horizontal (ACROSS) Sort Fields

You can apply vertical bar graphs to different columns above a common ACROSS sort field. The entire set of values for each column is grouped over an ACROSS sort field that has bar graphs applied. Therefore, the longest bar graph corresponds to the maximum value of the entire set of values.

This action is acceptable for separate column values that have ranges that are close. Many times, however, there is a marked discrepancy between the sets of values for separate columns. The following image illustrates such a discrepancy.



As you can see from the figure above, the values for the Dollar Sales field (\$11,392,310.00 to \$11,710,379.00) is much larger in magnitude than the set of values for the Difference field (\$206,292.00 to -\$184,622.00). Also notice that the vertical bar graphs associated with the Difference values all but disappear when graphed against the entire set of values.

To display separate vertical bar graphs based on the set of values for each column, use the GRAPHSCALE StyleSheet attribute. This attribute modifies data visualization graphics to use the minimum and maximum values for each column below a common ACROSS sort field to construct a distinct vertical bar graph.

Syntax: How to Set Orientation for Visualization Bars

The VISBARORIENT parameter enables you to set horizontal or vertical orientation for visualization bars for ACROSS columns.

```
SET VISBARORIENT = {H|V}
```

where:

H  
Indicates horizontal bar orientation for visualization bars.

V  
Indicates vertical bar orientation for visualization bars. This is the default value.



































**Example: Setting Orientation for Visualization Bars**

The following report creates vertical bars for the ACROSS column values (DOLLARS, BUDDOLLARS, UNITS, BUDUNITS).

```
SET VISBARORIENT=V
TABLE FILE GGSales
SUM DOLLARS BUDDOLLARS UNITS BUDUNITS
BY REGION
ACROSS CATEGORY
WHERE CATEGORY EQ 'Coffee' OR 'Food'
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
TYPE =REPORT , BORDER=light,$
GRAPHTYPE=DATA, ACROSSCOLUMN=DOLLARS,
GRAPHCOLOR=GREEN,BACKCOLOR=RGB(#ffff00), $
GRAPHTYPE=DATA, ACROSSCOLUMN=BUDDOLLARS, GRAPHCOLOR=RGB(255 0 0), $
GRAPHTYPE=DATA, ACROSSCOLUMN=UNITS, GRAPHCOLOR=blue, $
GRAPHTYPE=DATA, ACROSSCOLUMN=BUDUNITS, GRAPHCOLOR=thistle, $
ENDSTYLE
END
```

The output is:

Region	Coffee				Food			
	Dollar Sales	Budget Dollars	Unit Sales	Budget Units	Dollar Sales	Budget Dollars	Unit Sales	Budget Units
Midwest	 4178513	 4086032	 332777	 335526	 4338271	 4220721	 341414	 339263
Northeast	 4164017	 4252462	 335778	 335920	 4379994	 4453907	 353368	 351431
Southeast	 4415408	 4431429	 350948	 355693	 4308731	 4409288	 349829	 351509
West	 4473517	 4523963	 356763	 358784	 4202337	 4183244	 340234	 335361

The following report creates horizontal bars for the ACROSS column values ( DOLLARS and BUDDOLLARS).

```
SET VISBARORIENT=H
TABLE FILE GGSales
SUM DOLLARS BUDDOLLARS
BY REGION
ACROSS CATEGORY
WHERE CATEGORY EQ 'Coffee' OR 'Food'
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
TYPE =REPORT ,BORDER=light,$
GRAPHTYPE=DATA, ACROSSCOLUMN=DOLLARS,
GRAPHCOLOR=GREEN,BACKCOLOR=RGB(#ffff00), $
GRAPHTYPE=DATA, ACROSSCOLUMN=BUDDOLLARS, GRAPHCOLOR=RGB(255 0 0), $
ENDSTYLE
END
```

The output is:

Region	Category							
	Coffee				Food			
	Dollar Sales		Budget Dollars		Dollar Sales		Budget Dollars	
Midwest	4178513		4086032		4338271		4220721	
Northeast	4164017		4252462		4379994		4453907	
Southeast	4415408		4431429		4308731		4409288	
West	4473517		4523963		4202337		4183244	

Applying Scaling to Data Visualization Bar Graphs

The GRAPHSCALE parameter specifies the relative bar graph scaling for multiple report columns under a common ACROSS sort field in which you have applied data visualization graphics. GRAPHSCALE can only be set for an entire report (TYPE=REPORT).

Syntax: How to Apply Scaling to Data Visualization Bar Graphs

```
TYPE=REPORT, GRAPHSCALE={UNIFORM|DISTINCT}
```

where:

```
TYPE=REPORT
```

Specifies that the declaration applies to the entire report, and not to a specific bar graph within the report.

```
GRAPHSCALE
```

Specifies the relative bar graph scaling for multiple report columns under a common ACROSS sort field in which you have applied data visualization graphics. GRAPHSCALE is a report-lever setting (TYPE=REPORT).

UNIFORM

Scales each vertical bar graph based on the minimum and maximum values of the entire set of values compiled from each ACROSS column in which you have applied data visualization graphics.

DISTINCT

Scales each vertical bar graph based on the distinct minimum and maximum values for each ACROSS column in which you have applied data visualization graphics.

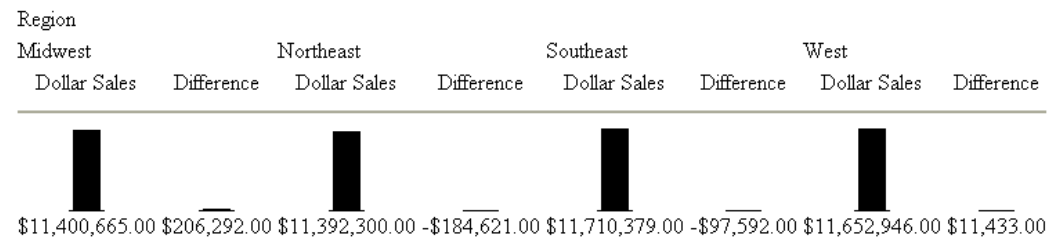
Example: Using GRAPHSCALE to Display Distinct Vertical Bar Graphs

The following report request displays vertical bar graphs for two columns (DOLLARS and DIFFERENCE) associated with a common ACROSS field (REGION):

```
DEFINE FILE GGSales
Difference/D12.2M=DOLLARS-BUDDOLLARS;
END

TABLE FILE GGSales
SUM DOLLARS/D12.2M Difference
ACROSS REGION
ON TABLE SET STYLE *
TYPE=REPORT,GRID=OFF,$
GRAPHTYPE=DATA, ACROSSCOLUMN=N1,$
GRAPHTYPE=DATA,ACROSSCOLUMN=N2,$
ENDSTYLE
END
```

This request produces the following report output:



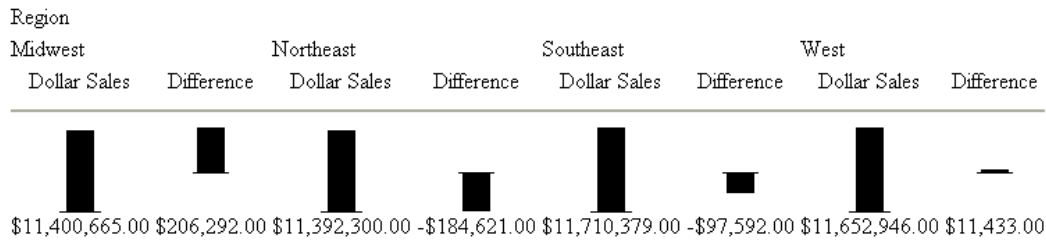
Since the GRAPHSCALE attribute is not specified, the default setting UNIFORM is applied to the report. This setting uses the entire set of values (values from Dollar Sales and Difference) to plot the bar graphs for both columns.

The following request is the same as the above request, except it has the GRAPHSCALE=DISTINCT attribute included in the StyleSheet.

```
DEFINE FILE GGSales
Difference/D12.2M=DOLLARS-BUDDOLLARS;
END

TABLE FILE GGSales
SUM DOLLARS/D12.2M Difference
ACROSS REGION
ON TABLE SET STYLE *
GRAPHTYPE=DATA, ACROSSCOLUMN=N1,$
GRAPHTYPE=DATA, ACROSSCOLUMN=N2,$
TYPE=REPORT, GRAPHSCALE=DISTINCT,$
ENDSTYLE
END
```

Notice the difference in the output:



Now each bar graph is plotted based on the set of values for each field.

Working With Mailing Labels and Multi-Pane Pages

You can print sheets of laser printer mailing labels by dividing each page into a matrix of sub-pages, each corresponding to a single label. Each page break in the report positions the printer at the top of the next label.

Multi-pane printing places a whole report on a single printed page. You can create columns or rows so that when text overflows on one page, it appears in the next column or row on the same page rather than on the next page.

These features apply to a PDF or PS report.

**Reference: Attributes for Mailing Labels and Multi-Pane Printing**

In addition to the attributes in the table, you can use standard margin attributes (for example, LEFTMARGIN or TOPMARGIN) to position the entire sheet of labels at once, creating an identical margin for each sheet.

Attribute	Description	Applies to
PAGEMATRIX	Sets the number of columns and rows of labels on a page.	PDF PS
ELEMENT	Sets the width and height of each label, expressed in the unit of measurement specified by the UNITS parameter.	PDF PS
GUTTER	Sets the horizontal and vertical distance between each label, expressed in the unit of measurement specified by the UNITS parameter.	PDF PS
MATRIXORDER	Sets the order in which the labels are printed.	PDF PS
LABELPROMPT	Sets the position of the first label on the mailing label sheet.	PDF PS

**Procedure: How to Set Up a Report to Print Mailing Labels**

1. Create the label as a page heading.
2. Sort the labels but use NOPRINT to suppress sort field display. Only the fields embedded in the page heading will print.
3. Insert a page break on a sort field to place each new field value on a separate label.
4. Suppress default page numbers and associated blank lines from the beginning of each page (SET PAGE-NUM=NOPAGE).

**Syntax:**      **How to Print Mailing Labels or a Multi-Pane Report**

```
[TYPE=REPORT,] PAGEMATRIX=(c r), ELEMENT=(w h), [GUTTER=(x y),]  
[MATRIXORDER={VERTICAL|HORIZONTAL},] [LABELPROMPT={OFF|ON},] $
```

where:

**TYPE=REPORT**

Applies the settings to the entire report. Not required, as it is the default.

*c*

Is the number of columns of labels across the page.

Enclose the values *c* and *r* in parentheses, and do not include a comma between them.

*r*

Is the number of rows of labels down the page.

*w*

Is the width of each label.

Enclose the values *w* and *h* in parentheses, and do not include a comma between them.

*h*

Is the height of each label.

**GUTTER**

Is the distance between each label.

*x*

Is the horizontal distance between each label.

Enclose the values *x* and *y* in parentheses, and do not include a comma between them.

*y*

Is the vertical distance between each label.

**MATRIXORDER**

Is the order in which the labels are printed.

**VERTICAL**

Prints the labels down the page.

**HORIZONTAL**

Prints the labels across the page.

**LABELPROMPT**

Is the position of the first label on the mailing label sheet.

**OFF**

Starts the report on the first label on the sheet. OFF is the default value.

**ON**

Prompts you at run time for the row and column number at which to start printing. All remaining labels follow consecutively. This feature allows partially used sheets of labels to be re-used.

**Example: Printing Mailing Labels**

The following report prints on 8 1/2 x 11 sheets of address labels.

```
SET ONLINE-FMT = PDF
TABLE FILE EMPLOYEE
BY LAST_NAME NOPRINT BY FIRST_NAME NOPRINT
ON FIRST_NAME PAGE-BREAK
HEADING
"<FIRST_NAME <LAST_NAME"
"<ADDRESS_LN1"
"<ADDRESS_LN2"
"<ADDRESS_LN3"
ON TABLE SET PAGE-NUM NOPAGE
ON TABLE SET STYLE LABEMP
END
```

The labels have the following dimensions, defined in the StyleSheet LABEMP:

```
UNITS=IN, PAGESIZE=LETTER, LEFTMARGIN=0.256, TOPMARGIN=0.5,
PAGEMATRIX=(2 5), ELEMENT=(4 1), GUTTER=(0.188 0), $
```

The first page of labels prints as follows:

JOHN BANNING  
160 LOMBARDO AVE.  
APT 4C  
FREEPORT NY 11520

DIANE JONES  
  
235 MURRAY HIL PKWY  
RUTHERFORD NJ 07073

ROSEMARIE BLACKWOOD  
MRS. P. JONES  
3704 FARRAGUT RD.  
BROOKLYN NY 11210

JOHN MCCOY  
ASSOCIATED  
2 PENN PLAZA  
NEW YORK NY 10001

BARBARA CROSS  
APT 2G  
147-15 NORTHERN BLD  
FLUSHING NY 11354

ROGER MCKNIGHT  
APT 4D  
117 HARRISON AVE.  
ROSELAND NJ 07068

MARY GREENSPAN  
  
13 LINDEN AVE.  
JERSEY CITY NJ 07300

ANTHONY ROMANS  
  
271 PRESIDENT ST.  
FREEPORT NY 11520

JOAN IRVING  
APT 2J  
123 E 32 ST.  
NEW YORK NY 10001

MARY SMITH  
ASSOCIATED  
2 PENN PLAZA  
NEW YORK NY 10001



**Example: Printing a Multi-Pane Report**

This request divides the first report page in two columns so that the second report page appears in the second column of the first page. A PAGE-BREAK creates a multi-page report for the purpose of this example.

```

SET ONLINE-FMT = PDF
TABLE FILE EMPLOYEE
PRINT LAST_NAME AND CURR_SAL BY

DEPARTMENT
ON DEPARTMENT PAGE-BREAK
HEADING
"PAGE <TABPAGE NO>"
ON TABLE SET STYLE *
UNITS=IN, PAGESIZE=LETTER, PAGEMATRIX=(2 1), ELEMENT=(3.5 8.0),
MATRIXORDER=VERTICAL, $
TYPE=REPORT, SIZE=8, $
ENDSTYLE
END

```

The report prints as:

PAGE 1			PAGE 2		
DEPARTMENT	LAST_NAME	CURR_SAL	DEPARTMENT	LAST_NAME	CURR_SAL
MIS	SMITH	\$13,200.00	PRODUCTION	STEVENS	\$11,000.00
	JONES	\$18,480.00		SMITH	\$9,500.00
	MCCOY	\$18,480.00		BANNING	\$29,700.00
	BLACKWOOD	\$21,780.00		IRVING	\$26,862.00
	GREENSPAN	\$9,000.00		ROMANS	\$21,120.00
	CROSS	\$27,062.00		MCKNIGHT	\$16,100.00



## Using Headings, Footings, Titles, and Labels

---

After you have selected the data for a report, you can make it more meaningful by adding headings, footings, titles, and labels. Headings and footings supply key information, such as the purpose of a report and its audience. They also provide structure, helping the user navigate to the detail sought. Titles and labels identify individual pieces of data, ensuring correct interpretation. These components supply context for data and enhance the visual appeal of a report.

For information about adding contextual information to specialized report types, see [Creating a Graph](#) on page 1657, [Creating Financial Reports With Financial Modeling Language \(FML\)](#) on page 1729, and [Creating a Free-Form Report](#) on page 1809.

### In this chapter:

- ☐ [Creating Headings and Footings](#)
- ☐ [Including an Element in a Heading or Footing](#)
- ☐ [Displaying Syntax Components in Heading Objects](#)
- ☐ [Repeating Headings and Footings on Panels in PDF Report Output](#)
- ☐ [Customizing a Column Title](#)
- ☐ [Controlling Column Title Underlining Using a SET Command](#)
- ☐ [Controlling Column Title Underlining Using a StyleSheet Attribute](#)
- ☐ [Creating Labels to Identify Data](#)
- ☐ [Formatting a Heading, Footing, Title, or Label](#)
- ☐ [Applying Font Attributes to a Heading, Footing, Title, or Label](#)
- ☐ [Adding Borders and Grid Lines](#)
- ☐ [Justifying a Heading, Footing, Title, or Label](#)
- ☐ [Choosing an Alignment Method for Heading and Footing Elements](#)
- ☐ [Aligning a Heading or Footing Element in an HTML, XLSX, EXL2K, PDF, PPTX, or DHTML Report](#)
- ☐ [Aligning a Heading or Footing Element Across Columns in an HTML or PDF Report](#)
- ☐ [Aligning Content in a Multi-Line Heading or Footing](#)
- ☐ [Positioning Headings, Footings, or Items Within Them](#)

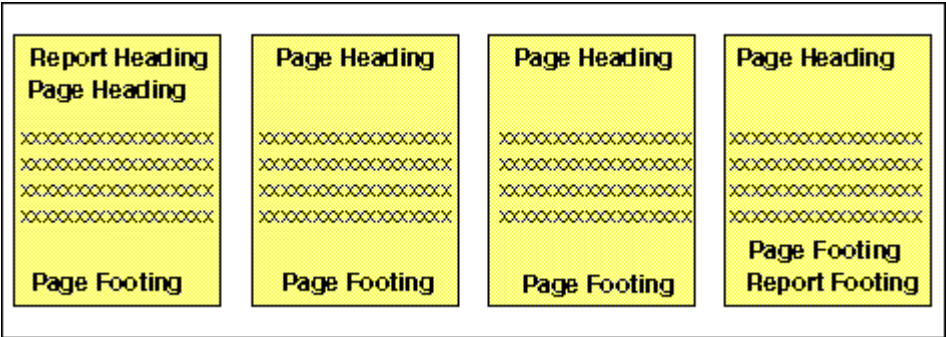
- ❑ [Controlling the Vertical Positioning of a Heading or Footing](#)
- ❑ [Placing a Report Heading or Footing on Its Own Page](#)

## Creating Headings and Footings

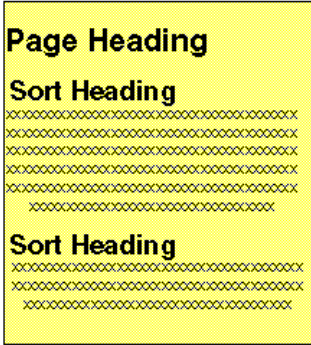
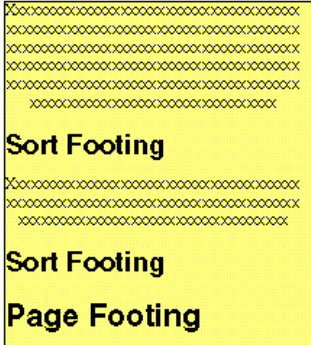
There are several types of headings and footings:

- ❑ Report titles. These are titles you define that display in your browser's title bar when you run a report or graph in HTML or, as the worksheet tab name in an EXL2K report. For details see, [Creating a Custom Report or Worksheet Title](#) on page 1436.
- ❑ A report heading, which appears at the top of the first page of a report and a report footing, which appears on the last page of a report. For details on report headings and footings, see [Creating a Report Heading or Footing](#) on page 1438.
- ❑ A page heading, which appears at the top of every page of a report and a page footing, which appears at the bottom of every page of a report. For details on page headings and footings, see [Creating a Page Heading or Footing](#) on page 1445.
- ❑ A sort heading, which appears in the body of a report to identify the beginning of a group of related data. And a sort footing, which appears in the body to identify the end of a group of related data. For details on sort headings and footings, see [Creating a Sort Heading or Footing](#) on page 1456.

The following sample report contains a report heading at the beginning of the report and a report footing at the end of the report. It also contains a page heading and page footing on every page of the report.



The following sample report contains sort headings and sort footings, as well as, a page heading and page footing for reference.

<p>A sort heading looks like this:</p> 	<p>A sort footing looks like this:</p> 
--	---

### Limits for Headings and Footings

The following limitations apply to report headings and footings, page headings and footings, and sort headings and footings:

- ☐ The space for headings, footings, subheadings, and subfootings is allocated dynamically. WebFOCUS imposes no limit on the amount of space used.
- ☐ The maximum number of sort headings plus sort footings in one request is 64.
- ☐ The maximum limit of nested headings is 64.
- ☐ If your code for a single heading or footing line is broken into multiple lines in the report request, you can indicate that they are all a single line of heading using the <OX spot marker. For more information, see [Extending Heading and Footing Code to Multiple Lines in a Report Request](#) on page 1434.
- ☐ A maximum of 128 objects, text items or embedded fields, can be placed in a heading, footing, subhead, or subfoot.
- ☐ For PDF and Postscript reports, the heading or footing lines must fit within the maximum report width to be displayed properly. Also, in order for the report body to be displayed, the number of heading or footing lines must leave room on the page for at least one detail line (including column titles).

## Extending Heading and Footing Code to Multiple Lines in a Report Request

A single line heading or footing code, between double quotation marks, can be a maximum of 32K characters. However, in some editors the maximum length of a line of code in a procedure is 80 characters. In cases like this, you can use the <OX spot marker to continue your heading onto the next line. The heading or footing content and spacing appears exactly as it would if typed on a single line.

Even if you do not need to extend your code beyond the 80-character line limit, this technique offers convenience, since shorter lines may be easier to read on screen and to print on printers.

### ***Procedure:*** How to Extend Heading or Footing Code to Multiple Lines in a Report Request

To extend the length of a single-line heading or footing beyond 80 characters:

1. Begin the heading or footing with double quotation marks (").
2. Split the heading or footing content into multiple lines of up to 76 characters each, using the <OX spot marker at any point up to the 76th character to continue your heading onto the next line. (The four remaining spaces are required for the spot marker itself, and a blank space preceding it.)
3. The heading or footing line can contain a maximum of 410 characters, with each line ending in an <OX spot marker.
4. Place the closing double quotation marks at the end of the final line of heading or footing code.

You can use this technique to create a report heading or footing, page heading or footing, or sort heading or footing of up to 410 characters.

### ***Example:*** Extending Heading and Footing Code to Multiple Lines in a Report Request

This request creates a sort heading coded on two lines. The <OX spot marker positions the first character on the continuation line immediately to the right of the last character on the previous line. (No spaces are inserted between the spot marker and the start of a continuation line.)

```
SET ONLINE-FMT = HTML
SET PAGE-NUM = OFF
JOIN STORE_CODE IN CENTCOMP TO STORE_CODE IN CENTORD
```

```

TABLE FILE CENTCOMP
HEADING
"Century Corporation Orders Report"
PRINT PROD_NUM QUANTITY LINEPRICE
BY STORE_CODE NOPRINT
BY ORDER_NUM
ON STORE_CODE SUBHEAD
"Century Corporation orders for store <STORENAME <0X
  (store # <STORE_CODE|) in <STATE|."
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
TYPE=HEADING, FONT='ARIAL', STYLE=BOLD, $
TYPE=SUBHEAD, OBJECT=FIELD, ITEM=2, STYLE=ITALIC, $
TYPE=SUBHEAD, OBJECT=FIELD, ITEM=3, STYLE=BOLD, $
ENDSTYLE
END

```

The partial output is:

### Century Corporation Orders Report

<u>Order Number:</u>	<u>Product Number#:</u>	<u>Quantity:</u>	<u>Line Total</u>
Century Corporation orders for store Audio Expert (store # 1003CA) in CA.			
74610	1012	268	\$222,234.35
	1028	323	\$29,215.62
	1032	339	\$27,208.12
	1034	339	\$154,943.38
	1036	339	\$93,266.39
Century Corporation orders for store Audio Expert (store # 1003CO) in CO.			
39274	1006	179	\$54,201.50
	1008	179	\$29,034.16
	1020	179	\$45,457.59
	1032	400	\$30,861.24
	1034	400	\$143,744.49
Century Corporation orders for store Audio Expert (store # 1003CT) in CT.			
35995	1008	13	\$2,132.45
	1020	13	\$3,895.38
	1022	146	\$52,159.13
	1024	146	\$45,306.84
	1030	146	\$18,525.49

**Tip:** You can use this technique to create a heading of up to 410 characters. Although demonstrated here for a sort heading, you can use this technique with any heading or footing line.

### Creating a Custom Report or Worksheet Title

You can create a report title that:

- ☐ Overrides the default report title (FOCUS Report) that appears in the title bar of your browser in an HTML report or graph.
- ☐ Replaces the default worksheet tab name with the name you specify in an EXL2K report.

The worksheet tab names for an Excel Table of Contents report are the BY field values that correspond to the data on the current worksheet. If the user specifies the TITLETEXT keyword in the stylesheet, it will be ignored.

- ☐ Excel limits the length of worksheet titles to 31 characters. The following special characters cannot be used: ':', '?', '\*', and '/'.
- ☐ If you want to use date fields as the bursting BY field, you can include the - character instead of the / character. The - character is valid in an Excel tab title. However, if you do use the / character, WebFOCUS will substitute it with the - character.

### **Syntax:** How to Create a Custom Report Title

Add the following declaration to your WebFOCUS StyleSheet:

```
TYPE=REPORT, TITLETEXT='title', $
```

where:

*title*

Is the text for your title.

The maximum number of characters for:

- ☐ The worksheet tab name in an EXL2K report is 31. Any text that exceeds 31 characters will be truncated.
- ☐ The browser title for an HTML report or graph is 95. This is a limit imposed by the browser.

Text specified in the title is placed in the file as is and is not encoded. Special characters, such as <, >, and &, should not be used since they have special meaning in HTML and may produce unpredictable results.



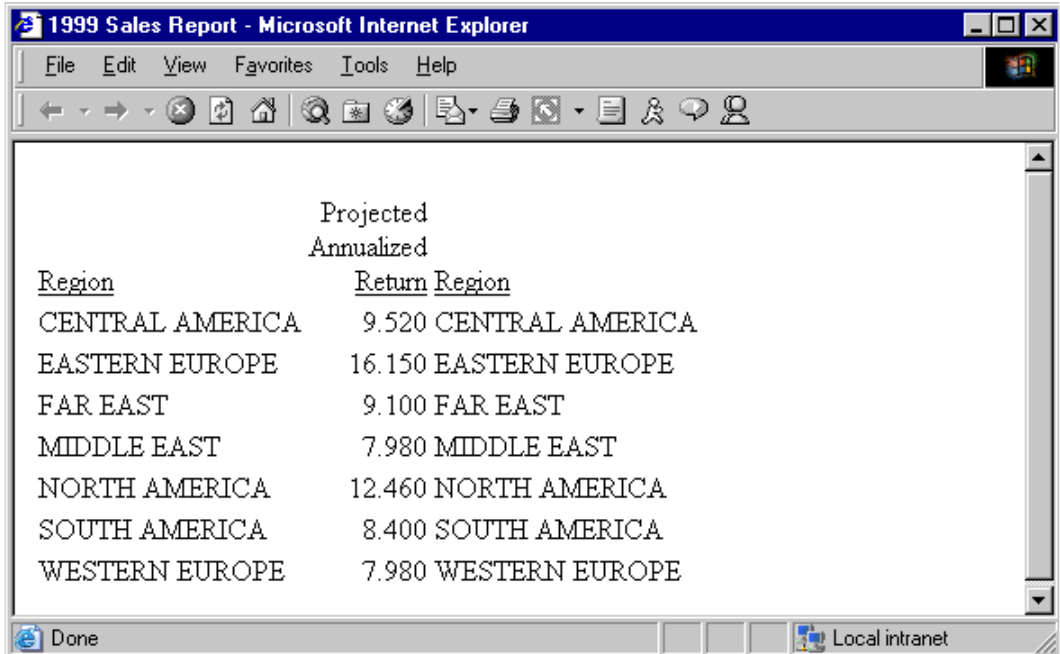
**Note:** The words "Microsoft Internet Explorer" are always appended to any HTML report title.

**Example:**    **Creating a Custom Report Title in an HTML Report**

The following illustrates how you can replace the default report title in an HTML report using the TITLETEXT attribute in your StyleSheet.

```
TABLE FILE SHORT
SUM PROJECTED REGION
BY REGION
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, TITLETEXT='1999 Sales Report', $
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:



The screenshot shows a Microsoft Internet Explorer window titled "1999 Sales Report - Microsoft Internet Explorer". The browser's address bar is empty, and the status bar at the bottom indicates "Done" and "Local intranet". The main content area displays a report with the following data:

<u>Region</u>	<u>Projected Annualized Return</u>	<u>Region</u>
CENTRAL AMERICA	9.520	CENTRAL AMERICA
EASTERN EUROPE	16.150	EASTERN EUROPE
FAR EAST	9.100	FAR EAST
MIDDLE EAST	7.980	MIDDLE EAST
NORTH AMERICA	12.460	NORTH AMERICA
SOUTH AMERICA	8.400	SOUTH AMERICA
WESTERN EUROPE	7.980	WESTERN EUROPE

**Example:**    **Creating a Custom Report Title in an EXL2K Report**

The following illustrates how you can replace the default worksheet tab name in an EXL2K report using the TITLETEXT attribute in your StyleSheet.

```
TABLE FILE SHORT
SUM PROJECTED_RETURN
BY REGION
ON TABLE PCHOLD FORMAT EXL2K
ON TABLE SET STYLE *
TYPE=REPORT, TITLETEXT='1999 Sales Report', $
ENDSTYLE
END
```

The output is:

	A	B	C	D	E
		Projected Annualized Return			
1	Region				
2	CENTRAL AMERICA	9.520			
3	EASTERN EUROPE	16.150			
4	FAR EAST	9.100			
5	MIDDLE EAST	7.980			
6	NORTH AMERICA	12.460			
7	SOUTH AMERICA	8.400			
8	WESTERN EUROPE	7.980			
9					

1999 Sales Report

**Creating a Report Heading or Footing**

A report heading appears before the first page and is one of the most important components of a report. It provides a unique name to a report and identifies its purpose or content. A short, single-line report heading may meet the needs of your user, or you may include multiple lines of appropriate information.

A report footing appears after the last page of a report. You might add a report footing to signal the end of data so the user knows that the report is complete. A report footing can also provide other information, such as the author of the report.

A report heading or footing can include text, fields, Dialogue Manager variables, images, and spot markers.

**Syntax:**      **How to Create a Report Heading**

Include the following syntax in a request. Each heading or footing line must begin and end with a double quotation mark.

```
ON TABLE [PAGE-BREAK AND] SUBHEAD
  "content ... "
["content ... "]
.
.
.
["content ... "]
```

where:

**PAGE-BREAK**

Is an optional command that creates the report heading on the first page by itself, followed by the page or pages of data. If you do not use PAGE-BREAK, the report heading appears on the first page of the report, followed by a page heading if one is supplied, and column titles. For related information, see [Placing a Report Heading or Footing on Its Own Page](#) on page 1605.

**SUBHEAD**

Is the command required to designate a report heading.

**content**

Heading or footing content can include the following elements, between double quotation marks. (If the ending quotation mark is omitted, all subsequent lines of the request are treated as part of the report heading.)

**text**

Is text that appears on the first page of a report. You can include multiple lines of text.

The text must start on a line by itself, following the SUBHEAD command.

Text can be combined with variables and spot markers.

For related information, see [Limits for Headings and Footings](#) on page 1433.

**variable**

Can be any one or a combination of the following:

**Fields** (real data source fields, virtual fields created with the DEFINE command in a Master File or report request, calculated values created with the COMPUTE command in a request, or a system field such as TABPAGENO). You can qualify data source fields with certain prefix operators.

**Dialogue Manager variables.**

[Images](#). You can include images in a heading or footing.

For details, see [Including an Element in a Heading or Footing](#) on page 1469.

### *spot marker*

Enables you to position items, to identify items to be formatted, and to extend code beyond the 80-character line limit of some text editors.

`<+0>` divides a heading or footing into items for formatting. For details, see [Identifying a Heading, Footing, Title, or FML Free Text](#) on page 1191.

`</n` specifies skipped lines. For details, see [Controlling the Vertical Positioning of a Heading or Footing](#) on page 1598.

`<-n` to position the next character on the line. For details, see [Using Spot Markers to Refine Positioning](#) on page 1593.

`<0x` continues a heading or footing specification on the next line of the request. For details, see [Extending Heading and Footing Code to Multiple Lines in a Report Request](#) on page 1434.

**Note:** When a closing spot marker is immediately followed by an opening spot marker (`><`), a single space text item will be placed between the two spot markers (`> <`). This must be considered when applying formatting.

### [Blank lines](#)

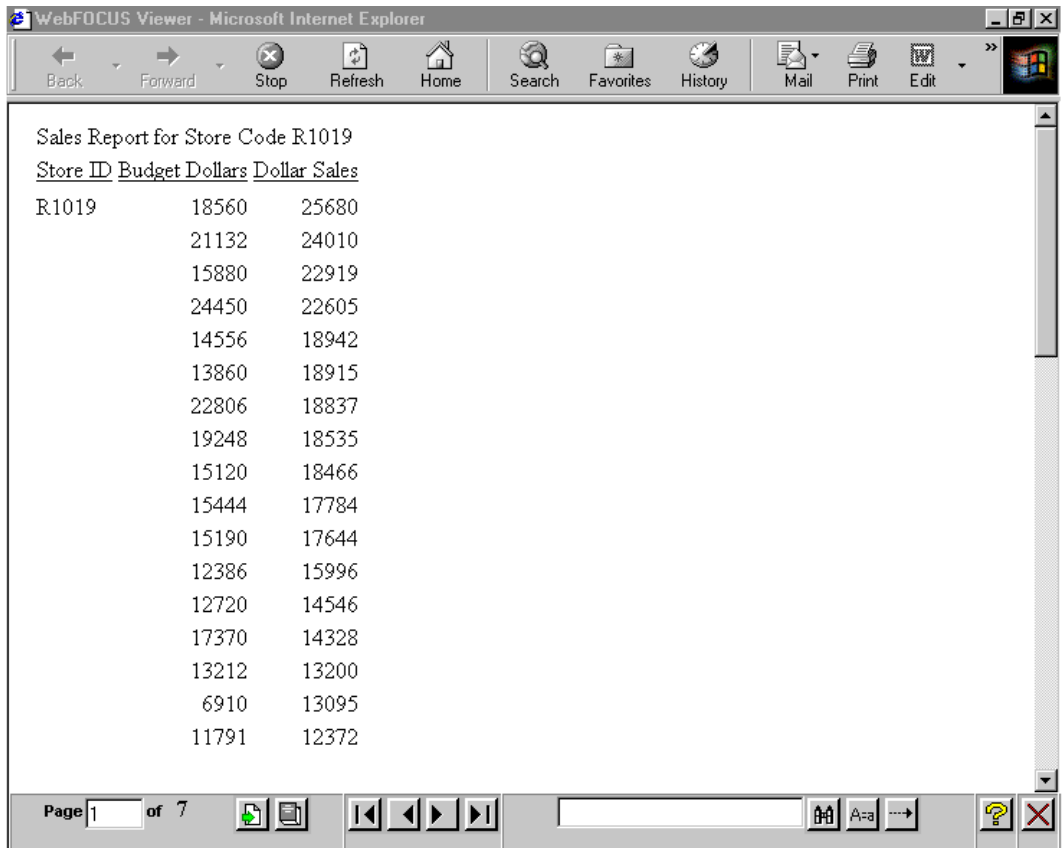
If you omit all text, variables, and spot markers, you have a blank heading or footing line (for example, " ") which you can use to skip a line in the heading or footing. (You can also skip a line using a vertical spot marker, such as `</1.`)

### **Example:** Creating a Single-Line Report Heading

This request creates a single-line report heading that identifies the content of the report.

```
TABLE FILE GGSALES
PRINT BUDDOLLARS DOLLARS
BY STCD
WHERE STCD EQ 'R1019'
ON TABLE SUBHEAD
  "Sales Report for Store Code R1019"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET WEBVIEWER ON
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF,$
ENDSTYLE
END
```

The output illustrates the placement of a report heading on a multi-page HTML report. The report heading is at the top of the first page.

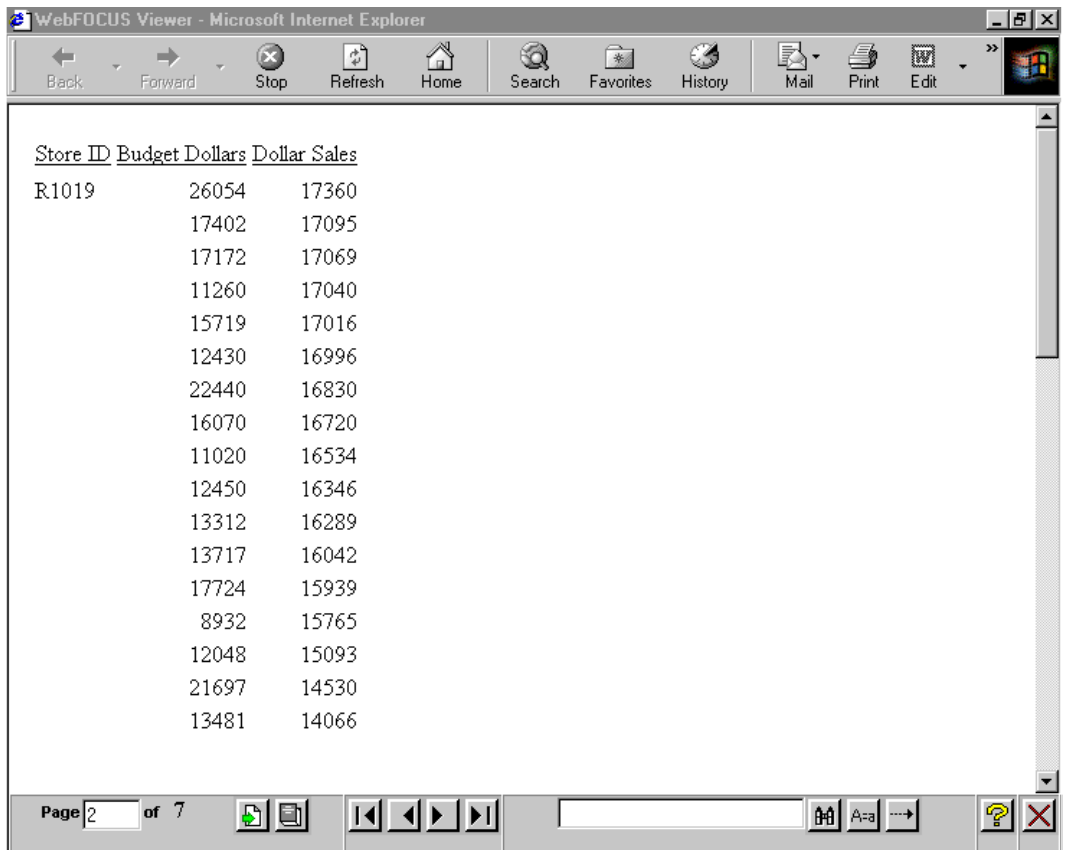


Sales Report for Store Code R1019

<u>Store ID</u>	<u>Budget Dollars</u>	<u>Dollar Sales</u>
R1019	18560	25680
	21132	24010
	15880	22919
	24450	22605
	14556	18942
	13860	18915
	22806	18837
	19248	18535
	15120	18466
	15444	17784
	15190	17644
	12386	15996
	12720	14546
	17370	14328
	13212	13200
	6910	13095
	11791	12372

Page 1 of 7

Subsequent pages do not contain a heading.



<u>Store ID</u>	<u>Budget Dollars</u>	<u>Dollar Sales</u>
R1019	26054	17360
	17402	17095
	17172	17069
	11260	17040
	15719	17016
	12430	16996
	22440	16830
	16070	16720
	11020	16534
	12450	16346
	13312	16289
	13717	16042
	17724	15939
	8932	15765
	12048	15093
	21697	14530
	13481	14066

**Tip:** If you do not see the navigation arrows, click the maximize button.

**Syntax:**      **How to Create a Report Footing**

Include the following syntax in a request. Each heading or footing line must begin and end with a double quotation mark.

```
ON TABLE [PAGE-BREAK AND] SUBFOOT
  "content ... "
["content ... "]
.
.
.
["content ... "]
```

where:

**PAGE-BREAK**

Is an optional command that creates the report footing after the last page by itself. If you do not include PAGE-BREAK, the report footing appears as the last line of the report. For related information, see [Placing a Report Heading or Footing on Its Own Page](#) on page 1605.

**SUBFOOT**

Is the command required to designate a report footing.

*content*

Heading or footing content can include the following elements, between double quotation marks. (If the ending quotation mark is omitted, all subsequent lines of the request are treated as part of the report footing unless you are using the <OX spot marker.)

*text*

Is text that appears on the last page of a report. You can include multiple lines of text.

The text must start on a line by itself, following the SUBFOOT command.

Text can be combined with variables and spot markers.

For related information, see [Limits for Headings and Footings](#) on page 1433.

*variable*

Can be any one or a combination of the following:

**Fields** (real data source fields, virtual fields created with the DEFINE command in a Master File or report request, calculated values created with the COMPUTE command in a request, or a system field such as TABPAGENO). You can qualify data source fields with certain prefix operators.

**Dialogue Manager variables.**

[Images](#). You can include images in a heading or footing.

For details, see [Including an Element in a Heading or Footing](#) on page 1469.

### *spot marker*

Enables you to position items, to identify items to be formatted, and to extend code beyond the 80-character line limit of the text editor.

`<+0>` divides a heading or footing into items for formatting. For details, see [Identifying a Heading, Footing, Title, or FML Free Text](#) on page 1191.

`</n` specifies skipped lines. For details, see [Controlling the Vertical Positioning of a Heading or Footing](#) on page 1598.

`<-n` to position the next character on the line. For details, see [Using Spot Markers to Refine Positioning](#) on page 1593.

`<0x` continues a heading or footing specification on the next line of the request. For details, see [Extending Heading and Footing Code to Multiple Lines in a Report Request](#) on page 1434.

**Note:** When a closing spot marker is immediately followed by an opening spot marker (`><`), a single space text item will be placed between the two spot markers (`> <`). This must be considered when applying formatting.

### [Blank lines](#)

If you omit all text, variables, and spot markers, you have a blank heading or footing line (for example, " ") which you can use to skip a line in the heading or footing. (You can also skip a line using a vertical spot marker, such as `</1.`)

## **Example:** Creating a Single-Line Report Footing

This request creates a single-line report footing that identifies the author of the report.

```
TABLE FILE GGSALES
PRINT UNITS
WHERE UNITS GE 1400
BY STCD BY REGION
WHERE REGION EQ 'Northeast'
ON TABLE SUBFOOT
"AUTHOR: MARY SMITH"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET WEBVIEWER ON
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF,$
ENDSTYLE
END
```



The output illustrates the placement of a report footing on a multi-page HTML report. The report footing follows the data on the last page.



**Tip:** If you do not see the navigation arrows, click the maximize button.

## Creating a Page Heading or Footing

A page heading appears at the top of every page of a report, and a page footing appears at the bottom of every page.

Add a page heading to identify and reinforce the report content and purpose from page to page, or include a variable that customizes the heading on each page. For example, consider a report with employee bank account information, arranged by department. Information for each department appears on a separate page. The page heading for this report identifies the department addressed on each page (for example, ACCOUNT REPORT FOR PRODUCTION DEPARTMENT).

Add a page footing to supply information that warrants repetition on each page, such as the date of the report, or a reminder that it is confidential. You can also use a page footing to supply descriptive information about a report, such as PRELIMINARY or DRAFT COPY.

A page heading or footing can include text, fields, Dialogue Manager variables, images, and spot markers.

In addition, you can use page heading and footing syntax to create a free-form (non-tabular) report, in which you position data on a page using a layout of your own design. See [Creating a Free-Form Report](#) on page 1809 for details.

A TABLE request can have more than one page heading or footing. For each heading or footing, a WHEN clause against the data being retrieved can determine whether the heading or footing displays on the report output.

In a heading, the data for the WHEN clause and data field values displayed in the heading are based on the first line on the page. In a footing, the data for the WHEN clause and the data field values displayed in the footing are based on the last line on the page.

The CONDITION StyleSheet attribute enables you to identify a specific WHEN clause so that you can style each heading or footing separately. For information, see [Identifying a Heading or Footing](#) on page 1195.

### **Syntax:**      **How to Create a Page Heading**

Include the following syntax in a request. Each heading or footing line must begin and end with a double quotation mark.

```
[HEADING [CENTER]]  
  "content ... "  
["content ... "  
.  
.  
.  
["content ... "]
```

where:

#### **HEADING**

Is an optional command if you place the text before the first display command (for example, PRINT or SUM); otherwise, it is required to identify the text as a page heading.

#### **CENTER**

Is an optional command that centers the page heading over the report data. For details, see [How to Center a Page Heading or Footing Using Legacy Formatting](#) on page 1535.

*content*

Heading or footing content can include the following elements, between double quotation marks. (If the ending quotation mark is omitted, all subsequent lines of the request are treated as part of the page heading.)

*text*

Is text for the page heading. You can include multiple lines of text.

The text must start on a line by itself, following the **HEADING** command.

Text can be combined with variables and spot markers.

For related information, see [Limits for Headings and Footings](#) on page 1433.

*variable*

Can be any one or a combination of the following:

**Fields** (real data source fields, virtual fields created with the **DEFINE** command in a Master File or report request, calculated values created with the **COMPUTE** command in a request, or a system field such as **TABPAGENO**). You can qualify data source fields with certain prefix operators.

**Dialogue Manager variables.**

**Images.** You can include images in a heading or footing.

For details, see [Including an Element in a Heading or Footing](#) on page 1469.

*spot marker*

Enables you to position items, to identify items to be formatted, and to extend code beyond the 80-character line limit of the text editor.

**<+0>** divides a heading or footing into items for formatting. For details, see [Identifying a Heading, Footing, Title, or FML Free Text](#) on page 1191.

**</n** specifies skipped lines. For details, see [Controlling the Vertical Positioning of a Heading or Footing](#) on page 1598.

**<-n** to position the next character on the line. For details, see [Using Spot Markers to Refine Positioning](#) on page 1593.

**<0x** continues a heading or footing specification on the next line of the request. For details, see [Extending Heading and Footing Code to Multiple Lines in a Report Request](#) on page 1434.

**Note:** When a closing spot marker is immediately followed by an opening spot marker (**><**), a single space text item will be placed between the two spot markers (**> <**). This must be considered when applying formatting.

Blank lines

If you omit all text, variables, and spot markers, you have a blank heading or footing line (for example, " ") which you can use to skip a line in the heading or footing. (You can also skip a line using a vertical spot marker, such as </1.)

Example: Creating a Single-Line Page Heading

This request omits the command HEADING since the page heading text precedes the display command PRINT. The page heading includes text and an embedded field.

```
TABLE FILE EMPLOYEE
  "ACCOUNT REPORT FOR DEPARTMENT <DEPARTMENT"
PRINT CURR_SAL BY DEPARTMENT BY HIGHEST BANK_ACCT
  BY EMP_ID
ON DEPARTMENT PAGE-BREAK
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF,$
ENDSTYLE
END
```

The output illustrates the placement of a page heading on a multi-page HTML report. The page heading appears on both pages of the report, identifying the department to which the data applies. See [How to Include a Field Value in a Heading or Footing](#) on page 1470 for information on embedded field values. The first page of data applies to the MIS department.

ACCOUNT REPORT FOR DEPARTMENT MIS			
<u>DEPARTMENT</u>	<u>BANK_ACCT</u>	<u>EMP_ID</u>	<u>CURR_SAL</u>
MIS	163800144	818692173	\$27,062.00
	122850108	326179357	\$21,780.00
	40950036	117593129	\$18,480.00
		112847612	\$13,200.00
		219984371	\$18,480.00
		543729165	\$9,000.00

The second page of data applies to the PRODUCTION department.

#### ACCOUNT REPORT FOR DEPARTMENT PRODUCTION

<u>DEPARTMENT</u>	<u>BANK ACCT</u>	<u>EMP ID</u>	<u>CURR SAL</u>
PRODUCTION	819000702	123764317	\$26,862.00
	136500120	451123478	\$16,100.00
	160633	119329144	\$29,700.00
		071382660	\$11,000.00
		119265415	\$9,500.00
		126724188	\$21,120.00

### **Syntax:** How to Create a Page Footing

Include the following syntax in a request. Each heading or footing line must begin and end with a double quotation mark.

```
FOOTING [CENTER] [BOTTOM]
  "content ... "
["content ... "]
.
.
.
["content ... "]
```

where:

#### FOOTING

Is the required command that identifies the content as a page footing.

#### CENTER

Is an optional command that centers the page footing over the report data. For details on CENTER, see [How to Center a Page Heading or Footing Using Legacy Formatting](#) on page 1535.

#### BOTTOM

Is an optional command that places the footing at the bottom of the page. If you omit BOTTOM, the page footing appears two lines below the report data. For details on BOTTOM, see [How to Position a Page Footing at the Bottom of a Page](#) on page 1603.

#### content

Heading or footing content can include the following elements, between double quotation marks. (If the ending quotation mark is omitted, all subsequent lines of the request are treated as part of the footing unless you are using the <OX spot marker.)

#### text

Is text for the page footing. You can include multiple lines of text.

The text must start on a line by itself, following the FOOTING command.

Text can be combined with variables and spot markers.

For related information, see [Limits for Headings and Footings](#) on page 1433.

### *variable*

Can be any one of, or a combination of the following:

[Fields](#) (real data source fields, virtual fields created with the DEFINE command in a Master File or report request, calculated values created with the COMPUTE command in a request, or a system field such as TABPAGENO). You can qualify data source fields with certain prefix operators.

[Dialogue Manager variables](#).

[Images](#). You can include images in a heading or footing.

For details, see [Including an Element in a Heading or Footing](#) on page 1469.

### *spot marker*

Enables you to position items, to identify items to be formatted, and to extend code beyond the 80-character line limit of the text editor.

[<+0>](#) divides a heading or footing into items for formatting. For details, see [Identifying a Heading, Footing, Title, or FML Free Text](#) on page 1191.

[</n](#) specifies skipped lines. For details, see [Controlling the Vertical Positioning of a Heading or Footing](#) on page 1598.

[<-n](#) positions the next character on the line. For details, see [Using Spot Markers to Refine Positioning](#) on page 1593.

[<0x](#) continues a heading or footing specification on the next line of the request. For details, see [Extending Heading and Footing Code to Multiple Lines in a Report Request](#) on page 1434.

**Note:** When a closing spot marker is immediately followed by an opening spot marker (><), a single space text item will be placed between the two spot markers (> <). This must be considered when applying formatting.

### *Blank lines*

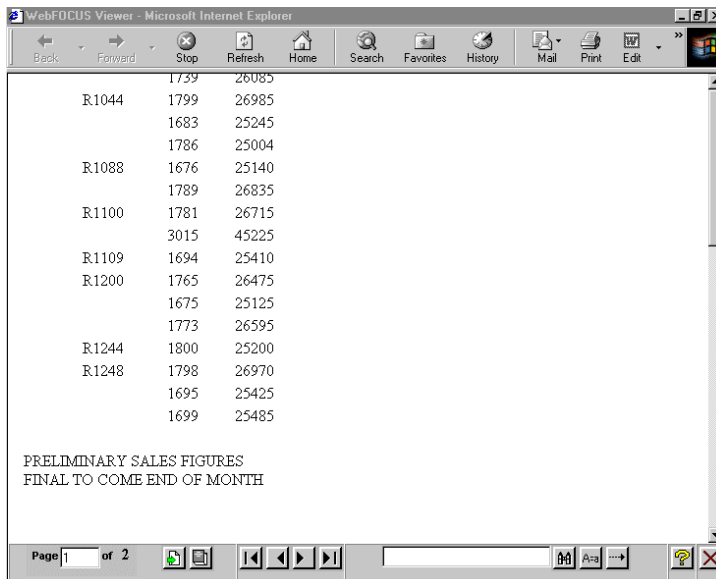
If you omit all text, variables, and spot markers, you have a blank heading or footing line (for example, " ") which you can use to skip a line in the heading or footing. (You can also skip a line using a vertical spot marker, such as [</1.](#))

**Example: Creating a Multiple-Line Page Footing**

This request creates a two-line page footing that identifies the data as preliminary and indicates when the final report will be available.

```
TABLE FILE GGSales
PRINT UNITS DOLLARS
BY CATEGORY BY STCD
WHERE TOTAL DOLLARS GE 25000
FOOTING
"PRELIMINARY SALES FIGURES"
"FINAL TO COME END OF MONTH"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET WEBVIEWER ON
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF,$
ENDSTYLE
END
```

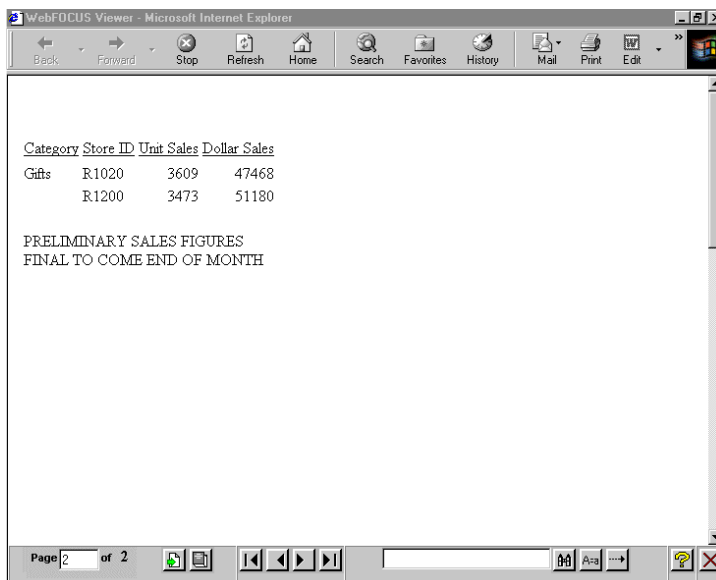
The partial output illustrates the placement of page footings on a multi-page HTML report. The page footing appears on both pages of the report.



	1739	26085
R1044	1799	26985
	1683	25245
	1786	25004
R1088	1676	25140
	1789	26835
R1100	1781	26715
	3015	45225
R1109	1694	25410
R1200	1765	26475
	1675	25125
	1773	26595
R1244	1800	25200
R1248	1798	26970
	1695	25425
	1699	25485

PRELIMINARY SALES FIGURES  
FINAL TO COME END OF MONTH

Page 1 of 2



Category	Store ID	Unit Sales	Dollar Sales
Gifts	R1020	3609	47468
	R1200	3473	51180

PRELIMINARY SALES FIGURES  
FINAL TO COME END OF MONTH

Page 2 of 2

**Tip:** If you do not see the navigation arrows, click the maximize button.



**Syntax:** How to Specify a Heading or Footing With a WHEN Clause

```
{HEADING [CENTER] | FOOTING}
" text_and_data1 "
.
.
" text_and_datan "
WHEN expression
```

where:

*text\_and\_data1*, *text\_and\_datan*

Is the text and data for each heading or footing line.

*expression*

Is an expression that resolves to TRUE or FALSE (1 or 0). If its value resolves to TRUE, the heading or footing is displayed. If the expression resolves to FALSE, the heading or footing is not displayed.

**Reference:** Usage Notes for Multiple Headings

- ☐ HEADING CENTER and FOOTING CENTER apply only to the specific heading or footing in which they are specified.
- ☐ A request can have a total of 120 headings, 120 footings, 120 subheadings, and 120 subfootings.
- ☐ Once you use the BOTTOM option on a footing, all subsequent footings also go to the bottom of the page.

**Freezing HTML Headings, Footings, and Column Titles**

You may want to scroll the data in a report while freezing headings, column titles, and footings in order to see the context of the report output while scrolling.

Using StyleSheet attributes, you can set aside a scrollable area for HTML report output.

The HTML HFREEZE reporting feature is supported with the browser versions listed in [Web Browser Support for WebFOCUS](#). The HFREEZE feature is listed in the 8.1 Browser Support Matrix in the HTML Reporting Features section JavaScript components row.

**Reference:** Usage Notes for HTMLARCHIVE With HFREEZE

WebFOCUS interactive reporting features must have a connection to the WebFOCUS client in order to access the components required to operate successfully.

HTMLARCHIVE can be used to create self-contained HTML pages with user-defined images when client access is not available.

To generate HTML pages containing user-defined images that can operate interactively, use one of the following commands:

```
SET HTMLREMBEDIMG=ON
SET HTMLARCHIVE=ON
```

Define BASEURL to point directly to the host machine where these files can be accessed using the following syntax:

```
SET BASEURL=http://{hostname:portnumber}
```

For more information on SET BASEURL, see [Specifying a Base URL](#) on page 813.

### **Syntax:** How to Create a Scrollable Area in an HTML Report

```
TYPE=REPORT,HFREEZE={OFF|ON|TOP|BOTTOM},[SCROLLHEIGHT={AUTO|nn[.n]}], $
```

where:

**HFREEZE=OFF**

Does not freeze the heading, column titles, grand totals, and footing. OFF is the default value.

**HFREEZE=ON**

Freezes the heading, column titles, grand totals, and footing.

**HFREEZE=TOP**

Freezes the heading and column titles.

**HFREEZE=BOTTOM**

Freezes the grand totals and footing.

**SCROLLHEIGHT=AUTO**

Automatically sizes the HFREEZE report within the browser page or the frame within the page. For mobile devices, the default scroll height is set to AUTO so it sizes to the current device.

**nn[.n]**

Is the height, in inches, of the scrollable area. The default for non-mobile devices is 4 inches.

**Note:** When the browser is manually resized after initial display of the HFREEZE report, the HFREEZE report size remains the same. Select the browser refresh option to reload the HFREEZE report, which will run the JavaScript that calculates the HFREEZE report size within the current page or frame.

**Reference:** **HFREEZE With Blank Column Titles**

The HTML HFREEZE reporting feature supports blank column titles. The vertical HFREEZE scroll bar will be aligned with the first row of report data.

**Reference:** **Usage Notes for Freezing Areas of HTML Report Output**

Report headers and footers can be frozen, while the data lines scroll in HTML reports only. For all other output formats, the StyleSheet attribute HFREEZE is ignored. The request must include the setting ON TABLE SET HTMLCSS ON, which is the default.

The following features are supported with HFREEZE:

☐ SET SUBTOTALS=ABOVE

In a standard HTML report, the TOTALS follow the data lines. With SUBTOTALS BELOW, the GRANDTOTAL row is anchored at the bottom of the frozen frame. With SUBTOTALS ABOVE, the totals precede the data lines, so you will see the GRANDTOTAL display as the first row at the top of the report. In this type of report, the frame will not contain the frozen GRANDTOTAL row, as this information has been presented above the report.

☐ HEADALIGN=BODY (the alignment grid)

The following HTML features are not supported with HFREEZE:

☐ HFREEZE does not support the placement of images in subheadings and subfootings within the frozen area in Internet Explorer. HFREEZE does support the placement of images in subheadings and subfootings within the frozen area in Mozilla Firefox®, Google Chrome™, and Microsoft Edge®.

☐ Accordion reports

☐ WRAP StyleSheet attribute

☐ Custom HTML tags or JavaScript

☐ Compound Reports

## Creating a Sort Heading or Footing

A sort heading is text that precedes a change in a sort field value, identifying the beginning of a group of related data. A sort footing is text that follows a change in a sort field value, identifying the end of a group of related data.

A sort heading or footing, which appears in the body of a report, helps you identify different areas of detail in a report. Sort headings or footings can include text, fields, Dialogue Manager variables, images, and spot markers.

By including a WHEN phrase in a request, you can generate a message, implemented as a sort heading or footing, for data that meets the criterion you define. For details on conditional formatting, see [Controlling Report Formatting](#) on page 1139.

If you are using a RECAP command to create subtotal values in a calculation, you can replace the default RECAP label with a more meaningful sort footing by following the RECAP command for a field with a SUBFOOT command for that field. For details about the RECAP command, see [Including Totals and Subtotals](#) on page 369.

If one or more data fields are embedded in the sort footing, you can *omit* a display command from the report request since, by default, data fields in headings and footings are summed. If, however, a request does contain an explicit SUM command and a display field is also specified in the sort footing, the field in the sort footing is summed. You can omit the display command from other types of headings and footings as well. Note that the data for headings is taken from the first sort group and the data for footings is taken from the last sort group. For related information, see [Limits for Headings and Footings](#) on page 1433.

By default, WebFOCUS generates a blank line before a subheading or subfooting. You can eliminate these automatic blank lines by issuing the SET DROPBLNKLINE=ON command.

### **Reference:** Alignment of Subheadings and Subfootings

By default, with SQUEEZE=ON, the right margin used for borders and bgcolor for subheadings and subfootings is defined based on the maximum width of all heading, footing, subheading, and subfooting lines. The length of subheading and subfooting lines is tied to the lengths of the page heading and page footing, not to the size of the data columns in the body of the report. The ALIGN-BORDERS=BODY attribute in a StyleSheet allows you to align the subheadings and subfootings with the data/report body on PDF report output instead of the other heading elements.

**Syntax:**      **How to Create a Sort Heading**

Each heading or footing line must begin and end with a double quotation mark, unless you are using the line continuation spot marker (<ox).

```

BY fieldname SUBHEAD [NEWPAGE]
  "content ... "
  ["content ... "]
  .
  .
  .
  ["content ... "]
  [WHEN expression;]
BY fieldname
ON fieldname SUBHEAD [NEWPAGE]
  "content ... "
  ["content ... "]
  .
  .
  .
  ["content ... "]
  [WHEN expression;]

```

OR

```

BY fieldname
ON fieldname SUBHEAD [NEWPAGE]
  "content ... "
  ["content ... "]
  .
  .
  .
  ["content ... "]
  [WHEN expression;]

```

where:

*fieldname*

Is the sort field before which the heading text appears.

*content*

Heading or footing content can include the following elements, between double quotation marks. (If the ending quotation mark is omitted, all subsequent lines of the request are treated as part of the heading.)

*text*

Is text for the sort heading. You can include multiple lines of text.

The text must start on a line by itself, following the SUBHEAD command.

Text can be combined with variables and spot markers.

For related information, see [Limits for Headings and Footings](#) on page 1433.

*variable*

Can be any one of, or a combination of, the following:

**Fields** (real data source fields, virtual fields created with the DEFINE command in a Master File or report request, calculated values created with the COMPUTE command in a request, or a system field such as TABPAGENO). You can qualify data source fields with certain prefix operators.

**Dialogue Manager variables.**

**Images.** You can include images in a heading or footing.

For details, see [Including an Element in a Heading or Footing](#) on page 1469.

*spot marker*

Enables you to position items, to identify items to be formatted, and to extend code beyond the 80-character line limit of the text editor.

**<+0>** divides a heading or footing into items for formatting. For details, see [Identifying a Heading, Footing, Title, or FML Free Text](#) on page 1191.

**</n** specifies skipped lines. For details, see [Controlling the Vertical Positioning of a Heading or Footing](#) on page 1598.

**<-n** to position the next character on the line. For details, see [Using Spot Markers to Refine Positioning](#) on page 1593.

**<0x** continues a heading or footing specification on the next line of the request. For details, see [Extending Heading and Footing Code to Multiple Lines in a Report Request](#) on page 1434.

**Note:** When a closing spot marker is immediately followed by an opening spot marker (><), a single space text item will be placed between the two spot markers (> <). This must be considered when applying formatting.

**WHEN** *expression*

Specifies a condition under which a sort heading is displayed, as determined by a logical expression. You must place the WHEN phrase on a line following the text.

For details on conditional formatting, see [Controlling Report Formatting](#) on page 1139. For related information, see [Using Expressions](#) on page 431.

**Blank lines**

If you omit all text, variables, and spot markers, you have a blank heading or footing line (for example, " ") which you can use to skip a line in the heading or footing. (You can also skip a line using a vertical spot marker, such as </1.)

**NEWPAGE**

Inserts a new page after the sort heading. Column titles appear on every page.

You can use NEWPAGE with PDF reports. In HTML reports, blank space is added instead of a new page.

**Example: Creating a Sort Heading When a Product Description Changes**

This request displays a sort heading each time the product description changes. The sort heading includes text and an embedded field.

```
TABLE FILE GGPRODS
PRINT PACKAGE_TYPE AND UNIT_PRICE
WHERE UNIT_PRICE GT 50
BY PRODUCT_DESCRIPTION NOPRINT BY PRODUCT_ID
ON PRODUCT_DESCRIPTION SUBHEAD
"Summary for <PRODUCT_DESCRIPTION>"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The sort heading identifies the product that the next line of data applies to.

<u>Product Code</u>	<u>Package</u>	<u>Unit Price</u>
Summary for Coffee Grinder		
G 110	Case	125.00
Summary for Coffee Pot		
G121	Case	140.00
Summary for French Roast		
B142	Pounds	81.00
Summary for Hazelnut		
B141	Pounds	58.00
Summary for Kona		
B144	Pounds	76.00
Summary for Thermos		
G104	Case	96.00

See [Including a Field Value in a Heading or Footing](#) on page 1470 for information on embedded field values.

**Example: Creating a Conditional Sort Heading**

This request displays a sort heading for a category only if its sales fall below \$17,000,000.

```
TABLE FILE GGSales
SUM DOLLARS
BY CATEGORY SUBHEAD
  "<CATEGORY ALERT: SALES FALL BELOW $17,000,000"
WHEN DOLLARS LT 17000000;
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

Sales for the category Gifts fall below the specified amount, as the sort heading warns. No other category is preceded by a sort heading.

<u>Category</u>	<u>Dollar Sales</u>
Coffee	17231455
Food	17229333
Gifts ALERT: SALES FALL BELOW \$17,000,000	
Gifts	11695502

See [Including a Field Value in a Heading or Footing](#) on page 1470 for information on embedded field values.



**Syntax:**      **How to Create a Sort Footing**

Each heading or footing line must begin and end with a double quotation mark.

For a single sort field, use the syntax

```
BY fieldname SUBFOOT [WITHIN] [MULTILINES] [NEWPAGE]
  "content ... "
[ "content ... " ]
.
.
.
[ "content ... " ]
[WHEN expression;]
BY fieldname
ON fieldname SUBFOOT [WITHIN] [MULTILINES] [NEWPAGE]
  "content ... "
[ "content ... " ]
.
.
.
[ "content ... " ]
[WHEN expression;]
```

For multiple sort fields, use the syntax

```
BY fieldname
ON fieldname SUBFOOT [MULTILINES] [NEWPAGE]
  "content ... "
[ "content ... " ]
.
.
.
[ "content ... " ]
[WHEN expression;]
```

where:

*fieldname*

Is the sort field after which the footing text appears.

WITHIN

Causes the fields in the SUBFOOT to be calculated within each value of *fieldname*.

Without this option, a field in the SUBFOOT is taken from the last line of report output above the subfooting.

MULTILINES

Suppresses the sort footing when there is only one line of data for a sort field value. (MULTI-LINES is a synonym for MULTILINES.)

### *content*

Heading or footing content can include the following elements, between double quotation marks. (If the ending quotation mark is omitted, all subsequent lines of the request are treated as part of the sort footing.)

### *text*

Is text that appears on the first page of a report. You can include multiple lines of text.

The text must start on a line by itself, following the SUBFOOT command.

Text can be combined with variables and spot markers.

For related information, see [Limits for Headings and Footings](#) on page 1433.

### *variable*

Can be any one or a combination of the following:

[Fields](#) (real data source fields, a virtual fields created with the DEFINE command in a Master File or report request, calculated values created with the COMPUTE command in a request, a system field such as TABPAGENO). You can qualify data source fields with certain prefix operators.

[Dialogue Manager variables](#).

[Images](#). You can include images in a heading or footing.

For details, see [Including an Element in a Heading or Footing](#) on page 1469.

### *spot marker*

Enables you to position items, to identify items to be formatted, and to extend code beyond the 80-character line limit of the text editor.

[<+0>](#) divides a heading or footing into items for formatting. For details, see [Identifying a Heading, Footing, Title, or FML Free Text](#) on page 1191.

[</n](#) specifies skipped lines. For details, see [Controlling the Vertical Positioning of a Heading or Footing](#) on page 1598.

[<-n](#) to position the next character on the line. For details, see [Using Spot Markers to Refine Positioning](#) on page 1593.

[<0x](#) continues a heading or footing specification on the next line of the request. For details, see [Extending Heading and Footing Code to Multiple Lines in a Report Request](#) on page 1434.

**Note:** When a closing spot marker is immediately followed by an opening spot marker (><), a single space text item will be placed between the two spot markers (> <). This must be considered when applying formatting.

**WHEN** *expression*

Specifies a condition under which a sort footing is displayed, as determined by a logical expression. You must place the WHEN phrase on a line following the text.

For details on conditional formatting, see [Controlling Report Formatting](#) on page 1139. For related information, see [Using Expressions](#) on page 431.

**Blank lines**

If you omit all text, variables, and spot markers, you have a blank heading or footing line (for example, " ") which you can use to skip a line in the heading or footing. (You can also skip a line using a vertical spot marker, such as </1.)

**NEWPAGE**

Inserts a new page before the sort footing.

**Example: Creating a Sort Footing When a Product Description Changes**

This request displays a sort footing each time the product description changes.

```
TABLE FILE GGPRODS
PRINT PACKAGE_TYPE AND UNIT_PRICE
WHERE UNIT_PRICE GT 50
BY PRODUCT_DESCRIPTION NOPRINT BY PRODUCT_ID
ON PRODUCT_DESCRIPTION SUBFOOT
"Summary for <PRODUCT_DESCRIPTION"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

Product Code	Package	Unit Price
G110	Case	125.00
Summary for Coffee Grinder		
G121	Case	140.00
Summary for Coffee Pot		
B142	Pounds	81.00
Summary for French Roast		
B141	Pounds	58.00
Summary for Hazelnut		
B144	Pounds	76.00
Summary for Kona		
G104	Case	96.00
Summary for Thermos		

See [How to Include a Field Value in a Heading or Footing](#) on page 1470 for information on embedded field values.

**Example: Creating a Conditional Sort Footing With Multiple Sort Options**

This report lists orders, order dates, and order totals for the Century Corporation. It uses conditional sort footings to distinguish between orders that total more than \$200,000 and less than \$200,000.

Notice that one sort phrase (ON ORDER\_NUM) specifies several sort-related options (two different SUBFOOT phrases), and that each option has its own WHEN phrase.

```
TABLE FILE CENTORD
HEADING
"Order Revenue"
" "
SUM ORDER_DATE LINEPRICE AS 'Order,Total:'
BY HIGHEST 5 ORDER_NUM
ON ORDER_NUM
  SUBFOOT
    "--- Order total is less than $200,000 ---"
    " "
    WHEN LINEPRICE LT 200000;
  SUBFOOT
    "+++ Order total is greater than or equal to $200,000 +++"
    " "
    WHEN LINEPRICE GE 200000;
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The following report appears.

### Order Revenue

<u>Order Number:</u>	<u>Date Of Order:</u>	<u>Order Total:</u>
94710	2001/01/02	\$406,964.24
+++ Order total is greater than or equal to \$200,000 +++		
94680	2001/01/02	\$421,916.60
+++ Order total is greater than or equal to \$200,000 +++		
94670	2001/01/02	\$513,868.76
+++ Order total is greater than or equal to \$200,000 +++		
94550	2001/01/02	\$496,323.64
+++ Order total is greater than or equal to \$200,000 +++		
94530	2001/01/02	\$3,472.41
--- Order total is less than \$200,000 ---		

### **Example:** Suppressing a Sort Footing

This request suppresses the sort footing for any product that has only one line of data (that is, a product that was only ordered one time on 01/01/96).

```
TABLE FILE GGORDER
PRINT QUANTITY
BY PRODUCT_CODE NOPRINT BY PRODUCT_DESCRIPTION
WHERE ORDER_DATE EQ '01/01/96'
WHERE STORE_CODE EQ 'R1019'
WHERE PRODUCT_DESCRIPTION EQ 'Hazelnut' OR 'Biscotti' OR 'Croissant'
ON PRODUCT_CODE SUBFOOT MULTILINES
"<PRODUCT_DESCRIPTION has multiple orders."
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

In the output, the sort footing for Biscotti is suppressed.

<u>Product</u>	<u>Ordered Units</u>
Hazelnut	300
	117
Hazelnut has multiple orders.	
Biscotti	399
Croissant	93
	433
Croissant has multiple orders.	

**Example: Replacing the Default RECAP Label With a Sort Footing**

In this request, a SUBFOOT command for the field DEPARTMENT follows a RECAP command for that field. The RECAP command creates subtotal values for the calculation.

```
TABLE FILE SHORT
SUM BALANCE AS 'Dollars' ENGLAND_POUND AS 'Sterling'
BY REGION
WHERE REGION EQ 'FAR EAST' OR 'CENTRAL AMERICA' OR 'WESTERN EUROPE';
BY COUNTRY NOPRINT
RECAP EURO/D16=BALANCE * 1.03;
SUBFOOT
" "
"Balance of investments for <COUNTRY> in Euros is <EURO>."
" "
END
```

The sort footing text (for example, "Balance of investments for FRANCE in Euros is 87,336,971.") replaces the default label for the RECAP value (\*\* EURO 87,336,971).

<u>Region</u>	<u>Dollars</u>	<u>Sterling</u>
CENTRAL AMERICA	727,810	443,964

Balance of investments for GUATEMALA in Euros is 749,644 .

39,472,857	24,078,443
------------	------------

Balance of investments for HONDURAS in Euros is 40,657,043 .

FAR EAST	88,068,897	53,722,027
----------	------------	------------

Balance of investments for HONG KONG in Euros is 90,710,964 .

114,591,945	69,901,086
-------------	------------

Balance of investments for JAPAN in Euros is 118,029,703 .

WESTERN EUROPE	125,757,198	76,711,891
----------------	-------------	------------

Balance of investments for ENGLAND in Euros is 129,529,914 .

84,793,176	51,723,837
------------	------------

Balance of investments for FRANCE in Euros is 87,336,971 .



**Example: Omitting a Display Command in a Sort Footing**

This request creates a complete report as a sort footing. It does not require a display command because the sort footing content contains the data fields DEPARTMENT and SALARY. By default, the field SALARY is summed in the sort footing.

```
TABLE FILE EMPLOYEE
BY DEPARTMENT NOPRINT SUBFOOT
" <DEPARTMENT DEPARTMENT TOTAL SALARY IS <SALARY"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

```
MIS DEPARTMENT TOTAL SALARY IS $160,177.00
```

```
PRODUCTION DEPARTMENT TOTAL SALARY IS $172,752.00
```

**Reference: Usage Notes for Subfoots**

- ☐ SUBFOOT WITHIN is useful where a prefixed field within a sort break would result in a single value (for example, AVE., MIN, MAX). Use of PCT. or APCT. displays only the last value from the sort group.
- ☐ SUBFOOT WITHIN "<prefix.fieldname " does not result in the same value as SUBTOTAL prefix. The SUBFOOT WITHIN creates a display field that operates on the original input records. SUBTOTAL with a prefix operates on the internal matrix (so AVE. is the average of the SUMS or, if a display field had the prefix AVE., the average of the averages). SUBFOOT WITHIN "<AVE.field " generates an overall average.
- ☐ Prefix operators are not supported on alphanumeric fields in a WITHIN phrase.
- ☐ The ST. prefix operator is not supported in a SUBFOOT WITHIN phrase.

**Including an Element in a Heading or Footing**

You can customize a heading or footing by including:

- ☐ A field value. See [Including a Field Value in a Heading or Footing](#) on page 1470 and [Including a Text Field in a Heading or Footing](#) on page 1477.

- ❑ A page number. See [Including a Page Number in a Heading or Footing](#) on page 1479.
- ❑ A Dialogue Manager variable. See [Including a Dialogue Manager Variable in a Heading or Footing](#) on page 1479.
- ❑ An image. See [Including an Image in a Heading or Footing](#) on page 1481.

### Including a Field Value in a Heading or Footing

You can include a field name in heading or footing text. When the request is run, the output includes the field value. The result is a customized heading or footing with specific data identification for the user.

While you can use this technique in any report, it is essential if you are creating a free-form report. For details, see [Creating a Free-Form Report](#) on page 1809.

For requests with multiple display and sort field sets, fields in a report heading or footing, or page heading or footing, are evaluated as if they were objects of the first display command. Fields in a sort heading or footing are evaluated as part of the first display command in which they are referenced. If a field is not referenced, it is evaluated as part of the last display command.

You can use a prefix operator to derive a field value in a heading or footing. However, the DST., MDE., and MDN. prefix operators are not supported in headings or footings in requests that have an ACROSS phrase or multiple display commands. For a list of operations you can perform with prefix operators, see [Displaying Report Data](#) on page 43.

Two operators are specifically designed for use with a sort footing:

- ❑ ST. produces a subtotal value of a numeric field at a sort break in a report.
- ❑ CT. produces a cumulative total of a numeric field.

### **Syntax:** How to Include a Field Value in a Heading or Footing

```
<[prefix_operator] fieldname<fieldname[>]
```

or

```
<fieldname[>]
```

where:

```
<fieldname
```

Places the field value in the heading or footing, and suppresses trailing blanks in an alphanumeric field for all values of SET STYLEMODE.

*<fieldname>*

Places the field value in the heading or footing, and retains trailing blanks in an alphanumeric field if SET STYLEMODE = FIXED. Suppresses trailing blanks for all other values of SET STYLEMODE. PDF output retains trailing blanks regardless of the STYLEMODE setting.

*prefix\_operator*

Performs a calculation directly on the value of a field. A prefix operator is applied to a single field, and affects only that field.

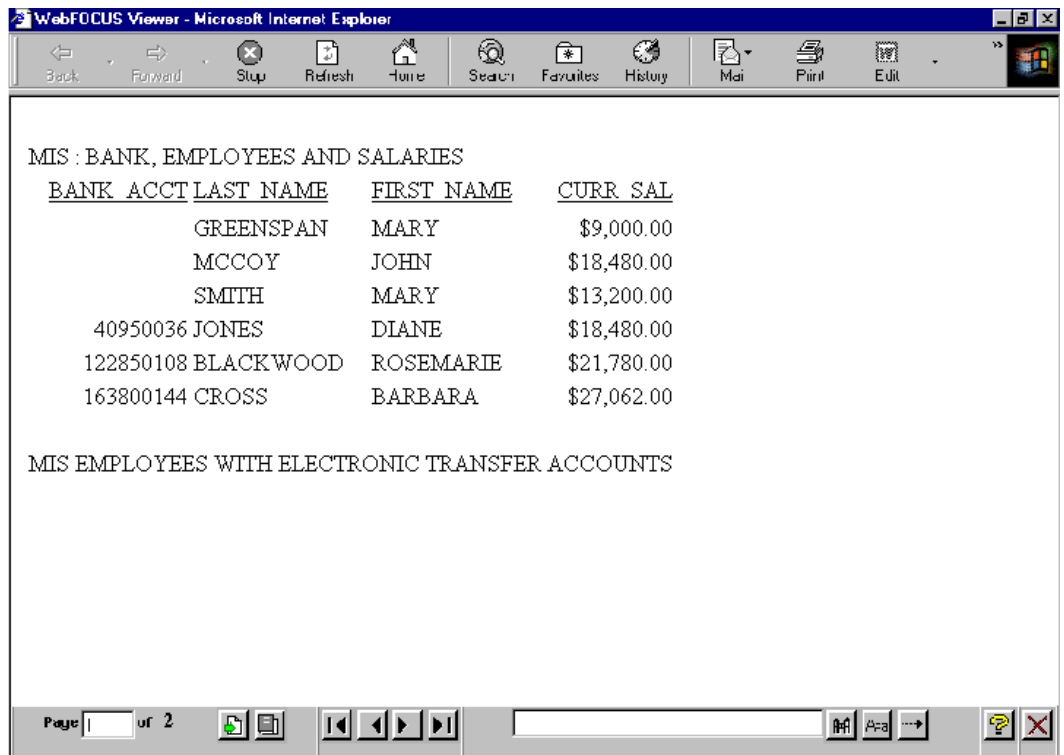
**Note:** To display the caret character (<) as text in a heading or footing, use two consecutive caret symbols (<<).

**Example:**     **Including the Department Name in a Page Heading and Footing**

This request includes the field name DEPARTMENT in both the page heading and footing text. The command HEADING is not required in the request because the page heading text appears before the command PRINT.

```
TABLE FILE EMPLOYEE
  "<DEPARTMENT : BANK, EMPLOYEES AND SALARIES"
PRINT CURR_SAL
  BY DEPARTMENT NOPRINT BY BANK_ACCT
  BY LAST_NAME BY FIRST_NAME
  ON DEPARTMENT PAGE-BREAK
FOOTING
  "<DEPARTMENT EMPLOYEES WITH ELECTRONIC TRANSFER ACCOUNTS"
  ON TABLE SET ONLINE-FMT HTML
  ON TABLE SET WEBVIEWER ON
  ON TABLE SET PAGE-NUM OFF
  ON TABLE SET STYLESHEET *
  TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output displays the output for a multi-page HTML report. On the first page of output, the value of DEPARTMENT in the page heading and footing is MIS.



MIS : BANK, EMPLOYEES AND SALARIES

<u>BANK ACCT</u>	<u>LAST NAME</u>	<u>FIRST NAME</u>	<u>CURR SAL</u>
	GREENSPAN	MARY	\$9,000.00
	MCCOY	JOHN	\$18,480.00
	SMITH	MARY	\$13,200.00
40950036	JONES	DIANE	\$18,480.00
122850108	BLACKWOOD	ROSEMARIE	\$21,780.00
163800144	CROSS	BARBARA	\$27,062.00

MIS EMPLOYEES WITH ELECTRONIC TRANSFER ACCOUNTS

On the second page of output, the value of DEPARTMENT is PRODUCTION.

PRODUCTION : BANK, EMPLOYEES AND SALARIES

<u>BANK ACCT</u>	<u>LAST NAME</u>	<u>FIRST NAME</u>	<u>CURR SAL</u>
	ROMANS	ANTHONY	\$21,120.00
	SMITH	RICHARD	\$9,500.00
	STEVENS	ALFRED	\$11,000.00
160633	BANNING	JOHN	\$29,700.00
136500120	MCKNIGHT	ROGER	\$16,100.00
819000702	IRVING	JOAN	\$26,862.00

PRODUCTION EMPLOYEES WITH ELECTRONIC TRANSFER ACCOUNTS

Page 2 of 2

**Note:** If you do not see the navigation arrows, click the maximize button.

### **Example:** Displaying a Less Than Symbol in a Heading

The following request computes the difference between REVENUE\_US and COGS\_US and displays those rows in which the difference is less than 100,000.

```
TABLE FILE WF_RETAIL_LITE
HEADING CENTER
" Difference << 100,000"
" "
SUM COGS_US REVENUE_US
COMPUTE Difference/D20.2 = REVENUE_US - COGS_US;
BY PRODUCT_CATEGORY
WHERE TOTAL Difference LT 100000
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

The heading displays the text *Difference < 100,000*, as shown in the following image.

Difference < 100,000			
Product			
<u>Category</u>	<u>Cost of Goods</u>	<u>Revenue</u>	<u>Difference</u>
Computers	\$109,281.00	\$179,761.46	70,480.46
Televisions	\$227,820.00	\$286,160.68	58,340.68
Video Production	\$180,540.00	\$259,179.84	78,639.84

**Example:**    **Retaining Trailing Blanks in an Alphanumeric Field**

Trailing blanks are not retained in standard HTML output. When the output type is HTML, STYLEMODE is set to FULL by default. To retain trailing blanks in the alphanumeric field DEPARTMENT, the STYLEMODE setting has been changed to FIXED in this request and the delimiters < and > have been included around the field name in the sort footing text.

```
SET STYLEMODE = FIXED
TABLE FILE EMPLOYEE
SUM SALARY
BY DEPARTMENT SUBFOOT
"<DEPARTMENT> DEPARTMENT TOTAL SALARY IS <SALARY"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

Values for DEPARTMENT appear in the sort footing as MIS and PRODUCTION.

DEPARTMENT	SALARY		
-----	-----		
MIS	\$160,177.00		
MIS	DEPARTMENT TOTAL SALARY IS		\$160,177.00
PRODUCTION	\$172,752.00		
PRODUCTION	DEPARTMENT TOTAL SALARY IS		\$172,752.00

**Note:** SET STYLEMODE=FIXED turns off the HTML formatting of your browser for that report. The resulting report displays in a fixed font without colors and other web capabilities.

**Example: Using the Prefix Operator TOT in a Page Heading**

This request uses the prefix operator TOT to generate grand totals for three fields.

```

DEFINE FILE SALES
ACTUAL_SALES/D8.2 = UNIT_SOLD - RETURNS;
SALES/F5.1 = 100 * ACTUAL_SALES / UNIT_SOLD;
END
TABLE FILE SALES
"SUMMARY OF ACTUAL SALES"
"UNITS SOLD <TOT.UNIT_SOLD"
"RETURNS <TOT.RETURNS"
"TOTAL SOLD <TOT.ACTUAL_SALES"
  "
  "BREAKDOWN BY PRODUCT"
PRINT UNIT_SOLD AND RETURNS AND ACTUAL_SALES
BY PROD_CODE
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT PDF
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END

```

The totals appear in the page heading.

```
SUMMARY OF ACTUAL SALES
UNITS SOLD    645
RETURNS    58
TOTAL SOLD    587.00
```

BREAKDOWN BY PRODUCT			
<u>PROD_CODE</u>	<u>UNIT_SOLD</u>	<u>RETURNS</u>	<u>ACTUAL_SALES</u>
B10	60	10	50.00
	30	2	28.00
	13	1	12.00
B12	40	3	37.00
	29	1	28.00
B17	29	2	27.00
	20	2	18.00
B20	15	0	15.00
	25	1	24.00
C13	25	3	22.00
C17	12	0	12.00
C7	45	5	40.00
	40	0	40.00
D12	27	0	27.00
	20	3	17.00
E1	30	4	26.00
E2	80	9	71.00
E3	70	8	62.00
	35	4	31.00

### **Example:** Using Multiple Prefix Operators in a Page Heading

This request uses the prefix operators MAX, MIN, AVE, and TOT. It does not require a display command because the page heading text contains data fields.

```
TABLE FILE SALES
"MOST UNITS SOLD WERE <MAX.UNIT_SOLD"
"LEAST UNITS SOLD WERE <MIN.UNIT_SOLD"
"AVERAGE UNITS SOLD WERE <AVE.UNIT_SOLD"
"TOTAL UNITS SOLD WERE <TOT.UNIT_SOLD"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```



The prefix operators generate summary data in the page heading.

```
MOST UNITS SOLD WERE 80
LEAST UNITS SOLD WERE 12
AVERAGE UNITS SOLD WERE 33
TOTAL UNITS SOLD WERE 645
```

### **Example:** Using Multiple Prefix Operators in a Sort Footing

This request uses the prefix operators CNT and AVE in a sort footing. The output does not contain columns of data. All data is included in the sort footing itself.

```
TABLE FILE EMPLOYEE
BY DEPARTMENT NOPRINT SUBFOOT
"NUMBER OF EMPLOYEES IN DEPARTMENT <DEPARTMENT = <CNT.LAST_NAME"
"WITH AVERAGE SALARY OF <AVE.CURR_SAL"
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The sort footing is a summary report on the number of employees in a department and their average salary.

The prefix operators generate summary data in the page heading.

```
NUMBER OF EMPLOYEES IN DEPARTMENT MIS = 6
WITH AVERAGE SALARY OF $18,000.33
NUMBER OF EMPLOYEES IN DEPARTMENT PRODUCTION = 6
WITH AVERAGE SALARY OF $19,047.00
```

### **Including a Text Field in a Heading or Footing**

You can include one or more text fields in a heading or footing. A text field has the attribute FORMAT=TXn in a Master File.

### **Reference:** Limits for Text Fields in a Heading or Footing

- ☐ You cannot embed a text field in a Financial Modeling Language (FML) report. For details, see [Creating Financial Reports With Financial Modeling Language \(FML\)](#) on page 1729.
- ☐ You cannot apply the StyleSheet attribute WRAP to a text field, since a text field is separated into multiple lines by the character count specified in the FORMAT attribute in the Master File.

**Syntax:**      **How to Include a Text Field in a Heading or Footing**

```
<TEXTFLD
```

**Example:**      **Including a Text Field in a Sort Footing**

In this example, you create a Master File named TXTFLD.MAS and a corresponding FOCUS data source named TXTFLD.FOC.

1. Create and save the Master File.

```
FILENAME = TXTFLD, SUFFIX = FOC,$
SEGNAME=TXTSEG, SEGTYPE = S1,$
    FIELDNAME = CATALOG, FORMAT = A10, $
    FIELDNAME = TEXTFLD,      FORMAT = TX50,$
```

2. Create and save the following MODIFY procedure. This procedure creates and populates the data source in a Windows environment.

```
CREATE FILE TXTFLD
MODIFY FILE TXTFLD
FIXFORM CATALOG/10 TEXTFLD
DATA
COURSE100 This course provides the junior programmer
with the skills needed to code simple reports.%%
COURSE200 This course provides the advanced programmer with
techniques helpful in developing complex
applications.%%
END
```

3. Run the MODIFY procedure to populate the data source.
4. Create and save the following report request.

```
TABLE FILE TXTFLD
BY CATALOG SUBFOOT
" <TEXTFLD"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF,$
ENDSTYLE
END
```

5. Run the report request.

The output is:

```

                                CATALOGCOURSE 100
This course provides the junior programmer with
the skills needed to code simple reports
COURSE 200
This course provides the advanced programmer with
techniques helpful in developing complex
applications.
```

The horizontal space occupied by the text field is determined by the number of characters specified in the FORMAT attribute in the Master File. In the sample Master File, TX50 means 50 characters wide.

**Tip:** Since the heading in this example includes a single embedded text field, the default alignment is satisfactory. However, to include text to introduce the embedded field or add another embedded field, you may align items in your output to improve readability.

## Including a Page Number in a Heading or Footing

You can include a system-generated page number in a heading or footing. For details, see [Laying Out the Report Page](#) on page 1249.

## Including a Dialogue Manager Variable in a Heading or Footing

You can include a variable whose values are unknown until run time in a heading or footing. This technique allows you to customize the heading or footing by supplying a different value each time the procedure executes.

Variables fall into two categories:

- ☐ System and statistical variables are predefined and their values are automatically supplied by the system when a procedure references them. System and statistical variables have names that begin with &. For example, &DATE generates the current date in report output.
- ☐ Local (&) and global (&&) variables, whose user-defined values must be supplied at run time:
  - ☐ A local variable retains its values during the execution of one procedure. Values are lost after the procedure finishes processing. Values are *not* passed to other invoked procedures that contain the same variable name.
 

A local variable is identified by a single ampersand followed by the variable name.
  - ☐ A global variable retains its value for the duration of the connection to the WebFOCUS Reporting Server and is passed from the execution of one invoked procedure to the next.

Because a new session is created on the WebFOCUS Reporting Server each time a request is submitted, values for global variables are not retained between report requests. This means that you can use the same global variable in more than one procedure as long as these procedures are called in the same request.

A global variable is identified by a double ampersand followed by the variable name.

**Note:** To avoid conflicts, do not name local or global variables beginning with Date, IBI, or WF. Variable names beginning with these values are reserved for Information Builder use.

For details on Dialogue Manager variables, see the *Developing Reporting Applications* manual.

### **Syntax:** How to Include a Dialogue Manager Variable in a Heading or Footing

`&[&]variable`

where:

`&`

Introduces a local Dialogue Manager variable.

`&&`

Introduces a global Dialogue Manager variable.

`variable`

Is a variable whose value is supplied by the system or by a user at run time.

### **Example:** Including the Current Date in a Report Heading

This request includes today's date on the second line of the report heading, highlighted in bold.

```
TABLE FILE GGSales
PRINT BUDDOLLARS DOLLARS
BY STCD
WHERE STCD EQ 'R1019'
ON TABLE SUBHEAD
"Sales Report for Store Code R1019"
"&DATE"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF,$
TYPE=TABLEHEADING, LINE=1, FONT='TIMES', SIZE=10, STYLE=BOLD, $
TYPE=TABLEHEADING, LINE=2, COLOR=BLUE, $
ENDSTYLE
END
```

The output is:

### Sales Report for Store Code R1019

06/07/02

<u>Store ID</u>	<u>Budget Dollars</u>	<u>Dollar Sales</u>
R1019	18560	25680
	21132	24010
	15880	22919
	24450	22605
	14556	18942
	13860	18915
	22806	18837
	19248	18535
	15120	18466
	15444	17784

**Note:** You can modify the format of the date. Some formats are:

Variable	Display Format
&DATEtrMMDYY	2002, December 11
&DATEMDYY	12/11/2002
&DATEtrMDYY	December 11, 2002
&DATEQYY	Q4 2002

### Including an Image in a Heading or Footing

A StyleSheet enables you to include an image in a heading or footing. An image, such as a logo, gives corporate identity to a report, or provides visual appeal.

For details on adding and positioning images, see [Laying Out the Report Page](#) on page 1249.

## Displaying Syntax Components in Heading Objects

You can automatically display syntax components from your report or chart request in heading objects by adding one or more of the following attributes:

- ☐ **<REQUEST.FILTERS.** Lists the WHERE and IF conditions in the request.
- ☐ **<REQUEST.VERB\_OBJECTS.** Lists the display fields referenced in the request.
- ☐ **<REQUEST.SORT\_KEYS.** Lists all sort fields in the request.
- ☐ **<REQUEST.BYKEYS.** Lists all BY sort fields in the request.
- ☐ **<REQUEST.ACROSSKEYS.** Lists all ACROSS sort fields in the request.
- ☐ **<REQUEST.VERB\_OBJECTS\_CONTEXT.** Lists the display command syntax used for each display field.
- ☐ **<REQUEST.SORT\_KEYS\_CONTEXT.** Lists all sort phrases in the request.

**Note:** The syntax component breaks onto multiple lines if the heading line length extends beyond the width of the report or chart container.

**Example: Displaying Report Syntax Components**

The following request displays all available syntax components from the report request in the heading. The spot markers (<+0>) are used to separate heading items so they can be styled separately in the StyleSheet.

```
TABLE FILE WF_RETAIL_LITE
HEADING
"Display Objects: <+0> <REQUEST.VERB_OBJECTS"
"Sort Fields: <+0> <REQUEST.SORT_KEYS"
"BY Fields: <+0> <REQUEST.BYKEYS"
"ACROSS Fields: <+0> <REQUEST.ACROSSKEYS"
"Filters: <+0> <REQUEST.FILTERS"
" "
"Display Commands: <+0> <REQUEST.VERB_OBJECTS_CONTEXT"
"Sort Phrases: <+0> <REQUEST.SORT_KEYS_CONTEXT"
" "
" "
SUM COGS_US REVENUE_US
COMPUTE RATIO/D12.2=REVENUE_US/COGS_US;
BY PRODUCT_CATEGORY SUBTOTAL COGS_US
ACROSS BUSINESS_REGION ACROSS-TOTAL
WHERE BUSINESS_REGION NE 'Oceania'
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
TYPE=REPORT, COLOR = BLUE, SIZE=10, GRID=OFF,$
TYPE=HEADING, ITEM=1, OBJECT=TEXT, FONT=Courier, COLOR=BLUE, STYLE=BOLD,$
TYPE=HEADING, ITEM=2, FONT=Courier, COLOR=TEAL, STYLE=ITALIC,$
TYPE=TITLE, FONT=ARIAL, STYLE=BOLD, COLOR=NAVY,$
TYPE=ACROSSTITLE, FONT=ARIAL, STYLE=BOLD, COLOR=NAVY,$
TYPE=ACROSSVALUE, FONT=ARIAL, STYLE=ITALIC, COLOR=NAVY, SIZE=10,$
ENDSTYLE
END
```

The output is shown in the following image.

Display Objects: 'Cost of Goods' Revenue  
Sort Fields: 'Product,Category' 'Customer,Business,Region'  
BY Fields: 'Product,Category'  
ACROSS Fields: 'Customer,Business,Region'  
Filters: WHERE 'Customer,Business,Region' NE 'Oceania';  
  
Display Commands: WRITE 'Cost of Goods' Revenue  
Sort Phrases: BY 'Product,Category' SUBTOTAL COGS\_US ACROSS 'Customer,Business,Region' ACROSS-TOTAL

Product Category	Customer,Business,Region EMEA			North America			South America			TOTAL		
	Cost of Goods	Revenue	RATIO	Cost of Goods	Revenue	RATIO	Cost of Goods	Revenue	RATIO	Cost of Goods	Revenue	RATIO
Accessories	\$143,987.00	\$211,477.99	1.47	\$171,306.00	\$249,935.10	1.46	\$25,723.00	\$35,445.49	1.38	\$341,016.00	\$496,838.58	4.31
*TOTAL Accessories	\$143,987.00			\$171,306.00			\$25,723.00			\$341,016.00		
Camcorder	\$187,000.00	\$280,235.28	1.50	\$242,967.00	\$349,755.15	1.44	\$22,686.00	\$36,276.20	1.60	\$452,653.00	\$666,266.63	4.54
*TOTAL Camcorder	\$187,000.00			\$242,967.00			\$22,686.00			\$452,653.00		
Computers	\$47,616.00	\$78,240.81	1.64	\$53,329.00	\$87,843.34	1.65	\$7,845.00	\$12,869.35	1.64	\$108,790.00	\$178,953.50	4.93
*TOTAL Computers	\$47,616.00			\$53,329.00			\$7,845.00			\$108,790.00		
Media Player	\$328,401.00	\$424,907.47	1.29	\$386,681.00	\$501,551.53	1.30	\$60,183.00	\$77,727.74	1.29	\$775,265.00	\$1,004,186.74	3.88
*TOTAL Media Player	\$328,401.00			\$386,681.00			\$60,183.00			\$775,265.00		
Stereo Systems	\$361,756.00	\$514,689.93	1.42	\$412,036.00	\$585,129.52	1.42	\$81,823.00	\$114,226.95	1.40	\$855,615.00	\$1,214,046.40	4.24
*TOTAL Stereo Systems	\$361,756.00			\$412,036.00			\$81,823.00			\$855,615.00		
Televisions	\$100,443.00	\$123,921.37	1.23	\$101,212.00	\$128,882.05	1.27	\$25,800.00	\$33,057.86	1.28	\$227,455.00	\$285,861.28	3.79
*TOTAL Televisions	\$100,443.00			\$101,212.00			\$25,800.00			\$227,455.00		
Video Production	\$78,722.00	\$112,141.72	1.42	\$89,489.00	\$129,247.23	1.44	\$11,740.00	\$16,872.90	1.44	\$179,951.00	\$258,261.85	4.31
*TOTAL Video Production	\$78,722.00			\$89,489.00			\$11,740.00			\$179,951.00		
TOTAL	\$1,247,925.00			\$1,457,020.00			\$235,800.00			\$2,940,745.00		

Repeating Headings and Footings on Panels in PDF Report Output

When the columns presented on PDF reports cannot be displayed on a single page, the pages automatically panel. Paneling places subsequent columns for the same page on *overflow pages*. These overflow pages are generated until the entire width of the report is presented, after which the next vertical page is generated with a new page number and its associated horizontal panels.

In order to make panels following the initial panel more readable, you can designate that heading elements from the initial panel should be repeated on each subsequent panel using the HEADPANEL=ON StyleSheet attribute.

When paneling occurs, if default page numbering is used, the page number presented will include both the page number and the panel number (for example, 1.1, 1.2, 1.3). Turning HEADPANEL on will also cause the panel designation to be included in TABPAGE.NO.

HEADPANEL can be designated for the entire report, causing all headings and footings to be replicated on the paneled pages. It can also be turned on for just individual headings, footings, subheadings, or subfootings.



HEADPANEL causes borders from the initial page to be replicated on the paneled pages. Additional control of subheading and subfooting borders can be gained through the use of ALIGN-BORDERS which allows for the designation that subitem borders should align with the body of the data rather than the page or report headings. For more information about using ALIGN-BORDERS with HEADPANEL see [How to Align Subheading and Subfooting Margins With the Report Body](#) on page 1334.

### **Syntax:** How to Repeat Heading Elements on Panels

```
TYPE={REPORT|headfoot [BY=sortcolumn]}, HEADPANEL={ON|OFF}, $
```

where:

#### **REPORT**

Repeats all report headings, footings, page headings, page footings, subheadings, and subfootings.

#### ***headfoot***

Identifies a heading or footing. Select from:

- ☐ **TABHEADING**, which is a report heading. This appears once at the beginning of the report and is generated by ON TABLE SUBHEAD.
- ☐ **TABFOOTING**, which is a report footing. This appears once at the end of the report and is generated by ON TABLE SUBFOOT.
- ☐ **HEADING**, which is a page heading. This appears at the top of every report page and is generated by HEADING.
- ☐ **FOOTING**, which is a page footing. This appears at the bottom of every report page and is generated by FOOTING.
- ☐ **SUBHEAD**, which is a sort heading. This appears at the beginning of a vertical (BY) sort group (generated by ON *sortfield* SUBHEAD).
- ☐ **SUBFOOT**, which is a sort footing. This appears at the end of a vertical (BY) sort group (generated by ON *sortfield* SUBFOOT).

#### **BY**

When there are several sort headings or sort footings, each associated with a different vertical sort (BY) column, this enables you to identify which sort heading or sort footing you wish to format.

If there are several sort headings or sort footings associated with different vertical sort (BY) columns, and you omit this attribute and value, the formatting will be applied to all of the sort headings or footings.

*sortcolumn*

Specifies the vertical sort (BY) column associated with one of the report sort headings or sort footings.

*ON*

Repeats the specified heading or footing elements on each panel.

*OFF*

Displays heading or footing elements on the first panel only. OFF is the default value.

Note that the HEADPANEL=ON attribute can only be applied to the entire heading or footing, not individual lines or items within the heading or footing.

**Example: Repeating All Headings and Footings on Report Panels**

The following request against the GGSALES data source sums units sold, budgeted units sold, dollar sales, and budgeted sales by region, state, city, category, and product. The report has a page heading and, for each region, a subfooting.

```
TABLE FILE GGSALES
HEADING
"PRODUCT SALES REPORT"
" "
"Page<TABPAGENO"
" "
SUM UNITS BUDUNITS DOLLARS BUDDOLLARS
BY REGION NOPRINT
BY ST BY CATEGORY BY PRODUCT
ON REGION SUBFOOT
" "
" SUBFOOT FOR REGION <REGION "
" SUBTOTAL BUDDOLLARS: <ST.BUDDOLLARS SUBTOTAL DOLLARS: <ST.DOLLARS "
" "
ON TABLE SET BYPANEL ON
ON TABLE SET PAGE ON
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE = REPORT, HEADPANEL=OFF,$
ENDSTYLE
END
```

The request sets BYPANEL ON, so each panel displays the sort field values. However, since HEADPANEL=OFF for the entire report, the first panel for page 1 has the heading and the subfooting, but the second panel does not.

The output for page 1 panel 1 has the heading and subfooting, as shown in the following image. Note that with HEADPANEL=OFF, TABPAGENO does not include the panel number.

**PRODUCT SALES REPORT**

**Page 1**

State	Category	Product	Unit Sales	Budget Units	Dollar Sales
IL	Coffee	Espresso	32227	32416	420439
		Latte	77244	79015	978140
	Food	Biscotti	29413	30001	378413
		Croissant	43300	43271	549366
		Scone	45355	45091	595069
	Gifts	Coffee Grinder	19339	19224	233293
		Coffee Pot	15785	16015	204828
		Mug	30157	30881	378754
		Thermos	14651	14137	187901
	MD	Coffee	Espresso	32596	32787
Latte			77247	77141	966951
Food		Biscotti	29188	28764	368077
		Croissant	48941	49327	613871
		Scone	37602	36573	481953
Gifts		Coffee Grinder	14614	14779	181570
		Coffee Pot	14887	14978	190153
		Mug	27040	26817	343852
		Thermos	15592	15903	195686
TX		Coffee	Espresso	36321	36666
	Latte		76932	77501	978245
	Food	Biscotti	27504	27074	345238
		Croissant	46941	47059	597887
		Scone	33170	33112	418198
	Gifts	Coffee Grinder	16440	16625	204292
		Coffee Pot	16564	16774	204897
		Mug	29521	29374	366137
		Thermos	16244	16779	194319

**SUBFOOT FOR REGION Midwest**

**SUBTOTAL SUBDOLLARS: 11194373 SUBTOTAL DOLLARS: 11400665**

CT	Coffee	Capuccino	32386	31098	158995
		Espresso	32482	32674	279173
	Food	Latte	74623	74427	926052
		Biscotti	46214	46404	589155
		Croissant	45847	46335	581489
		Scone	22378	21034	283874
	Gifts	Coffee Grinder	11691	12117	149908
		Coffee Pot	15523	15120	208209
		Mug	31728	32415	392967
		Thermos	17568	17667	221827
MA	Coffee	Capuccino	15358	15672	174144
		Espresso	19698	19888	248156
	Food	Latte	74572	73874	917737
		Biscotti	47064	48246	570191
		Croissant	41029	41351	497234
		Scone	25262	23774	332486
	Gifts	Coffee Grinder	14383	15384	177828

The output for page 1 panel 2 does not have the heading or subfooting, as shown in the following image.

<u>State</u>	<u>Category</u>	<u>Product</u>	<u>Budget Dollars</u>
IL	Coffee	Espresso	401477
		Latte	964787
	Food	Biscotti	385369
		Croissant	528255
		Scone	567231
	Gifts	Coffee Grinder	241711
		Coffee Pot	208255
		Mug	388612
		Thermos	181159
MO	Coffee	Espresso	416875
		Latte	921336
	Food	Biscotti	360403
		Croissant	592609
		Scone	478691
	Gifts	Coffee Grinder	171501
		Coffee Pot	191451
		Mug	324488
		Thermos	189484
TX	Coffee	Espresso	439880
		Latte	941677
	Food	Biscotti	321857
		Croissant	587869
		Scone	398437
	Gifts	Coffee Grinder	200241
		Coffee Pot	214301
		Mug	383050
		Thermos	193367
CT	Coffee	Capuccino	141574
		Espresso	299854
		Latte	953855
	Food	Biscotti	587501
		Croissant	580168
		Scone	269221
	Gifts	Coffee Grinder	159620
		Coffee Pot	197051
		Mug	424333
		Thermos	219025
MA	Coffee	Capuccino	192747
		Espresso	254310
		Latte	941438
	Food	Biscotti	616766
		Croissant	519322
		-	-----

The following output shows panels 1 and 2 if the StyleSheet declaration is changed to set HEADPANEL=ON for the entire report (TYPE=REPORT, HEADPANEL=ON, \$). The heading and subfooting are repeated on each panel. With HEADPANEL=ON, TABPAGENO includes the panel number.

# PRODUCT SALES REPORT

Page 1.1

State	Category	Product	Unit Sales	Budget Units	Dollar Sales
IL	Coffee	Espresso	12217	12416	420419
		Latte	77344	79015	278140
	Food	Biscotti	29413	30001	178412
		Croissant	41300	43271	549166
	Gifts	Scone	45355	45021	595069
		Coffee Grinder	19339	19224	233292
		Coffee Pot	15785	16035	204828
		Mug	30157	30881	376754
		Thermos	14651	14137	187901
MO	Coffee	Espresso	12596	12787	419143
		Latte	77347	77141	266991
	Food	Biscotti	29188	28764	168077
		Croissant	48941	49327	613871
	Gifts	Scone	37602	36573	481953
		Coffee Grinder	14614	14779	181570
		Coffee Pot	14807	14970	190153
		Mug	27040	26837	343852
		Thermos	15592	15203	195686
TX	Coffee	Espresso	16321	16666	455165
		Latte	76932	77501	238245
	Food	Biscotti	27504	27074	145218
		Croissant	46941	47059	587887
	Gifts	Scone	11170	12112	418198
		Coffee Grinder	16440	16625	204292
		Coffee Pot	16564	16774	204897
		Mug	29521	29374	366117
		Thermos	16344	16779	194319

SUBFOOT FOR REGION Midwest

SUBTOTAL RUDDOLLARS: 11194373 SUBTOTAL DOLLARS: 11400665

CT	Coffee	Cappuccino	12386	11028	156995
		Espresso	22482	23676	279173
		Latte	74623	74427	236052
	Food	Biscotti	46214	46404	589155
		Croissant	45847	46335	581489
		Scone	22378	21038	283874
	Gifts	Coffee Grinder	11691	12117	149908
		Coffee Pot	15923	15190	208209
		Mug	11728	12415	192967
		Thermos	17568	17667	221827
MA	Coffee	Cappuccino	15358	15672	174144
		Espresso	19698	19888	248156
		Latte	74572	73874	217717
	Food	Biscotti	47664	48246	570391
		Croissant	41029	41351	497214
		Scone	25363	23774	312486
	Gifts	Coffee Grinder	14382	15184	177940

## PRODUCT SALES REPORT

Page 1.2

State	Category	Product	Budget Dollars
IL	Coffee	Espresso	401477
		Latte	964787
	Food	Biscotti	185369
		Croissant	528299
		Scone	567291
	Gifts	Coffee Grinder	241711
		Coffee Pot	208299
Mug		188612	
Thermos		181199	
MO	Coffee	Espresso	416879
		Latte	921796
	Food	Biscotti	160409
		Croissant	592609
		Scone	478691
	Gifts	Coffee Grinder	171901
		Coffee Pot	191491
Mug		124488	
Thermos		189484	
TX	Coffee	Espresso	419880
		Latte	941677
	Food	Biscotti	121897
		Croissant	587869
		Scone	198497
	Gifts	Coffee Grinder	200341
		Coffee Pot	214901
Mug		183090	
Thermos		192267	

SUBFOOT FOR REGION Midwest

SUBTOTAL SUBDOLLARS: 11194373 SUBTOTAL DOLLARS: 11400665

CT	Coffee	Cappuccino	141574
		Espresso	292834
		Latte	953899
	Food	Biscotti	587501
		Croissant	580168
		Scone	269221
	Gifts	Coffee Grinder	199620
		Coffee Pot	197091
		Mug	424222
MA	Coffee	Thermos	219029
		Cappuccino	192747
		Espresso	254310
	Food	Latte	941428
		Biscotti	616766
		Croissant	519222
	Gifts	Scone	112004
		Coffee Grinder	187686
		Coffee Pot	184071
	Mug	401617	

**Example: Repeating a Subfoot on Panels in PDF Report Output**

The following request against the GGSALES data source specifies the HEADPANEL=ON attribute only for the subfoot, not for the entire report. Notice that this request uses the default page numbering (ON TABLE SET PAGE ON) rather than TABPAGENO to present the page numbers on each page.

```
TABLE FILE GGSALES
HEADING
"  PRODUCT SALES REPORT"
"  "
SUM UNITS BUDUNITS DOLLARS BUDDOLLARS
BY REGION NOPRINT
BY ST BY CITY  BY CATEGORY BY PRODUCT
ON REGION SUBFOOT
"  "
"  SUBFOOT FOR REGION <REGION  "
"  SUBTOTAL BUDDOLLARS: <ST.BUDDOLLARS SUBTOTAL DOLLARS:  <ST.DOLLARS  "
"  "
ON TABLE SET BYPANEL ON
ON TABLE SET PAGE ON
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE = SUBFOOT, HEADPANEL=ON,$

ENDSTYLE
END
```

Panel 1 displays both the heading and the subfooting, as shown in the following image.

```

PAGE      1.1

PRODUCT SALES REPORT

State City           Category Product           Unit Sales
----
IL    Chicago        Coffee Espresso         32137
                                           Latte            77344
                                           Food Biscotti        29413
                                           Croissant        42300
                                           Scone            45355
                                           Gifts Coffee Grinder 19339
                                           Coffee Pot       15785
                                           Mug              30157
                                           Thermos          14651
MO    St. Louis      Coffee Espresso         32596
                                           Latte            77347
                                           Food Biscotti        29188
                                           Croissant        48941
                                           Scone            37602
                                           Gifts Coffee Grinder 14614
                                           Coffee Pot       14807
                                           Mug              27040
                                           Thermos          15592
TX    Houston        Coffee Espresso         36321
                                           Latte            76932
                                           Food Biscotti        27504
                                           Croissant        46941
                                           Scone            33170
                                           Gifts Coffee Grinder 16440
                                           Coffee Pot       16564
                                           Mug              29521
                                           Thermos          16344

SUBFOOT FOR REGION Midwest
SUBTOTAL BUDDOLLARS: 11194373 SUBTOTAL DOLLARS: 11400665

CT    New Haven      Coffee Capuccino         12386
                                           Espresso         22482
                                           Latte            74623
                                           Food Biscotti        46214
                                           Croissant        45847
                                           Scone            22378
                                           Gifts Coffee Grinder 13691
                                           Coffee Pot       15523
                                           Mug              31728
                                           Thermos          17568

```



Panel 2 displays only the subfooting, not the heading, as shown in the following image.

PAGE 1.2

<u>State</u>	<u>City</u>	<u>Category</u>	<u>Product</u>	<u>Budget Units</u>		
IL	Chicago	Coffee	Espresso	32416		
			Latte	79015		
		Food	Biscotti	30001		
			Croissant	43271		
			Scone	45091		
		Gifts	Coffee Grinder	19224		
			Coffee Pot	16035		
			Mug	30881		
			Thermos	14137		
MO	St. Louis	Coffee	Espresso	32787		
			Latte	77141		
		Food	Biscotti	28764		
			Croissant	49327		
			Scone	36573		
		Gifts	Coffee Grinder	14779		
			Coffee Pot	14970		
			Mug	26837		
			Thermos	15903		
TX	Houston	Coffee	Espresso	36666		
			Latte	77501		
		Food	Biscotti	27074		
			Croissant	47050		
			Scone	32112		
		Gifts	Coffee Grinder	16625		
			Coffee Pot	16774		
			Mug	29374		
			Thermos	16779		
SUBFOOT FOR REGION Midwest						
SUBTOTAL BUDDOLLARS: 11194373 SUBTOTAL DOLLARS: 11400665						
CT	New Haven	Coffee	Capuccino	11098		
			Espresso	23676		
			Latte	74427		
		Food	Biscotti	46404		
			Croissant	46335		
			Scone	21038		
		Gifts	Coffee Grinder	13117		
			Coffee Pot	15190		
			Mug	32415		
			Thermos	17667		

Since the page heading is not repeated, if you use the <TABPAGENO system variable to place the page number in the heading, it will not display the panel number and will not display on the second panel.

```
TABLE FILE GGSales
HEADING
"  PRODUCT SALES REPORT PAGE <TABPAGENO"
"  "
SUM UNITS BUDUNITS DOLLARS BUDDOLLARS
BY REGION NOPRINT
BY ST BY CITY BY CATEGORY BY PRODUCT
ON REGION SUBFOOT
"  "
"  SUBFOOT FOR REGION <REGION "
"  SUBTOTAL BUDDOLLARS: <ST.BUDDOLLARS SUBTOTAL DOLLARS: <ST.DOLLARS "
"  "
ON TABLE SET BYPANEL ON
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE = SUBFOOT, HEADPANEL=ON,$

ENDSTYLE
END
```

The first panel displays the page number in the heading, without the panel number, as shown in the following image.

```

PRODUCT SALES REPORT PAGE      1

State  City                    Category  Product                    Unit Sales
-----
IL     Chicago                Coffee    Espresso                   32237
                                           Latte                      77344
                                           Food    Biscotti                   29413
                                           Croissant                  43300
                                           Soona                      45355
                                           Gifts   Coffee Grinder             19339
                                           Coffee Pot                  15785
                                           Mug                        30157
                                           Thermos                    14651
MO     St. Louis              Coffee    Espresso                   32596
                                           Latte                      77347
                                           Food    Biscotti                   29188
                                           Croissant                  48941
                                           Soona                      37602
                                           Gifts   Coffee Grinder             14614
                                           Coffee Pot                  14807
                                           Mug                        27040
                                           Thermos                    15592
TX     Houston                Coffee    Espresso                   36321
                                           Latte                      76932
                                           Food    Biscotti                   27504
                                           Croissant                  46941
                                           Soona                      33170
                                           Gifts   Coffee Grinder             16440
                                           Coffee Pot                  16564
                                           Mug                        29521
                                           Thermos                    16344

SUBFOOT FOR REGION Midwest
SUBTOTAL BUDDOLLARS: 11194373 SUBTOTAL DOLLARS: 11400665

CT     New Haven              Coffee    Capuccino                  12386
                                           Espresso                   22482
                                           Latte                      74623
                                           Food    Biscotti                   46214
                                           Croissant                  45847
                                           Soona                      22378
                                           Gifts   Coffee Grinder             13691
                                           Coffee Pot                  15523
                                           Mug                        31728
                                           Thermos                    17568

```

The second panel does not display the heading and therefore, does not display the embedded page number, as shown in the following image.

<u>State</u>	<u>City</u>	<u>Category</u>	<u>Product</u>	<u>Budget</u>	<u>Units</u>		
IL	Chicago	Coffee	Espresso	32416			
			Latte	79015			
		Food	Biscotti	30001			
			Croissant	43271			
			Scone	45091			
		Gifts	Coffee Grinder	19224			
			Coffee Pot	16035			
			Mug	30881			
			Thermos	14137			
MO	St. Louis	Coffee	Espresso	32787			
			Latte	77141			
		Food	Biscotti	28764			
			Croissant	49327			
			Scone	36573			
		Gifts	Coffee Grinder	14779			
			Coffee Pot	14970			
			Mug	26837			
			Thermos	15903			
TX	Houston	Coffee	Espresso	36666			
			Latte	77501			
		Food	Biscotti	27074			
			Croissant	47050			
			Scone	32112			
		Gifts	Coffee Grinder	16625			
			Coffee Pot	16774			
			Mug	29374			
			Thermos	16779			
SUBFOOT FOR REGION Midwest							
SUBTOTAL BUDDOLLARS: 11194373 SUBTOTAL DOLLARS: 11400665							
CT	New Haven	Coffee	Capuccino	11098			
			Espresso	23676			
			Latte	74427			
		Food	Biscotti	46404			
			Croissant	46335			
			Scone	21038			
		Gifts	Coffee Grinder	13117			
			Coffee Pot	15190			
			Mug	32415			
			Thermos	17667			

**Example: Repeating Styled Headings and Footings on Paneled Pages**

The following request against the GGSALES data source has a report heading, a page heading with an image, a footing, a subheading, a subfooting, and a subtotal.

```

SET BYPANEL=ON
DEFINE FILE GGSALES
SHOWCATPROD/A30 = CATEGORY || ' / ' || PRODUCT;
END
TABLE FILE GGSALES
SUM
    DOLLARS/I8M AS ''
BY REGION
BY ST
BY CITY
ACROSS SHOWCATPROD AS 'Product Sales'

ON REGION SUBHEAD
" "
"Subheading Region <REGION"
" "
ON REGION SUBTOTAL AS '*TOTAL'
ON REGION SUBFOOT WITHIN
" "
"Subfooting Region <REGION"
" "
ON TABLE SUBHEAD
"Report Heading"
HEADING
"Page <TABPAGENO  "
" "
" "
" "
FOOTING
" "
"PAGE FOOTING "
ON TABLE SUBFOOT
" "
"Report Footing"
ON TABLE SET PAGE-NUM OFF
-*ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
    UNITS=IN,
    SQUEEZE=ON,
    ORIENTATION=PORTRAIT,
$

```

```
TYPE=REPORT,
  FONT='ARIAL',
  SIZE=9,
  HEADPANEL=ON,
  BORDER=ON,
$
TYPE=TITLE,
  STYLE=BOLD,
$
TYPE=TABHEADING,
  SIZE=20,
  STYLE=BOLD,
$
TYPE=TABFOOTING,
  SIZE=20,
  STYLE=BOLD,
$
TYPE=HEADING,
  SIZE=12,
  STYLE=BOLD,
$
TYPE=HEADING,
  LINE=1,
  JUSTIFY=RIGHT,
$
TYPE=HEADING,
  LINE=2,
  JUSTIFY=RIGHT,
$
TYPE=HEADING,
  LINE=3,
  JUSTIFY=RIGHT,
$
TYPE=HEADING,
  LINE=4,
  JUSTIFY=RIGHT,
$
TYPE=HEADING,
  LINE=5,
  JUSTIFY=RIGHT,
$
TYPE=HEADING,
  IMAGE=smplogol.gif,
  POSITION=(+0.000000 +0.000000),
$
TYPE=FOOTING,
  SIZE=12,
  STYLE=BOLD,
  JUSTIFY=RIGHT,
$
TYPE=SUBHEAD,
  SIZE=10,
  STYLE=BOLD,
$
```

```
TYPE=SUBFOOT,
    SIZE=10,
    STYLE=BOLD,
$
TYPE=SUBTOTAL,
    BACKCOLOR=RGB(210 210 210),
$
TYPE=ACROSSVALUE,
    SIZE=9,
    WRAP=ON,
$
TYPE=ACROSSTITLE,
    STYLE=BOLD,
$
TYPE=GRANDTOTAL,
    BACKCOLOR=RGB(210 210 210),
    STYLE=BOLD,
$
ENDSTYLE
END
```

Since HEADPANEL=ON for the entire report, both panels display all of the heading and footing elements.

The following image shows page 1 panel 1.

Report Heading

Information Builders

The Standard for Enterprise Business Intelligence

Page 1.1

			Product Sales				
			Coffee /Capuccino	Coffee /Espresso	Coffee /Latte	Food /Black&B	Food /Croissant
Region	State	City					
Subheading Region Midwest							
Midwest	IL	Chicago	-	\$420,439	\$975,340	\$378,412	\$549,386
	MO	St. Louis	-	\$419,143	\$968,981	\$368,077	\$613,871
	TX	Houston	-	\$465,366	\$938,245	\$345,236	\$587,887
*TOTAL Midwest			\$0	\$1,294,947	\$2,883,566	\$1,091,727	\$1,751,124
Subfooting Region Midwest							
Subheading Region Northeast							
Northeast	CT	New Haven	\$158,995	\$279,373	\$926,062	\$589,355	\$561,489
	MA	Boston	\$174,344	\$248,356	\$917,737	\$570,391	\$497,234
	NY	New York	\$208,756	\$322,378	\$926,006	\$642,259	\$622,095
*TOTAL Northeast			\$542,095	\$850,107	\$2,771,815	\$1,802,005	\$1,670,818
Subfooting Region Northeast							
Subheading Region Southeast							
Southeast	FL	Orlando	\$317,027	\$256,539	\$889,687	\$511,597	\$802,076
	GA	Atlanta	\$352,161	\$317,389	\$907,365	\$555,231	\$861,806
	TN	Memphis	\$274,812	\$279,844	\$820,584	\$438,889	\$638,477
*TOTAL Southeast			\$944,000	\$853,772	\$2,617,636	\$1,505,717	\$1,902,359
Subfooting Region Southeast							



The following image shows page 1 panel 2.

Report Heading

Information Builders

The Standard for Enterprise Business Intelligence

Page 1.2

			Product Sales				
			Food /Scooter	Gifts /Coffee Grinder	Gifts /Coffee Pot	Gifts /Mug	Gifts /Thermos
Region	State	City					
Subheading Region Midwest							
Midwest	IL	Chicago	\$595,069	\$233,292	\$204,628	\$376,754	\$167,801
	MO	St. Louis	\$461,953	\$181,570	\$180,153	\$343,852	\$195,696
	TX	Houston	\$418,386	\$204,292	\$204,687	\$386,337	\$194,319
*TOTAL Midwest			\$1,485,420	\$619,154	\$599,878	\$1,086,943	\$577,806
Subfooting Region Midwest							
Subheading Region Northeast							
Northeast	CT	New Haven	\$263,874	\$169,908	\$208,209	\$392,967	\$221,827
	MA	Boston	\$332,488	\$177,940	\$184,119	\$401,944	\$203,436
	NY	New York	\$290,811	\$161,352	\$196,462	\$349,300	\$178,636
*TOTAL Northeast			\$887,171	\$509,200	\$590,790	\$1,144,211	\$604,009
Subfooting Region Northeast							
Subheading Region Southeast							
Southeast	FL	Orlando	\$311,638	\$217,204	\$212,057	\$409,466	\$195,526
	GA	Atlanta	\$273,420	\$217,254	\$232,552	\$355,447	\$227,462
	TN	Memphis	\$315,399	\$171,319	\$200,694	\$337,790	\$209,449
*TOTAL Southeast			\$900,655	\$605,777	\$645,303	\$1,102,703	\$632,457
Subfooting Region Southeast							

## Customizing a Column Title

A column title identifies the data in a report. Use the AS phrase to change the default column title for customized data identification or more desirable formatting. You can change a column title:

- ☐ In a request.
- ☐ In a Master File.

A column title defaults to the field name in the Master File. For a calculated value (one created with COMPUTE), the title defaults to the field name in the request.

### ***Example:*** Using Default Column Titles

Consider this request:

```
TABLE FILE EMPDATA
SUM SALARY
BY DEPT
BY LASTNAME
WHERE DEPT IS 'SALES' OR 'CONSULTING' OR 'ACCOUNTING'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The report output illustrates these default column titles:

- ☐ The column title for the field named in the display command (SUM) is SALARY.
- ☐ The column titles for the fields named in the BY phrases are DEPT and LASTNAME.

The output is:

<u>DEPT</u>	<u>LASTNAME</u>	<u>SALARY</u>
ACCOUNTING	ANDERSON	\$32,400.00
	LOPEZ	\$26,400.00
	SANCHEZ	\$83,000.00
	SOPENA	\$79,000.00
	WANG	\$62,500.00
CONSULTING	ELLNER	\$35,900.00
	WANG	\$49,500.00
	WHITE	\$40,900.00
SALES	ADDAMS	\$54,100.00
	CASSANOVA	\$70,000.00
	CHISOLM	\$43,000.00
	DUBOIS	\$43,600.00
	LASTRA	\$115,000.00
	MEDINA	\$39,000.00
	OLSON	\$30,500.00

### **Reference:** Limits for Column Titles

- ☐ The width allotted for column titles has no limit other than the memory available.
- ☐ A column title for a styled output format can contain up to 16 lines of text.
- ☐ You can replace a column title for a field named in an ACROSS phrase by only one line of text.

### **Syntax:** How to Customize a Column Title in a Request

```
fieldname AS 'title_line_1 [, title_line_2, ...]'
```

where:

*fieldname*

Is a field named in a display command (such as PRINT or SUM), ACROSS phrase, or BY phrase.

*title\_line\_1,title\_line\_2*

Is the customized column title, enclosed in single quotation marks.

To specify a multiple-line column title, separate each line with a comma.

To customize a column title for a calculated value, use the syntax:

```
COMPUTE fieldname[/format] = expression AS 'title'
```

For related information, see [Creating Temporary Fields](#) on page 277.

**Tip:**

- ❑ To suppress the display of a column title, enter two consecutive single quotation marks without the intervening space. For example:

```
PRINT LAST_NAME AS ''
```

- ❑ To display underscores, enclose blanks in single quotation marks.
- ❑ If you use an AS phrase for a calculated value, repeat the COMPUTE command before referencing the next computed field.
- ❑ Multi-line column titles created with the AS phrase are not supported for ACROSS fields.

**Example: Customizing Column Titles in a Request**

This request customizes the column titles for the field named in the SUM command (SALARY), and the fields named in the BY phrases (DIV and DEPT).

```
TABLE FILE EMPDATA
SUM SALARY AS 'Total,Salary'
BY DIV AS 'Division'
BY DEPT AS 'Department'
WHERE DIV EQ 'NE' OR 'SE' OR 'CORP'
HEADING
"Current Salary Report"
ON TABLE SET STYLE *
TYPE=REPORT,GRID=OFF,$
ENDSTYLE
END
```

The output is:

Current Salary Report		
<u>Division</u>	<u>Department</u>	<u>Total Salary</u>
CORP	ACCOUNTING	\$283,300.00
	MARKETING	\$153,200.00
NE	CUSTOMER SUPPORT	\$81,800.00
	MARKETING	\$139,800.00
	SALES	\$82,600.00
SE	CONSULTING	\$85,400.00
	CUSTOMER SUPPORT	\$62,500.00
	MARKETING	\$113,000.00
	PERSONNEL	\$80,500.00
	PROGRAMMING & DVLPMT	\$49,000.00

**Example: Suppressing a Column Title**

This request suppresses the column title for LAST\_NAME. It also illustrates a multiple-line column title (EMPLOYEE NUMBER) for the data for EMP\_ID.

```
TABLE FILE EMPLOYEE
PRINT FIRST_NAME AS 'NAME' AND LAST_NAME AS ''
BY DEPARTMENT
BY EMP_ID AS 'EMPLOYEE,NUMBER'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

EMPLOYEE			
<u>DEPARTMENT</u>	<u>NUMBER</u>	<u>NAME</u>	
MIS	112847612	MARY	SMITH
	117593129	DIANE	JONES
	219984371	JOHN	MCCOY
	326179357	ROSEMARIE	BLACKWOOD
	543729165	MARY	GREENSPAN
	818692173	BARBARA	CROSS
PRODUCTION	071382660	ALFRED	STEVENS
	119265415	RICHARD	SMITH
	119329144	JOHN	BANNING
	123764317	JOAN	IRVING
	126724188	ANTHONY	ROMANS
	451123478	ROGER	MCKNIGHT

**Example:** Customizing a Column Title for a Calculated Value

This request customizes the column title for the calculated value REV.

```
TABLE FILE SALES
SUM UNIT_SOLD RETAIL_PRICE
COMPUTE REV/D12.2M = UNIT_SOLD * RETAIL_PRICE;AS 'GENERATED REVENUE'
BY PROD_CODE
WHERE CITY EQ 'NEW YORK'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

<u>PROD CODE</u>	<u>UNIT SOLD</u>	<u>RETAIL PRICE</u>	<u>GENERATED REVENUE</u>
B10	30	\$ .85	\$25.50
B17	20	\$1.89	\$37.80
B20	15	\$1.99	\$29.85
C17	12	\$2.09	\$25.08
D12	20	\$2.09	\$41.80
E1	30	\$ .89	\$26.70
E3	35	\$1.09	\$38.15

### Customizing a Column Title in a Master File

You can change the default column title using the optional TITLE attribute for a field. Any formatting you apply to the field will be applied to its customized title.

See the *Developing Reporting Applications* manual for details on the TITLE attribute.

### Distinguishing Between Duplicate Field Names

The command SET QUALTITLES determines whether or not duplicate field names appear as qualified column titles in report output. A qualified column title distinguishes between identical field names by including the segment.

Column titles specified in an AS phrase are used when duplicate field names are referenced in a MATCH command, or when duplicate field names exist in a HOLD file.

#### **Syntax:** How to Distinguish Between Duplicate Field Names

```
SET QUALTITLES = {ON|OFF}
```

where:

#### **ON**

Enables qualified column titles when duplicate field names exist and SET FIELDNAME is set to NEW (the default). For information on SET commands, see the *Developing Reporting Applications* manual.

#### **OFF**

Disables qualified column titles. OFF is the default value.

## Controlling Column Title Underlining Using a SET Command

The SET TITLELINE command allows you to control whether column titles are underlined for report output.

### **Syntax:** How to Control Column Title Underlining Using a SET Command

```
SET TITLELINE = (ON|OFF|SKIP)
```

```
ON TABLE SET TITLELINE (ON|OFF|SKIP)
```

where:

ON

Underlines column titles. ON is the default value.

OFF

Replaces the underline with a blank line.

SKIP

Omits both the underline and the line on which the underline would have displayed.

**Note:** ACROSSLINE is a synonym for TITLELINE.

### **Example:** Controlling Column Title Underlining Using a SET Command

The following request has a BY and an ACROSS field.

```
SET TITLELINE=ON
TABLE FILE GGSales
SUM UNITS BY PRODUCT
ACROSS REGION
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
ENDSTYLE
END
```



With the default value (ON) for SET TITLELINE, the column titles are underlined.

	Region			
	Midwest	Northeast	Southeast	West
Product				
Biscotti	86105	145242	119594	70436
Capuccino	.	44785	73264	71168
Coffee Grinder	50393	40977	47083	48081
Coffee Pot	47156	46185	49922	47432
Croissant	139182	137394	156456	197022
Espresso	101154	68127	68030	71675
Latte	231623	222866	209654	213920
Mug	86718	91497	88474	93881
Scone	116127	70732	73779	72776
Thermos	46587	48870	48976	45648

With SET TITLELINE=OFF, the column titles are not underlined, but the blank line where the underlines would have been is still there.

	Region			
	Midwest	Northeast	Southeast	West
Product				
Biscotti	86105	145242	119594	70436
Capuccino	.	44785	73264	71168
Coffee Grinder	50393	40977	47083	48081
Coffee Pot	47156	46185	49922	47432
Croissant	139182	137394	156456	197022
Espresso	101154	68127	68030	71675
Latte	231623	222866	209654	213920
Mug	86718	91497	88474	93881
Scone	116127	70732	73779	72776
Thermos	46587	48870	48976	45648

With SET TITLELINE=SKIP, both the underlines and the blank line are removed.

	Region			
	Midwest	Northeast	Southeast	West
Product				
Biscotti	86105	145242	119594	70436
Capuccino	.	44785	73264	71168
Coffee Grinder	50393	40977	47083	48081
Coffee Pot	47156	46185	49922	47432
Croissant	139182	137394	156456	197022
Espresso	101154	68127	68030	71675
Latte	231623	222866	209654	213920
Mug	86718	91497	88474	93881
Scone	116127	70732	73779	72776
Thermos	46587	48870	48976	45648

Controlling Column Title Underlining Using a StyleSheet Attribute

The TITLELINE attribute allows you to control whether column titles are underlined for report output.

Syntax: How to Control Column Title Underlining Using a StyleSheet Attribute

```
TYPE={REPORT|TITLE}, TITLELINE = (ON|OFF|SKIP)
```

where:

ON Underlines column titles. ON is the default value.

OFF Replaces the underline with a blank line.

SKIP Omits both the underline and the line on which the underline would have displayed.

**Example: Controlling Column Title Underlining Using a StyleSheet Attribute**

The following request has a BY and an ACROSS field.

```
TABLE FILE GGSALES
SUM UNITS BY PRODUCT
ACROSS REGION
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
TYPE=REPORT, TITELINE=ON, GRID=OFF, FONT=ARIAL,$
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
ENDSTYLE
END
```

With the default value (ON) for TITELINE, the column titles are underlined.

	Region			
	Midwest	Northeast	Southeast	West
Product				
Biscotti	86105	145242	119594	70436
Capuccino	.	44785	73264	71168
Coffee Grinder	50393	40977	47083	48081
Coffee Pot	47156	46185	49922	47432
Croissant	139182	137394	156456	197022
Espresso	101154	68127	68030	71675
Latte	231623	222866	209654	213920
Mug	86718	91497	88474	93881
Scone	116127	70732	73779	72776
Thermos	46587	48870	48976	45648

With TITLELINE=OFF, the column titles are not underlined, but the blank line where the underlines would have been is still there.

	Region			
	Midwest	Northeast	Southeast	West
Product				
Biscotti	86105	145242	119594	70436
Capuccino	.	44785	73264	71168
Coffee Grinder	50393	40977	47083	48081
Coffee Pot	47156	46185	49922	47432
Croissant	139182	137394	156456	197022
Espresso	101154	68127	68030	71675
Latte	231623	222866	209654	213920
Mug	86718	91497	88474	93881
Scone	116127	70732	73779	72776
Thermos	46587	48870	48976	45648

With TITLELINE=SKIP, both the underlines and the blank line are removed.

	Region			
	Midwest	Northeast	Southeast	West
Product				
Biscotti	86105	145242	119594	70436
Capuccino	.	44785	73264	71168
Coffee Grinder	50393	40977	47083	48081
Coffee Pot	47156	46185	49922	47432
Croissant	139182	137394	156456	197022
Espresso	101154	68127	68030	71675
Latte	231623	222866	209654	213920
Mug	86718	91497	88474	93881
Scone	116127	70732	73779	72776
Thermos	46587	48870	48976	45648

## Creating Labels to Identify Data

Labels enable you to provide meaningful and distinct names for the following report elements that are otherwise identified by generic labels:

- ❑ Row and column totals. See [Creating a Label for a Row or Column Total](#) on page 1513.
- ❑ Subtotals for sort groups. See [Creating a Label for a Subtotal and a Grand Total](#) on page 1515.
- ❑ Rows in a financial report. See [Creating a Label for a Row in a Financial Report](#) on page 1520.

## Creating a Label for a Row or Column Total

A label for a row or column total identifies the sum of values for two or more fields. A label draws attention to the total. It is particularly important that you create a label for a row or column total if you have both in one report.

For related information, see [Including Totals and Subtotals](#) on page 369.

### **Syntax:** How to Create a Label for a Row or Column Total

```
fieldname [AND] ROW-TOTAL[/justification[/format] [AS 'label']]
fieldname [AND] COLUMN-TOTAL[/justification] [AS 'label']
```

or

```
fieldname [AND] COLUMN-TOTAL[/justification] [AS 'label']
```

where:

*fieldname*

Is a field named in a display command.

*justification*

Is the alignment of the label. Valid values are:

**L** which left justifies the label.

**R** which right justifies the label.

**C** which centers the label.

For related information, see [Justifying a Label for a Row or Column Total](#) on page 1542.

*format*

Is the format of the row or column total. When fields with the same format are summed, the format of the total is the same as the format of the fields. When fields

with different formats are summed, the default D12.2 is used for either the row or column total.

*label*

Is the customized row or column total label. The default label is TOTAL.

You can also specify a row or column total with the ON TABLE phrase. With this syntax, you cannot include field names with ROW-TOTAL. Field names are optional with COLUMN-TOTAL.

```
ON TABLE ROW-TOTAL[/justification[/format] [AS 'label']]
ON TABLE COLUMN-TOTAL[/justification] [AS 'label']
[fieldname fieldname fieldname]
```

If a request queries a field created with COMPUTE, the value of that field is included in a row or column total. Keep that in mind when customizing a label that identifies the total.

**Example: Creating a Label for a Row and Column Total**

This request creates the label Total Population by State for the row total, and the label Total Population by Gender for the column total. The format D12 for ROW-TOTAL displays that data with commas.

```
TABLE FILE GGDEMOG
PRINT MALEPOP98 FEMPOP98
ROW-TOTAL/D12 AS 'Total Population by State'
BY ST
WHERE (ST EQ 'WY' OR 'MT')
ON TABLE COLUMN-TOTAL AS 'Total Population by Gender'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

<u>State</u>	<u>Male Population</u>	<u>Female Population</u>	<u>Total Population by State</u>
MT	426357	438836	865,193
WY	245023	245897	490,920
Total Population by Gender	671380	684733	1,356,113

**Example: Creating a Row Total Label With ACROSS**

This request adds the populations of two states, sorts the information using the ACROSS phrase, and labels the row totals as Total by Gender. There are two row totals within the Total by Gender column, Male Population and Female Population.

```
TABLE FILE GGDEMOG
SUM MALEPOP98/D12 FEMPOP98/D12
ROW-TOTAL AS 'Total by Gender'
ACROSS ST
WHERE ST EQ 'WY' OR 'MT';
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

State				Total by Gender	
MT		WY			
Male Population	Female Population	Male Population	Female Population	Male Population	Female Population
426,357	438,836	245,023	245,897	671,380	684,733

**Creating a Label for a Subtotal and a Grand Total**

Frequently, a report contains detailed information for a sort group, and it is useful to provide a subtotal for such a group, and a grand total for all groups at the end of the report.

For related information see [Including Totals and Subtotals](#) on page 369.

**Syntax: How to Create a Label for a Subtotal or a Grand Total**

```
{BY|ON} fieldname {SUB-TOTAL|SUBTOTAL|COLUMN-TOTAL} [MULTILINES]
      [field1 [AND] field2...] [AS 'label'] [WHEN expression];
```

where:

*fieldname*

Is a sort field named on a BY or ON phrase.

MULTILINES

Suppresses a subtotal when there is only one value at a sort break. After it is specified, MULTILINES suppresses the subtotal for every sort break with only one detail line. MULTI-LINES is a synonym for MULTILINES.

*field1 field2*

Are specific fields that will be subtotaled. A specified field overrides the default, which includes all numeric display fields.

*AS 'label'*

Is the customized label for the subtotal. You cannot change the default label for a higher level sort field if using SUB-TOTAL.

*WHEN expression*

Specifies a conditional subtotal as determined by a logical expression. For details, see [Using Expressions](#) on page 431.

### **Example:** Creating a Label for a Subtotal and a Grand Total

This request creates a customized label for the subtotal, which is the total dollar amount deducted from employee paychecks for city taxes per department; and the grand total which is the total dollar amount for both departments.

```
TABLE FILE EMPLOYEE
SUM DED_AMT BY DED_CODE BY DEPARTMENT
BY BANK_ACCT
WHERE DED_CODE EQ 'CITY'
WHERE BANK_ACCT NE 0
ON DEPARTMENT SUBTOTAL AS 'Total City Deduction for'
ON TABLE COLUMN-TOTAL AS '**GRAND TOTAL**'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```



In the output, the department values MIS and PRODUCTION are included by default in the customized subtotal label.

<u>DED CODE</u>	<u>DEPARTMENT</u>	<u>BANK ACCT</u>	<u>DED AMT</u>
CITY	MIS	40950036	\$14.00
		122850108	\$31.75
		163800144	\$82.70
Total City Deduction for MIS			\$128.45
	PRODUCTION	160633	\$7.42
		136500120	\$18.25
		819000702	\$60.20
Total City Deduction for PRODUCTION			\$85.87
**GRAND TOTAL**			\$214.32

**Example: Creating a Label for the Subtotal of a Specific Field**

This request creates a customized label, Order Total, for the subtotal for LINEPRICE. It uses the default label TOTAL for the grand total.

```
TABLE FILE CENTORD
PRINT PNUM QUANTITY LINEPRICE
  BY ORDER_NUM SUBTOTAL LINEPRICE AS 'Order Total'
WHERE ORDER_NUM EQ '28003' OR '28004';
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

<u>Order Number:</u>	<u>Product Number#:</u>	<u>Quantity:</u>	<u>Line Total</u>
28003	1006	1	\$501.60
	1008	1	\$239.18
	1020	1	\$324.06
	1032	1	\$103.23
	1034	1	\$520.40
Order Total 28003			\$1,688.47
28004	1006	1	\$471.83
	1016	1	\$282.19
	1018	1	\$462.96
	1022	1	\$401.30
	1028	1	\$110.34
	1036	1	\$295.00
Order Total 28004			\$2,023.62
TOTAL			\$3,712.09

**Syntax:** How to Create a Label for the Subtotal of a Calculated Value

```
{BY|ON} fieldname {SUMMARIZE|RECOMPUTE} [MULTILINES]
      [field1 [AND] field2...] [AS 'label'] [WHEN expression:]
ON TABLE {SUMMARIZE|RECOMPUTE}
```

where:

*fieldname*

Is a sort field named on a BY or ON phrase.

**MULTILINES**

Suppresses a subtotal when there is only one value at a sort break. After it is specified, MULTILINES suppresses the subtotal for every sort break with only one detail line. MULTI-LINES is a synonym for MULTILINES.

*field1 field2*

Are specific fields that will be subtotaled. Specified fields override the default, which includes all numeric display fields.

**AS** *'label'*

Is the customized label for the subtotal. You cannot change the default label for a higher level sort field if using SUMMARIZE.

**WHEN** *expression*

Specifies a conditional subtotal as determined by a logical expression. For details, see [Using Expressions](#) on page 431.

You can also generate a subtotal with the ON TABLE phrase:

**ON TABLE** {SUMMARIZE|RECOMPUTE}

### **Example: Creating a Label for the Subtotal of a Calculated Value**

This request creates a customized label for the subtotal, including the calculation for the field DG\_RATIO, created with COMPUTE.

```
TABLE FILE EMPLOYEE
SUM GROSS DED_AMT AND COMPUTE
DG_RATIO/F4.2 = DED_AMT / GROSS;
BY DEPARTMENT BY BANK_ACCT
WHERE BANK_ACCT NE 0
ON DEPARTMENT SUMMARIZE AS 'SUBTOTAL FOR '
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

In the output, the department values MIS and PRODUCTION are included by default in the customized subtotal label. The default grand total label is TOTAL.

<u>DEPARTMENT</u>	<u>BANK ACCT</u>	<u>GROSS</u>	<u>DED</u>	<u>AMT</u>	<u>DG</u>	<u>RATIO</u>
MIS	40950036	\$6,099.50	\$2,866.18			.47
	122850108	\$9,075.00	\$6,307.00			.69
	163800144	\$22,013.75	\$15,377.40			.70
SUBTOTAL FOR MIS		\$37,188.25	\$24,550.58			.66
PRODUCTION	160633	\$2,475.00	\$1,427.24			.58
	136500120	\$9,130.00	\$3,593.92			.39
	819000702	\$17,094.00	\$11,949.44			.70
SUBTOTAL FOR PRODUCTION		\$28,699.00	\$16,970.60			.59
TOTAL		\$65,887.25	\$41,521.18			.63

### Creating a Label for a Row in a Financial Report

Financial Modeling Language (FML) meets the special needs associated with creating, calculating, and presenting financially oriented data. FML reports are structured on a row-by-row basis. This organization gives you greater control over the data incorporated into a report and over its presentation.

You identify rows by labels that you can customize for accurate data identification and format to enhance the visual appearance and clarity of the data.

For details on FML reports, see [Creating Financial Reports With Financial Modeling Language \(FML\)](#) on page 1729.

### Formatting a Heading, Footing, Title, or Label

You can use a variety of strategies for enhancing the context-building elements in a report, that is, the headings, footings, column and row titles, and labels you assign to row and column totals and to subtotals. These additions enhance the visual appeal of a report and communicate a sense of completeness, using font and color for emphasis and distinctions, and alignment to add structural clarity and facilitate comprehension.

You can:

- ☐ Apply font characteristics to column and row titles, to headings, footing, and elements in them, and to subtotals, grand totals, and subtotal calculations. For more information, see [Applying Font Attributes to a Heading, Footing, Title, or Label](#) on page 1522.
- ☐ Add borders around headings and footings and grid lines around headings, footings, and column titles. For more information, see [Laying Out the Report Page](#) on page 1249.

You can also add space between heading or footing content and grid lines. For more information, see [Controlling the Vertical Positioning of a Heading or Footing](#) on page 1598.

- ☐ Left-justify, right-justify, or center a heading, footing, or individual lines in a multi-line heading or footing, column titles, and subtotals. For more information, see [Justifying a Heading, Footing, Title, or Label](#) on page 1529. See also [How to Center a Page Heading or Footing Using Legacy Formatting](#) on page 1535 and [How to Justify a Column Title Using a StyleSheet](#) on page 1538.

You can also justify a label for a row or column total. For more information, see [Justifying a Label for a Row or Column Total](#) on page 1542.

- ☐ Align a heading or footing, or elements within a heading or footing, based on:
  - ☐ Data column position in the main HTML table or on column position in an embedded HTML table created for the heading or footing in the report. These alignment techniques are supported for HTML reports. For more information, see [Aligning a Heading or Footing Element in an HTML, XLSX, EXL2K, PDF, PPTX, or DHTML Report](#) on page 1548.
  - ☐ Explicit width and justification specifications for multi-line headings and footings, including unit measurements (like inches) to enforce the alignment of decimal points in stacked numeric or alphanumeric data. These alignment techniques are supported for HTML and PDF reports. For more information, see [Aligning Content in a Multi-Line Heading or Footing](#) on page 1573.
  - ☐ An absolute or relative starting position, defined by either a unit measurement (like inches) or a column position. These alignment techniques are supported for PDF reports, although some features are also supported for HTML when used with an internal cascading style sheet. For more information, see [Positioning Headings, Footings, or Items Within Them](#) on page 1584.

For assistance in determining which of these approaches best suits your needs, see [Choosing an Alignment Method for Heading and Footing Elements](#) on page 1546.

- ☐ Control the vertical positioning of a heading or footing. For more information, see [Controlling the Vertical Positioning of a Heading or Footing](#) on page 1598.

- ❑ Position a report or sort heading or footing on a separate page. See [Placing a Report Heading or Footing on Its Own Page](#) on page 1605.

### Applying Font Attributes to a Heading, Footing, Title, or Label

You can specify font family, size, color, and style for any report element you can identify in a StyleSheet:

- ❑ Row and column titles. Styling is applied to either the default title or a customized title.
- ❑ Headings and footings, and elements within them, including specific lines in a multi-line heading or footing, items in a line, text strings, and embedded fields. Note that you can also specify background color for individual elements.
- ❑ Labels for subtotals, grand totals, subtotal calculations, and row totals. Styling applies to default or customized names.
- ❑ Page numbers in a heading or footing.
- ❑ Underlines and skipped lines (not supported in HTML reports).

For detailed syntax, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167. For details on font options, including size, color, and style, see [Formatting Report Data](#) on page 1611.

#### **Example:** Applying Font Characteristics to a Report Heading and Column Titles

This request uses a StyleSheet to select 12-point Arial bold for the report heading (Sales Report), and 10-point Arial italic for the default column titles (Category, Product, Unit Sales, Dollar Sales), based on the HTML point scale, which differs from standard point sizes. See [Formatting Report Data](#) on page 1611.

For an HTML report, the font name must be enclosed in single quotation marks. The StyleSheet attribute TYPE = TABHEADING identifies the report heading, and the attribute TYPE = TITLE identifies the column titles.

```
TABLE FILE GGSales
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT
ON TABLE SUBHEAD
"Sales Report"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE = REPORT, GRID=OFF, $
TYPE = TABHEADING, FONT = 'ARIAL', SIZE = 12, STYLE = BOLD, $
TYPE = TITLE, FONT = 'ARIAL', SIZE = 10, STYLE = ITALIC, $
ENDSTYLE
END
```

The output is:

## Sales Report

<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

**Example:   Setting Font Size for a Report Heading Using an Internal Cascading Style Sheet**

An internal cascading style sheet enables you to specify an absolute size, measured in points, rather than the corresponding HTML point scale, thereby providing greater control over the appearance of fonts in a report. See [Formatting Report Data](#) on page 1611 and [Controlling Report Formatting](#) on page 1139.

This request generates an internal cascading style sheet and specifies font characteristics for the report heading.

```
TABLE FILE GGSales
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT
ON TABLE SUBHEAD
"Sales Report"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLESHEET *
TYPE = REPORT, GRID=OFF, $
TYPE = TABHEADING, FONT = 'ARIAL', SIZE = 12, STYLE = BOLD, $
TYPE = TITLE, FONT = 'ARIAL', SIZE = 10, STYLE = ITALIC, $
ENDSTYLE
END
```

The output is:

Sales Report			
<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829



**Example: Applying Font Styles to a System Variable in a Report Heading**

This request includes the system variable &DATE in the heading. Styling is italic to distinguish it from the rest of the heading text, which is bold. The spot marker <+0> creates two items in the heading so that each one can be formatted separately.

```
TABLE FILE GGSALES
PRINT BUDDOLLARS DOLLARS
BY STCD
WHERE STCD EQ 'R1019'
ON TABLE SUBHEAD
"Sales Report for Store Code R1019 <+0>&DATE"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF,$
TYPE=TABHEADING, FONT='TIMES', SIZE=10, STYLE=BOLD,$
TYPE=TABHEADING, ITEM=2, STYLE=ITALIC,$
ENDSTYLE
END
```

The partial output is:

<b>Sales Report for Store Code R1019 06/13/02</b>		
<u>Store ID</u>	<u>Budget Dollars</u>	<u>Dollar Sales</u>
R1019	18560	25680
	21132	24010
	15880	22919
	24450	22605
	14556	18942
	13860	18915
	22806	18837
	19248	18535
	15120	18466
	15444	17784
	15190	17644
	12386	15996
	12720	14546
	17370	14328

Adding Borders and Grid Lines

You can add borders and grid lines to headings, footings, titles and labels. For detailed syntax, see [Laying Out the Report Page](#) on page 1249.

*Example:* Adding a Grid Around a Report Heading in a PDF Report

This request generates a PDF report with a grid around the heading, created with the GRID attribute, to set the heading off from the body of the report.

```
TABLE FILE GGSales
SUM BUdUnits UNITS BUdDollars Dollars
BY Category
ON TABLE SUBHEAD
"SALES REPORT"
"(CONFIDENTIAL)"
"December 2001 </1"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT PDF
ON TABLE SET SQUEEZE ON
ON TABLE SET STYLESHEET *
TYPE = TABHEADING, JUSTIFY = CENTER, GRID=ON, $
ENDSTYLE
END
```

The output is:

SALES REPORT ** (CONFIDENTIAL) ** December 2001				
Category	Budget Units	Unit Sales	Budget Dollars	Dollar Sales
Coffee	1385923	1376266	17293886	17231455
Food	1377564	1384845	17267160	17229333
Gifts	931007	927880	11659732	11695502

**Example: Emphasizing Column Titles With Horizontal Lines in a PDF Report**

This request generates a PDF report with horizontal lines, created with the HGRID attribute, above and below the column titles.

```
TABLE FILE GGSales
SUM BUDUNITS UNITS BUDDOLLARS DOLLARS
BY CATEGORY
ON TABLE SUBHEAD
"SALES REPORT"
"**(CONFIDENTIAL)**"
"December 2001 </1"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT PDF
ON TABLE SET SQUEEZE ON
ON TABLE SET STYLESHEET *
TYPE = TABHEADING, JUSTIFY = CENTER, FONT=ARIAL, SIZE=12, $
TYPE = TITLE, HGRID=ON, $
END
```

The output is:

SALES REPORT  
 \*\*(CONFIDENTIAL)\*\*  
 December 2001

Category	Budget Units	Unit Sales	Budget Dollars	Dollar Sales
Coffee	1385923	1376266	17293886	17231455
Food	1377564	1384845	17267160	17229333
Gifts	931007	927880	11659732	11695502

### ***Example:*    Formatting a Border Around a Report Heading**

This request generates an HTML report with a heavy red dotted line around the entire report heading.

```
TABLE FILE GGSales
SUM BUdUNITS UNITS BUdDOLLARS DOLLARS
BY CATEGORY
ON TABLE SUBHEAD
"</1 Sales Report"
"***CONFIDENTIAL**"
"December 2002 </1"
ON TABLE SET PAGE=NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
TYPE=TABLEHEADING, STYLE=BOLD, JUSTIFY=Center, BORDER=HEAVY,
BORDER-COLOR=RED, BORDER-STYLE=DOTTED, $
ENDSTYLE
END
```

The output is:

**Sales Report**  
**\*\*CONFIDENTIAL\*\***  
**December 2002**

<u>Category</u>	<u>Budget Units</u>	<u>Unit Sales</u>	<u>Budget Dollars</u>	<u>Dollar Sales</u>
Coffee	1385923	1376266	17293886	17231455
Food	1377564	1384845	17267160	17229333
Gifts	931007	927880	11659732	11695502

**Tip:** You can use the same BORDER syntax to generate this output in a PDF or PS report.

**Example: Formatting a Report Heading With Top and Bottom Borders**

This request generates a light blue line above the heading and a heavy double line of the same color below the heading. The request does not specify border lines for the left and right sides of the heading.

```
TABLE FILE GGSales
SUM BUDUNITS UNITS BUDDOLLARS DOLLARS
BY CATEGORY
ON TABLE SUBHEAD
"/1 Sales Report"
**CONFIDENTIAL**
"December 2002 /1"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
TYPE=TABLEDING, JUSTIFY=Center, BORDER-TOP-LIGHT, BORDER-COLOR-BLUE,
BORDER-BOTTOM-HEAVY, BORDER-BOTTOM-STYLE=DOUBLE,$
ENDSTYLE
END
```

The output is:

---

Sales Report				
**CONFIDENTIAL**				
December 2002				

---

<u>Category</u>	<u>Budget Units</u>	<u>Unit Sales</u>	<u>Budget Dollars</u>	<u>Dollar Sales</u>
Coffee	1385923	1376266	17293886	17231455
Food	1377564	1384845	17267160	17229333
Gifts	931007	927880	11659732	11695502

**Tip:** You can use the same BORDER syntax to generate this output in a PDF or PS report.

**Justifying a Heading, Footing, Title, or Label**

You can left-justify, right-justify, or center the following report elements:

- ☐ A heading or footing. See [Justifying a Heading or Footing](#) on page 1530.
- ☐ A column title. See [Justifying a Column Title](#) on page 1537.

- ❑ A label for a row or column total. See [Justifying a Label for a Row or Column Total](#) on page 1542.
- ❑ A label for a subtotal or grand total. See [Justifying a Label for a Subtotal or Grand Total](#) on page 1544.

In addition, you can use justification syntax in combination with other StyleSheet syntax to align headings and footings with other report elements, based on either unit measurements or relationships to other report elements, such as columns. For a summary of these options, see [Choosing an Alignment Method for Heading and Footing Elements](#) on page 1546.

### Justifying a Heading or Footing

You can left-justify, right-justify, or center a heading or footing in a StyleSheet. By default, a heading or footing is left justified. In addition, you can justify an individual line or lines in a multiple-line heading or footing.

To center a page heading or footing over the report data, you can use a legacy formatting technique that does not require a StyleSheet; simply include the CENTER command in a HEADING or FOOTING command.

**Justification behavior in HTML and PDF.** For HTML reports, justification is implemented with respect to the report width. That means a centered heading is centered over the report content. In contrast, for PDF reports the default justification area is the page width, rather than the report width, resulting in headings and footings that are not centered on the report. In most cases, you can achieve justification based on report width in a PDF report by adding the command SET SQUEEZE=ON to your request. This command improves the appearance of the report by eliminating excessive white space between columns and implements justification over the report content. However, if the heading is wider than the report, it will be centered on the page, even when SQUEEZE=ON.

**Tip:** You can also use justification syntax in combination with other StyleSheet syntax to align headings, footings, and items in them with other report elements, based on either unit measurements or relationships to other columns. For a summary of these options, see [Choosing an Alignment Method for Heading and Footing Elements](#) on page 1546.

### **Syntax:** How to Justify a Heading or Footing in a StyleSheet

```
TYPE = headfoot, [LINE = line_#,] JUSTIFY = option, $
```

where:

*headfoot*

Is the type of heading or footing. Valid values are TABHEADING, TABFOOTING, HEADING, FOOTING, SUBHEAD, and SUBFOOT.

*line\_#*

Optionally identifies a line by its position in the heading or footing so that you can individually align it. If a heading or footing has multiple lines and you omit this option, the value supplied for JUSTIFY applies to all lines.

*option*

Is the type of justification. Valid values are:

LEFT which left justifies the heading or footing. LEFT is the default value.

RIGHT which right justifies the heading or footing.

CENTER which centers the heading or footing.

For an alternative way to center a page heading or footing without a StyleSheet, see [How to Center a Page Heading or Footing Using Legacy Formatting](#) on page 1535.

**Note:** JUSTIFY is not supported with WRAP.

**Example:**     **Justifying a Report Heading**

This request centers the report heading PRODUCT REPORT, using the attribute JUSTIFY = CENTER.

```
TABLE FILE GGPRODS
SUM UNITS BY PRODUCT_DESCRIPTION BY PRODUCT_ID BY VENDOR_NAME
ON TABLE SUBHEAD
"PRODUCT REPORT"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE = REPORT, GRID=OFF, $
TYPE = REPORT, COLUMN = VENDOR_NAME, JUSTIFY = CENTER, $
TYPE = TABHEADING, JUSTIFY = CENTER, $
ENDSTYLE
END
```

The output is:

PRODUCT REPORT			
<u>Product</u>	<u>Product Code</u>	<u>Vendor Name</u>	<u>Unit Price</u>
Biscotti	F102	Delancey Bakeries	17.00
Coffee Grinder	G110	Appliance Craft	125.00
Coffee Pot	G121	Appliance Craft	140.00
Croissant	F103	West Side Bakers	28.00
French Roast	B142	European Specialities,	81.00
Hazelnut	B141	Coffee Connection	58.00
Kona	B144	Evelina Imports, Ltd	76.00
Mug	G100	NY Ceramic Supply	26.00
Scone	F101	Ridgewood Bakeries	13.00
Thermos	G104	ThermoTech, Inc	96.00

**Tip:** If you wish to run this report in PDF format, add the code ON TABLE SET SQUEEZE ON to eliminate excessive white space between columns and to center the heading over the report.

For more information on justifying a column title, see [Justifying a Column Title](#) on page 1537.

### **Example:** Justifying Individual Lines in a Multiple-Line Report Heading

In this request, heading line 1 (SALES REPORT) is centered, heading line 2 (\*\*CONFIDENTIAL\*\*) is also centered, and heading line 3 (December 2001) is right justified.

```
TABLE FILE GGSALES
SUM BUDDUNITS UNITS BUDDOLLARS DOLLARS
BY CATEGORY
ON TABLE SUBHEAD
"SALES REPORT"
"**(CONFIDENTIAL)**"
"December 2001"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE = REPORT, GRID=OFF, $
TYPE = TABHEADING, LINE = 1, JUSTIFY = CENTER, $
TYPE = TABHEADING, LINE = 2, JUSTIFY = CENTER, $
TYPE = TABHEADING, LINE = 3, JUSTIFY = RIGHT, $
ENDSTYLE
END
```



The output is:

SALES REPORT				
**(CONFIDENTIAL)**				
December 2001				
<u>Category</u>	<u>Budget Units</u>	<u>Unit Sales</u>	<u>Budget Dollars</u>	<u>Dollar Sales</u>
Coffee	1385923	1376266	17293886	17231455
Food	1377564	1384845	17267160	17229333
Gifts	931007	927880	11659732	11695502

**Tip:** To run this report in PDF format, add the code ON TABLE SET SQUEEZE ON to eliminate excessive white space between columns and to center the heading over the report.

**Example: Centering All Lines in a Multiple-Line Report Heading**

This request centers all lines in a multiple-line report heading using the single StyleSheet attribute for the entire heading.

```
TABLE FILE GGSales
SUM BUDUNITS UNITS BUDDOLLARS DOLLARS
BY CATEGORY
ON TABLE SUBHEAD
"SALES REPORT"
"**(CONFIDENTIAL)**"
"December 2001"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE = REPORT, GRID=OFF, $
TYPE = TABHEADING, JUSTIFY = CENTER, $
ENDSTYLE
END
```

The output is:

```

      SALES REPORT
    ***(CONFIDENTIAL)**
      December 2001

Category Budget Units Unit Sales Budget Dollars Dollar Sales
Coffee      1385923 1376266    17293886 17231455
Food        1377564 1384845    17267160 17229333
Gifts       931007  927880     11659732 11695502
```

**Tip:** To run this report in PDF format, add the code ON TABLE SET SQUEEZE ON to eliminate excessive white space between columns and to center the heading over the report.

**Reference:** **Justification Regions and Behavior**

The region in which text is justified depends on the relationship of the sizes of certain elements in the report:

- ☐ When SQUEEZE=ON, the maximum width of all the heading types in the report is calculated. This value is called MaxHeadWidth.

If MaxHeadWidth is less than or equal to the total width of the columns of the report, headings are justified in the space over the report columns.

If MaxHeadWidth exceeds the total width of the columns of the report, headings are centered and right-justified in the entire width of the page.

- ☐ When SQUEEZE=OFF, the maximum width of all the headings are not pre-calculated. Headings are centered in the entire width of the page.

With a styled, multiple-panel report (in which the width exceeds one page), headings can only appear in the first panel. Thus, the preceding calculations deal with the total width of the columns in the first panel rather than the total width of all the columns in the report.

**Syntax:**      **How to Center a Page Heading or Footing Using Legacy Formatting**

```
{HEADING|FOOTING} CENTER
  "content ... "
["content ... "]
.
.
.
["content ... "]
```

where:

**HEADING**

Is a page heading.

**FOOTING**

Is a page footing.

**CENTER**

Centers the page heading or footing over or under the report data.

*content*

Heading or footing content can include the following elements, between double quotation marks. If the ending quotation mark is omitted, all subsequent lines of the request are treated as part of the heading or footing.

*text*

Is text for the heading or footing. You can include multiple lines of text.

The text must start on a line by itself, following the HEADING or FOOTING command.

Text can be combined with variables and spot markers.

For related information, see [Limits for Headings and Footings](#) on page 1433.

*variable*

Can be any one or a combination of the following:

**Fields** (real data source fields, a virtual fields created with the DEFINE command in a Master File or report request, calculated values created with the COMPUTE command in a request, a system field such as TABPAGENO). You can qualify data source fields with certain prefix operators.

**Dialogue Manager variables.**

**Images.** You can include images in a heading or footing.

For details, see [Including an Element in a Heading or Footing](#) on page 1469.

*spot marker*

Enables you to position items, to identify items to be formatted, and to extend code beyond the 80-character line limit of the text editor.

<+0> divides a heading or footing into items for formatting. For details, see [Identifying a Heading, Footing, Title, or FML Free Text](#) on page 1191.

</n specifies skipped lines. For details, see [Controlling the Vertical Positioning of a Heading or Footing](#) on page 1598.

<-n to position the next character on the line. For details, see [Using Spot Markers to Refine Positioning](#) on page 1593.

<0x continues a heading or footing specification on the next line of the request. For details, see [Extending Heading and Footing Code to Multiple Lines in a Report Request](#) on page 1434.

**Note:** When a closing spot marker is immediately followed by an opening spot marker (><), a single space text item will be placed between the two spot markers (> <). This must be considered when applying formatting.

**Blank lines**

If you omit all text, variables, and spot markers, you have a blank heading or footing line (for example, " ") which you can use to skip a line in the heading or footing. (You can also skip a line using a vertical spot marker, such as </1.)

**Tip:** Do not use the command CENTER with the StyleSheet attribute JUSTIFY = CENTER. A single method will generate the desired result.

**Example: Centering a Page Heading**

This request uses the command CENTER in the page heading syntax.

```
TABLE FILE EMPLOYEE
HEADING CENTER
"ACCOUNT REPORT FOR DEPARTMENT <DEPARTMENT"
PRINT CURR_SAL BY DEPARTMENT BY HIGHEST BANK_ACCT
BY EMP_ID
ON DEPARTMENT PAGE-BREAK
ON TABLE SET PAGE-NUM OFF
ON TABLE SET WEBVIEWER ON
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE=REPORT, SIZE=10, GRID=OFF,$
ENDSTYLE
END
```

The page heading is centered over the report data, as shown in the first page of output.

ACCOUNT REPORT FOR DEPARTMENT MIS					
<u>DEPARTMENT</u>	<u>BANK</u>	<u>ACCT</u>	<u>EMP</u>	<u>ID</u>	<u>CURR SAL</u>
MIS	163800144	818692173			\$27,062.00
	122850108	326179357			\$21,780.00
	40950036	117593129			\$18,480.00
		112847612			\$13,200.00
		219984371			\$18,480.00
		543729165			\$9,000.00

**Tip:** If you do not see the navigation arrows, click the maximize button.

## Justifying a Column Title

You can left-justify, right-justify, or center a column title for a display field, BY field, ACROSS field, or calculated value using a StyleSheet.

If a title is specified with an AS phrase in a request, or with the TITLE attribute in a Master File, that title will be justified, as specified for the field in StyleSheet syntax, if such syntax exists in the request. For related information, see [Customizing a Column Title](#) on page 1501.

**Justification behavior in HTML and PDF.** For HTML reports, justification is implemented with respect to the report width. That means a centered column title is centered over a report column. In contrast, for PDF reports the default justification area is the page width, rather than the report width, resulting in column titles that are not centered over the report column. You can achieve justification based on report width in a PDF report by adding the command SET SQUEEZE=ON to your request. This command improves the appearance of the report by eliminating excessive white space between columns and implements justification over the report content.

You can also justify a column title for a display or BY field using legacy formatting methods. However, when legacy formatting is applied to an ACROSS field, data values, not column titles, are justified as specified. See [How to Justify a Column Title for a Display or BY Field Using Legacy Formatting](#) on page 1541.

**Syntax:**      **How to Justify a Column Title Using a StyleSheet**

To justify a column title for a vertical sort column (generated by BY) or a display column (generated by PRINT, LIST, SUM, or COUNT), the StyleSheet syntax is

```
TYPE=TITLE, [COLUMN=column,] JUSTIFY=option, $  
TYPE=ACROSSTITLE, [ACROSS=column,] JUSTIFY=option, $  
TYPE=ACROSSVALUE, [COLUMN=column,] JUSTIFY=option, $
```

To justify a horizontal sort column title (generated by ACROSS), the StyleSheet syntax is

```
TYPE=ACROSSTITLE, [ACROSS=column,] JUSTIFY=option, $
```

To justify an ACROSS value or a ROW-TOTAL column title in an HTML report, use

```
TYPE=ACROSSVALUE, [COLUMN=column,] JUSTIFY=option, $
```

where:

**TITLE**

Specifies a vertical sort (BY) title or a display field title.

*column*

Specifies the column whose title you wish to justify. If you omit this attribute and value, the formatting will be applied to all of the report's column titles. For details on identifying columns, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

**ACROSSTITLE**

Specifies a horizontal sort (ACROSS) title.

**ACROSSVALUE**

Specifies a horizontal sort (ACROSS) value or a ROW-TOTAL column title.

*option*

Is the type of justification. Valid values are:

**LEFT** which left justifies the column title. This value is the default for an alphanumeric field.

**RIGHT** which right justifies the column title. This value is the default for a numeric or date field.

**CENTER** which centers the column title. You cannot center an ACROSSTITLE in a PDF report.

**Note:** JUSTIFY is not supported with WRAP.

### **Example:** Using a StyleSheet to Justify Column Titles for Display and BY Fields

This request centers the column titles for STORE\_NAME and ADDRESS1. The default column title for STORE\_NAME is Store Name, as specified in the Master File with the TITLE attribute. The default column title for ADDRESS1 is Contact, also specified in the Master File. The request right-justifies the column title for STATE, which is specified in the AS phrase as St. Each column is identified by its field name and justified separately.

```
TABLE FILE GGSTORES
PRINT STORE_NAME STATE AS 'St' BY ADDRESS1
WHERE STATE EQ 'CA'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET SQUEEZE ON
ON TABLE SET STYLE SHEET *
TYPE = REPORT, GRID=OFF, $
TYPE=TITLE, COLUMN=STORE_NAME, JUSTIFY=CENTER, $
TYPE=TITLE, COLUMN=STATE, JUSTIFY=RIGHT, $
TYPE=TITLE, COLUMN=ADDRESS1, JUSTIFY=CENTER, $
ENDSTYLE
END
```

The output is:

<u>Contact</u>	<u>Store Name</u>	<u>St</u>
JEFF DARFELL	GOTHAM GRINDS #1244	CA
PAT SMILEY	GOTHAM GRINDS #1040	CA

**Example:**    **Using a StyleSheet to Justify a Column Title for ACROSS and ROW-TOTAL Fields**

This request centers the column title, State, created by the ACROSS phrase over the two values (MT and WY) and the row total column title, Total by Gender, over the two row totals (Male Population and Female Population). Notice that each across value functions as a title for one or more columns in the report.

```
TABLE FILE GGDEMOG
SUM MALEPOP98 FEMPOP98
ROW-TOTAL/D12 AS 'Total by Gender'
ACROSS ST
WHERE ST EQ 'WY' OR 'MT';
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
TYPE=ACROSSTITLE, JUSTIFY-CENTER, FONT='TIMES', SIZE=11, STYLE-BOLD, $
TYPE=ACROSSVALUE, COLUMN=N5, JUSTIFY-CENTER, $
ENDSTYLE
END
```

The output is:

State					
MT		WY		Total by Gender	
Male Population	Female Population	Male Population	Female Population	Male Population	Female Population
426357	438836	245023	245897	671380	684733

**Example:**    **Using a StyleSheet to Justify a Column Title for a Calculated Value**

This request identifies the column title of the calculated value and left justifies it over the data.

```
TABLE FILE SALES
SUM UNIT_SOLD RETAIL_PRICE
COMPUTE REV/D12.2M = UNIT_SOLD * RETAIL_PRICE;
BY PROD_CODE
WHERE CITY EQ 'NEW YORK'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
TYPE=TITLE, COLUMN=REV, STYLE-BOLD, JUSTIFY-LEFT, $
ENDSTYLE
END
```



The output is:

<u>PROD CODE</u>	<u>UNIT SOLD</u>	<u>RETAIL PRICE</u>	<u>REV</u>
B10	30	\$ .85	\$25.50
B17	20	\$1.89	\$37.80
B20	15	\$1.99	\$29.85
C17	12	\$2.09	\$25.08
D12	20	\$2.09	\$41.80
E1	30	\$ .89	\$26.70
E3	35	\$1.09	\$38.15

**Note:** To run this report in PDF format, add the code `ON TABLE SET SQUEEZE ON` to eliminate excessive white space between columns and to justify column titles properly over the data.

### **Syntax:** How to Justify a Column Title for a Display or BY Field Using Legacy Formatting

*fieldname/justification [/format] [AS 'title']*

where:

*fieldname*

Is the name of the field.

*justification*

Is the type of justification. Valid values are:

**L** which left justifies the column title. This value is the default for an alphanumeric field.

**R** which right justifies the column title. This value is the default for a numeric or date field.

**C** which centers the column title.

*/format*

Is an optional format specification for the field. For a display field, you can combine the justification value with the format value (in either order) to adjust the width of the column data or to specify display options.

*AS 'title'*

Is an optional customized column title.

**Tip:** For an ACROSS field, this syntax justifies data values, not column titles. For syntax that will justify the title, see [How to Justify a Column Title Using a StyleSheet](#) on page 1538.

**Example: Using Legacy Formatting to Justify Column Titles for Display and BY Fields**

This request centers the column titles for STORE\_NAME and ADDRESS1. The default column title for STORE\_NAME is Store Name, as specified in the Master File with the TITLE attribute. The default column title for ADDRESS1 is Contact, also specified in the Master File. The request right justifies the column title for STATE, which is specified in the AS phrase as St.

```
TABLE FILE GGSTORES
PRINT STORE_NAME/C STATE/R AS 'St' BY ADDRESS1/C
WHERE STATE EQ 'CA'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE = REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

<u>Contact</u>	<u>Store Name</u>	<u>St</u>
JEFF DARFELL	GOTHAM GRINDS #1244	CA
PAT SMILEY	GOTHAM GRINDS #1040	CA

**Note:** Add the syntax, ON TABLE SET SQUEEZE ON to your request if you are using PDF format.

**Justifying a Label for a Row or Column Total**

You can left-justify, right-justify, or center a label for a row or column total. For related information, see [Creating Labels to Identify Data](#) on page 1513.

**Syntax: How to Justify a Label for a Row or Column Total Using Legacy Formatting**

```
ROW-TOTAL/justification [/format] [AS 'label']
COLUMN-TOTAL/justification [AS 'label']
```

or

```
COLUMN-TOTAL/justification [AS 'label']
```

where:

*justification*

Is the type of justification. Valid values are:

L which Left justifies the label.

**R** which right justifies the label.

**C** which centers the label.

*/format*

Is an optional format specification for a row total. You can combine the alignment value with the format value (in either order) to adjust the width of the column data or specify display options.

**AS 'label'**

Is an optional customized label.

### **Example: Centering a Label for a Row Total**

This request creates the stacked label Total, Population, by State for the row total and centers it. The format D12 for ROW-TOTAL displays commas by default.

```
TABLE FILE GGDEMOG
PRINT MALEPOP98 FEMPOP98
ROW-TOTAL/C/D12 AS 'Total,Population,by State'
BY ST
WHERE (ST EQ 'WY' OR 'MT')
ON TABLE COLUMN-TOTAL AS 'Total by Gender'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE = REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

<u>State</u>	<u>Male Population</u>	<u>Female Population</u>	Total Population by State
MT	426357	438836	865,193
WY	245023	245897	490,920
Total by Gender	671380	684733	1,356,113

## Justifying a Label for a Subtotal or Grand Total

You cannot directly justify a customized label for a subtotal. However, for HTML, EXL2K, or XLSX report output, if columns are being totaled or subtotaled by the one subtotal command, and you do not specify a column in the StyleSheet, formatting is applied to the totals and subtotals of all columns and to the labeling text that introduces the total and subtotal values. For related information, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

### *Example:* Justifying Subtotal and Grand Total Labels

This request subtotals the numeric columns in the report and right-justifies the output, including the text of the label that precedes the values for the subtotals. Since numeric output is right-justified by default, in this example the justification specifications in the StyleSheet are used to reposition the labels. The default label for the automatically generated grand total is also right-justified.

```
TABLE FILE EMPLOYEE
SUM DED_AMT BY DED_CODE BY DEPARTMENT
BY BANK_ACCT
WHERE DED_CODE EQ 'CITY'
WHERE BANK_ACCT NE 0
ON DEPARTMENT SUBTOTAL AS 'Total City Deduction for'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
TYPE-SUBTOTAL, STYLE=BOLD, JUSTIFY=RIGHT,$
TYPE-GRANDTOTAL, STYLE=BOLD, JUSTIFY=RIGHT,$
ENDSTYLE
END
```

The output is:

<u>DED CODE</u>	<u>DEPARTMENT</u>	<u>BANK ACCT</u>	<u>DED AMT</u>
CITY	MIS	40950036	\$14.00
		122850108	\$31.75
		163800144	\$82.70
<b>Total City Deduction for MIS</b>			<b>\$128.45</b>
	PRODUCTION	160633	\$7.42
		136500120	\$18.25
		819000702	\$60.20
<b>Total City Deduction for PRODUCTION</b>			<b>\$85.87</b>
<b>TOTAL</b>			<b>\$214.32</b>

### Choosing an Alignment Method for Heading and Footing Elements

To align text and data in headings and footings based on factors other than left/right/center justification, consider the following descriptions before deciding which alignment method best suits your needs.

Alignment Method	Applies to ...	When to use...	Related Methods
<b>1) StyleSheet</b> <b>Attributes:</b>  HEADALIGN  COLSPAN  JUSTIFY  <b>Details:</b> See <a href="#">Aligning a Heading or Footing Element in an HTML, XLSX, EXL2K, PDF, PPTX, or DHTML Report</a> on page 1548.	HTML  XLSX  EXL2K  PDF  PPTX  DHTML	<b>To align heading or footing items in HTML and EXL2K reports:</b> If you expect to display reports in HTML or EXL2K format, use HEADALIGN options to align heading and footing items with either columns in the HTML table for the body of the report or with cells in an embedded HTML table. The browser handles alignment based on your specifications, without requiring unit measurements, which are required with WIDTH and JUSTIFY.  <b>To align heading or footing items in PDF reports:</b> If you expect to display reports in PDF format, use the HEADALIGN=BODY option to align heading and footing items with columns in the report body.  <b>To specify a heading or footing item that spans multiple columns:</b> You can combine HEADALIGN syntax with the COLSPAN attribute to achieve this result. For details, see <a href="#">Aligning a Heading or Footing Element Across Columns in an HTML or PDF Report</a> on page 1566.	

Alignment Method	Applies to ...	When to use...	Related Methods
<b>2) StyleSheet Attributes:</b> WIDTH JUSTIFY  <b>Details:</b> See <a href="#">Aligning Content in a Multi-Line Heading or Footing</a> on page 1573.	HTML PDF PS	<p><b>For portability between HTML and PDF:</b> To code a request that can be used without revision to produce identical output in HTML (with internal cascading style sheets) and in PDF, use WIDTH and JUSTIFY attributes in your StyleSheet. These settings can be applied to report, page, and sort headings and footings.</p> <p><b>To align heading or footing items:</b> Used together, WIDTH and JUSTIFY allow you to align specific items in the heading, rather than entire headings or footings or entire heading or footing lines, where the implied justification width is the total width of the report panel. To right- or center-justify an item in a heading or footing, you must know the width of the area you want to justify it in. That information is provided by the WIDTH attribute.</p> <p><b>To align decimal points in a multi-line heading or footing:</b> Use this technique to align decimal points in data that has varying numbers of decimal places. You define the width of the decimal item, then you measure how far in from the right side of a column you want to position the decimal point. This places the decimal point in the same position in a column, regardless of the number of decimal places displayed to its right.</p>	For an HTML, PDF, or EXL2K report, you can align specific items with HEADALIGN options.

Alignment Method	Applies to ...	When to use...	Related Methods
<b>3) StyleSheet Attribute:</b> POSITION <b>Details:</b> See <a href="#">Positioning Headings, Footings, or Items Within Them</a> on page 1584 and <a href="#">Laying Out the Report Page</a> on page 1249.	PDF PS HTML (limited)	<p><b>To set starting positions for headings or footings, or items within them:</b> Use POSITION syntax to specify absolute and relative starting positions.</p> <p>In HTML, with an internal cascading style sheet, you can use POSITION to specify the starting point for a heading or footing line. You can also position an image in a heading or footing.</p> <p><b>To align heading and footing items with columns:</b> Use POSITION syntax to align a heading item with a column position. For example, the syntax</p> <pre>TYPE=SUBHEAD, LINE=1, ITEM=3, POSITION=SALES, \$</pre> <p>places ITEM 3 of the sort heading at the horizontal position where the column SALES is.</p>	<p>For a PDF report, you can accomplish most positioning with WIDTH and JUSTIFY.</p> <p>For an HTML or PDF report, you can align a heading item with a column by setting the HEADALIGN attribute to BODY.</p>

## Aligning a Heading or Footing Element in an HTML, XLSX, EXL2K, PDF, PPTX, or DHTML Report

For HTML output (and for Excel 2000 output, which uses HTML alignment), you can position text and field items in headings and footings using HEADALIGN options. These options work within the limitations of HTML and browser technologies to provide a significant degree of formatting flexibility. Here is how HEADALIGN works.

For PDF output, you can use the HEADALIGN=BODY option to align heading and footing elements with the report body.



For HTML or Excel 2000 output, when HEADALIGN is set either to BODY or INTERNAL, output is laid out as an HTML table, which means that the browser determines the widths of the columns, thereby limiting the precise positioning of items. A basic rule governs the placement of heading or footing items: each item (text or embedded field) is placed in sequence into the next HTML table cell (<TD>). When HEADALIGN is set to NONE, the default, all the items in the heading or footing are strung together, inside a single cell. The browser stretches the heading table and the report table to accommodate the length of the text.

You can exercise control over the placement of items by overriding the default and choosing either BODY or INTERNAL:

- ❑ HEADALIGN=BODY puts heading item cells in the same HTML table (for HTML or EXL2K output) as the body of the report, ensuring that the items in the heading and the data in the body of the report line up naturally since they have the same column widths. For PDF output, HEADALIGN=BODY aligns heading items with data columns. This is a simple and useful way to align heading items with columns of data. For example, suppose that you have computed subtotal values that you want to include in a sort footing. Using HEADALIGN=BODY, you can align the subtotals in the same columns as the data that is being totaled.
- ❑ HEADALIGN=INTERNAL puts the heading items in an HTML table of its own. This allows the heading items to be aligned vertically with each other, independent of the data, since the widths of the heading items do not affect the width of the report columns and vice versa.
- ❑ In EXL2K formatted reports, with headings or footings containing multiple items that are separated by spot markers without spaces, the spot marker adds an additional space between the items within the text in the cell. The workaround is to use XLSX formatted reports, instead.

To compare sample output, see [Comparing Output Generated With HEADALIGN Options](#) on page 1554.

To break a text string into multiple parts for manipulation across columns, you can use <+0> spot markers in the request. For details, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

You can use HEADALIGN options in conjunction with the COLSPAN attribute. COLSPAN allows heading items to span multiple table columns, thereby providing additional flexibility in how you can design your headings. For details, see [Aligning a Heading or Footing Element Across Columns in an HTML or PDF Report](#) on page 1566.

If there is more than one heading or footing type in a report, you can individually align any element within each of them using this syntax.

**Tip:** For a summary of other alignment methods, see [Choosing an Alignment Method for Heading and Footing Elements](#) on page 1546.

**Syntax:**      **How to Align a Heading or Footing Element in an HTML or PDF Report**

`TYPE = {REPORT|headfoot}, HEADALIGN = option, $`

where:

**REPORT**

Applies the chosen alignment to all heading and footing elements in a report.

*headfoot*

Is the type of heading or footing. Valid values are TABHEADING, TABFOOTING, HEADING, FOOTING, SUBHEAD, and SUBFOOT.

*option*

Is the type of alignment. Valid values are:

**NONE** which places heading items in HTML reports in an embedded HTML table inside the main (body) table, and strings together, in a single cell of the embedded table, all the heading items (text and fields) on a line. In PDF reports, this uses the default alignment heading alignment. NONE is the default value.

**INTERNAL** which places heading items in an HTML table of its own, with each item in a separate cell. This allows the heading items to be aligned vertically with each other, independent of the data columns. The widths of the heading items do not affect the widths of the report columns and vice versa.

**Note:** HEADALIGN=INTERNAL is not supported in PDF reports.

**BODY** which aligns heading items with data columns. For HTML output, this places the items in the cells of the same HTML table as the body of the report. Since they have the same column widths, the items in the heading and the data in the body of the report line up naturally. For PDF output, this aligns the heading or footing elements with the data columns.

**Note:** HEADALIGN=BODY does not support paneling.

You can combine HEADALIGN options with the COLSPAN attribute to allow heading items to span multiple HTML table columns. For details, see [How to Align a Heading or Footing Element in an HTML or PDF Report](#) on page 1550.

**Example: Aligning Subfooting Items With Report Columns in PDF Report Output**

In the following request against the GGORDER data source, the subfooting has a text object ("Total") and a field object (ST.QUANTITY). The subfooting aligns the items with their report columns using TYPE=SUBFOOT, HEADALIGN=BODY, \$. The text object is placed in the second report column using the <+0 spot marker, and the field object is placed in the third report column using another <+0 spot marker. Then the text item is left aligned (the default) with its report column. The field object is right aligned with its report column.

```
TABLE FILE GGORDER
PRINT QUANTITY
ORDER_NUMBER ORDER_DATE STORE_CODE
BY PRODUCT_CODE BY PRODUCT_DESCRIPTION
WHERE ORDER_DATE EQ '01/01/96'
WHERE STORE_CODE EQ 'R1019'
ON PRODUCT_CODE SUBFOOT
" <+0 Total: <+0 <ST.QUANTITY"
ON TABLE SET PAGE-NUM OFF
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLESHEET *
TYPE = SUBFOOT,HEADALIGN=BODY, $
TYPE = SUBFOOT,OBJECT=TEXT,STYLE = BOLD, $
TYPE = SUBFOOT,OBJECT=FIELD,JUSTIFY=RIGHT,STYLE = BOLD, $
ENDSTYLE
END
```

The output shows that the text *Total* is aligned with the product names and the subtotal field object is right aligned with the *Ordered Units* column.

<u>Product Code</u>	<u>Product</u>	<u>Ordered Units</u>	<u>Order Number</u>	<u>Order Date</u>	<u>Store Code</u>
B141	Hazelnut	300	1	01/01/96	R1019
		117	2	01/01/96	R1019
	<b>Total:</b>	<b>417</b>			
B142	French Roast	126	3	01/01/96	R1019
		86	4	01/01/96	R1019
		49	5	01/01/96	R1019
	<b>Total:</b>	<b>261</b>			
F101	Scone	37	11	01/01/96	R1019
		325	12	01/01/96	R1019
	<b>Total:</b>	<b>362</b>			
F102	Biscotti	399	13	01/01/96	R1019
	<b>Total:</b>	<b>399</b>			
F103	Croissant	93	14	01/01/96	R1019
		433	15	01/01/96	R1019
	<b>Total:</b>	<b>526</b>			
G100	Mug	289	6	01/01/96	R1019
		231	7	01/01/96	R1019
	<b>Total:</b>	<b>520</b>			
G104	Thermos	347	8	01/01/96	R1019
	<b>Total:</b>	<b>347</b>			
G110	Coffee Grinder	265	9	01/01/96	R1019
	<b>Total:</b>	<b>265</b>			
G121	Coffee Pot	309	10	01/01/96	R1019
	<b>Total:</b>	<b>309</b>			

### **Example:** Using OVER With HEADALIGN=BODY in a PDF Report

When aligning heading elements with the data line using HEADALIGN=BODY, the first row of fields serves as the anchor data row. Each heading line contains the number of columns presented in the anchor data row. Any additional columns that may appear on other data lines are not presented. If the first row of data contains fewer data value cells than other data rows, you will be unable to add alignment columns within headings for these additional columns.

In the following example, the first row (the anchor data row) contains a single value. Items placed in headings to correspond with column two that appears on subsequent rows are not displayed.

```
SET LAYOUTGRID=ON
TABLE FILE GGSales
"Product<+0>"
"Units<+0>Dollars"
SUM
PRODUCT AS ''
OVER
UNITS/D8C AS '' DOLLARS/D12.2CM AS ''
BY PRODUCT NOPRINT
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, SQUEEZE=ON, FONT=ARIAL, SIZE=8, LEFTGAP=.1, RIGHTGAP=.1,
GAPINTERNAL=ON, LEFTMARGIN=1, $
TYPE=REPORT, BORDER=ON, $
TYPE=HEADING, BORDERALL=ON, HEADALIGN=BODY, $
TYPE=HEADING, LINE=1, ITEM=1, COLSPAN=2, WIDTH=2, JUSTIFY=LEFT, $
TYPE=HEADING, LINE=2, ITEM=1, WIDTH=1, JUSTIFY=LEFT, $
TYPE=HEADING, LINE=2, ITEM=2, WIDTH=1, JUSTIFY=LEFT, $
TYPE=REPORT, COLUMN=PRODUCT(2), SQUEEZE=2, $
TYPE=REPORT, COLUMN=UNITS, SQUEEZE=1, $
TYPE=REPORT, COLUMN=DOLLARS, SQUEEZE=1, $
END
```

The output shows that the heading lines have one column each, while the data lines alternate between one column and two columns.

PAGE 1	
Product	
Units	
Biscotti	
421,377	\$5,283,317.00
Capuccino	
189,217	\$2,381,590.00
Coffee Grinder	
186,534	\$2,337,567.00
Coffee Pot	
190,695	\$2,449,585.00
Croissant	
630,054	\$7,749,902.00
Espresso	
308,986	\$3,906,243.00
Latte	
878,063	\$10,943,622.00
Mug	
360,570	\$4,522,521.00
Scone	
333,414	\$4,216,114.00
Thermos	

**Example:** Comparing Output Generated With HEADALIGN Options

The requests that follow illustrate the differences in alignment with each HEADALIGN setting. The grid lines are exposed in the output to help distinguish the HTML table created for the body of the report from the embedded HTML tables created for the heading in some variations.

All HEADALIGN settings are compatible with COLSPAN syntax, which allows heading items to span multiple columns.

```
TABLE FILE CAR
SUM SALES BY COUNTRY BY CAR BY MODEL
ON COUNTRY SUBHEAD
"This is my subhead"
" "
"Country is:<COUNTRY Car is:<CAR"
"Model is:<MODEL"
IF COUNTRY EQ 'ENGLAND'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLESHEET *
TYPE=SUBHEAD, HEADALIGN=OPTION, $
TYPE=SUBHEAD, LINE=1, ITEM=1, COLSPAN=4, JUSTIFY=CENTER, $
ENDSTYLE
END
```

**HEADALIGN=NONE** without the second TYPE=SUBHEAD declaration highlighted in the request syntax creates a separate table with default left alignment. The text and fields in each heading line are strung together in a single HTML table cell.

```
TYPE=SUBHEAD, HEADALIGN=NONE, $
```

COUNTRY	CAR	MODEL	SALES
This is my subhead			
Country is:ENGLAND Car is:JAGUAR			
Model is:V12XKE AUTO			
ENGLAND	JAGUAR	V12XKE AUTO	0
		XJ12L AUTO	12000
	JENSEN	INTERCEPTOR III	0
	TRIUMPH	TR7	0

**HEADALIGN=NONE with COLSPAN**

```
TYPE=SUBHEAD, HEADALIGN=NONE, $
TYPE=SUBHEAD, LINE=1, ITEM=1, COLSPAN=4, JUSTIFY=CENTER, $
```

COUNTRY	CAR	MODEL	SALES
This is my subhead			
Country is:ENGLAND Car is:JAGUAR			
Model is:V12XKE AUTO			
ENGLAND	JAGUAR	V12XKE AUTO	0
		XJ12L AUTO	12000
		JENSEN	INTERCEPTOR III
		TRIUMPH	TR7

The first line is centered across all four columns of the internal table, based on the COLSPAN=4 setting.

**HEADALIGN=INTERNAL** creates a separate HTML table. Columns are generated based on the number of items (text and fields) in the heading. Each item is placed in a separate cell. These columns do *not* correspond to those in the HTML table for the body of the report.

```
TYPE=SUBHEAD, HEADALIGN=INTERNAL, $
```

COUNTRY	CAR	MODEL	SALES
This is my subhead			
Country is: ENGLAND Car is:JAGUAR			
Model is: V12XKE AUTO			
ENGLAND	JAGUAR	V12XKE AUTO	0
		XJ12L AUTO	12000
		JENSEN	INTERCEPTOR III
		TRIUMPH	TR7



Country is aligned with Model in the first column of the internal table. The value of <COUNTRY is aligned with the value of <MODEL in the second column.

#### HEADALIGN=INTERNAL with COLSPAN

```
TYPE=SUBHEAD, HEADALIGN=INTERNAL, $
TYPE=SUBHEAD, LINE=1, ITEM=1, COLSPAN=4, JUSTIFY=CENTER, $
```

COUNTRY	CAR	MODEL	SALES
This is my subhead			
Country is:	ENGLAND	Car is:	JAGUAR
Model is:	V12XKE AUTO		
ENGLAND	JAGUAR	V12XKE AUTO	0
		XJ12L AUTO	12000
	JENSEN	INTERCEPTOR III	0
	TRIUMPH	TR7	0

The first line is centered across all 4 columns of the internal table, based on the COLSPAN=4 setting.

**HEADALIGN=BODY** places the heading lines within the cells of the main HTML table. As a result, the columns of the heading correspond to the columns of the main table.

`TYPE=SUBHEAD, HEADALIGN=BODY, $`

COUNTRY	CAR	MODEL	SALES
This is my subhead			
Country is:	ENGLAND	Car is:	JAGUAR
Model is:	V12XKE AUTO		
ENGLAND	JAGUAR	V12XKE AUTO	0
		XJ12L AUTO	12000
	JENSEN	INTERCEPTOR III	0
	TRIUMPH	TR7	0

Country is aligned with Model in the first column of the main (body) HTML table. The value of <COUNTRY is aligned with the value of <MODEL in the second column.

**HEADALIGN=BODY with COLSPAN**

```
TYPE=SUBHEAD, HEADALIGN=BODY, $
TYPE=SUBHEAD, LINE=1, ITEM=1, COLSPAN=4, JUSTIFY=CENTER, $
```

COLSPAN controls the cross-column alignment of the first row of the heading.

COUNTRY	CAR	MODEL	SALES
This is my subhead			
Country is:	ENGLAND	Car is:	JAGUAR
Model is:	V12XKE AUTO		
ENGLAND	JAGUAR	V12XKE AUTO	0
		XJ12L AUTO	12000
	JENSEN	INTERCEPTOR III	0
	TRIUMPH	TR7	0

**Example: Aligning Elements in a Sort Footing With Data Columns**

This request creates an HTML report using HEADALIGN = BODY to align the two elements of the sort footing (TOTAL IS and the value) with each of the two data columns (Product and Ordered Units). JUSTIFY = RIGHT, which applies to the entire sort footing, right justifies each sort footing element under the data column.

```
TABLE FILE GGORDER
PRINT QUANTITY
BY PRODUCT_CODE NOPRINT BY PRODUCT_DESCRIPTION
WHERE ORDER_DATE EQ '01/01/96'
WHERE STORE_CODE EQ 'R1019'
ON PRODUCT_CODE SUBFOOT
"TOTAL IS: <ST.QUANTITY"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE = REPORT, GRID = OFF, $
TYPE = SUBFOOT, HEADALIGN = BODY, JUSTIFY = RIGHT, $
TYPE = SUBFOOT, OBJECT = FIELD, STYLE = BOLD, $
ENDSTYLE
END
```

The partial output is:

<u>Product</u>	<u>Ordered Units</u>
Hazelnut	300
	117
TOTAL IS:	<b>417</b>
French Roast	126
	86
	49
TOTAL IS:	<b>261</b>
Scone	37
	325
TOTAL IS:	<b>362</b>

**Example: Aligning Elements in a Page Heading Using a Separate HTML Table**

This request creates an embedded HTML table for a page heading, within the HTML table that governs alignment in the body of the report. This table has three rows and three columns to accommodate all the heading elements.

In the first line of the heading, a spot marker (<+0>) creates two text elements: the first element is blank, and the second element is Gotham Grinds, Inc. In the output, the second element appears in the second cell of the first row of the embedded table. For related information, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

The second and fourth lines of the heading are blank.

The spot markers in the third line of the heading split it into three text elements: Orders Report, blank, Run on: &DATE. In the output, each element appears in a cell in the third row of the embedded HTML table, in the order specified in the request.

```
TABLE FILE GGORDER
HEADING
  " <+0>Gotham Grinds, Inc."
  " "
  "Orders Report <+0> <+0> Run on: &DATE"
  " "
PRINT ORDER_NUMBER ORDER_DATE STORE_CODE QUANTITY
BY PRODUCT_CODE BY PRODUCT_DESCRIPTION
IF RECORDLIMIT EQ 10
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE = REPORT, GRID = ON, $
TYPE = HEADING, HEADALIGN = INTERNAL, STYLE = BOLD, $
ENDSTYLE
END
```

GRID=ON in the request enables you to see the embedded HTML table for the heading, and the main HTML table for the body of the report.

The output is:

Gotham Grinds, Inc.					
Orders Report			Run on: 06/19/02		
Product Code	Product	Order Number	Order Date	Store Code	Ordered Units
B141	Hazelnut	1	01/01/96	R1019	300
		2	01/01/96	R1019	117
B142	French Roast	3	01/01/96	R1019	126
		4	01/01/96	R1019	86
		5	01/01/96	R1019	49
G100	Mug	6	01/01/96	R1019	289
		7	01/01/96	R1019	231
G104	Thermos	8	01/01/96	R1019	347
G110	Coffee Grinder	9	01/01/96	R1019	265
G121	Coffee Pot	10	01/01/96	R1019	309

Notice that the positioning is maintained when the grid is hidden (off).

### **Gotham Grinds, Inc.**

#### **Orders Report**

**Run on: 04/20/01**

Product <u>Code</u>	<u>Product</u>	Order <u>Number</u>	Order <u>Date</u>	Store <u>Code</u>	Ordered <u>Units</u>
B141	Hazelnut	1	01/01/96	R1019	300
		2	01/01/96	R1019	117
B142	French Roast	3	01/01/96	R1019	126
		4	01/01/96	R1019	86
		5	01/01/96	R1019	49
G100	Mug	6	01/01/96	R1019	289
		7	01/01/96	R1019	231
G104	Thermos	8	01/01/96	R1019	347
G110	Coffee Grinder	9	01/01/96	R1019	265
G121	Coffee Pot	10	01/01/96	R1019	309

**Example: Aligning a Text Field With a Column in a Sort Footing**

This example uses a Master File and the MODIFY procedure created in the example named [Including a Text Field in a Sort Footing](#) on page 1478. Rerun that example and return here to align the text field.

The request uses HEADALIGN=BODY to align the text field lines in a sort footing. With this setting, each element in the footing is aligned with a column in the main HTML table generated for the report: the first element (the text Course Description:) is aligned with the first data column, CATALOG. The embedded field is aligned in a second column. The grid is turned on in this example to make the alignment easier to see.

```
TABLE FILE TXTFLD
BY CATALOG SUBFOOT
"Course Description: <TEXTFLD"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE = REPORT, GRID = ON, $
TYPE = SUBFOOT, HEADALIGN = BODY, $
ENDSTYLE
END
```

The output displays a new value for the text field each time the value of CATALOG changes.

CATALOG	
COURSE 100	
Evening Course	
Course Description:	This course provides the junior programmer with
	the skills needed to code simple reports.
COURSE 200	
Evening Course	
Course Description:	This course provides the advanced programmer with
	techniques helpful in developing complex
	applications.



**Example: Aligning and Styling a Text Field in a Sort Footing**

This example uses a Master File and the MODIFY procedure created in the example named [Including a Text Field in a Sort Footing](#) on page 1478. Rerun that example and return here to align the text field. This request applies boldface type to the second line of a multiple-line sort footing, which includes the text Course Description as well as the text of the field TEXTFLD. Line 1 of the sort footing is the text Evening Course.

```
TABLE FILE TEXTFLD
BY DESCRIPTION AS 'CATALOG' SUBFOOT
"Evening Course"
"Course Description: <TEXTFLD"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE = REPORT, GRID = OFF, $
TYPE = SUBFOOT, HEADALIGN = BODY, $
TYPE = SUBFOOT, LINE = 2, STYLE = BOLD, $
ENDSTYLE
END
```

The output is:

CATALOG

COURSE 100

Evening Course

**Course Description: This course provides the junior programmer with  
the skills needed to code simple reports.**

COURSE 200

Evening Course

**Course Description: This course provides the advanced programmer with  
techniques helpful in developing complex  
applications.**

If the StyleSheet instead identifies the text field as an object for styling

```
TYPE = SUBFOOT, HEADALIGN = BODY, $
TYPE = SUBFOOT, LINE = 2, OBJECT = FIELD, STYLE = BOLD, $
```

then only the text in TEXTFLD is bold.

CATALOG

COURSE 100

Evening Course

Course Description: **This course provides the junior programmer with the skills needed to code simple reports.**

COURSE 200

Evening Course

Course Description: **This course provides the advanced programmer with techniques helpful in developing complex applications.**

## Aligning a Heading or Footing Element Across Columns in an HTML or PDF Report

With HEADALIGN=BODY, each heading or footing element is aligned with a data column in an HTML or PDF report. With HEADALIGN=INTERNAL, each element is continued in a column of an HTML table created and aligned specifically for the report heading or footing. By default, every heading or footing element (ITEM) is placed in the first available column. However, you can position an item to span multiple columns using the COLSPAN attribute. For details about HEADALIGN options, see [Aligning a Heading or Footing Element in an HTML, XLSX, EXL2K, PDF, PTX, or DHTML Report](#) on page 1548.

You must specify the HEADALIGN and COLSPAN attributes in two separate StyleSheet declarations, since HEADALIGN applies to an entire heading or footing, while COLSPAN applies to a specific item in a heading or footing.

### **Syntax:** How to Align a Heading or Footing Element Across Columns in an HTML or PDF Report

*TYPE* = *headfoot*, [*subtype*,] COLSPAN = *n*, \$

where:

*headfoot*

Is the type of heading or footing. Valid values are TABHEADING, TABFOOTING, HEADING, FOOTING, SUBHEAD, and SUBFOOT.

*subtype*

Are additional attributes that identify the report component. These options can be used separately or in combination, depending upon the degree of specificity required to identify an element. Valid values are:

- ❑ **LINE**, which identifies a line by its position in a heading or footing. Identifying individual lines enables you to format each line differently.

If a heading or footing has multiple lines and you apply a StyleSheet declaration that does not specify LINE, the declaration is applied to *all* lines. Blank lines are counted when interpreting the value of LINE.

- ❑ **OBJECT**, which identifies an element in a heading or footing as a text string or field value. Valid values are TEXT or FIELD. TEXT may represent free text or a Dialogue Manager amper (&) variable.

It is not necessary to specify OBJECT=TEXT unless you are styling both text strings and embedded fields in the same heading or footing.

- ❑ **ITEM**, which identifies an item by its position in a line. To divide a heading or footing line into items, you can use the <+0> spot marker. For more information, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

To determine the ITEM for an OBJECT, follow these guidelines:

- ❑ When used with OBJECT=TEXT, count only the text strings from left to right.
- ❑ When used with OBJECT=FIELD, count only values from left to right.
- ❑ When used without OBJECT, count text strings and field values from left to right.

If you apply a StyleSheet declaration that specifies ITEM, the number is counted from the beginning of each line in the heading or footing, not just from the beginning of the first line.

**COLSPAN**

Is an attribute that aligns an item in the width spanned by multiple columns.

*n*

Is the column with which the specified item is aligned.

**Example: Centering a Page Heading Across Three Columns**

In this request, HEADALIGN=INTERNAL creates a three-column embedded HTML table for the heading. The COLSPAN attribute then centers the first line of the heading, Gotham Grinds, Inc., over the report, spanning the three columns in the embedded HTML table.

```
TABLE FILE GGORDER
HEADING
"Gotham Grinds, Inc."
" "
"Orders Report <+0> <+0> Run on: &DATE"
" "

PRINT ORDER_NUMBER ORDER_DATE STORE_CODE QUANTITY
BY PRODUCT_CODE BY PRODUCT_DESCRIPTION
IF RECORDLIMIT EQ 10
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE = REPORT, GRID = OFF, $
TYPE = HEADING, HEADALIGN = INTERNAL, $
TYPE = HEADING, LINE=1, COLSPAN=3, STYLE = BOLD, JUSTIFY-CENTER, $
TYPE = HEADING, LINE=3, ITEM=3, JUSTIFY=RIGHT, $
ENDSTYLE
END
```

The output is:

### Gotham Grinds, Inc.

#### Orders Report

Run on: 06/19/02

<u>Product</u>		<u>Order</u>	<u>Order</u>	<u>Store</u>	<u>Ordered</u>
<u>Code</u>	<u>Product</u>	<u>Number</u>	<u>Date</u>	<u>Code</u>	<u>Units</u>
B141	Hazelnut	1	01/01/96	R1019	300
		2	01/01/96	R1019	117
B142	French Roast	3	01/01/96	R1019	126
		4	01/01/96	R1019	86
		5	01/01/96	R1019	49
G100	Mug	6	01/01/96	R1019	289
		7	01/01/96	R1019	231
G104	Thermos	8	01/01/96	R1019	347
G110	Coffee Grinder	9	01/01/96	R1019	265
G121	Coffee Pot	10	01/01/96	R1019	309

**Example: Aligning a Field Value Across Multiple Columns**

In this request, HEADALIGN=BODY aligns the sort footing in the same HTML table as the body of the report. COLSPAN = 5 positions the first item in the sort footing (the text Total) in the fifth column of the HTML table. The second item in the sort footing (the field <ST.QUANTITY) is positioned in the next available column.

The HEADALIGN attribute is on a separate line from the COLSPAN attribute because it applies to the entire sort footing (and consequently to both items), whereas COLSPAN applies to the single item Total.

```
TABLE FILE GGORDER
PRINT ORDER_NUMBER ORDER_DATE STORE_CODE QUANTITY
BY PRODUCT_CODE BY PRODUCT_DESCRIPTION
WHERE ORDER_DATE EQ '01/01/96'
WHERE STORE_CODE EQ 'R1019'
ON PRODUCT_CODE SUBFOOT
"Total: <ST.QUANTITY"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLE SHEET *
TYPE = REPORT, GRID = OFF, $
TYPE = SUBFOOT, HEADALIGN = BODY, JUSTIFY = RIGHT, STYLE = BOLD, $
TYPE = SUBFOOT, OBJECT = TEXT, COLSPAN = 5, $
ENDSTYLE
END
```

The partial output is:

<u>Product</u>	<u>Order</u>	<u>Order</u>	<u>Store</u>	<u>Ordered</u>	
<u>Code</u>	<u>Product</u>	<u>Number</u>	<u>Date</u>	<u>Code</u>	<u>Units</u>
B141	Hazelnut	1	01/01/96	R1019	300
		2	01/01/96	R1019	117
				<b>Total:</b>	<b>417</b>
B142	French Roast	3	01/01/96	R1019	126
		4	01/01/96	R1019	86
		5	01/01/96	R1019	49
				<b>Total:</b>	<b>261</b>
F101	Scone	11	01/01/96	R1019	37
		12	01/01/96	R1019	325
				<b>Total:</b>	<b>362</b>

**Example: Aligning a Field Value Across Multiple Columns in a PDF Report**

In this request, HEADALIGN=BODY aligns the sort footing in the same grid as the body of the report. COLSPAN=5 positions the first item in the sort footing (the text *Total*) in the fifth column of the report output. The second item in the sort footing (the field <ST.QUANTITY) is positioned in the next available column. The subfooting items are right justified.

The HEADALIGN attribute is on a separate line from the COLSPAN attribute because it applies to the entire sort footing (and consequently to both items), whereas COLSPAN applies only to the text item *Total*.

```
TABLE FILE GGORDER
PRINT ORDER_NUMBER ORDER_DATE STORE_CODE QUANTITY
BY PRODUCT_CODE BY PRODUCT_DESCRIPTION
WHERE ORDER_DATE EQ '01/01/96'
WHERE STORE_CODE EQ 'R1019'
ON PRODUCT_CODE SUBFOOT
"Total:<ST.QUANTITY"
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLESHEET *
TYPE=REPORT, FONT=ARIAL, SQUEEZE=ON,$
TYPE = SUBFOOT, HEADALIGN = BODY, JUSTIFY = RIGHT, STYLE = BOLD, $
TYPE = SUBFOOT, ITEM=1, COLSPAN = 5, $
ENDSTYLE
END
```

The output shows that the first item in the sort footing (the text *Total*) is in the fifth column of the report output. The second item in the sort footing (the field <ST.QUANTITY) is positioned in the next available column.

<u>Product Code</u>	<u>Product</u>	<u>Order Number</u>	<u>Order Date</u>	<u>Store Code</u>	<u>Ordered Units</u>
B141	Hazelnut	1	01/01/96	R1019	300
		2	01/01/96	R1019	117
				<b>Total:</b>	<b>417</b>
B142	French Roast	3	01/01/96	R1019	126
		4	01/01/96	R1019	86
		5	01/01/96	R1019	49
				<b>Total:</b>	<b>261</b>
F101	Scone	11	01/01/96	R1019	37
		12	01/01/96	R1019	325
				<b>Total:</b>	<b>362</b>
F102	Biscotti	13	01/01/96	R1019	399
				<b>Total:</b>	<b>399</b>
F103	Croissant	14	01/01/96	R1019	93
		15	01/01/96	R1019	433
				<b>Total:</b>	<b>526</b>
G100	Mug	6	01/01/96	R1019	289
		7	01/01/96	R1019	231
				<b>Total:</b>	<b>520</b>
G104	Thermos	8	01/01/96	R1019	347
				<b>Total:</b>	<b>347</b>
G110	Coffee Grinder	9	01/01/96	R1019	265
				<b>Total:</b>	<b>265</b>
G121	Coffee Pot	10	01/01/96	R1019	309
				<b>Total:</b>	<b>309</b>



## Aligning Content in a Multi-Line Heading or Footing

The HEADALIGN and COLSPAN syntax described in [Aligning a Heading or Footing Element in an HTML, XLSX, EXL2K, PDF, PPTX, or DHTML Report](#) on page 1548 is specific to HTML reports. This topic describes how you can design reports that are printable across HTML and PDF formats. Using the WIDTH and JUSTIFY syntax in a StyleSheet, you can:

- ❑ Align vertical sets of text or data as columnar units.
- ❑ Combine columnar formatting with line-by-line formatting.
- ❑ Align decimal points when the data displayed has varying numbers of decimal places. See [Aligning Decimals in a Multi-Line Heading or Footing](#) on page 1578.

You can apply WIDTH and JUSTIFY attributes to report headings and footings, page headings and footings, and sort headings and footings, using either mono-space or proportional fonts.

These techniques rely on internal cascading style sheets, which support WebFOCUS StyleSheet attributes that were not previously available for HTML reports. The syntax associated with these techniques resolves the problem of having to format headings differently for HTML reports (using HEADALIGN and COLSPAN) and PDF and PS reports (using POSITION and spot markers).

While the WIDTH and JUSTIFY attributes are particularly useful when you need to format a multi-line heading or footing, or align stacked decimals, you can also use this syntax to position items in an individual heading or footing line.

**Tip:** For a summary of other alignment methods, see [Choosing an Alignment Method for Heading and Footing Elements](#) on page 1546.

### **Syntax:** How to Align Heading Text and Data in Columns

For a multi-line report or page heading or footing, use the syntax:

```
TYPE=headfoot, WRAP=OFF, $
TYPE=headfoot, [LINE=line_#], ITEM=item_#, [OBJECT={TEXT|FIELD}],
  WIDTH=width, [JUSTIFY=option], $
```

For a multi-line sort heading or footing, use the syntax:

```
TYPE=headfoot, WRAP=OFF, $
TYPE={SUBHEAD|SUBFOOT}, [BY=sortfield] [LINE=line_#], ITEM=item_#,
  [OBJECT={TEXT|FIELD}], WIDTH=width, [JUSTIFY=option], $
```

where:

### *headfoot*

Is the type of heading or footing. Valid values are TABHEADING, TABFOOTING, HEADING, FOOTING, SUBHEAD, and SUBFOOT.

### *sortfield*

When TYPE=SUBHEAD or SUBFOOT, you can specify alignment for the sort heading or sort footing associated with a particular sort field. If no sort field is specified, formatting is applied to the sort headings or footings associated with all sort fields.

### LINE

Is an optional entry that identifies a line by its position in a heading or footing. Identifying individual lines enables you to format each one differently.

If a heading or footing has multiple lines and you apply a StyleSheet declaration that does not specify LINE, the declaration is applied to *all* lines. Blank lines are counted when interpreting the value of LINE.

You can use LINE in combination with ITEM.

### ITEM

Is a required entry when you are using WIDTH to control alignment. An item can identify either:

- ☐ A vertical set of text or data that you wish to align as a columnar unit. You must identify each vertical unit as an item.
- ☐ An item's position in a line. You must identify each line element as an item. See [Line and Item Formatting in a Multi-Line Heading or Footing](#) on page 1575 for information about acceptable variations.

You can use either or both approaches for a single heading or footing.

To divide a heading or footing line into items, you can use the <+0> spot marker. See, [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167. The number of items you can identify is limited by the cumulative widths of the items in the heading or footing, within the physical boundaries of the report page.

You can use ITEM in conjunction with OBJECT to refine the identification of an element whose width you want to define. To determine the ITEM for an OBJECT, follow these guidelines:

- ☐ When used with OBJECT=TEXT, count only the text strings from left to right.
- ☐ When used with OBJECT=FIELD, count only values from left to right.
- ☐ When used without OBJECT, count text strings and field values from left to right.

If you apply a StyleSheet declaration that specifies ITEM, the number is counted from the beginning of each line in the heading or footing, not just from the beginning of the first line.

#### OBJECT

Is an optional entry that identifies an element in a heading or footing as a text string or field value. Valid values are TEXT or FIELD. TEXT may represent free text or a Dialogue Manager ampersand (&) variable.

It is not necessary to specify OBJECT=TEXT unless you are styling both text strings and embedded fields in the same heading or footing.

#### width

Is the measurement expressed in units (inches by default), which is required to accommodate the longest text string or field value associated with a numbered item. For details, see [How to Measure for Column Width](#) on page 1578.

#### option

Is the type of justification. Valid values are:

[LEFT](#) which left justifies the heading or footing. LEFT is the default value.

[RIGHT](#) which right justifies the heading or footing.

[CENTER](#) which centers the heading or footing.

#### DECIMAL (*n*)

Is the measurement expressed in units (inches by default), which specifies how far in from the right side of a column to place the decimal point. With this specification, you can locate the decimal point in the same position within a column, regardless of the number of decimal places displayed to its right.

The measurement will be a portion of the width specified for this item. For details, see [How to Measure for Column Width](#) on page 1578.

**Note:** JUSTIFY is not supported with WRAP.

### **Reference:** Line and Item Formatting in a Multi-Line Heading or Footing

Line formatting maximizes your control over the items you identify on each line:

- ☐ You can align and stack the same number of items with uniform widths. For example,

Line 1	Item 1	Item 2	Item 3
Line 2	Item 1	Item 2	Item 3

- ☐ You can also align different numbers of items as long as the items on each line have the same starting point and the same cumulative width.

Line 1			
Line 2			

Do not use HEADALIGN or COLSPAN syntax, which are specific to HTML reports and may conflict with WIDTH and JUSTIFY settings.

For HTML reports, turn WRAP OFF (ON is the default) to ensure proper processing of WIDTH and JUSTIFY.

**Example: Aligning Data and Text in a Multi-Line Heading or Footing**

In the following free-form report, content is defined entirely in the sort heading, where text and data are stacked to support comparison among countries. Each set of data is aligned vertically, to appear as a column. To achieve this affect, each vertical unit is identified as an item: the first column of text is *item 1*, the next column of data is *item 2*, and so on.

Note especially the last column, in which decimal data with different numbers of decimal places is lined up on the decimal point to facilitate reading and comparison.

Country:	ARGENTINA	Exchange Rate:	35.00
Type:	ST. NOTES	Projected Return:	4.200
Holder:	COMM	Balance:	42,167,880.00
Country:	BRAZIL	Exchange Rate:	39.55
Type:	ST. NOTES	Projected Return:	4.200
Holder:	COMM	Balance:	45,493,501.00
Country:	CANADA	Exchange Rate:	49.00
Type:	ST. NOTES	Projected Return:	3.990
Holder:	COMM	Balance:	56,212,634.00
Country:	CZECH. REP	Exchange Rate:	1,192.45
Type:	ST. NOTES	Projected Return:	12.110
Holder:	COMM	Balance:	51,121,201.00

The chart below breaks out the structure of the previous report:

Item1:	Item 2:	Item 3:	Item 4:
Text	Data values	Text	Values with decimal places
Country	ARGENTINA BRAZIL, and so on	Exchange Rate	<i>nn.dd</i>
Type	ST.NOTES	Projected Return	<i>n.ddd</i>
Holder	COMM	Balance	<i>nn,nnn,nnn.dd</i>

For each item, you specify the width of the column and the justification of its content, as illustrated in the following code.

```

DEFINE FILE SHORT
BALANCE/D14.2=BALANCE;
END
TABLE FILE SHORT
BY COUNTRY NOPRINT SUBHEAD
"Country:<COUNTRY Exchange Rate:<EXCHANGE_RATE"
"Type:<TYPE Projected Return:<PROJECTED_RETURN"
"Holder:<HOLDER Balance:<BALANCE"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE SHEET *
TYPE=REPORT, FONT='TIMES', $
TYPE=REPORT, GRID=OFF, $
TYPE=SUBHEAD, ITEM=1, WIDTH=1.00, JUSTIFY=RIGHT, $
TYPE=SUBHEAD, ITEM=2, WIDTH=1.25, JUSTIFY=RIGHT, $
TYPE=SUBHEAD, ITEM=3, WIDTH=1.25, JUSTIFY=RIGHT,$
TYPE=SUBHEAD, ITEM=4, WIDTH=1.5, JUSTIFY=DECIMAL(.6),$
ENDSTYLE
END

```

This procedure produces a three-line sort heading, broken out as four items, each with a measured width and defined justification. The decimal item (4) uses a variation on standard justification to line up the decimal points. For details, see [How to Align Heading Text and Data in Columns](#) on page 1573 and [Aligning Decimals in a Multi-Line Heading or Footing](#) on page 1578.

**Note:** To take advantage of this feature for an HTML report, you must turn on internal cascading style sheets (SET HTMLCSS=ON). This command enables WebFOCUS StyleSheet attributes that were not previously available for HTML reports. This line of code is ignored for a PDF report.

Aligning Decimals in a Multi-Line Heading or Footing

The ability to align heading content in a multi-line heading based on width and justification values has special benefit in reports that contain data with different numbers of decimal places. For example, if a figure is in dollars, it is formatted with a decimal point and two places for zeroes. If in Swiss francs, it is formatted with a decimal place and four zeroes. If in yen, the decimal is at the end with no zeroes. In addition, sometimes the currency or units do not vary, but the number of digits of decimal precision varies.

By aligning the decimal points in a vertical stack, you can more easily read and compare these numbers, as illustrated in the following output:

Floating decimal points		Aligned decimal points	
Bond -----	Face Value -----	Bond -----	Face Value -----
Galosh Ltd.	22375.5784596	Galosh Ltd.	22375.5784596
Mukluk Inc.	1212345.457	Mukluk Inc.	1212345.457
Overshoe Inc.	232.45484	Overshoe Inc.	232.45484

The technique uses a width specification for the item that contains decimals, combined with a variation on standard left/right/center justification to achieve the proper decimal alignment. For the syntax that generates this output, see [How to Align Heading Text and Data in Columns](#) on page 1573.

Procedure: How to Measure for Column Width

Determining the width of a heading or footing item is a three-step process:

1. Identify the maximum number of characters in a text string or field.
2. For a text string, simply count the characters. For a field, refer to the format specification in the Master File or in a command such as a DEFINE.
3. Measure the physical space in units (for example, in inches) that is required to display the number of characters identified in step 1, based on the size of the font you are using. For example, the following value of the COUNTRY field would measure as follows:

Font	Font size	Comparison	Inches
Helvetica	10	England	.5
Times New Roman	10	England	.44

Font	Font size	Comparison	Inches
Courier	10	England	.56

**Tip:** Consider using a consistent set of fonts in your reports to make your measurements reusable.

### ***Procedure:* How to Measure for Decimal Alignment**

After you have determined the width of an item, you can do a related measurement to determine the physical space required to display decimal data with a varying number of digits to the right of the decimal point.

1. Determine the maximum number of decimal places you need to accommodate to the right of the decimal place, plus the decimal point itself.
2. Measure the physical space in units (for example, in inches) that is required to display the number of characters identified in step 1, based on the size of the font you are using.

### **Combining Column and Line Formatting in Headings and Footings**

By combining column and line formatting, you can create complex reports in which different ranges of lines in the same heading or footing have different numbers of aligned columns in different locations.

### ***Example:* Combining Column and Line Formatting to Align Items in a Sort Heading**

This request produces a free-form report in which content is defined in a seven-line sort heading. Text and data is stacked in two groupings:

- ☐ The first grouping identifies the country and region (continent).
- ☐ The second grouping provides financial information for each country/region pair.

Although this is a single sort heading, our goal is to format the information in each grouping a bit differently to provide emphasis and facilitate comparison. The request also demonstrates a coding technique that makes formatting changes easier for the report designer. See the annotations following the code for details.

As you review the sample request, keep in mind that a heading can contain two kinds of items: text and embedded fields. A text item consists of any characters, even a single blank, between embedded fields and/or spot markers. In particular, if you have a single run of text that you want to treat as two items, you can separate the two items using a <+0> spot marker. For example, in the heading line:

```
" <+0>Country:<COUNTRY"
```

item #1 is a single blank space.

item #2, separated by the <+0> spot marker, is the text Country:

item #3 is the embedded field <COUNTRY.

For details about the <+0> spot marker, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

#### Request and annotations:

```

DEFINE FILE SHORT
BALANCE/D14.2=BALANCE;
END
TABLE FILE SHORT
BY COUNTRY NOPRINT SUBHEAD
1. " <+0>Country:<COUNTRY"
2. " <+0>Region:<REGION"
   " "
3. "Type:<TYPE <+0>Exchange Rate:<EXCHANGE_RATE"
4. "Holder:<HOLDER <+0>Projected Return:<PROJECTED_RETURN"
5. "Risk class:<RISK_CLASS <+0>Balance:<BALANCE"
   " "
   ON TABLE SET PAGE-NUM OFF
6. ON TABLE SET HTMLCSS ON
   ON TABLE SET STYLE SHEET *
   TYPE=REPORT, FONT='TIMES', $
   TYPE=REPORT, GRID=OFF, $
   -* Bottom section of subhead:
7. TYPE=SUBHEAD, ITEM=1, WIDTH=1.00, JUSTIFY=RIGHT, $
8. TYPE=SUBHEAD, ITEM=2, WIDTH=1.25, JUSTIFY=RIGHT, $
9. TYPE=SUBHEAD, ITEM=3, WIDTH=.5, $
10. TYPE=SUBHEAD, ITEM=4, WIDTH=1.25, JUSTIFY=RIGHT, $
11. TYPE=SUBHEAD, ITEM=5, WIDTH=1.5, JUSTIFY=DECIMAL(.6), $
   -* Top section of subhead (overrides above ITEM defaults
   -*   for lines 1 and 2):
12. -SET &INDENT=1.5;
13. TYPE=SUBHEAD, LINE=1, ITEM=1, WIDTH=&INDENT, $
14. TYPE=SUBHEAD, LINE=1, ITEM=2, WIDTH=1, JUSTIFY=LEFT, $
15. TYPE=SUBHEAD, LINE=1, ITEM=3, SIZE=14, WIDTH=2, JUSTIFY=LEFT, $
16. TYPE=SUBHEAD, LINE=2, ITEM=1, WIDTH=&INDENT, $
17. TYPE=SUBHEAD, LINE=2, ITEM=2, WIDTH=1, JUSTIFY=LEFT, $
18. TYPE=SUBHEAD, LINE=2, ITEM=3, WIDTH=2, JUSTIFY=LEFT, $
ENDSTYLE
END

```



The output highlights the key information and its relationship by aligning text and data, including decimal data in which decimal points are aligned for easy comparison.

	Country:	<b>ARGENTINA</b>	
	Region:	SOUTH AMERICA	
Type:	ST. NOTES	Exchange Rate:	35.00
Holder:	COMM	Projected Return:	4.200
Risk class:	Medium	Balance:	42,167,880.00
	Country:	<b>BRAZIL</b>	
	Region:	SOUTH AMERICA	
Type:	ST. NOTES	Exchange Rate:	39.55
Holder:	COMM	Projected Return:	4.200
Risk class:	Medium	Balance:	45,493,501.00
	Country:	<b>CZECH. REP</b>	
	Region:	EASTERN EUROPE	
Type:	ST. NOTES	Exchange Rate:	1,192.45
Holder:	COMM	Projected Return:	12.110
Risk class:	High	Balance:	51,121,201.00
	Country:	<b>ENGLAND</b>	
	Region:	WESTERN EUROPE	
Type:	ST. NOTES	Exchange Rate:	21.35
Holder:	COMM	Projected Return:	4.060
Risk class:	Medium	Balance:	125,757,198.00

Line #	Description
<b>1-2</b>	Defines the content for the <i>top</i> , two-line section of the sort heading. Each line contains three items: the first is a blank area (denoted by a space, separated from the next item by a <+0> spot marker), the second contains text, the third contains data values related to the text.

Line #	Description
3-5	Defines the content for the <i>bottom</i> , three-line section of the sort heading. Each line contains five items: text, data values related to the text, a blank column (denoted by a space, separated from the next item by a null spot marker), text, data values related to the text.
6	Turns on internal cascading style sheets, a requirement for these formatting options. This command enables WebFOCUS StyleSheet attributes that were not previously available for HTML reports. This line of code is ignored for a PDF report.

Line #	Description
7-11	<p>Specifies the basic formatting characteristics for the sort heading by breaking the content into five columns, each identified as an item with a defined width, and justification information for all but the empty column.</p> <p><b>Important:</b> Had additional formatting code (annotated as 12-17) not been included in the request, the specifications annotated as 7-11 would have applied to the entire sort heading (that is, the formatting of the three columns in the top section of the heading would have been based on the specifications for the first three columns described below). However, that is not the effect we want to achieve, so a second section of StyleSheet code is defined to override this formatting for lines 1 and 2 of the sort heading. See annotations 12-18.</p> <p>The formatting of the <i>bottom</i>, three-line section of the heading is controlled by the following specifications:</p> <p>Item 1 identifies a columnar unit that contains text (that is, Type, Holder, Risk Class). It has a defined width of 1 inch and the text is right justified.</p> <p>Item 2 identifies a columnar unit that contains data values related to the text in item 1. It has a defined width of 1.25 inches and the data is right justified.</p> <p>Item 3 identifies a columnar unit that contains blank space and serves as a separator between columns. It has a width of .5 inches. Justification is not relevant.</p> <p>Item 4 identifies a columnar unit that contains text (e.g., Exchange Rate, Projected Return, Balance). It has a defined width of 1.25 inches and the text is right justified.</p> <p>Item 5 identifies a columnar unit that contains a decimal value. The width of the column that contains the value is 1.5 inches, with the decimal point anchored .6 inches in from the right edge of that column.</p> <p>The common width and justification definitions enforce the proper alignment of each item.</p>

Line #	Description
12	<p>Defines a variable called &amp;INDENT, with a width setting of 1.5 inches. This variable defines the width of the blank area (item 1) at the beginning of lines 1 and 2 of the sort heading.</p> <p>Defining the width as a variable enables you to experiment with different widths simply by changing the value in one location. For a complex report, this technique can potentially save a lot of development time. For details, see the documentation on Dialogue Manager in the <i>Developing Reporting Applications</i> manual.</p>
13-18	<p>Specifies line-by-line formatting for the <i>top</i>, two-line section of the sort heading. This code overrides the previous formatting for lines 1 and 2 of the sort heading because it specifies a line number.</p> <p>Item 1 on each line refers to the blank area. The width is defined as a variable and implemented based on the current value of &amp;INDENT.</p> <p>Item 2 on each line refers to the text area. It has a defined width of 1 inch and the text is left justified.</p> <p>Item 3 on each line refers to the data values. It has a defined width of 2 inches and the data is left justified.</p> <p>The common width and justification definitions enforce the proper alignment of each item.</p> <p>Notice that item 1 in line 15 defines a font size for the data values associated with the COUNTRY field. All other items on both lines use a default font. Line-by-line formatting enables you to define a unique characteristic for a single item.</p>

## Positioning Headings, Footings, or Items Within Them

For a PDF, PS, or HTML report, you can use the POSITION attribute in a StyleSheet to specify a starting position for a heading or footing, expressed as a unit measurement. For HTML, this capability requires an internal cascading style sheet. For details on selecting an alignment method, see [Choosing an Alignment Method for Heading and Footing Elements](#) on page 1546.

In addition, for a PDF or PS report, you can use the POSITION attribute to specify an absolute or relative starting position for an element within a heading or footing or to align an item in a heading or footing with a report column. An absolute starting position is the distance from the left margin of the report. A relative starting position is the distance from the preceding object. For the first item on a heading line this is the left margin of the report.

In an HTML report, you can use related syntax and an internal cascading style sheet to position an image in a heading or footing. For details on images, see [Laying Out the Report Page](#) on page 1249.

### **Syntax:** How to Set a Starting Position for a Heading or Footing

Use the following syntax to specify a starting position for an entire heading or footing in relation to the left margin of a report.

```
TYPE = headfoot, POSITION = position, $
```

where:

*headfoot*

Is the type of heading or footing. Valid values are TABHEADING, TABFOOTING, HEADING, FOOTING, SUBHEAD, and SUBFOOT.

*position*

Is the desired distance from the left, expressed by the UNITS attribute (the default is INCHES).

**Note:** In an HTML report, this syntax must be used in conjunction with an internal cascading style sheet.

### **Example:** Setting a Starting Position for a Report Heading in PDF

This request positions the report heading 1.25 inches from the left margin.

```
SET ONLINE-FMT=PDF
TABLE FILE GGSales
PRINT BUDDOLLARS DOLLARS
BY STCD
WHERE BUDDOLLARS GE 25000
WHERE STCD EQ 'R1019'
ON TABLE SUBHEAD
"Sales Report"
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE SET SQUEEZE ON
ON TABLE SET STYLESHEET *
TYPE = TABHEADING, POSITION = 1.25, $
ENDSTYLE
END
```

The output is:

Sales Report		
<u>Store ID</u>	<u>Budget Dollars</u>	<u>Dollar Sales</u>
R1019	28965	21905
	25350	19305
	26054	17360
	27210	17380

**Example:** Setting a Starting Position for a Report Heading in HTML

The request generates an internal cascading style sheet as part of its HTML code, enabling the use of the POSITION attribute to specify a starting position for the heading, Sales Report, 1.5 inches from the left margin.

```
SET ONLINE-FMT = HTML
TABLE FILE GGSales
PRINT BUDDOLLARS DOLLARS
BY STCD
WHERE BUDDOLLARS GE 25000
WHERE STCD EQ 'R1019'
ON TABLE SUBHEAD
"Sales Report"
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
TYPE = TABHEADING, POSITION = 1.5, $
ENDSTYLE
END
```

The output is:

Sales Report		
<u>Store ID</u>	<u>Budget Dollars</u>	<u>Dollar Sales</u>
R1019	28965	21905
	25350	19305
	26054	17360
	27210	17380

**Syntax:**      **How to Set a Starting Position for a Heading or Footing Element**

For a PDF or PS report, use the following syntax to specify a starting position for a heading or footing element in relation to the preceding item

```
TYPE = headfoot, [subtype,] POSITION = {+|-}option, $
```

where:

*headfoot*

Is the type of heading or footing. Valid values are TABHEADING, TABFOOTING, HEADING, FOOTING, SUBHEAD, and SUBFOOT.

*subtype*

Are additional attributes that identify the report component. These options can be used separately or in combination, depending upon the degree of specificity you need to fully identify an element. Valid values are:

- ☐ **LINE**, which identifies a line by its position in a heading or footing. Identifying individual lines enables you to format each line differently.

If a heading or footing has multiple lines and you apply a StyleSheet declaration that does not specify LINE, the declaration is applied to *all* lines. Blank lines are counted when interpreting the value of LINE.

- ☐ **ITEM**, which identifies an item by its position in a line. To divide a heading or footing line into items, you can use the <+0> spot marker. For details, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

To determine an ITEM number for an OBJECT, follow these guidelines:

- ☐ When used with OBJECT=TEXT, count only the text strings from left to right.
- ☐ When used with OBJECT=FIELD, count only values from left to right.
- ☐ When used without OBJECT, count text strings and field values from left to right.

If you apply a StyleSheet declaration that specifies ITEM, the number is counted from the beginning of each line in the heading or footing, not just from the beginning of the first line.

- ☐ **OBJECT**, which identifies an element in a heading or footing as a text string or field value. Valid values are TEXT or FIELD. TEXT may represent free text or a Dialogue Manager amper (&) variable.

It is not necessary to specify OBJECT=TEXT unless you are styling both text strings and embedded fields in the same heading or footing.

### *option*

Is the alignment method. Valid values are:

- ❑ **position**, which is the desired distance, expressed by the UNITS attribute (the default is inches) for absolute positioning.
- ❑ **+**, which starts the heading or footing element at the specified distance to the *right* of the preceding item. For the first item in a heading or footing, the preceding item is the left margin of the report.
- ❑ **-**, which starts the heading or footing element at the specified distance to the *left* of the preceding item. This is useful if you want to overlap images in a heading.
- ❑ **column\_title**, which aligns the heading or footing element with the first character of the designated column.

### **Example:** Setting an Absolute Starting Position for a Heading Item

This request uses the spot marker <+0> to divide the report heading into three text strings. It starts the third text string, 1st Qtr 2001, 3 inches from the left report margin. This technique can be used in PDF as well as PS reports.

```
SET ONLINE-FMT = PDF
TABLE FILE GGSales
SUM UNITS DOLLARS BY CATEGORY BY PRODUCT
ON TABLE SUBHEAD
"Sales Report - <+0>All Products<+0> 1st Qtr 2001"
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE SET SQUEEZE ON
ON TABLE SET STYLESHEET *
TYPE = TABHEADING, OBJECT = TEXT, ITEM=1, SIZE = 12, STYLE = BOLD, $
TYPE = TABHEADING, OBJECT = TEXT, ITEM=2, STYLE = BOLD, $
TYPE = TABHEADING, OBJECT = TEXT, ITEM=3, POSITION = 3, $
ENDSTYLE
END
```



The output is:

Sales Report - All Products		1st Qtr 2001	
Category	Product	Unit Sales	Dollar Sales
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

**Example: Setting a Relative Starting Position for a Heading Item**

This request uses the spot marker <+0> to divide the report heading into three text strings. It starts the third text string, 1st Qtr 2001, one inch to the right of the previous item on the heading line. Inches is the default unit of measure. This technique can be used in PDF as well as PS reports.

```

SET ONLINE-FMT = PDF
TABLE FILE GGSales
SUM UNITS DOLLARS BY CATEGORY BY PRODUCT
ON TABLE SUBHEAD
"Sales Report - <+0>All Products<+0> 1st Qtr 2001"
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE SET SQUEEZE ON
ON TABLE SET STYLESHEET *
TYPE = TABHEADING, OBJECT = TEXT, ITEM=1, SIZE = 12, STYLE = BOLD, $
TYPE = TABHEADING, OBJECT = TEXT, ITEM=2, STYLE = BOLD, $
TYPE = TABHEADING, OBJECT = TEXT, ITEM=3, POSITION = +1, $
ENDSTYLE
END

```

The output is:

Sales Report - All Products			1st Qtr 2001
Category	Product	Unit Sales	Dollar Sales
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Boone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

**Example:** Aligning a Heading Item With a Column

This request uses the spot marker <+0> to divide the report heading into three text strings. It starts the second text string at the horizontal position where the column UNITS (Unit Sales) is. This technique can be used in PDF as well as PS reports.

```

SET ONLINE-FMT = PDF
TABLE FILE GGSales
SUM UNITS DOLLARS BY CATEGORY BY PRODUCT
ON TABLE SUBHEAD
"Sales Report - <+0>All Products<+0> 1st Qtr 2001"
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE SET SQUEEZE ON
ON TABLE SET STYLESHEET *
TYPE = TABHEADING, LINE=1, ITEM=2, POSITION=UNITS, $
ENDSTYLE
END

```

The output is:

Sales Report -		All Products 1st Qtr 2001	
<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

**Tip:** In this request the column (UNITS) is identified by name. However, there are other ways to identify a column that you wish to format. See [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

## Using PRINTPLUS

PRINTPLUS includes enhancements to display alternatives offered by WebFOCUS. For example, you can place a FOOTING after a SUBFOOT in your report. PRINTPLUS provides the flexibility to produce the exact report you desire.

The PRINTPLUS parameter must be set to ON to use the following TABLE capabilities:

- ☐ PAGE-BREAK is handled internally to provide the correct spacing of pages. For example, if a new report page is started and an instruction to skip a line at the top of the new page is encountered, WebFOCUS knows to suppress the blank line and start at the top of the page.
- ☐ NOSPLIT is handled internally. (Use NOSPLIT to force a break at a specific spot.)
- ☐ You can perform RECAPs in cases where pre-specified conditions are met.
- ☐ A Report SUBFOOT now prints above the footing instead of below it.
- ☐ Data displays correctly in subfoots when IF/WHERE TOTAL or BY HIGHEST is used.
- ☐ BY field actions are linked with BY field options so they appear on the same page. The footing no longer splits on two pages.
- ☐ Footings and Subfoots always appear on a page with at least one data item, and will never split between two pages.

- ❑ Printing beyond the length of the page no longer occurs.
- ❑ Splitting of fields linked by OVER onto separate pages no longer occurs.
- ❑ There is no reserved space for conditional output. The output page is fully used.
- ❑ The order of sort fields is no longer relevant.

**Note:** PRINTPLUS is not supported for StyleSheets. A warning message is generated in this case.

**Syntax:**      **How to Use PRINTPLUS**

Issue the command

```
SET PRINTPLUS = {ON|OFF}
```

**Example:**      **Using PRINTPLUS With SUBFOOT and FOOTING**

With PRINTPLUS on, the SUBFOOT prints first, followed by the FOOTING.

```
SET PRINTPLUS = ON
TABLE FILE CAR
  PRINT CAR MODEL
  BY SEATS BY COUNTRY
  IF COUNTRY EQ ENGLAND OR FRANCE OR ITALY
  ON TABLE SUBFOOT
  " "
  " SUMMARY OF CARS IN COUNTRY BY SEATING CAPACITY"
  FOOTING
  " RELPMEK CAR SURVEY "
ON TABLE SET STYLE *
TYPE=REPORT,GRID=OFF,$
ENDSTYLE
END
```

The output is:

SEATS	COUNTRY	CAR	MODEL
-----	-----	---	-----
2	ENGLAND	TRIUMPH	TR7
	ITALY	ALFA ROMEO	2000 GT VELOCE
		ALFA ROMEO	2000 SPIDER VELOCE
		MASERATI	DORA 2 DOOR
4	ENGLAND	JAGUAR	V12XKE AUTO
		JENSEN	INTERCEPTOR III

	ITALY	ALFA ROMEO	2000 4 DOOR BERLINA
5	ENGLAND	JAGUAR	XJ12L AUTO
	FRANCE	PEUGEOT	504 4 DOOR

SUMMARY OF CARS IN COUNTRY BY SEATING CAPACITY  
RELPMEX CAR SURVEY

## Using Spot Markers to Refine Positioning

You can employ several types of spot markers to refine the positioning of headings and footings, and elements within them, in HTML and PDF reports that use proportional fonts. For maximum control, you can combine spot markers with other alignment techniques. See [Choosing an Alignment Method for Heading and Footing Elements](#) on page 1546.

The following spot markers enable you to position items and to identify items to be formatted:

- ❑ **<+0>** divides a heading or footing into items for formatting.

To divide a heading or footing into items that can be formatted separately, place the **<+0>** spot marker after the text string or field you wish to specify. It will not add any additional spaces to your heading or footing. For details, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

- ❑ **</n** specifies skipped lines.

To specify skipped lines in a heading or footing, place the **</n>** spot marker on the same line as the text in the request. If you place it on a line by itself, WebFOCUS counts the line the spot marker is on plus the number of skip-lines you designate. For details, see [Controlling the Vertical Positioning of a Heading or Footing](#) on page 1598.

- ❑ **<-n** controls the positioning of a character immediately following a field.

**Note:** When a closing spot marker is immediately followed by an opening spot marker (**><**), a single space text item will be placed between the two spot markers (**> <**). This must be considered when applying formatting.

You can also use spot markers to position heading and footing elements at fixed and relative column locations. Several spot markers control positioning based on the pre-defined width of a character in a monospace font. This is a legacy formatting technique that is not supported for proportional fonts.

**Example: Positioning a Character Immediately After a Field in an HTML Report**

This request generates an HTML report in which the closing parenthesis and the period in the sort heading follow immediately after the STORE\_CODE and STATE fields, respectively. This behavior is controlled by the <-1 spot markers, which indicate a relative starting position from the preceding object. Without these spot markers to indicate that the punctuation characters should follow the preceding objects, an extra space would appear in each of these positions in the display.

```
SET ONLINE-FMT = HTML
SET PAGE-NUM = OFF
JOIN STORE_CODE IN CENTCOMP TO STORE_CODE IN CENTORD

TABLE FILE CENTCOMP
HEADING
"Century Corporation Orders Report </1"
PRINT PROD_NUM QUANTITY LINEPRICE
BY STORE_CODE NOPRINT
BY ORDER_NUM
ON STORE_CODE SUBHEAD
"Century Corporation orders for store <STORENAME <0X
(store # <STORE_CODE<-1 ) in <STATE|. </1"
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The partial output is:

Century Corporation Orders Report

<u>Order Number:</u>	<u>Product Number#:</u>	<u>Quantity:</u>	<u>Line Total</u>
Century Corporation orders for store Audio Expert (Store # 1003CA) in CA.			
74610	1012	268	\$222,234.35
	1028	323	\$29,215.62
	1032	339	\$27,208.12
	1034	339	\$154,943.38
	1036	339	\$93,266.39
Century Corporation orders for store Audio Expert (Store # 1003CO) in CO.			
39274	1006	179	\$54,201.50
	1008	179	\$29,034.16
	1020	179	\$45,457.59
	1032	400	\$30,861.24
	1034	400	\$143,744.49

**Example: Positioning a Character Immediately After a Field in a PDF or PS Report**

In a PDF report, an embedded field in a heading or footing must be followed by a space in the request to be recognized for processing. However, in the output the space may not be desirable. This example demonstrates two techniques for positioning punctuation characters immediately after a field in a PDF report.

The first technique uses the POSITION attribute in a StyleSheet to position the closing parenthesis immediately after the STORE\_CODE value. The second technique uses the <-1 spot marker to position the period immediately after the STATE value. The POSITION measurement is based on the UNITS designation POINTS. Experimentation demonstrated that -7 points moves the closing parenthesis to the proper location after the field, using the default proportional font and size.

```
SET ONLINE-FMT = PDF
SET PAGE-NUM = OFF
JOIN STORE_CODE IN CENTCOMP TO STORE_CODE IN CENTORD

TABLE FILE CENTCOMP
HEADING
"Century Corporation Orders Report"
PRINT PROD_NUM QUANTITY LINEPRICE
BY STORE_CODE NOPRINT
BY ORDER_NUM
ON STORE_CODE SUBHEAD
"Century Corporation orders for store <STORENAME (store # <STORE_CODE )
in <0X <STATE <-1 . </1"
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, UNITS=POINTS, $
TYPE=SUBHEAD, OBJECT=TEXT, ITEM=3, POSITION= -7, $
ENDSTYLE
END
```

The output is:

Century Corporation Orders Report			
<u>Order Number:</u>	<u>Product Number#:</u>	<u>Quantity:</u>	<u>Line Total</u>
Century Corporation orders for store Audio Expert (store # 1003CA) in CA.			
74610	1012	268	\$222,234.35
	1028	323	\$29,215.62
	1032	339	\$27,208.12
	1034	339	\$154,943.38
	1036	339	\$93,266.39
Century Corporation orders for store Audio Expert (store # 1003CO) in CO.			
39274	1006	179	\$54,201.50
	1008	179	\$29,034.16
	1020	179	\$45,457.59
	1032	400	\$30,861.24
	1034	400	\$143,744.49
Century Corporation orders for store Audio Expert (store # 1003CT) in CT.			



Without the spot marker and position measurement, the output would have looked like this:

Century Corporation Orders Report			
Order Number:	Product Number#:	Quantity:	Line Total
Century Corporation orders for store Audio Expert (store # 1003CA) in CA .			
74610	1012	268	\$222,234.35
	1028	323	\$29,215.62
	1032	339	\$27,208.12
	1034	339	\$154,943.38
	1036	339	\$93,266.39
Century Corporation orders for store Audio Expert (store # 1003CO) in CO .			
39274	1006	179	\$54,201.50
	1008	179	\$29,034.16
	1020	179	\$45,457.59
	1032	400	\$30,861.24
	1034	400	\$143,744.49
Century Corporation orders for store Audio Expert (store # 1003CT) in CT .			

### **Example:** Customizing Position Measurements for Font Attributes

This request uses a 12-point Helvetica font. Experimentation demonstrated that the POSITION value of -2 moves the text in this font and size to the required position.

```
SET ONLINE-FMT = PDF
SET PAGE-NUM = OFF
JOIN STORE_CODE IN CENTCOMP TO STORE_CODE IN CENTORD

TABLE FILE CENTCOMP
HEADING
"CENTURY CORPORATION ORDERS REPORT"
PRINT PROD_NUM QUANTITY LINEPRICE
BY STORE_CODE NOPRINT WHERE STORE_CODE EQ '1003NY' OR '1003CT' OR
'1003NJ'
BY ORDER_NUM
ON STORE_CODE SUBHEAD
"CENTURY CORPORATION ORDERS FOR STORE <STORENAME (store # <STORE_CODE)>0X
IN <STATE <-1 . </1"
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, UNITS=POINTS, FONT='Helvetica', SIZE=12, $
TYPE=SUBHEAD, OBJECT=TEXT, ITEM=3, POSITION= -2, $
ENDSTYLE
END
```

**Tip:** You cannot use the POSITION attribute to position a heading element in an HTML report. However, you can achieve the same result by placing the horizontal spot markers <-1 immediately after the fields STORE\_CODE and STATE. Do not add a space between the field and the character that will follow it.

The output is:

CENTURY CORPORATION ORDERS REPORT			
Order Number:	Product Number#:	Quantity:	Line Total
CENTURY CORPORATION ORDERS FOR STORE Audio Expert (store # 1003CT) IN CT			
35995	1008	13	\$2,132.45
	1020	13	\$3,895.38
	1022	146	\$52,159.13
	1024	146	\$45,306.84
	1030	146	\$18,525.49
CENTURY CORPORATION ORDERS FOR STORE Audio Expert (store # 1003NJ) IN NJ			
35514	1004	279	\$35,050.34
	1008	118	\$17,200.98
	1012	146	\$109,416.14
	1020	118	\$31,570.60
	1022	146	\$44,677.60
	1024	146	\$44,749.87
	1030	251	\$32,518.53
CENTURY CORPORATION ORDERS FOR STORE Audio Expert (store # 1003NY) IN NY			
48883	1008	12	\$1,836.34
	1020	12	\$2,883.95
	1022	157	\$57,051.68
	1024	157	\$41,985.00
	1030	157	\$20,189.56

Controlling the Vertical Positioning of a Heading or Footing

You can use several vertical positioning techniques to enhance the appearance and readability of a report:

- ❑ In a report generated in HTML, PDF, or other report formats, you can add one or more lines above or below heading or footing text using spot markers and free text lines. See [How to Add Blank Lines to a Heading or Footing](#) on page 1599.
- ❑ In a PDF report you can control the space above or below a heading or footing line or the distance between text and the surrounding grid lines in a heading or footing line using the TOPGAP and BOTTOMGAP attributes in a StyleSheet. See [How to Control Vertical Spacing in a Heading or Footing](#) on page 1600. For full information on TOPGAP and BOTTOMGAP, see [Laying Out the Report Page](#) on page 1249.
- ❑ In a PDF report, you can position a page footing at the bottom of a page, rather than directly after the report data. See [How to Position a Page Footing at the Bottom of a Page](#) on page 1603.

**Syntax:**      **How to Add Blank Lines to a Heading or Footing**

Use the following syntax options to add blank lines above or below, or within a heading or footing, where:

`</n`

Is a spot marker that specifies the number of lines to skip. It is best to put the spot marker on the same line as the text in the request. If you place the spot marker `</n` on a line by itself, it will add that line in addition to the designated number of skipped lines.

`" "`

Indicates a separate line in the heading or footing, with blank content.

You can use these techniques separately or in combination.

**Example:**      **Adding Blank Lines Above and Below a Report Heading**

This request creates an HTML report with one blank line between each line of the page heading and two blank lines between the page heading and the actual report. The first blank line is added as an empty text line, The next blank lines are added with the skip-line spot marker.

```
TABLE FILE GGSales
SUM BUdunits UNITS BUdDOLLARS DOLLARS
BY CATEGORY
ON TABLE SUBHEAD
"SALES REPORT"
" "
*** (CONFIDENTIAL) **</1"
"December 2002 </2"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, TOPMARGIN=0, $
TYPE = TABHEADING, JUSTIFY = CENTER, $
ENDSTYLE
END
```

The output is:

SALES REPORT  
\*\*(CONFIDENTIAL)\*\*  
DECEMBER 2002

<u>Category</u>	<u>Budget Units</u>	<u>Unit Sales</u>	<u>Budget Dollars</u>	<u>Dollar Sales</u>
Coffee	1385923	1376266	17293907	17231465
Food	1377564	1384845	17267160	17229344
Gifts	931007	927880	11659732	11695502

**Syntax:**      **How to Control Vertical Spacing in a Heading or Footing**

In a PDF report, you can use the TOPGAP and BOTTOMGAP attributes to control spacing above or below a heading or footing line or the distance between heading or footing text and the grid lines above and below them.

**Note:** You can use TOPGAP and BOTTOMGAP with multi-line headings. Keep in mind that between heading lines the top *and* bottom gap will be inserted, making the spacing between lines greater than the spacing at the top and bottom of the heading.

`TYPE=headfoot, {TOPGAP|BOTTOMGAP}=gap, $`

where:

*headfoot*

Is the type of heading or footing. Valid values are TABHEADING, TABFOOTING, HEADING, FOOTING, SUBHEAD, and SUBFOOT.

*TOPGAP*

Indicates how much space to add above a report component.

*BOTTOMGAP*

Indicates how much space to add below a report component.

*gap*

Is the amount of blank space, in the unit of measurement specified by the UNITS parameter (inches, by default).

In the absence of grids, the default value is 0.

In the presence of grids, the default value increases to provide space between the grid and the text.

**Example: Adding Blank Space to Separate Heading Text From Grid Lines in a PDF Report**

This request generates a PDF report with blank space added above and below the report heading to separate the text from the upper and lower grid lines. The space above is added by the TOPGAP attribute. The space below is added by the BOTTOMGAP attribute.

```
TABLE FILE GGSALES
SUM BUDUNITS UNITS BUDDOLLARS DOLLARS
BY CATEGORY
ON TABLE SUBHEAD
"SALES REPORT <+0>December 2001"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT PDF
ON TABLE SET SQUEEZE ON
ON TABLE SET STYLESHEET *
TYPE = TABHEADING, GRID=ON, JUSTIFY=CENTER, TOPGAP=.25, BOTTOMGAP=.25, $
TYPE = TABHEADING, FONT='TIMES', SIZE=12, STYLE=BOLD, $
TYPE = TABHEADING, ITEM=2, SIZE=10, STYLE=ITALIC, $
ENDSTYLE
END
```

The output is:

SALES REPORT <i>December 2001</i>				
Category	Budget Units	Unit Sales	Budget Dollars	Dollar Sales
Coffee	1385923	1376266	17293886	17231455
Food	1377564	1384845	17267160	17229333
Gifts	931007	927880	11659732	11695502

**Example: Adjusting Vertical Spacing Below a Sort Footing**

The request generates a PDF report in which the sort footings are bolded for emphasis and space is added below each footing to visually tie the footing text to the preceding data.

```
TABLE FILE GGPRODS
PRINT PACKAGE_TYPE AND UNIT_PRICE
WHERE UNIT_PRICE GT 50
BY PRODUCT_DESCRIPTION NOPRINT BY PRODUCT_ID
ON PRODUCT_DESCRIPTION SUBFOOT
"Summary for <PRODUCT_DESCRIPTION>"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT PDF
ON TABLE SET SQUEEZE ON
ON TABLE SET STYLESHEET *
TYPE=SUBFOOT, STYLE=BOLD, BOTTOMGAP=.25, $
ENDSTYLE
END
```

The output is:

<u>Product Code</u>	<u>Package</u>	<u>Unit Price</u>
G110	Case	125.00
<b>Summary for Coffee Grinder</b>		
G121	Case	140.00
<b>Summary for Coffee Pot</b>		
B142	Pounds	81.00
<b>Summary for French Roast</b>		
B141	Pounds	58.00
<b>Summary for Hazelnut</b>		
B144	Pounds	76.00
<b>Summary for Kona</b>		

**Syntax:**      **How to Position a Page Footing at the Bottom of a Page**

You can position a page footing at the bottom of a page. By default, a page footing appears two lines below the report data.

```
FOOTING [BOTTOM]
  "content ..."
["content ..." ]
.
.
.
["content ..." ]
```

where:

**FOOTING**

Is the required command that identifies the content as a page footing.

**BOTTOM**

Is an optional command that places the footing at the bottom of the page. If you omit BOTTOM, the page footing appears two lines below the report data. **Note:** FOOTING BOTTOM is not supported in an HTML report or by the WebFOCUS Viewer.

*content*

Footing content can include the following elements, between double quotation marks. (If the ending quotation mark is omitted, all subsequent lines of the request are treated as part of the footing.)

*text*

Is the footing text. You can include multiple lines of text.

The text must start on a line by itself, following the FOOTING command.

Text can be combined with variables and spot markers.

For related information, see [Limits for Headings and Footings](#) on page 1433.

*variable*

Can be any one or a combination of the following:

**Fields** (real data source fields, virtual fields created with the DEFINE command in a Master File or report request, calculated values created with the COMPUTE command in a request, system fields such as TABPAGENO). You can qualify data source fields with certain prefix operators.

**Dialogue Manager variables.**

**Images.** You can include images in a heading or footing.

For details, see [Including an Element in a Heading or Footing](#) on page 1469.

### *spot marker*

Enables you to position items, to identify items to be formatted, and to extend code beyond the 80-character line limit of the text editor.

<+0> divides a heading or footing into items for formatting. For details, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

</n specifies skipped lines. For details, see [Controlling the Vertical Positioning of a Heading or Footing](#) on page 1598.

<-n to position the next character on the line. For details, see [Using Spot Markers to Refine Positioning](#) on page 1593.

<0x continues a heading or footing specification on the next line of the request. For details, see [Extending Heading and Footing Code to Multiple Lines in a Report Request](#) on page 1434.

**Note:** When a closing spot marker is immediately followed by an opening spot marker (><), a single space text item will be placed between the two spot markers (> <). This must be considered when applying formatting.

### *Blank lines*

If you omit all text, variables, and spot markers, you have a blank heading or footing line (for example, " ") which you can use to skip a line in the heading or footing. (You can also skip a line using a vertical spot marker, such as </1.)

**Note:** The maximum number of sort headings and sort footings in one request is 33.

### **Example: Positioning a Page Footing at the Bottom of a Page**

This request produces a PDF report in which the page footing appears at the bottom of the page, rather than in its default position, two lines below the report data.

```
TABLE FILE GGSALES
PRINT UNITS DOLLARS
BY CATEGORY BY STCD
WHERE TOTAL DOLLARS GE 25000
FOOTING BOTTOM
"PRELIMINARY SALES FIGURES"
ON TABLE SET ONLINE-FMT PDF
ON TABLE SET PAGE-NUM OFF
END
```



The following output shows the end of the report, with the footing.

	R1088	1676	25140
		1789	26835
	R1100	1781	26715
		3015	45225
	R1109	1694	25410
	R1200	1765	26475
		1675	25125
		1773	26595
	R1244	1800	25200
	R1248	1798	26970
		1695	25425
		1699	25485
Gifts	R1020	3609	47468
	R1200	3473	51180

#### PRELIMINARY SALES FIGURES

## Placing a Report Heading or Footing on Its Own Page

In a PDF report or an HTML report displayed in the WebFOCUS Viewer, you can request that a report heading or footing appear on its own page to set off important information. For example, you might want to create a cover page that flags salary information as *Confidential* and is separate from the actual data.

### **Syntax:** How to Position a Report Heading or Footing on Its Own Page

Each heading or footing line must begin and end with a double quotation mark.

```
ON TABLE PAGE-BREAK AND {SUBHEAD|SUBFOOT}
"content ... "
["content ... "]
.
.
.
["content ... "]
```

where:

#### PAGE-BREAK

Determines when a new page starts. Use with the SET LINES command to control the length of a printed page.

#### SUBHEAD

Generates a report heading.

### `SUBFOOT`

Generates a report footing.

### *content*

Heading or footing content can include the following elements, between double quotation marks. If the ending quotation mark is omitted, all subsequent lines of the request are treated as part of the report heading.

### *text*

Is the heading or footing text. You can include multiple lines of text.

The text must start on a line by itself, following the SUBHEAD or SUBFOOT command.

Text can be combined with variables and spot markers.

For related information, see [Limits for Headings and Footings](#) on page 1433.

### *variable*

Can be any one or a combination of the following:

`Fields` (real data source fields, virtual fields created with the DEFINE command in a Master File or report request, calculated values created with the COMPUTE command in a request, system fields such as TABPAGENO). You can qualify data source fields with certain prefix operators.

`Dialogue Manager variables`.

`Images`. You can include images in a heading or footing.

For details, see [Including an Element in a Heading or Footing](#) on page 1469.

### *spot marker*

Enables you to position items, to identify items to be formatted, and to extend code beyond the 80-character line limit of the text editor.

`<+0>` divides a heading or footing into items for formatting. For details, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

`</n` specifies skipped lines. For details, see [Controlling the Vertical Positioning of a Heading or Footing](#) on page 1598.

`<-n` positions the next character on the line. For details, see [Using Spot Markers to Refine Positioning](#) on page 1593.

`<0x` continues a heading or footing specification on the next line of the request. For details, see [Extending Heading and Footing Code to Multiple Lines in a Report Request](#) on page 1434.

**Note:** When a closing spot marker is immediately followed by an opening spot marker (><), a single space text item will be placed between the two spot markers (> <). This must be considered when applying formatting.

#### Blank lines

If you omit all text, variables, and spot markers, you have a blank heading or footing line (for example, " ") which you can use to skip a line in the heading or footing. (You can also skip a line using a vertical spot marker, such as </1.)

### **Example:** Positioning a Report Heading on a Separate Page

Using PAGE-BREAK, this request generates a two-page report, with important information by itself on the first page.

```
TABLE FILE CENTORD
SUM ORDER_DATE LINEPRICE AS 'Order,Total:'
BY HIGHEST 5 ORDER_NUM
ON TABLE PAGE-BREAK AND SUBHEAD
"CONFIDENTIAL COMPANY INFORMATION"
"March 2003"
HEADING
"Order Revenue"
" "
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT PDF
END
```

The first page of output identifies the confidential nature of the report and the date.

```
CONFIDENTIAL COMPANY INFORMATION
March 2003
```

The second page of output contains the column titles and data.

#### Order Revenue

<u>Order Number:</u>	<u>Date Of Order:</u>	<u>Order Total:</u>
94710	2001/01/02	\$406,964.24
94680	2001/01/02	\$421,916.60
94670	2001/01/02	\$513,868.76
94550	2001/01/02	\$496,323.64
94530	2001/01/02	\$3,472.41

**Tip:** To produce comparable results in HTML, include the following code in the request to turn on the WebFOCUS Viewer.

```
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET WEBVIEWER ON
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The first page will display the report heading. You can navigate to the second page for the report data.

### **Example:** Positioning a Report Footing on a Separate Page

Using PAGE-BREAK, this request generates a two-page report with the report footing, which signals the end of the report, on a page by itself.

```
TABLE FILE CENTORD
HEADING
"Order Revenue"
" "
SUM ORDER_DATE LINEPRICE AS 'Order,Total:'
BY HIGHEST 5 ORDER_NUM
ON TABLE PAGE-BREAK AND SUBFOOT
"END OF REPORT"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT PDF
END
```

The first page of output contains the column titles and data.

#### Order Revenue

<u>Order Number:</u>	<u>Date Of Order:</u>	<u>Order Total:</u>
94710	2001/01/02	\$406,964.24
94680	2001/01/02	\$421,916.60
94670	2001/01/02	\$513,868.76
94550	2001/01/02	\$496,323.64
94530	2001/01/02	\$3,472.41

The last page of output signals the end of the report.

```
END OF REPORT
```

**Note:** To produce comparable results in HTML, include the following code in the request to turn on the WebFOCUS Viewer.

```
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET WEBVIEWER ON
ON TABLE SET STYLE SHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The first page of output will contain the column titles and data. You can navigate to the last page to see END OF REPORT.



# Chapter 23

## Formatting Report Data

---

This chapter covers information about formatting and positioning text in a report. You can select the size, color, font, and style for the text of your report, in addition to being able to adjust the position of text within a report.

### In this chapter:

- ❑ [Specifying Font Format in a Report](#)
  - ❑ [Specifying Background Color in a Report](#)
  - ❑ [Alternating Background Color By Wrapped Line](#)
  - ❑ [Specifying Data Format in a Report](#)
  - ❑ [Positioning Data in a Report](#)
- 

### Specifying Font Format in a Report

Using StyleSheet attributes, you can enhance the appearance of a report by specifying the font, size, and color of the font. Font format can be designated for a report as a whole, or for headings, footings, and columns individually.

#### **Syntax:** How to Specify Font Size in a Report

To specify a font size, use the following syntax within a StyleSheet.

```
TYPE = type, [subtype,] SIZE=pts, $
```

where:

*type*

Is the report component you wish to affect, such as REPORT, HEADING, or TITLE.

*subtype*

Is any additional attribute, such as COLUMN, ACROSS, ITEM etc. that is needed to identify the report component that you are formatting. See [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167 for more information about how to specify different report components.

*pts*

Is the size of the font in points. The default value is 10, which corresponds to the HTML default font size 3. For more information on the correlation between point size and html font size, see [Usage Notes for Changing Font Size](#) on page 1612.

**Example: Specifying Font Size in a Report**

In the following report request, the point size of column titles is set to 12:

```
TABLE FILE GGSales
ON TABLE SET PAGE-NUM OFF
SUM UNITS DOLLARS BY CATEGORY
ON TABLE SET STYLE *
TYPE=TITLE, SIZE=12, $
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

<u>Category</u> <u>Unit Sales</u> <u>Dollar Sales</u>		
Coffee	1376266	17231465
Food	1384845	17229344
Gifts	927880	11695502

**Reference: Usage Notes for Changing Font Size**

Point size is fixed, except in an HTML report. Relative point size uses a different scale than HTML font size. The following table lists the point size and the corresponding HTML font size:

Size in Points	Corresponding HTML Font Size
8 or smaller	1
9	2
<u>10</u>	<u>3</u>
11	4
12	5
13	6



Size in Points	Corresponding HTML Font Size
14 or larger	7

### **Syntax:** How to Specify Bold or Italic Font Style in a Report

To specify a font style, use the following syntax within a StyleSheet.

```
TYPE=type, [subtype,] STYLE=[+|-]txtsty[{+|-}txtsty], $
```

where:

*type*

Is the report component you wish to affect, such as REPORT, HEADING, or TITLE.

*subtype*

Is any additional attribute, such as COLUMN, ACROSS, ITEM etc. that is needed to identify the report component that you are formatting. See [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167 for more information about how to specify different report components.

*txtsty*

Is one of the following values: NORMAL, BOLD, ITALIC. The default value is NORMAL.

Note that if you specify a style that is not supported for the font you are using, the specified font will display without the style.

+

Enables you to specify a combination of font styles. You can add additional font styles to an attribute that already has one or more font styles applied to it.

-

Enables you to remove a font style from an attribute.

### **Example:** Specifying Font Style in a Report

In the following report, the column titles are specified to have bold and italic font styles:

```
TABLE FILE GGSales
SUM UNITS DOLLARS BY CATEGORY
ON TABLE SET STYLE *
TYPE=TITLE, STYLE=BOLD+ITALIC, $
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

<i><u>Category</u></i>	<i><u>Unit Sales</u></i>	<i><u>Dollar Sales</u></i>
Coffee	1376266	17231465
Food	1384845	17229344
Gifts	927880	11695502

**Example:** Adding and Removing Inherited Font Style in a Report

In the following report request, the font styles bold and italics are specified for the entire report. The inherited italics are removed from the heading, and both styles are removed from the column titles:

```
TABLE FILE GGSales
HEADING
"Sales Report by Category"
SUM UNITS DOLLARS BY CATEGORY
ON TABLE SET STYLE *
TYPE=REPORT, STYLE=BOLD+ITALIC, $
TYPE=HEADING, STYLE=-ITALIC, $
TYPE=TITLE, STYLE=-BOLD-ITALIC, $
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

<b>Sales Report by Category</b>		
<u>Category</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
<i>Coffee</i>	<i>1376266</i>	<i>17231465</i>
<i>Food</i>	<i>1384845</i>	<i>17229344</i>
<i>Gifts</i>	<i>927880</i>	<i>11695502</i>

**Syntax:** How to Specify Font Color in a Report

To specify a color for the font of a report or a report component, use the following syntax within a StyleSheet.

```
TYPE=type, [subtype,] COLOR={color|RGB({r g b|#hexcolor})},$
```

where:

*type*

Is the report component you wish to affect, such as REPORT, HEADING, or TITLE.

*subtype*

Is any additional attribute, such as COLUMN, ACROSS, ITEM etc. that is needed to identify the report component that you are formatting. See [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167 for more information about how to specify different report components.

*color*

Is one of the preset color values such as GREY or GOLD. If the display or output device does not support colors, it substitutes shades of gray. The default value is BLACK. For a complete list of available color values, see [Color Values in a Report](#) on page 1615.

*RGB(*r g b*)*

Specifies the font color using a mixture of red, green, and blue.

(*r g b*) Is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense. Note that using the three color components in equal intensities results in shades of gray.

*RGB(#*hexcolor*)*

Is the hexadecimal value for the color. For example, FF0000 is the hexadecimal value for red. The hexadecimal digits can be in upper or lower case and must be preceded by a pound sign (#).

**Reference: Color Values in a Report**

The following chart lists all available color values that can be utilized with the syntax

*COLOR=color, or BACKCOLOR=color,*

where *color* is one of the following values:

AQUA (CYAN)	MEDIUM FOREST GREEN (OLIVE)
AQUAMARINE	MEDIUM GOLDENROD
BLACK	MEDIUM ORCHID
BLUE VIOLET	MEDIUM SLATE BLUE
CADET BLUE	MEDIUM SPRING GREEN
CORAL	MEDIUM TURQUOISE
CORNFLOWER BLUE	MEDIUM VIOLET RED

CYAN (AQUA)	MIDNIGHT BLUE
DARK GREEN	NAVY (NAVY BLUE)
DARK OLIVE GREEN	OLIVE (MEDIUM FOREST GREEN)
DARK ORCHID	ORANGE
DARK SLATE BLUE (PURPLE)	ORANGE RED
DARK SLATE GREY	ORCHID
DARK TURQUOISE	PALE GREEN
DIM GREY (GRAY, GREY)	PINK
FIREBRICK	PLUM
FOREST GREEN (GREEN)	PURPLE (DARK SLATE BLUE)
FUCHSIA (MAGENTA)	RED
GOLD	SALMON
GOLDENROD	SEA GREEN
GRAY (DIM GREY, GREY)	SIENNA
GREEN (FOREST GREEN)	SILVER
GREEN YELLOW	SKY BLUE
GREY (DIM GREY, GRAY)	SLATE BLUE
INDIAN RED	STEEL BLUE (TEAL)
KHAKI	TAN
LIGHT BLUE	TEAL (STEEL BLUE)
LIGHT GREY	THISTLE

LIGHT STEEL BLUE	TURQUOISE
LIME	VIOLET
LIME GREEN	VIOLET RED
MAGENTA (FUCHSIA)	WHEAT
MAROON	WHITE
MEDIUM AQUAMARINE	YELLOW
MEDIUM BLUE	YELLOW GREEN

## Specifying Fonts for Reports

You can specify your own fonts in a report by using the FONT attribute in a StyleSheet. If you are specifying a font for an HTML report, the web browser must support the font. If the web browser does not support the font, it reverts to its default behavior of using the default proportional font.

### **Syntax:** How to Specify Fonts in a Report

To specify a font for your report, use the following syntax within a StyleSheet.

```
TYPE=type, [subtype,] FONT='font[,font]', $
```

where:

*type*

Is the report component you wish to affect, such as REPORT, HEADING, or TITLE.

*subtype*

Is any additional attribute, such as COLUMN, ACROSS, ITEM etc. that is needed to identify the report component that you are formatting. See [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167 for more information about how to specify different report components.

*font*

Is the name of the font. You must enclose the value in single quotes. If you are creating an HTML report, you can specify more than one font within the single quotes to accommodate more than one browser.

**Note:** In an HTML report, specifying different fonts for several different report components significantly increases the size of the source code.

### **Example:** Specifying Multiple Fonts in an HTML Report

To control how a report looks on more than one platform, you can specify both a common Windows font and a common UNIX font in a request. The web browser searches for the first font in the list. If the browser does not find the first font, it searches for the next font in the list. If none of the fonts are identified, the browser uses the default proportional font.

In this example, the web browser first searches for the Arial font. If the browser does not find Arial, it searches for the Helvetica font. If neither font is identified, the browser uses the default proportional font.

```
TYPE=REPORT, FONT='ARIAL,HELVETICA', $
```

### **Syntax:** How to Specify the Default Browser Fonts for HTML Reports

A browser assigns specific fonts as the default proportional and default monospaced fonts. To specify a default browser font for an HTML report, you use the reserved names, DEFAULT-PROPORTIONAL and DEFAULT-FIXED in the StyleSheet of your report. The browser displays the report accordingly.

To select the default fixed or proportional font of the browser, use the following syntax. Note that you must specify TYPE to indicate which report components you wish to affect.

```
FONT={DEFAULT-PROPORTIONAL|DEFAULT-FIXED}, $
```

where:

**DEFAULT-PROPORTIONAL**

Specifies the default proportional font of the web browser.

**DEFAULT-FIXED**

Specifies the default monospaced font of the web browser.

**Example: Specifying Default Browser Fonts**

In this example, the web browser uses the default monospace font for the entire report except the report heading and the column headings. For these headings, the web browser uses the default proportional font.

```
TABLE FILE GGSales
HEADING
"Sales Report"
SUM UNITS DOLLARS BY CATEGORY BY PRODUCT
ON TABLE SET STYLE *
TYPE=REPORT, FONT=DEFAULT-FIXED, $
TYPE=TITLE, FONT=DEFAULT-PROPORTIONAL, $
TYPE=HEADING, FONT=DEFAULT-PROPORTIONAL, $
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

Sales Report			
<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	Capuccino	189217	2381600
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263328
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

**Specifying Background Color in a Report**

Using StyleSheet attributes, you can enhance the appearance of a report by specifying a background color. You can designate background colors for a report as a whole, or for headings, footings, and columns individually.

Additionally, alternating backcolors (banded) can be specified for the background of the data lines in a report. These alternating colors are not supported for stacked columns (OVER, FOLD-LINE).

**Syntax:**      **How to Specify Background Color in a Report**

To specify a color for the background of a report, use the following syntax within a StyleSheet.

Note that when using BACKCOLOR in a PDF report, extra space is added to the top, bottom, right, and left of each cell of data in the report. This is for readability and to prevent truncation.

```
TYPE=type, [subtype,] BACKCOLOR={color|RGB({r g b|#hexcolor)}}, $
```

where:

*type*

Is the report component you wish to affect, such as REPORT, HEADING, or TITLE. In a HEADING, FOOTING, SUBHEADING, or SUBFOOTING, you can specify a background color for individual elements.

*subtype*

Is any additional attribute, such as COLUMN, ACROSS, ITEM, that is needed to identify the report component that you are formatting. In a HEADING, FOOTING, SUBHEADING, or SUBFOOTING, you can specify a background color for individual elements. For more information about how to specify different report components, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

*color*

Is the background color, which fills the space of the specified report component. The default value is NONE. If you are creating a report in HTML format, background colors will only appear in web browsers that support them.

RGB (*r g b*)

Specifies the font color using a mixture of red, green, and blue.

(*r g b*) Is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense. Note that using the three color components in equal intensities results in shades of gray.

RGB (#*hexcolor*)

Is the hexadecimal value for the color. For example, FF0000 is the hexadecimal value for red. The hexadecimal digits can be in upper or lower case and must be preceded by a pound sign (#).



**Example: Specifying Background and Font Color in a Report**

You can use color in a report to emphasize important information in a report. In the following report request, the data in the Dollar Sales column has been specified as RED on the condition that the Dollars value is less than 2,500,000. The background color is set to LIGHT BLUE:

```
TABLE FILE GGSALES
ON TABLE SET PAGE-NUM OFF
HEADING
"Sales Report"
SUM UNITS DOLLARS BY CATEGORY BY PRODUCT
ON TABLE SET STYLE *
TYPE=REPORT, BACKCOLOR=LIGHT BLUE, $
TYPE=DATA, COLUMN=DOLLARS, COLOR=RED, WHEN=DOLLARS LT 2500000, $
TYPE=REPORT, GRID=OFF, $
TYPE=HEADING, JUSTIFY=CENTER, SIZE=12,$
ENDSTYLE
END
```

The output is:

Sales Report			
<u>Category</u>	<u>Product</u>	<u>Unit Sales</u>	<u>Dollar Sales</u>
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

## Syntax: How to Specify Alternating Data Background Color in a Report

To specify alternating colors for the background of the data in a report, use the following syntax within a StyleSheet.

Note that when using BACKCOLOR in a PDF report, extra space is added to the top, bottom, right, and left of each cell of data in the report. This is for readability and to prevent truncation.

```
TYPE=DATA, BACKCOLOR=( {c1|RGB({r1 g1 b1|#hc1})} {c2|RGB({r2 g2 b2|#hc2})} ),
$
```

where:

*c1, c2*

Are the background colors for the data in the report. The default value is NONE. If you are creating a report in HTML format, background colors will only appear in web browsers that support them. Color names that contain a space must be enclosed in single quotation marks, as a space is the delimiter between the alternating color values.

*RGB(r1 g1 b1), RGB(r2 g2 b2)*

Specifies the font colors using a mixture of red, green, and blue.

(*r g b*) Is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense. Note that using the three color components in equal intensities results in shades of gray.

*RGB(#hc1), RGB(#hc2)*

Are the hexadecimal values for the colors. For example, FF0000 is the hexadecimal value for red. The hexadecimal digits can be in upper or lower case and must be preceded by a pound sign (#).

## Example: Specifying Alternating Background Colors for the Data Lines in a Report

The following request against the GGSALES data source produces alternating light blue and white data rows on the report output:

```
TABLE FILE GGSALES
ON TABLE SET PAGE-NUM OFF
HEADING
"Sales Report"
SUM UNITS DOLLARS
BY CATEGORY
BY PRODUCT
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=DATA, BACKCOLOR=('LIGHT BLUE' WHITE), $
TYPE=HEADING, JUSTIFY=CENTER, SIZE=12, $
ENDSTYLE
END
```

The output is:

Sales Report			
Category	Product	Unit Sales	Dollar Sales
Coffee	Capuccino	189217	2381590
	Espresso	308986	3906243
	Latte	878063	10943622
Food	Biscotti	421377	5263317
	Croissant	630054	7749902
	Scone	333414	4216114
Gifts	Coffee Grinder	186534	2337567
	Coffee Pot	190695	2449585
	Mug	360570	4522521
	Thermos	190081	2385829

## Alternating Background Color By Wrapped Line

The ALTBACKPERLINE attribute alternates the background color by line for reports that use positioned drivers, for example PDF, DHTML, PPT, and PPTX. This enables you to wrap a long field value, and alternate the background color of each line for that value, independent of borders. In order to apply alternating background color per line, you need to explicitly add the SET ALTBACKPERLINE=ON command to procedures that use WRAP.

### **Reference:** Alternate Background Color By Wrapped Line

```
SET ALTBACKPERLINE = {ON|OFF}
```

where:

**ON**

Alternates background color by line.

**OFF**

Alternates background color by row. This is the default value.

**Example:**    **Alternating Background Color By Wrapped Line**

The following report request prints a COMPUTE field with a long line of text, using WRAP and without using the SET ALTBKPERLINE command. The default alternating background color is applied by row.

```
TABLE FILE WF_RETAIL PRINT
COMPUTE LONG_LINE/A1000 = 'This is a very long line of data, set up so that'
                           | 'it wraps onto the next line within a report'
                           | 'and to test the alternate color syntax'
                           | 'in style sheets.';
BY PRODUCT_CATEGORY BY PRODUCT_SUBCATEG BY MODEL
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET PAGE OFF
ON TABLE SET STYLE *
UNITS=IN, PAGESIZE='A4', LEFTMARGIN=0.19, TOPMARGIN=0.00,
BOTTOMMARGIN=0.00, SQUEEZE=ON, ORIENTATION=LANDSCAPE, $
TYPE=REPORT, FONT='ARIAL', SIZE=8, COLOR=BLACK, BACKCOLOR='NONE',
STYLE=NORMAL, $
TYPE=REPORT, COLUMN=N4, WRAP=1, $
TYPE=DATA, COLUMN=N4, BACKCOLOR=(RGB(235 240 178) RGB(255 255 255)), $
ENDSTYLE
END
```

The output is:

<u>Product Category</u>	<u>Product Subcategory</u>	<u>Model</u>	<u>LONG_LINE</u>
Accessories	Charger	B00D7MOHDO	This is a very long line of data, set up so thatit wraps onto the next line within a reportand to test the alternate color syntaxin style sheets.
		BCG34HRE4KN	This is a very long line of data, set up so thatit wraps onto the next line within a reportand to test the alternate color syntaxin style sheets.
	Headphones	Audio Technica ATHW5000	This is a very long line of data, set up so thatit wraps onto the next line within a reportand to test the alternate color syntaxin style sheets.
		Denon AHD5000	This is a very long line of data, set up so thatit wraps onto the next line within a reportand to test the alternate color syntaxin style sheets.

With the SET ALTBACKPERLINE=ON command added to the request, the alternating background color is applied by line, as shown in the following output.

<u>Product Category</u>	<u>Product Subcategory</u>	<u>Model</u>	<u>LONG_LINE</u>
Accessories	Charger	B00D7MOHDO	This is a very long line of data, set up so thatit wraps onto the next line within a reportand to test the alternate color syntaxin style sheets.
		BCG34HRE4KN	This is a very long line of data, set up so thatit wraps onto the next line within a reportand to test the alternate color syntaxin style sheets.
	Headphones	Audio Technica ATHW5000	This is a very long line of data, set up so thatit wraps onto the next line within a reportand to test the alternate color syntaxin style sheets.
			This is a very long line of data, set up so thatit wraps onto the next line within a reportand to test the alternate color syntaxin style sheets.
		Denon AHD5000	This is a very long line of data, set up so thatit wraps onto the next line within a reportand to test the alternate color syntaxin style sheets.
			This is a very long line of data, set up so thatit wraps onto the next line within a reportand to test the alternate color syntaxin style sheets.

Specifying Data Format in a Report

You can affect how data is represented in a report in several ways:

- ☐ You can change the format of the numeric values that appear in a column.
- ☐ You can determine whether or not you wish to use Continental Decimal Notation (CDN).
- ☐ You can set a specific character or set of characters to represent fields that do not contain data.

## Changing the Format of Values in a Report Column

A field format is defined in the Master File. You can, however, change the format of a report column. Field formats are described in full detail in the *Describing Data With WebFOCUS Language* manual.

### **Syntax:** How to Change Format of Values in a Column

```
fieldname [alignment] [/format]
```

where:

*fieldname*

Is a display field—that is, a field displayed by the PRINT, LIST, SUM, or COUNT command, a row-total, or a column-total.

*alignment*

Specifies the position of the column title.

/R specifies a right justified column title.

/L specifies a left justified column title.

/C specifies a centered column title.

*format*

Is any valid field format, preceded by a slash (/). Field formats are described in the *Describing Data With WebFOCUS Language* manual. Field formats cannot be used with a column total.

### **Example:** Changing the Format of Values in a Column

The UNIT\_PRICE field has a format of D7.2 as defined in the GGPRODS Master File. To add a floating dollar sign to the display, the field format can be redefined as follows:

```
TABLE FILE GGPRODS
PRINT UNIT_PRICE/D7.2M
END
```

The output is:

Unit	
Price	\$58.00
\$81.00	
\$76.00	
\$13.00	
\$17.00	
\$28.00	
\$26.00	
\$96.00	
\$125.00	
\$140.00	

**Example:**    **Using Multiple Format Specifications in a Column**

The following request illustrates column title justification with a format specification, a BY field specification, and an AS phrase specification:

```
TABLE FILE CAR
PRINT MODEL/A10 STANDARD/A15/R AS 'RJUST,STANDARD'
BY CAR/C
WHERE CAR EQ 'JAGUAR' OR 'TOYOTA'
END
```

The output is:

CAR	MODEL	RJUST STANDARD
JAGUAR	V12XKE AUT XJ12L AUTO	POWER STEERING RECLINING BUCKE WHITEWALL RADIA WRAP AROUND BUM 4 WHEEL DISC BR
TOYOTA	COROLLA 4	BODY SIDE MOLDI MACPHERSON STRU

**Reference:**    **Usage Notes for Changing Column Format**

- ❑ Each time you reformat a column, the field is counted twice against the limit for display fields in a single report. For details, see [Controlling Report Formatting](#) on page 1139.
- ❑ If you create an extract file from the report—that is, a HOLD, PCHOLD, SAVE, or SAVB file—the extract file will contain fields for both the original format and the redefined format, unless HOLDLIST=PRINTONLY. Extract files are described in [Saving and Reusing Your Report Output](#) on page 471.
- ❑ When the size of a word in a text field instance is greater than the format of the text field in the Master File, the word wraps to a second line, and the next word begins on the same line.



- ❑ You may specify justification for display fields, BY fields, and ACROSS fields. For ACROSS fields, data values, not column titles, are justified as specified.
- ❑ For display commands only, the justification parameter may be combined with a format specification. The format specification may precede or follow the justification parameter.
- ❑ If a title is specified with an AS phrase or in the Master File, that title will be justified as specified in FORMAT.
- ❑ When multiple ACROSS fields are requested, justification is performed on the lowest ACROSS level only. All other justification parameters for ACROSS fields are ignored.

### Controlling Missing Values for a Reformatted Field

When a field is reformatted in a request (for example, SUM *field/format*), an internal COMPUTE field is created to contain the reformatted field value and display on the report output. If the original field has a missing value, that missing value can be propagated to the internal field by setting the COMPMISS parameter ON. If the missing value is not propagated to the internal field, it displays a zero (if it is numeric) or a blank (if it is alphanumeric). If the missing value is propagated to the internal field, it displays the missing data symbol on the report output.

#### **Syntax:** How to Control Missing Values in Reformatted Fields

```
SET COMPMISS = {ON|OFF}
```

where:

ON

Propagates a missing value to a reformatted field. ON is the default value.

OFF

Displays a blank or zero for a reformatted field.

#### **Example:** Controlling Missing Values in Reformatted Fields

The following procedure prints the RETURNS field from the SALES data source for store 14Z. With COMPMISS OFF, the missing values display as zeros in the column for the reformatted field value.

**Note:** Before trying this example, you must make sure that the SALEMISS procedure, which adds missing values to the SALES data source, has been run.

```
SET COMPMISS = OFF
TABLE FILE SALES
PRINT RETURNS RETURNS/D12.2 AS 'REFORMATTED,RETURNS'
BY STORE_CODE
WHERE STORE_CODE EQ '14Z'
END
```

The output is:

STORE_CODE	RETURNS	REFORMATTED RETURNS
-----	-----	-----
14Z	2	2.00
	2	2.00
	0	.00
	.	.00
	4	4.00
	0	.00
	3	3.00
	4	4.00
	.	.00
	4	4.00

With COMPMISS ON, the column for the reformatted version of RETURNS displays the missing data symbol when a value is missing:

```
SET COMPMISS = ON
TABLE FILE SALES
PRINT RETURNS RETURNS/D12.2 AS 'REFORMATTED,RETURNS'
BY STORE_CODE
WHERE STORE_CODE EQ '14Z'
END
```

The output is:

STORE_CODE	RETURNS	REFORMATTED RETURNS
-----	-----	-----
14Z	2	2.00
	2	2.00
	0	.00
	.	.
	4	4.00
	0	.00
	3	3.00
	4	4.00
	.	.
	4	4.00

**Reference: Usage Notes for SET COMPMISS**

- ❑ If you create a HOLD file with COMPMISS ON, the HOLD Master File for the reformatted field indicates MISSING = ON (as does the original field). With COMPMISS = OFF, the reformatted field does NOT have MISSING = ON in the generated Master File.
- ❑ The COMPMISS parameter cannot be set in an ON TABLE command.

**Using Commas vs. Decimals (Continental Decimal Notation)**

The CDN parameter determines the characters used for punctuation in numbers. For information about the CDN parameter values, see the *Developing Reporting Applications* manual.

**Setting Characters to Represent Null and Missing Values**

You can alter the appearance of your report output by specifying your own string of characters that will appear when no data is available for a field.

**Syntax: How to Set Characters to Represent a Null or Missing Value**

To specify a string for NODATA fields, use the following syntax

```
ON TABLE SET NODATA character string
```

where:

*NODATA*

Indicates that a NODATA character will be set.

*character string*

Is the string of characters that you want to appear when no data is available for a field. The default value is a period (.).

**Syntax: How to Set the NODATA Character as a SET Command**

To specify a character for NODATA fields, use the following syntax

```
SET NODATA=character
```

where:

*character*

Is the character or characters that you want to appear when no data is available for a field. The maximum number of characters is 11. The default value is a period (.).

### **Example: Setting the NODATA Character in a Request**

This request changes the NODATA character for missing data from a period (default) to the word NONE.

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL
BY LAST_NAME BY FIRST_NAME
ACROSS DEPARTMENT
ON TABLE SET NODATA NONE
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, SQUEEZE=OFF,$
ENDSTYLE
END
```

This request produces the following report.

		<u>DEPARTMENT</u>	
<u>LAST_NAME</u>	<u>FIRST_NAME</u>	<u>MIS</u>	<u>PRODUCTION</u>
BANNING	JOHN	NONE	\$29,700.00
BLACKWOOD	ROSEMARIE	\$21,780.00	NONE
CROSS	BARBARA	\$27,062.00	NONE
GREENSPAN	MARY	\$9,000.00	NONE
IRVING	JOAN	NONE	\$26,862.00
JONES	DIANE	\$18,480.00	NONE
MCCOY	JOHN	\$18,480.00	NONE
MCKNIGHT	ROGER	NONE	\$16,100.00
ROMANS	ANTHONY	NONE	\$21,120.00
SMITH	MARY	\$13,200.00	NONE
	RICHARD	NONE	\$9,500.00
STEVENS	ALFRED	NONE	\$11,000.00

### **Using Conditional Grid Formatting in a Field**

You can use conditional grid formatting in order to emphasize a particular cell or field in a report.

**Example: Creating a Report Using Conditional Grid Formatting**

```
TABLE FILE CAR
SUM SALES BY CAR
ON TABLE SET STYLE *
ON TABLE PCHOLD FORMAT PDF
TYPE=DATA, COLUMN=SALES, GRID=HEAVY, WHEN=CAR EQ 'DATSUN', $
ENDSTYLE
END
```

The output is:

<u>CAR</u>	<u>SALES</u>
ALFA ROMEO	30200
AUDI	7800
BMW	80390
DATSUN	43000
JAGUAR	12000
JENSEN	0
MASERATI	0
PEUGEOT	0
TOYOTA	35030
TRIUMPH	0

**Positioning Data in a Report**

You can position data within a report by selecting a justification (right, left or center) of a column or by specifying whether or not you wish to have data wrap within a cell. For information on positioning a column on the page, see [Laying Out the Report Page](#) on page 1249.

**Controlling Wrapping of Report Data**

You can control the wrapping of report data in a report, thus preventing line breaks within report cells. When using HTML output, most web browsers will, by default, wrap alphanumeric report data that does not fit on a single line in a cell.

This bumps the contents of the cell onto a second line. A web browser wraps data based on its algorithmic settings. Use the WRAP attribute if you wish to suppress a web browser data wrapping.

By default, WRAP is set to ON for HTML output, allowing each individual browser to define the width of each column in the report. For PDF, PS, DHTML, PPT, and PPTX output, WRAP is set OFF by default. For these positioned output formats in which the location of each item in the report is explicitly defined, WRAP = ON is not a valid value, except when specified for ACROSSVALUE. For other elements of the report, such as headers, footers, titles, or data, define the width of wrapped lines by using a numerical value, as in WRAP = *n*.

In PDF and PostScript report output, you can control the line spacing in wrapped lines by using the WRAPGAP attribute.

### Wrapping Data in PDF Reports That Use the OVER Phrase

OVER allows the presentation of a single data record across multiple lines within a report. By default, when OVER is defined within a request, the report shifts from a columnar presentation to a row level presentation. The field titles are displayed to the left of each value, rather than at the top of each column. This layout was not designed to be aligned in any specific fashion but to allow for the presentation of multiple elements of data within a small area. In many cases, reports that place columns over each other use blank AS names in order to align the columns properly. You can use the WRAP attribute to wrap data in PDF reports that use OVER and this technique works well with blank AS names.

### Wrapping Data in PDF Reports That Use the ACROSS Phrase

In a request that uses ACROSS, the output displays each value of the ACROSS field above the set of data columns applicable to that ACROSS value.

If the ACROSS value is longer than the width of its columns, you can wrap the ACROSS value within the width of its underlying columns.

By default, the width of each ACROSS value group (the ACROSS value and the data columns within) is defined as the largest of either the sum of the width of the data columns or the largest ACROSS value for that group. With wrapping, the size of each ACROSS wrap will be defined by the width defined based on this rule including all data columns and any non-wrapped across fields.

The width of each ACROSS column for a given ACROSS group is defined as the length of the largest value for that ACROSS group. A single width is used for each group so in groups where the values are shorter than the longest value, you will see a larger right gap within the cell.

For reports containing multiple ACROSS fields, you can wrap individual ACROSS fields or all of them. Each designated value will wrap within the defined ACROSS group.

### **Syntax:** How to Control Wrapping of Report Data

To control wrapping of text inside a report, use the following syntax within a StyleSheet.

```
TYPE=type, [subtype,] WRAP=value, $
```

where:

*type*

Is the report component you wish to affect, such as REPORT, HEADING, or TITLE.

*subtype*

Is any additional attribute, such as COLUMN, ACROSS, ITEM etc. that is needed to identify the report component that you are formatting. See [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167 for more information about how to specify different report components.

*value*

Is one of the following:

- ☐ **ON**, which turns on data wrapping. ON is the default value for HTML report output. For PDF, PS, DHTML, PPT, and PPTX report output, WRAP is set OFF by default. For these positioned output formats in which the location of each item in the report is explicitly defined, WRAP = ON is not a valid value, except when specified for ACROSSVALUE. For other elements of the report, such as headers, footers, titles, or data, define the width of wrapped lines by using a numerical value, as in WRAP = *n*. For HTML reports, WRAP is supported with all fields. For PDF reports, WRAP is supported only with embedded fields, not text.

**Note:** This setting is not supported when using WRAP with OVER in PDF report output.

- ☐ **OFF**, which turns off data wrapping. This is the default value for PDF, PS, DHTML, PPT, and PPTX report output.
- ☐ ***n***, which represents a specific numeric value that the column width can be set to. The value represents the measure specified with the UNITS parameter. This setting is supported for wrapping data in PDF reports that use the OVER phrase.

**Note:** WRAP=ON and WRAP=*n* are not supported with JUSTIFY.

### **Example:** Allowing the Web Browser to Wrap Report Data

The following example, with WRAP=ON, wraps report data based on the web browser functionality. Note that because this value is the default, there is no need to specify WRAP=ON in the report request syntax.

```
TABLE FILE GGPRODS
PRINT SIZE UNIT_PRICE PACKAGE_TYPE
VENDOR_CODE VENDOR_NAME
BY PRODUCT_ID BY PRODUCT_DESCRIPTION
ON TABLE SET STYLE *
TYPE=REPORT, GRID=ON, $
ENDSTYLE
END
```

**Note:** Wrap is determined by the size of your browser window, so you may need to shrink your window to see the example wrap the data as in the following image.

PAGE 1						
Product Code	Product	Size	Unit Price	Package	Vendor ID	Vendor Name
B141	Hazelnut	16	58.00	Pounds	V082	Coffee Connection
B142	French Roast	12	81.00	Pounds	V083	European Specialities,
B144	Kona	12	76.00	Pounds	V081	Evelina Imports, Ltd

Notice that records in the Vendor Name column break to a second line.

### **Example:** Suppressing the Wrapping of Report Data

The following report request, with **WRAP=OFF**, suppresses the web browser data wrapping:

```
TABLE FILE GGPRODS
PRINT SIZE UNIT_PRICE PACKAGE_TYPE
VENDOR_CODE VENDOR_NAME
BY PRODUCT_ID BY PRODUCT_DESCRIPTION
ON TABLE SET STYLE *
TYPE=REPORT, WRAP=OFF, $
TYPE=REPORT, GRID=ON, $
ENDSTYLE
END
```

The output is:

PAGE 1						
Product Code	Product	Size	Unit Price	Package	Vendor ID	Vendor Name
B141	Hazelnut	16	58.00	Pounds	V082	Coffee Connection
B142	French Roast	12	81.00	Pounds	V083	European Specialities,
B144	Kona	12	76.00	Pounds	V081	Evelina Imports, Ltd



**Example: Wrapping Columns With OVER**

The following request against the GGPRODS data source places the column VENDOR\_NAME on a new line with the OVER phrase. By default, wrap is turned off and must be defined explicitly within the StyleSheet:

```
TABLE FILE GGPRODS
PRINT SIZE UNIT_PRICE PACKAGE_TYPE OVER
VENDOR_NAME
BY PRODUCT_ID BY PRODUCT_DESCRIPTION
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, SQUEEZE=ON, $
ENDSTYLE
END
```

The partial output is shown in the following image.

PAGE 1

Product Code	Product					
B141	Hazelnut	Size 16	Unit Price	58.00	Package	Pounds
		Vendor Name	Coffee Connection			
B142	French Roast	Size 12	Unit Price	81.00	Package	Pounds
		Vendor Name	European Specialities,			
B144	Kona	Size 12	Unit Price	76.00	Package	Pounds
		Vendor Name	Evelina Imports, Ltd			
F101	Scone	Size 20	Unit Price	13.00	Package	Case
		Vendor Name	Ridgewood Bakeries			
F102	Biscotti	Size 24	Unit Price	17.00	Package	Case
		Vendor Name	Delancey Bakeries			
F103	Croissant	Size 20	Unit Price	28.00	Package	Case
		Vendor Name	West Side Bakers			
G100	Mug	Size 24	Unit Price	26.00	Package	Case
		Vendor Name	NY Ceramic Supply			
G104	Thermos	Size 16	Unit Price	96.00	Package	Case
		Vendor Name	ThermoTech, Inc			
G110	Coffee Grinder	Size 12	Unit Price	125.00	Package	Case
		Vendor Name	Appliance Craft			
G121	Coffee Pot	Size 8	Unit Price	140.00	Package	Case
		Vendor Name	Appliance Craft			

The following version of the request turns wrapping on and sets a column width of 1.5 for the VENDOR\_NAME column, which has been placed on a new line because of the OVER phrase:

```
TABLE FILE GGPRODS
PRINT SIZE UNIT_PRICE PACKAGE_TYPE OVER
VENDOR_NAME
BY PRODUCT_ID BY PRODUCT_DESCRIPTION
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, COLUMN=VENDOR_NAME, WRAP=1.5,$
ENDSTYLE
END
```

The partial output shows that the VENDOR\_NAME column now wraps. Notice that turning WRAP ON causes the OVER value, not the OVER TITLE, to wrap:

PAGE 1

<u>Product Code</u>	<u>Product</u>						
B141	Hazelnut	Size 16	Unit Price	58.00	Package	Pounds	
		Vendor Name	Coffee				
			Connection				
B142	French Roast	Size 12	Unit Price	81.00	Package	Pounds	
		Vendor Name	European				
			Specialities,				
B144	Kona	Size 12	Unit Price	76.00	Package	Pounds	
		Vendor Name	Evelina				
			Imports, Ltd				
F101	Scone	Size 20	Unit Price	13.00	Package	Case	
		Vendor Name	Ridgewood				
			Bakeries				
F102	Biscotti	Size 24	Unit Price	17.00	Package	Case	
		Vendor Name	Delancey				
			Bakeries				
F103	Croissant	Size 20	Unit Price	28.00	Package	Case	
		Vendor Name	West Side				
			Bakers				
G100	Mug	Size 24	Unit Price	26.00	Package	Case	
		Vendor Name	NY Ceramic				
			Supply				

### **Syntax:** How to Wrap ACROSS Values

Wrapping ACROSS Values is supported for HTML and PDF output formats.

```
TYPE=ACROSSVALUE, [ACROSS={fieldname|Nn|An}] WRAP={OFF|ON} , $
```

where:

#### ACROSS

If you have a request with multiple ACROSS fields, you can identify each field using the ACROSS identifier. You only need to include the ACROSS identifier if you have multiple ACROSS fields in your request.

#### *fieldname*

Specifies a horizontal sort row by its field name.

#### Nn

Identifies a column by its position in the report. To determine this value, count vertical sort (BY) fields, display fields, and ROW-TOTAL fields, from left to right, including NOPRINT fields.

#### An

Specifies a horizontal sort row by its position in the sequence of horizontal sort rows. To determine this value, count horizontal sort (ACROSS) fields. Cannot be combined with a field name specification in the same StyleSheet.

#### OFF

Turns off wrapping of the ACROSS values. OFF is the default value.

#### ON

Turns on wrapping of the ACROSS values.

**Note:** WRAP=ON is not supported with JUSTIFY.

### **Example:** Wrapping ACROSS Values in PDF Report Output

In the following request against the GGPRODS data source, VENDOR\_NAME is an ACROSS field:

```
TABLE FILE GGPRODS
HEADING
"  PRODUCT REPORT"
"  "
PRINT PRODUCT_ID UNIT_PRICE/D5
ACROSS VENDOR_NAME
BY SIZE
WHERE VENDOR_NAME GT 'B' AND VENDOR_NAME LT 'F'
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=REPORT, COLUMN=PRODUCT_ID, WIDTH=.25, $
TYPE=REPORT, COLUMN=UNIT_PRICE, WIDTH=.25, $
ENDSTYLE
END
```

As shown in the following image, the output is too wide for one panel because some of the ACROSS field values (vendor names) are longer than the sum of the product code and unit price columns under them.

PAGE 1.1

### PRODUCT REPORT

size	Vendor Name Coffee Connection Product Code	Unit Price	Delancey Product Code	Bakeries Unit Price	European Product Code	Specialities, Unit Price
12	.	.	.	.	B142	81
16	B141	58	.	.	.	.
24	.	.	F102	17	.	.

The following version of the request wraps the ACROSS values (TYPE=ACROSSVALUE, WRAP=ON , \$):

```
TABLE FILE GGPRODS
HEADING
" PRODUCT REPORT"
" "
PRINT PRODUCT_ID UNIT_PRICE/D5
ACROSS VENDOR_NAME
BY SIZE
WHERE VENDOR_NAME GT 'B' AND VENDOR_NAME LT 'F'
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=REPORT, COLUMN=PRODUCT_ID, WIDTH=.25, $
TYPE=REPORT, COLUMN=UNIT_PRICE, WIDTH=.25, $
TYPE = ACROSSVALUE, WRAP=ON,$
ENDSTYLE
END
```

The report now fits on one panel, as shown in the following image.

PAGE 1

## PRODUCT REPORT

	Vendor Name		Delancey Bakeries		European Specialities,		Evelina Imports, Ltd	
	Product	Unit	Product	Unit	Product	Unit	Product	Unit
Size	Code	Price	Code	Price	Code	Price	Code	Price
12	.	.	.	.	B142	81	.	.
	.		.	.	.	.	B144	76
16	B141	58	.	.	.	.	.	.
24	.	.	F102	17	.	.	.	.

**Reference:** OVER With Blank Column Titles

When OVER fields are defined with blank AS names (the value of the title of the column is set to empty ' '), they can be used to build a report with multiple data lines that present in an aligned grid fashion.

In this type of report, the column titles are usually indicated by adding multiple corresponding lines to the page headings rather than using the default titles that display to the left of the column field values. To present OVER fields with unique titles that take advantage of these new alignment features, you can place the column titles in independent fields and include them as fields within the given request.

**Example: Using OVER and WRAP With Blank AS Names**

The following example demonstrates using OVER with blank AS names and WRAP to build a multi-data line report:

```
TABLE FILE GGPRODS
PRINT PACKAGE_TYPE AS '' SIZE AS '' OVER
VENDOR_NAME AS ''
BY PRODUCT_ID AS ''
BY PRODUCT_DESCRIPTION AS ''
ON TABLE SUBHEAD
"Gotham Grinds"
"Products Details"
HEADING
" Code <+0>Description<+0>Size <+0>Package"
-*" <+0> <+0>Vendor"
" <+0>Vendor"
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=ARIAL, SIZE=10, SQUEEZE=ON,$
TYPE=REPORT, COLUMN=PACKAGE_TYPE, SQUEEZE=.5,$
TYPE=REPORT, COLUMN=VENDOR_NAME, WRAP=1,$
TYPE=REPORT, BORDER=ON,$
TYPE=HEADING, LINE=1, ITEM=1, BORDER=ON,$
TYPE=HEADING, LINE=1, ITEM=2, BORDER=ON, POSITION=PRODUCT_DESCRIPTION,$
TYPE=HEADING, LINE=1, ITEM=3, BORDER=ON, POSITION=SIZE,$
TYPE=HEADING, LINE=1, ITEM=4, BORDER=ON, POSITION=PACKAGE_TYPE,$
TYPE=HEADING, LINE=2, ITEM=1, BORDER=ON,$
TYPE=HEADING, LINE=2, ITEM=2, BORDER=ON, POSITION=PACKAGE_TYPE,$
ENDSTYLE
END
```

On the report output, the Package Type and Size have been placed over the vendor name. The page heading has the corresponding titles. In the heading, the titles Package and Size have also been placed over the title Vendor Name. Note that the vendor name data wraps to maintain the alignment.

PAGE 1

Gotham Grinds Products Details
-----------------------------------

Code	Description	Package	Size
		Vendor	

B141	Hazelnut	Pounds	16
		Coffee Connection	
B142	French Roast	Pounds	12
		European Specialities,	
B144	Kona	Pounds	12
		Evelina Imports, Ltd	
F101	Scone	Case	20
		Ridgewood Bakeries	
F102	Biscotti	Case	24
		Delancey Bakeries	
F103	Croissant	Case	20
		West Side Bakers	

### **Reference: OVER and WRAP With Non-Blank Column Titles**

The width of both the column title and the column data for each OVER value is determined by the single SQUEEZE or WRAP value. The title will automatically size to the same width as the wrapped data column. If the column title is wider than the width defined for the column wrap, you can either define a smaller title or add your titles as OVER fields that can be sized independently.

The following examples demonstrate how to build a report with OVER and WRAP that has column titles longer than the designated WRAP size.

**Example:**    **Using OVER and WRAP With Column Titles**

The following request defines two virtual fields to contain the column titles for the Product Name and Vendor Name fields. It then prints each virtual field next to its related data field and gives each a blank AS name. The first virtual field and data field are placed over the second virtual field and data field:

```
DEFINE FILE GGPRODS
TITLE_PROD/A20 = 'Product Description';
TITLE_VEND/A20 = 'Vendor Name';
END
TABLE FILE GGPRODS
PRINT TITLE_PROD AS '' PRODUCT_DESCRIPTION AS '' OVER
TITLE_VEND AS '' VENDOR_NAME AS ''
BY PRODUCT_ID AS ''
ON TABLE SUBHEAD
"Gotham Grinds"
"Products Details"
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=ARIAL, SIZE=10, SQUEEZE=ON,$
TYPE=REPORT, COLUMN=TITLE_PROD , SQUEEZE=1.25 ,$
TYPE=REPORT, COLUMN=TITLE_VEND , SQUEEZE=1.25 ,$
TYPE=REPORT, COLUMN=PRODUCT_DESCRIPTION, WRAP=.75 ,$
TYPE=REPORT, COLUMN=VENDOR_NAME, WRAP=.75 ,$
TYPE=REPORT, BORDER=ON, $
ENDSTYLE
END
```



The output shows that the titles and data align properly.

PAGE 1

Gotham Grinds Products Details		
B141	Product Description	Hazelnut
	Vendor Name	Coffee Connection
B142	Product Description	French Roast
	Vendor Name	European Specialities,
B144	Product Description	Kona
	Vendor Name	Evelina Imports, Ltd
F101	Product Description	Scone
	Vendor Name	Ridgewood Bakeries
F102	Product Description	Biscotti
	Vendor Name	Delancey Bakeries
F103	Product Description	Croissant
	Vendor Name	West Side Bakers
G100	Product Description	Mug
	Vendor Name	NY Ceramic Supply
G104	Product Description	Thermos

### **Syntax:** How to Control Spacing Between Wrapped Lines

You can use the WRAPGAP attribute in a StyleSheet to control spacing between wrapped lines in the data elements in PDF and PostScript report output.

`TYPE=DATA, WRAPGAP={ON|OFF|n}`

where:

**ON**

Does not leave any space between wrapped lines. ON is equivalent to specifying 0.0 for *n*.

**OFF**

Places wrapped data on the next line. OFF is the default value.

*n*

Is a number greater than or equal to zero that specifies how much space to leave between wrapped lines (using the unit of measurement specified by the UNITS attribute). Setting *n* to zero does not leave any space between wrapped lines, and is equivalent to specifying WRAPGAP=ON.

### ***Example:*** Specifying Spacing for Wrapped Lines

In the following request, wrapping is turned on for the ADDRESS\_LN3 column of the report:

```
TABLE FILE EMPLOYEE
PRINT ADDRESS_LN3
BY LAST_NAME BY FIRST_NAME
WHERE LAST_NAME LE 'CROSS'
  ON TABLE PCHOLD FORMAT PDF
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
type=report, grid=on, $
type=data, topgap=0.2, bottomgap=0.2, $
type=data, wrapgap=off, $
type=REPORT, column=ADDRESS_LN3, wrap=1.0 ,$
ENDSTYLE
END
```

With WRAPGAP=OFF, each wrapped line is placed on the next report line:

LAST NAME	FIRST NAME	ADDRESS LINE 1
BANNING	JOHN	FREEPORT
		NY 11520
BLACKWOOD	ROSEMARIE	NEW YORK
		NY 10001
		EDISON NJ
		08817
CROSS	BARBARA	BROOKLYN
		NY 11210
		NEW YORK
		NY 10001
		FLUSHING
		NY 11354

With WRAPGAP=ON, the wrapped lines are placed directly under each other:

LAST NAME	FIRST NAME	ADDRESS L!
BANNING	JOHN	FREEPORT NY 11520
BLACKWOOD	ROSEMARIE	NEW YORK NY 10001
		EDISON NJ 08817
		BROOKLYN NY 11210
CROSS	BARBARA	NEW YORK NY 10001
		FLUSHING NY 11354

**Reference: Usage Notes for WRAPGAP**

You can only specify WRAPGAP for columns that have wrapping enabled (WRAP attribute or parameter set to ON or a number). The TOPGAP and BOTTOMGAP attributes specify how much vertical space to leave above and below a report component. Increasing the values of these attributes makes a decrease in spacing between wrapped lines more noticeable.

**Justifying Report Columns**

You can adjust text within a column by specifying whether report columns are left justified, right justified, or centered. By default, alphanumeric columns are left justified, numeric columns are right justified, and heading and footing elements are left justified. However, you can change the default using the JUSTIFY attribute. For information on justifying column titles using /R /L and /C, see *Using Headings, Footings, Titles, and Labels* on page 1431.

**Syntax:**      **How to Justify a Report Column**

To left justify, right justify, or center a column, use the following syntax within a StyleSheet.

```
TYPE=type, [subtype,] [COLUMN=column,] JUSTIFY=option, $
```

where:

*type*

Is the report component you wish to affect, such as REPORT, HEADING, or TITLE.

*subtype*

Is any additional attribute, such as COLUMN, ACROSS, ITEM etc. that is needed to identify the report component that you are formatting. For more information about how to specify different report components, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

*column*

Is the column or group of columns you wish to justify. This attribute is only necessary if you wish to justify a specific column or set of columns. Omitting this attribute justifies the entire report.

*option*

Is the justification you wish to select:

- ☐ **LEFT**, which specifies that the column will be left justified.
- ☐ **RIGHT**, which specifies that the column will be right justified.
- ☐ **CENTER**, which specifies that the column will be centered.

**Note:** JUSTIFY is not supported with WRAP=ON or WRAP=*n*.

**Example:**      **Justifying Data in a Report Column**

The following example displays the StyleSheet syntax used to center the data in the Vendor Name column. The header is also center justified.

```
TABLE FILE GGPRODS
HEADING
"PRODUCT REPORT"
SUM UNITS BY PRODUCT_DESCRIPTION BY PRODUCT_ID BY VENDOR_NAME
ON TABLE SET STYLE *
TYPE=REPORT, COLUMN=VENDOR_NAME, JUSTIFY=CENTER, $
TYPE=HEADING, JUSTIFY=CENTER, $
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

PRODUCT REPORT			
<u>Product</u>	<u>Product Code</u>	<u>Vendor Name</u>	<u>Unit Price</u>
Biscotti	F102	Delancey Bakeries	17.00
Coffee Grinder	G110	Appliance Craft	125.00
Coffee Pot	G121	Appliance Craft	140.00
Croissant	F103	West Side Bakers	28.00
French Roast	B142	European Specialities,	81.00
Hazelnut	B141	Coffee Connection	58.00
Kona	B144	Evelina Imports, Ltd	76.00
Mug	G100	NY Ceramic Supply	26.00
Scone	F101	Ridgewood Bakeries	13.00
Thermos	G104	ThermoTech, Inc	96.00

## Field-Based Reformatting

Field-based reformatting allows you to apply different formats to each row in a single report column by using a field to identify the format that applies to each row. For example, you can use this technique to apply the appropriate decimal currency formats when each row represents a different country.

The field that contains the format specifications can be:

- ☐ A real field in the data source.
- ☐ A temporary field created with a DEFINE command.
- ☐ A DEFINE in the Master File.
- ☐ A COMPUTE command. If the field is created with a COMPUTE command, the command must appear in the request prior to using the calculated field for reformatting.

The field that contains the formats must be alphanumeric and be at least eight characters in length. Only the first eight characters are used for formatting.

The field-based format may specify a length longer than the length of the original field. However, if the new length is more than one-third larger than the original length, the report column width may not be large enough to hold the value (indicated by asterisks in the field).

You can apply a field-based format to any type of field. However, the new format must be compatible with the original format:

- ☐ A numeric field can be reformatted to any other numeric format with any edit format options.
- ☐ An alphanumeric field can be reformatted to a different length.
- ☐ Any date field can be reformatted to any other date format type.
- ☐ Any date-time field can be reformatted to any other date-time format.

If the field-based format is invalid or specifies an impermissible type conversion, the field displays with plus signs (+++++) on the report output. If the format field is blank or missing, the value is displayed without reformatting.

### ***Syntax:***

#### **How to Define and Apply a Format Field**

- ☐ With a DEFINE command:

```
DEFINE FILE filename
format_field/A8 = expression;
END
```

- ☐ In a Master File:

```
DEFINE format_field/A8 = expression; $
```

- ☐ In a request:

```
COMPUTE format_field/A8 = expression;
```

where:

*format\_field*

Is the name of the field that contains the format for each row.

*expression*

Is the expression that assigns the format values to the format field.

After the format field is defined, you can apply it in a report request:

```
TABLE FILE filename
displayfieldname/format_field[/just]
END
```

where:

*display*

Is any valid display command.

*fieldname*

Is a field in the request to be reformatted.

*format\_field*

Is the name of the field that contains the formats. If the name of the format field is the same as an explicit format, the explicit format will be used. For example, a field named I8 cannot be used for field-based reformatting because it will be interpreted as the explicit format I8.

*just*

Is a justification option, L, R, or C. The justification option can be placed before or after the format field, separated from the format by a slash.

**Example:   Displaying Different Decimal Places for Currency Values**

```
DEFINE FILE CAR
CFORMAT/A8 = DECODE COUNTRY('ENGLAND' 'D10.1' 'JAPAN' 'D10' ELSE
'D10.2');
END

TABLE FILE CAR
SUM SALES/CFORMAT/C DEALER_COST/CFORMAT
BY COUNTRY
END
```

The output is:

COUNTRY	SALES	DEALER_COST
-----	-----	-----
ENGLAND	12,000.0	37,853.0
FRANCE	.00	4,631.00
ITALY	30,200.00	41,235.00
JAPAN	78,030	5,512
W GERMANY	88,190.00	54,563.00



## Displaying Multi-Line An and AnV Fields

Using StyleSheet attributes, you can display An (character) and AnV (varchar) fields that contain line breaks on multiple lines in a PDF or PostScript report. Line breaks can be based on line-feeds, carriage-returns, or a combination of both. If you do not add these StyleSheet attributes, all line-feed and carriage-return formatting within these fields will be ignored, and all characters will be displayed on one line that wraps to fit the width of the report.

### **Syntax:** How to Display An and AnV Fields Containing Line Breaks on Multiple Lines

```
TYPE=REPORT,LINEBREAK= ' type' , $
```

where:

**REPORT**

Is the type of report component. TYPE must be REPORT. Otherwise an error will result.

**' type'**

Specifies that line breaks will be inserted in a report based on the following:

**LF** inserts a line break after each line-feed character found in all An and AnV fields.

**CR** inserts a line break after each carriage-return character found in all An and AnV fields.

**LF CR** inserts a line break after each combination of a line-feed character followed by a carriage-return character found in all An and AnV fields.

**CRLF** inserts a line break after each combination of a carriage-return character followed by a line-feed character found in all An and AnV fields.

**Note:** This feature is supported in PDF, PPTX, or PS formats.

**Example:**     **Displaying an Alphanumeric Field With Line Breaks in a PDF Report**

The following request defines an alphanumeric named ANLB field with a semicolon (in an EDCDIC environment) or a circumflex (in an ASCII environment) in the middle. The CTRAN function then replaces the semicolon or circumflex with a carriage return character and stores this string in a field named ANLBC. On the report output, this field displays on two lines:

```
DEFINE FILE EMPLOYEE
ANLB/A40 ='THIS IS AN An FIELD;WITH A LINE BREAK.';
ANLBC/A40 = CTRAN(40, ANLB, 094, 013  , ANLBC);
END
TABLE FILE EMPLOYEE
PRINT LAST_NAME ANLBC
WHERE LAST_NAME EQ 'BLACKWOOD'
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT,LINEBREAK='CR',$,
ENDSTYLE
END
```

The output is:

<u>LAST_NAME</u>	<u>ANLBC</u>
BLACKWOOD	THIS IS AN An FIELD WITH A LINE BREAK.

**Example:**     **Using an Alphanumeric Field With a Line Break in a Subfoot**

The following request defines an alphanumeric named ANLB field with a semicolon in the middle. The CTRAN function then replaces the semicolon (hex 094 in an EBCDIC environment, hex 059 in an ASCII environment) with a carriage return character and stores this string in a field named ANLBC. In the subfoot, this field displays on two lines.

The following report request is for an EBCDIC environment:

```

DEFINE FILE EMPLOYEE
ANLB/A40 ='THIS IS AN An FIELD;WITH A LINE BREAK.';
ANLBC/A40 = CTRAN(40, ANLB, 094, 013 , ANLBC);
END
TABLE FILE EMPLOYEE
PRINT FIRST_NAME
BY LAST_NAME
WHERE LAST_NAME EQ 'BLACKWOOD'
ON LAST_NAME SUBFOOT
" "
" <ANLBC "
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT,LINEBREAK='CR', $
ENDSTYLE
END

```

The following report request is for an ASCII environment:

```

DEFINE FILE EMPLOYEE
ANLB/A40 ='THIS IS AN An FIELD;WITH A LINE BREAK.';
ANLBC/A40 = CTRAN(40, ANLB, 059, 013 , ANLBC);
END
TABLE FILE EMPLOYEE
PRINT FIRST_NAME
BY LAST_NAME
WHERE LAST_NAME EQ 'BLACKWOOD'
ON LAST_NAME SUBFOOT
" "
" <ANLBC "
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT,LINEBREAK='CR', $
ENDSTYLE
END

```

The output is:

<u>LAST_NAME</u>	<u>FIRST_NAME</u>
BLACKWOOD	ROSEMARIE

THIS IS AN An FIELD  
WITH A LINE BREAK.



## Creating a Graph

---

Graphs often convey meaning more clearly than data listed in tabular format. Using the GRAPH command, you can easily transform almost any type of data into an effective graph that you can customize to suit your needs.

You can link your graph to other resources, or feature conditional styling to highlight specific data in your graph. You may also select from a multitude of graph styles, which include the standard graph formats bar, line, pie, and scatter as well as many variations on these types.

You can also represent data graphically using data visualization. For details, see [Displaying Report Data](#) on page 43.

### In this chapter:

- ☐ [Content Analysis: Determining Graphing Objectives](#)
  - ☐ [The GRAPH Command](#)
  - ☐ [Creating an HTML5 Graph](#)
  - ☐ [Selecting a Graph Type](#)
  - ☐ [Selecting Values for the X and Y Axes](#)
  - ☐ [Creating Multiple Graphs](#)
  - ☐ [Plotting Dates in Graphs](#)
  - ☐ [Refining the Data Set For Your Graph](#)
  - ☐ [Displaying Missing Data Values in a Graph](#)
  - ☐ [Applying Conditional Styling to a Graph](#)
  - ☐ [Linking Graphs to Other Resources](#)
  - ☐ [Adding Labels to a Graph](#)
  - ☐ [Applying Custom Styling to a Graph](#)
  - ☐ [Saving a Graph as an Image File](#)
  - ☐ [Printing a Graph](#)
- 

### Content Analysis: Determining Graphing Objectives

WebFOCUS offers a range of reporting tools that allow you to create reports that deliver critical information to your users. By selecting a tool that is well suited to your particular needs, you can design the information you deliver to users. One effective option with almost any type of data is a graphic presentation.

Graphs allow you to display multivariate or complex data efficiently, precisely, and in a way that a viewer can intuitively grasp. A graph is an effective presentation tool because it presents a visual idea, communicating meaningful changes in data to a user in a memorable way. By viewing your graph, a user can identify and track a change that you want them to notice.

Creating a meaningful graph is not simply a matter of applying aesthetics to your data. Instead, graphs allow you to design your presentation to capture the essential information in your data.

The first step in creating excellent graphics is determining your graphing objectives. You can break this process into several stages.

1. Assess your data:

Look for meaningful patterns or changes in the data. Does your data change most dramatically over time or in relationship to some other value? Are there two sets of data that you would like to compare to each other?

Determine what movement or changes you would like to highlight. Which of the patterns in the data would you most want the viewer to picture?

2. Select the graph type that best suits your argument and the overall shape of your data. Determine what will lead viewers to the cognitive task or connection that you want them to make.

3. Begin developing your graph.

4. Refine your graph:

Are the labels meaningful or useful?

How can the data be organized in a meaningful way? Consider customizing the scales you use with your graph.

## The GRAPH Command

GRAPH request syntax is similar to TABLE request syntax. To produce a graph instead of a tabular report, you need only substitute the command GRAPH for TABLE in the request. Thus, you can produce graphs by simply converting TABLE requests to GRAPH requests.

However, not every TABLE facility has a GRAPH counterpart, and there are some practical limitations on the amount of information that you can effectively display in a graph. When a TABLE request is converted in this manner, the various phrases that make up the body of the request take on special meanings that determine the format and layout of the graph. The type of graph produced by a GRAPH request depends on the display command used (SUM or PRINT), and the sort phrase(s) used (ACROSS or BY).

## Similarities Between GRAPH and TABLE

The GRAPH request elements generally follow the same rules as their TABLE counterparts:

- ❑ The word FILE and the file name must immediately follow the GRAPH command, unless they were previously specified in a SET command:

```
SET FILE=filename
```

You can specify any file available to WebFOCUS, including joined or cross-referenced structures.

- ❑ You can concatenate unlike data source files in a GRAPH request with the MORE command. For details, see [Universal Concatenation](#) on page 1093.
- ❑ The order of the phrases in the request does not affect the format of the graph. For example, the selection phrase may follow or precede the display command and sort phrase(s). The order of sort phrases does affect the format of a graph, just as the order of sort phrases in TABLE requests affects the appearance of reports.
- ❑ The word END must be on a line by itself to complete a request.
- ❑ All dates are displayed in MDY format unless they are changed to alphanumeric fields.

## Differences Between GRAPH and TABLE

There are a few notable syntactical differences between TABLE and GRAPH. Specifically, the following restrictions apply:

- ❑ A GRAPH request must contain a display command with at least one display field and at least one sort phrase (BY or ACROSS) in order to generate a meaningful graph.
- ❑ In GRAPH requests the object of the display command must always be a numeric field.
- ❑ No more than five display fields are permitted in a GRAPH request. Standard graph formats generally do not permit more variables to be displayed without rendering the graph unreadable.
- ❑ Several BY phrases can be used in a request, in which case multiple graphs are created. A single ACROSS phrase is allowed in a GRAPH request, and requests for certain graph forms can contain both ACROSS and BY phrases.
- ❑ The number of ACROSS values cannot exceed 64.
- ❑ The RUN option is not available as an alternative to END.

**Example:**    **Converting a TABLE Request to a GRAPH Request**

The following illustrates how a TABLE request can easily be converted into a GRAPH request by changing the TABLE command to a GRAPH command.

```
TABLE FILE GGORDER
HEADING CENTER
"SAMPLE TABLE"
SUM QUANTITY
BY PRODUCT_DESC AS 'Coffee Types'
WHERE PRODUCT_DESC EQ 'French Roast' OR 'Hazelnut' OR 'Kona'
END
```

The output is:

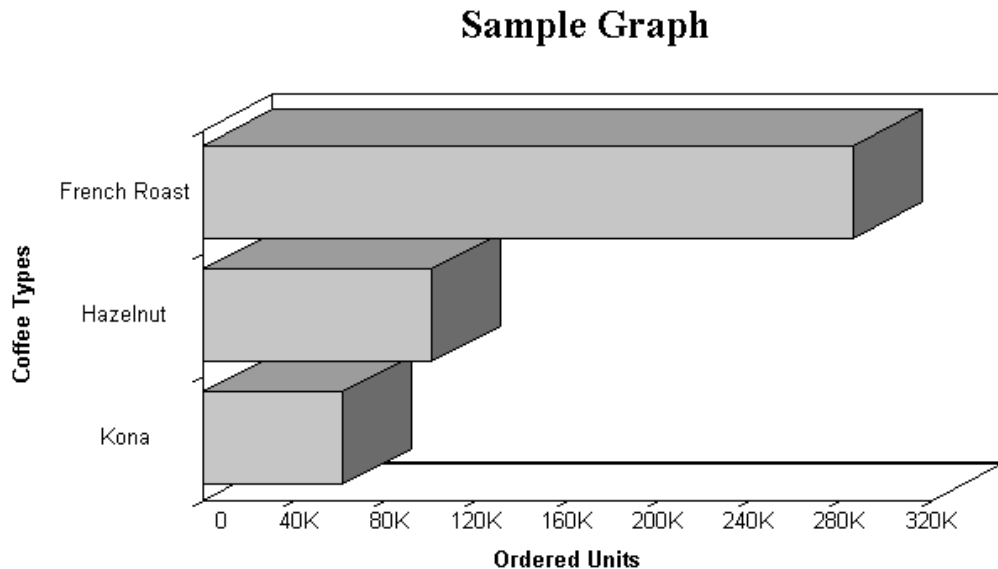
SAMPLE TABLE	
Coffee Types	Ordered Units
-----	-----
French Roast	285689
Hazelnut	100427
Kona	61498

The same request with a GRAPH command in place of the TABLE command is:

```
GRAPH FILE GGORDER
HEADING CENTER
"Sample Graph"
SUM QUANTITY
BY PRODUCT_DESC AS 'Coffee Types'
WHERE PRODUCT_DESC EQ 'French Roast' OR 'Hazelnut' OR 'Kona'
END
```



The output is:



## Creating an HTML5 Graph

WebFOCUS supports a graph output format that takes advantage of the HTML5 standard. The charts are rendered in the browser as high quality interactive vector graphics using a built-in JavaScript engine. Note that older browsers do not support all of the features of the HTML5 standard.

You can use the SET AUTOFIT command to make the HTML5 graph output resize to fit into the container in which it is placed.

### **Syntax:** How to Create HTML5 Graph Output

In your graph request, include the following commands

```
ON GRAPH PCHOLD FORMAT JSCHART
```

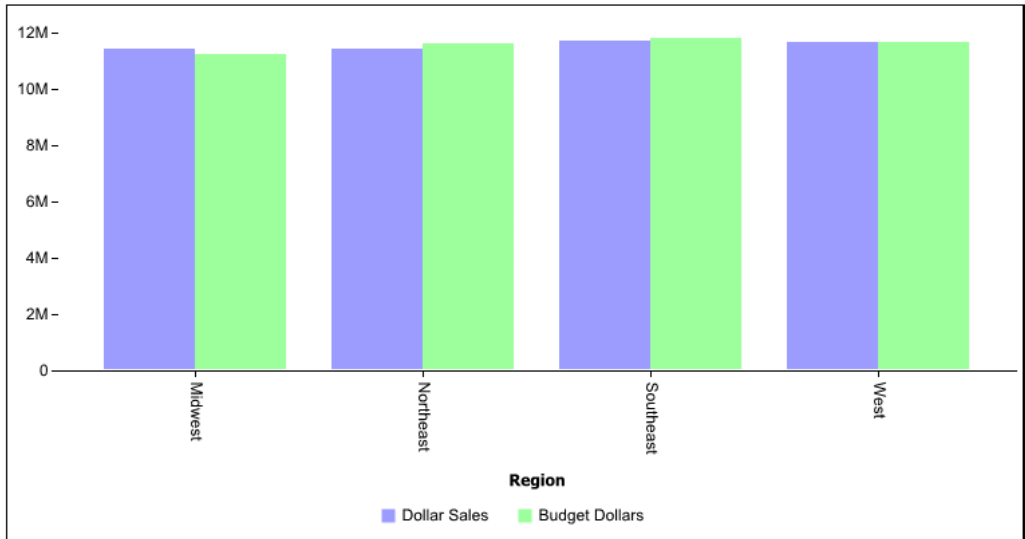
If the ON GRAPH PCHOLD FORMAT JSCHART command is not issued, server-side graphics are generated.

### **Example:** Creating an HTML5 Vertical Bar Graph

The following request against the GGSALES data source creates an HTML5 vertical bar graph:

```
GRAPH FILE GGSales
SUM DOLLARS BUDDOLLARS
BY REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
END
```

The output is:



### **Syntax:** How to Resize HTML5 Graph Output to Fit Its Container

```
ON GRAPH SET AUTOFIT {OFF|ON|RESIZE}
```

where:

OFF

Respects the dimensions specified by the HAXIS and VAXIS parameters.

ON

Always resizes the HTML5 graph output to fit its container.

RESIZE

Respects the dimensions specified by the HAXIS and VAXIS parameters initially, but resizes the graph output if the container is resized.

## Selecting a Graph Type

When creating a graph, it is important to select the appropriate graph type with which to display your data. You may select from a number of basic graph types, as well as refinements on these types. Basic graph types include line graphs (connected point plots), bar graphs, pie graphs, and scatter graphs. Use the brief descriptions (see [Graph Types](#) on page 1663) to select a graph type that suits the data set you are displaying and the change you want to highlight. Keep in mind that the data are the sets of numbers that you are displaying, and the scales are the numbers or variable measures displayed along the axes of the graph.

**Note:** When using a stacked chart of any type at least 2 series are required.

## Graph Types

Following are descriptions of the types of graphs you can create:

- ❑ **Line graphs.** Line graphs are useful for emphasizing the movement or trend of numerical data over time, since they allow a viewer to trace the evolution of a particular point by working backwards or interpolating. Highs and lows, rapid or slow movement, or a tendency towards stability are all types of trends that are well suited to a line graph.

Line graphs can also be plotted with two or more scales to suggest a comparison of the same value, or set of values, in different time periods. The number of scales your graph has depends on the type of graph you select. For details, see [Determining Graph Styles Using LOOKGRAPH](#) on page 1672.

- ❑ **Bar graphs.** A bar graph plots numerical data by displaying rectangular blocks against a scale. The length of a bar corresponds to a value or amount. Viewers can develop a clear mental image of comparisons among data series by distinguishing the relative heights of the bars. Use a bar graph to display numerical data when you want to present distributions of data. You can create horizontal as well as vertical bar graphs.
- ❑ **Pie graphs.** A pie graph emphasizes where your data fits in relation to a larger whole. Keep in mind that pie graphs work best when your data consists of several large sets. Too many variables divide the pie into small segments that are difficult to see. Use color or texture on individual segments to create visual contrast.
- ❑ **Scatter graphs.** Scatter graphs share many of the characteristics of basic line graphs. Data can be plotted using variable scales on both axes. When you use a scatter graph, your data is plotted using a basic line pattern. Use a scatter graph to visualize the density of individual data values around particular points or to demonstrate patterns in your data. A numeric X-axis, or sort field, will always yield a scatter graph by default.

It is important to note that scatter graphs and line graphs are distinguishable from one another only by virtue of their X-axis format. Line graphs can appear without connecting lines (making them look like scatter graphs) and scatter graphs can appear with connecting lines (making them look like line graphs).

- ☐ **Area graphs.** Area graphs are similar to line graphs except that the area between the data line and the zero line (or axis) is usually colored or textured. Area graphs allow you to stack data on top of each other. Stacking allows you to highlight the relationship between data series, showing how some data series approach or shadow a second series.
- ☐ **3D graphs.** 3D graphs add dimension to your graphing presentation. Dimensionality allows your viewers to recognize trends based on two or more data sets easily. 3D graphs also add impact to your presentation.
- ☐ **Bipolar graphs.** A bipolar graph is split along a horizontal line. This type of graph is useful for comparative trend analysis of widely disparate data values over time or other sort values.
- ☐ **Radar graphs.** This is a type of circular graph used when categories are cyclical. Radar graphs are essentially analogous to a line chart, except that the scale wraps around. Radar graphs work well with any data that are cyclical, such as the months of a year.

## Selecting Scales

After you have chosen a graph type, you should select an appropriate scale. A scale is a classification scheme or series of measures that you select for application to the axes of your graph. The scale provides the framework against which your data are plotted. When you choose an appropriate scale for your data, meaningful patterns can emerge, and when you modify a scale, the overall shape of your graph changes.

Steps or measures in the scale are represented along the axes of your graph by marks. The type of scale you choose determines the number of divisions along the scale. There are two general types of scales you can apply to the y-axis of your graph:

- ☐ Linear scales
- ☐ Logarithmic scales

A linear scale is a scale in which the values increase arithmetically. Each measure along the scale is one unit higher than the one that precedes it. Linear scales are useful when the data you are plotting are relatively small in range.

A logarithmic scale is a scale in which the values increase logarithmically. Each measure along the scale represents an exponential increase in the data value. Logarithmic scales are useful when you need to accommodate a large range of numbers.

**Syntax:**      **How to Select Scales**

To use logarithmic scales in your graph, add the following to your GRAPH request:

```
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
setYlLogScale( value );
*END
ENDSTYLE
END
```

where:

*value*

- Is one of the following:
- true* turns on logarithmic scaling.
- false* turns off logarithmic scaling. Linear scaling is used instead.

**Determining Graph Styles With Display Commands and Sort Phrases**

Each GRAPH request must include a sort phrase and at least one display field (up to five are allowed).

The fields, which are the subjects of the graph, may be real or virtual fields, with or without direct operation prefixes (AVE., MIN., MAX., etc.). They may also be calculated values.

**Note:** Display fields used only for calculations need not appear in the graph. You can use the NOPRINT or SUP-PRINT phrases to suppress the display of such fields. For details, see [Sorting Tabular Reports](#) on page 91.

By default, a particular combination of display commands and sort phrases determines the graph format. The combinations are:

Graph Type	Display Command and Sort Phrase
Line graph	<code>PRINT A {ACROSS BY} B</code> (where B is alphanumeric)

Graph Type	Display Command and Sort Phrase
Vertical bar graph	<code>SUM A ACROSS B</code>  (where B is alphanumeric)
Horizontal bar graph	<code>SUM A BY B</code>
Pie graph	<code>SET LOOKGRAPH=PIE</code> <code>SUM A {ACROSS BY} B</code>  or  <code>SET PIE=ON</code> <code>SUM A {ACROSS BY} B</code>
Scatter graph (without connecting lines)	<code>PRINT A ACROSS B</code>  (where B is numeric)
Scatter graph (with connecting lines)	<code>SUM A ACROSS B</code>  (where B is numeric)

You can override the default graph format using the LOOKGRAPH parameter. For details, see [Determining Graph Styles Using LOOKGRAPH](#) on page 1672.

**Syntax:**      **How to Create a Line Graph**

To create a line graph, issue a GRAPH request with the following display command and sort field combination

```
PRINT fieldname1 [AND] fieldname2...  
{ACROSS|BY} sortfield
```

where:

*fieldname1...*

Is the name of the field to be displayed on the Y-axis of the graph. There can be a maximum of 5 display fields in a GRAPH request.

AND

Is an optional phrase used to enhance readability. It can be used between any two field names and does not affect the graph.

*sortfield*

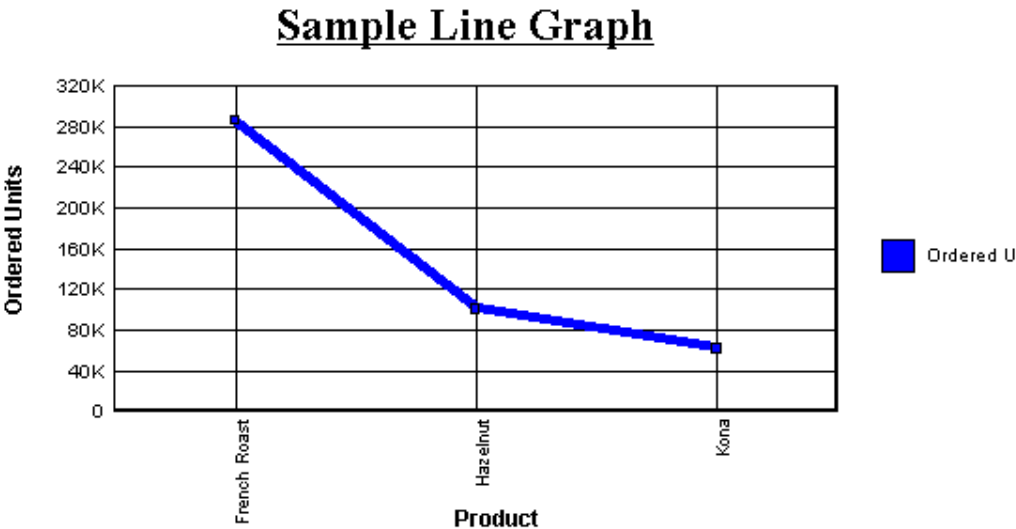
Is the name of the field to be displayed on the X-axis of the graph. This must be an alphanumeric field in order to generate a line graph. If the field specified is numeric, you can still create a line graph by using the LOOKGRAPH=LINE parameter. For details, see [Determining Graph Styles Using LOOKGRAPH](#) on page 1672.

**Example: Creating a Line Graph**

The following illustrates how to create a line graph using the LOOKGRAPH command:

```
SET LOOKGRAPH = LINE, GRID=ON
SET HAXIS=600, VAXIS=315
GRAPH FILE GGORDER
HEADING CENTER
"Sample Line Graph"
SUM QUANTITY
ACROSS PRODUCT_DESC
WHERE PRODUCT_DESC EQ 'French Roast' OR 'Hazelnut' OR 'Kona'
END
```

The output is:



**Syntax:**      **How to Create a Horizontal Bar Graph**

To create a horizontal bar graph, issue a GRAPH request with the following display command and sort field combination

```
SUM fieldname1 [AND] fieldname2...  
BY sortfield
```

where:

*fieldname1...*

Is the name of a field to be displayed on the Y-axis of the graph. There can be a maximum of 5 display fields in a GRAPH request.

AND

Is an optional phrase used to enhance readability. It can be used between any two field names and does not affect the graph.

*sortfield*

Is the name of a field to be displayed on the X-axis of the graph. A separate group of bars is created for each value of the BY field, and each group contains one bar for each display command (SUM) object.

**Syntax:**      **How to Create a Vertical Bar Graph**

To create a vertical bar graph, issue a GRAPH request with the following display command and sort field combination

```
SUM fieldname1 [AND] fieldname2...  
ACROSS sortfield
```

where:

*fieldname1...*

Is the name of the field to be displayed on the Y-axis of the graph. There can be a maximum of 5 display fields in a GRAPH request.

AND

Is an optional phrase used to enhance readability. It can be used between any two field names and does not affect the graph.

*sortfield*

Is the name of an alphanumeric field to be displayed on the X-axis of the graph.

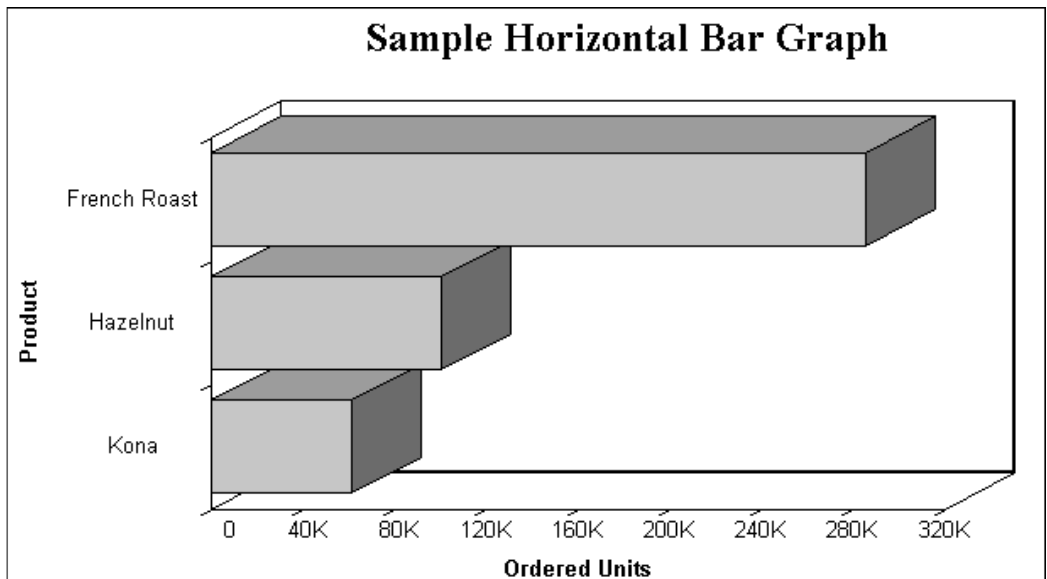


**Example: Creating a Horizontal Bar Graph**

The following illustrates how to create a horizontal bar graph:

```
GRAPH FILE GGORDER
HEADING CENTER
"Sample Horizontal Bar Graph"
SUM QUANTITY
BY PRODUCT_DESC
WHERE PRODUCT_DESC EQ 'French Roast' OR 'Hazelnut' OR 'Kona'
END
```

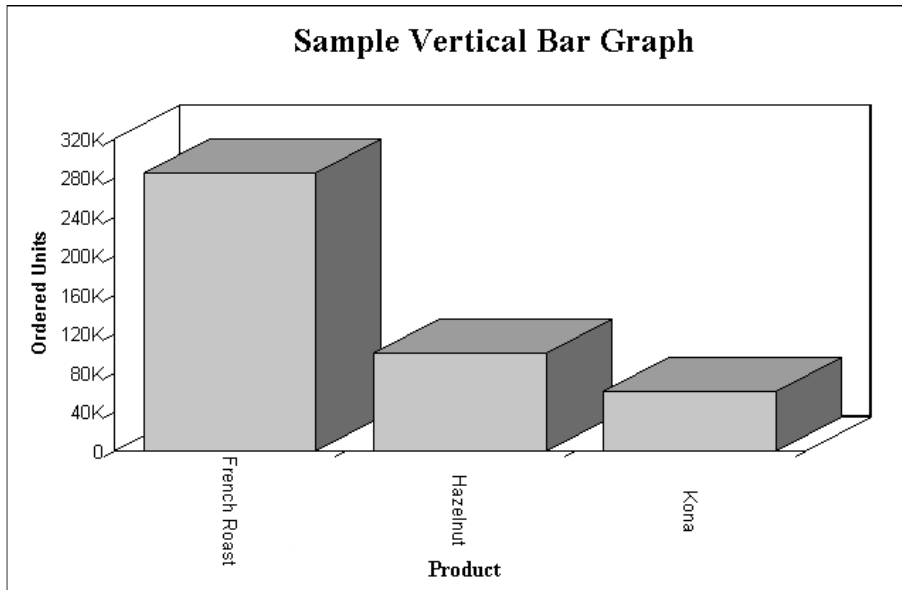
The output is:

**Example: Creating a Vertical Bar Graph**

The following illustrates how to create a vertical bar graph:

```
GRAPH FILE GGORDER
HEADING CENTER
"SAMPLE VERTICAL BAR GRAPH"
SUM QUANTITY
ACROSS PRODUCT_DESC
WHERE PRODUCT_DESC EQ 'French Roast' OR 'Hazelnut' OR 'Kona'
END
```

The output is:



### **Syntax:** How to Create a Pie Graph

To create a pie graph, issue a GRAPH request with the following SET command and display and sort field combination

```
SET LOOKGRAPH=PIE
SUM fieldname1 [AND] fieldname2...
{ACROSS|BY} sortfield
```

where:

*fieldname1...*

Is the name of the field to be displayed in the graph. There can be a maximum of 5 display fields in a GRAPH request.

AND

Is an optional phrase used to enhance readability. It can be used between any two field names and does not affect the graph.

*sortfield*

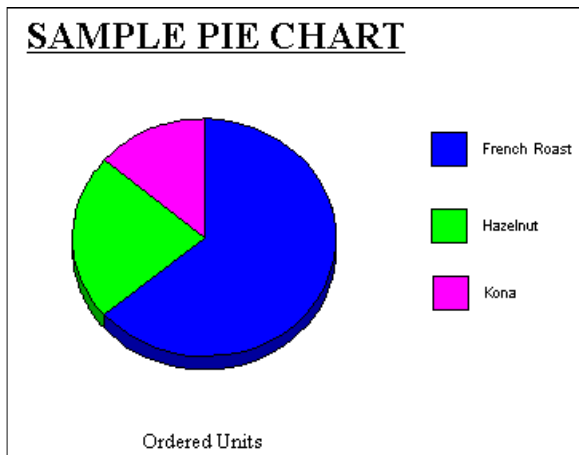
Is the name of the field to be displayed in the graph. Each value in the sort field will be represented by a section in the pie graph.

**Example: Creating a Pie Graph**

The following illustrates how to create a pie graph using a BY sort phrase and the LOOKGRAPH command:

```
SET LOOKGRAPH=PIE
GRAPH FILE GGORDER
HEADING CENTER
"SAMPLE PIE CHART"
SUM QUANTITY
BY PRODUCT_DESC AS COFFEES
WHERE PRODUCT_DESC EQ 'French Roast' OR 'Hazelnut' OR 'Kona'
END
```

The output is:

**Syntax: How to Create a Scatter Graph**

To create a scatter graph, issue a GRAPH request with the following display command and sort field combination

```
{PRINT|SUM} fieldname1 [AND] fieldname2...
ACROSS sortfield
```

where:

*fieldname*

Is the name of the field to be displayed in the graph. There can be a maximum of 5 display fields in a GRAPH request. When you specify more than one display field, they are represented by different symbols.

AND

Is an optional phrase used to enhance readability. It can be used between any two field names and does not affect the graph.

sortfield

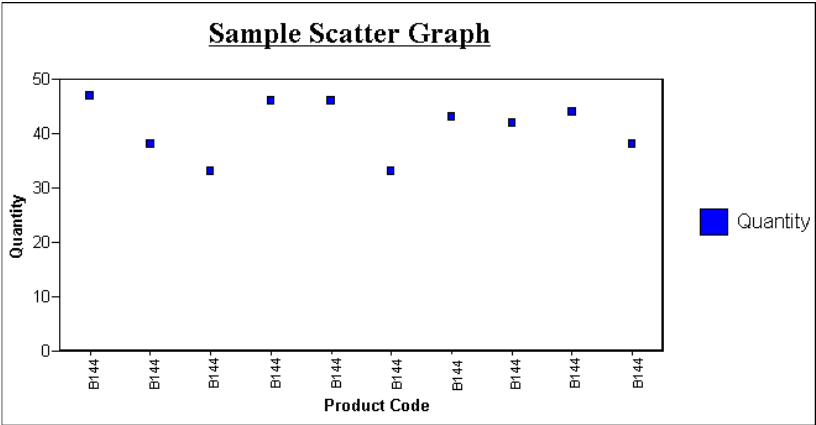
Is the name of the numeric field to be displayed on the X-axis of the graph.

Example: Creating a Scatter Graph

The following illustrates how to create a scatter graph:

```
GRAPH FILE GGORDER
HEADING CENTER
"Sample Scatter Graph"
PRINT QUANTITY AS 'Quantity'
ACROSS PRODUCT_CODE
WHERE PRODUCT_CODE EQ 'B144'
WHERE QUANTITY LT 51
END
```

The output is:



Determining Graph Styles Using LOOKGRAPH

By default, a particular combination of display commands and sort phrases determines the graph format. You can override the default graph format by using the LOOKGRAPH parameter.

The LOOKGRAPH parameter enables you to change the format of the graph without having to set individual control parameters or restructure the graph request. However, even if you use LOOKGRAPH, you can choose to set individual control parameters (for example, SET GRID=ON).

**Syntax:**      **How to Specify a Graph Style Using LOOKGRAPH**

`SET LOOKGRAPH= option`

where:

*option*

Specifies the graph style. For details on graph styles, see:

- ❑ [Style Options for Line Graphs](#) on page 1673.
- ❑ [Style Options for Bar Graphs](#) on page 1674.
- ❑ [Style Options for Pie Graphs](#) on page 1675.
- ❑ [Style Options for Scatter Graphs](#) on page 1676.
- ❑ [Style Options for Three-Dimensional Graphs](#) on page 1677.
- ❑ [Style Options for Area Graphs](#) on page 1678.
- ❑ [Style Options for Stock Charts](#) on page 1678.
- ❑ [Style Options for Polar Charts](#) on page 1680.
- ❑ [Style Options for Radar Charts](#) on page 1680.
- ❑ [Style Options for Bubble Charts](#) on page 1681.
- ❑ [Style Options for Spectral Charts](#) on page 1681.
- ❑ [Other Graph Types](#) on page 1681.
- ❑ [Options for HTML5-Only Chart Types](#) on page 1682.

**Reference:**    **Style Options for Line Graphs**

Choose one of the following LOOKGRAPH values to change the style of connected point plots:

SET LOOKGRAPH=	Description
LINE	A vertical connected point plot graph.
HLINE	A horizontal connected point plot graph.

SET LOOKGRAPH=	Description
HLINE2	A horizontal connected point plot graph with two axes.
HLINE2S	A horizontal connected point plot graph with two separate axes.
HLINSTK	A stacked horizontal connected point plot graph.
HLINSTK2	A stacked horizontal connected point plot graph with two axes.
HLNSTK2S	A stacked horizontal connected point plot graph with two separate axes.
HLNSTKPC	A stacked horizontal connected point plot graph showing percentages.
VLINE	A vertical connected point plot graph.
VLINE2	A vertical connected point plot graph with two axes.
VLINE2S	A vertical connected point plot graph with two separate axes.
VLINSTK	A stacked vertical connected point plot graph.
VLINSTK2	A stacked vertical connected point plot graph with two axes.
VLNSTK2S	A stacked vertical connected point plot graph with two separate axes.
VLNSTKPC	A stacked vertical connected point plot graph showing percentages.

**Reference: Style Options for Bar Graphs**

Choose one of the following LOOKGRAPH values to change the style of bar graphs:

SET LOOKGRAPH=	Description
BAR	A bar graph with the bars displayed beside each other.

SET LOOKGRAPH=	Description
STACK	A bar graph with stacked bars.
VBAR	A vertical bar graph.
VBAR2AX	A vertical bar graph with two axes.
VBAR2AXS	A vertical bar graph with two separate axes.
VBRSTK1	A stacked vertical bar graph.
VBRSTK2	A stacked vertical bar graph with two axes.
VBRSTK2S	A stacked vertical bar graph with two separate axes.
VBRSTKPC	A stacked vertical bar graph that shows percentages.
HBAR	A horizontal bar graph.
HBAR2AX	A horizontal bar graph with two axes.
HBAR2AXS	A horizontal bar graph with two separate axes.
HBRSTK1	A stacked horizontal bar graph.
HBRSTK2	A stacked horizontal bar graph with two axes.
HBRSTK2S	A stacked horizontal bar graph with two separate axes.
HBRSTKPC	A stacked horizontal bar graph that shows percentages.

**Reference: Style Options for Pie Graphs**

Choose one of the following LOOKGRAPH values to change the style of pie graphs:

SET LOOKGRAPH=	Description
PIE	A pie graph.

SET LOOKGRAPH=	Description
PIESINGL	A single pie graph.
PIEMULTI	Multiple pie graphs.
PIERING	A ring-shaped pie graph.
PIEMULPR	Multiple, ring-shaped pie graphs of proportional size.
PIEMULTP	Multiple pie graphs of proportional size.
PIEMULTR	Multiple, ring-shaped pie graphs.

### **Reference:** Style Options for Scatter Graphs

Choose one of the following LOOKGRAPH values to change the style of scatter graphs:

SET LOOKGRAPH=	Description
SCATTER	Produces a scatter graph.
SCATTERD	A dual scatter graph. Values from an additional data set are displayed on a second value (Y) axis.
SCATTRLs	A scatter graph that labels each data point with its exact numeric value.
SCATTRLd	A dual scatter graph that labels each data point with its exact numeric value.



**Reference: Style Options for Three-Dimensional Graphs**

Choose one of the following LOOKGRAPH values to change the style of three-dimensional graphs:

SET LOOKGRAPH=	Description
3DAREAG	A three-dimensional connected group area chart.
3DAREAS	A three-dimensional connected series area chart.
3DBAR	A two-dimensional bar graph with three-dimensional bars.
3D_BAR	A three-dimensional chart with bars.
3DCUBE	A three-dimensional bar graph in which all data points are blocks of identical size, hovering at the position that shows their data value.
3DGROUP	A three-dimensional chart with bars.
3DOCTAGN	A three-dimensional bar graph with octagon-shaped bars that have no roots.
3DPYRAMD	A three-dimensional pyramid chart.
3DRIBBNG	A three-dimensional connected group ribbon chart.
3DRIBBNS	A three-dimensional connected series ribbon chart.
3DSPHERE	A three-dimensional bar graph in which all data points are spheres of identical size, hovering at the position that shows their data value.
3DSTACK	A two-dimensional stack chart with three-dimensional type bars.
3DSURFCE	A three-dimensional surface chart that graphs all data points as a three-dimensional surface, like a rolling wave.
3DSURFHC	A three-dimensional honeycomb surface chart that graphs all data points as a three-dimensional surface using a honeycomb effect.

SET LOOKGRAPH=	Description
3DSURFSD	A three-dimensional surface chart with sides that graphs all data points as a three-dimensional surface with solid sides.

**Reference: Style Options for Area Graphs**

Choose one of the following LOOKGRAPH values to change the style of area graphs:

SET LOOKGRAPH=	Description
VAREA	A vertical area graph.
VAREASTK	A stacked vertical area graph.
VAREAR2	A vertical area graph with two axes.
VARESTK2	A stacked vertical area graph with two axes.
VARESTKP	A stacked vertical area graph that shows percentages.
HAREA	A horizontal area graph.
HAREAR2	A horizontal area graph with two axes.
HAREASTK	A stacked horizontal area graph.
HARESTK2	A stacked horizontal area graph with two axes.
HARESTKP	A stacked horizontal area graph that shows percentages.

**Reference: Style Options for Stock Charts**

Choose one of the following LOOKGRAPH values to change the style of stock charts:

SET LOOKGRAPH=	Description
STOCK	A stock chart.

SET LOOKGRAPH=	Description
STOCKH	<p>A high-low stock chart. Bars representing higher numeric values are placed behind bars representing lower numeric values. Only the top of the higher bar is visible, clearly illustrating the difference in value.</p> <p>The most popular application of this type of graph is to represent stock prices. Each bar represents the highest and lowest prices for a given stock on a given day.</p>
STOCKHB	A bipolar high-low stock chart. Values from different data sets are displayed on separate poles.
STOCKHD	A dual high-low stock chart. Values from an additional data set are displayed on a second value (Y) axis.
STOCKHCL	A high-low-close stock chart. The most popular application of this type of graph is to represent stock prices. Each bar represents the highest, lowest, and closing prices for a given stock on a given day.
STOCKHCB	<p>A bipolar high-low-close stock chart. Values from different data sets are displayed on separate poles.</p> <p>The most popular application of this type of graph is to represent stock prices. Each bar represents the highest, lowest, and closing prices for a given stock on a given day.</p>
STOCKHCD	A dual high-low-close stock chart. Values from an additional data set are displayed on a second value (Y) axis.
STOCKHOC	<p>A high-low-open-close stock chart.</p> <p>The most popular application of this type of graph is to represent stock prices. Each bar represents the highest, lowest, opening, and closing prices for a given stock on a given day.</p>
STOCKHOB	A bipolar high-low-open-close stock chart. Values from different data sets are displayed on separate poles.
STOCKHOD	A dual high-low-open-close stock chart. Values from an additional data set are displayed on a second value (Y) axis.

SET LOOKGRAPH=	Description
<a href="#">STOCKHV</a>	A high-low-volume stock chart.
<a href="#">STOCKHOV</a>	A high-low-open-close-volume stock chart.
<a href="#">STOCKC</a>	A candle stock chart.
<a href="#">STOCKHC</a>	A high-low candle stock chart.
<a href="#">STOCKCV</a>	A volume candle stock chart.
<a href="#">STOCKHCV</a>	A high-low-volume candle stock chart.

**Reference: Style Options for Polar Charts**

Choose one of the following LOOKGRAPH values to change the style of polar charts:

SET LOOKGRAPH=	Description
<a href="#">POLAR</a>	A polar chart that displays data points on a circle.
<a href="#">POLAR2</a>	A dual polar chart. Values from an additional data set are displayed on a second value (Y) axis.

**Reference: Style Options for Radar Charts**

Choose one of the following LOOKGRAPH values to change the style of radar charts:

SET LOOKGRAPH=	Description
<a href="#">RADARA</a>	A radar area chart.
<a href="#">RADARL</a>	A radar line chart.
<a href="#">RADARL2</a>	A dual radar line chart. Values from an additional data set are displayed on a second value (Y) axis.

**Reference: Style Options for Bubble Charts**

Choose one of the following LOOKGRAPH values to change the style of bubble charts:

SET LOOKGRAPH=	Description
BUBBLE	A bubble chart.
BUBBLED	A bubble chart with a dual axis.
BUBBLEDL	A bubble chart with a dual axis and labels.
BUBBLEL	A bubble chart with labels.

**Reference: Style Options for Spectral Charts**

Choose one of the following LOOKGRAPH values to change the style of spectral charts:

SET LOOKGRAPH=	Description
SPECTRAL	A spectral map chart. This is a chart with a row or column matrix of markers that is colored according to the data values.

**Reference: Other Graph Types**

SET LOOKGRAPH=	Description
GANTT	Provides a visual representation of project oriented time critical events. Gantt charts require six display fields and one sort field, in that order. Conditional styling and drill-down are not supported for GANTT charts.
POSITION	Product position charts provide a visual representation of market share and growth versus revenue and measurement (past, present, future). Product position charts require a set of three display fields.
VWATERFL	Vertical waterfall graph.

SET LOOKGRAPH=	Description
<a href="#">HWATERFL</a>	Horizontal waterfall graph.
<a href="#">PARETO</a>	Displays data following Pareto 80:20 rule. Pareto charts require only one display field.
<a href="#">MULTI3Y</a> <a href="#">MULTI4Y</a> <a href="#">MULTI5Y</a>	Stacks charts in order to make it easier to read, analyze and manage them.

**Reference: Options for HTML5-Only Chart Types**

The following LOOKGRAPH values are valid only when generating an HTML5 chart:

SET LOOKGRAPH	Description
<a href="#">BUBBLEMAP</a>	A bubblemap is a chart in which proportionally sized bubbles are displayed on relevant areas of the map.
<a href="#">CHOROPLETH</a>	a choropleth is a chart in which areas on a map are shaded or patterned in proportion to the value of the measure being represented,
<a href="#">MEKKO</a>	A Mekko chart is a variant of a stacked bar chart, in which the width of the bars is adjusted relative to its value in the data set.
<a href="#">PARABOX</a>	A Parabox (or parallel coordinates chart) is similar to a regular line chart, except that each group in the line chart has a unique and interactive numeric axis. Each line represents one series of data. Each vertical bar represents a numeric axis. You can click and drag along each of the axes to select (filter) the lines that pass through that part of the axis
<a href="#">STREAM</a>	A streamgraph is a simplified version of a stacked area chart. In a streamgraph, there are no axes or gridlines. The baseline is free, which makes it easier to perceive the thickness of any given layer across the data.

SET LOOKGRAPH	Description
<a href="#">TAGCLOUD</a>	A tagcloud is a visual representation of frequency. It displays only group labels. The size of each label is proportional to its data value.
<a href="#">TREEMAP</a>	A treemap chart displays hierarchical data as a set of nested rectangles.

## Selecting Values for the X and Y Axes

The values you select for the X- and Y-axes determine what data is included in the graph you are creating, and how it appears.

The X-axis value is determined by the sort phrase (BY or ACROSS) used in your GRAPH request. At least one sort phrase is required in every GRAPH request. When there are multiple BY phrases or when an ACROSS and BY phrase are included in the same request, multiple graphs are generated, one for each combination of values for the fields referenced in the request. For details, see [Creating Multiple Graphs](#) on page 1685.

The Y-axis value is determined from the display field associated with the display command (SUM or PRINT) issued in your GRAPH request.

You can also:

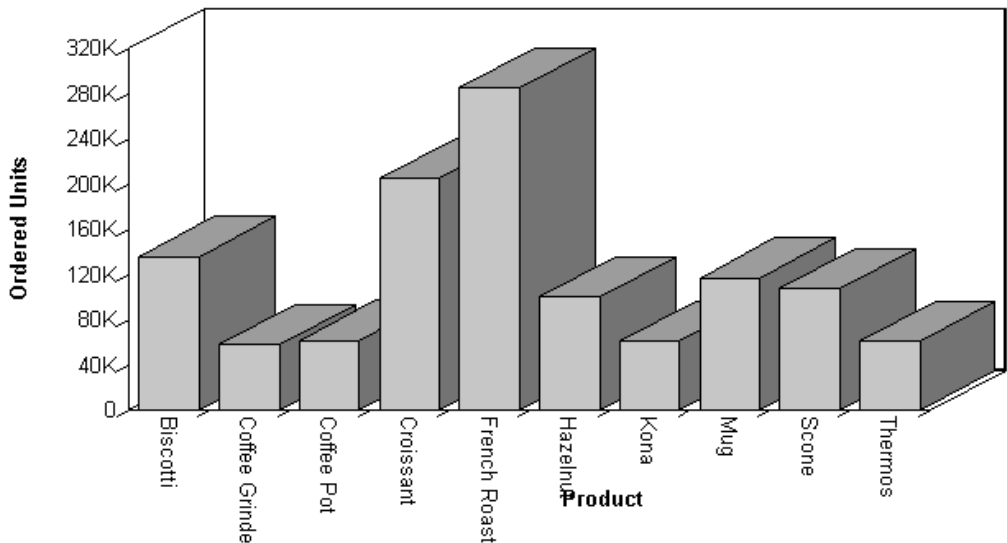
- ☐ Select a second horizontal category (X axis), which will produce multiple graphs. For details, see [Creating Multiple Graphs](#) on page 1685.
- ☐ Temporarily hide the display of a Y-axis field. For details, see [Hiding the Display of a Y-Axis Field](#) on page 1684.
- ☐ Interpolate X and Y axis values using linear regression. For details, see [Interpolating X and Y Axis Values Using Linear Regression](#) on page 1684.

### **Example:** Selecting Values for the X and Y Axes

The following illustrates how to set the X-axis (PRODUCT\_DESC) using an ACROSS phrase and the Y-axis (QUANTITY) with the display command SUM:

```
GRAPH FILE GGORDER
SUM QUANTITY AS 'Ordered Units'
ACROSS PRODUCT_DESC
END
```

The output is:



### Hiding the Display of a Y-Axis Field

You can hide the display of a Y-axis field in a graph. This option is useful when you want to temporarily remove a particular Y-axis field while retaining all of the original graph properties.

To temporarily hide the display of a Y-axis field, add the NOPRINT command to the field. Although the NOPRINT command applies to both verb objects and sort fields in a TABLE request, it only applies to verb objects in a GRAPH request.

### Interpolating X and Y Axis Values Using Linear Regression

You can interpolate X and Y axis values using basic linear regression. Basic linear regression involves the average of the summation of X and Y axis values to determine a linear equation that expresses the trend of the scatter diagram. Use the SET parameter GTREND to turn on basic linear regression in your graph.

GTREND is only available for use with scatter charts.

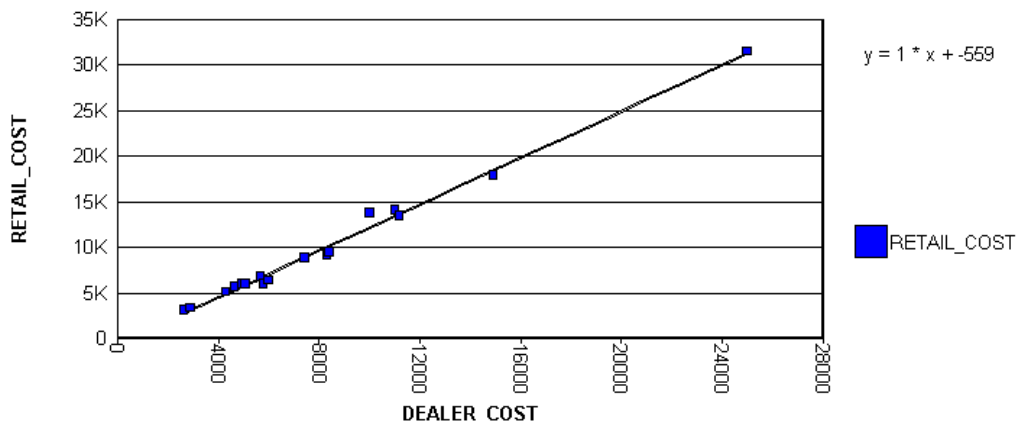


**Example: Interpolating X and Y Axis Values Using Linear Regression**

The following illustrates how to turn on linear regression in a scatter chart.

```
SET 3D=OFF
GRAPH FILE CAR
PRINT RC
ACROSS DC
ON GRAPH SET LOOKGRAPH SCATTER
ON GRAPH SET GTREND ON
END
```

The output is:

**Creating Multiple Graphs**

You can create multiple graphs by including secondary sort dimensions (fields).

By default, the number of graphs created depends on the number of values in the fields you designate in the sort (BY, ACROSS) phrases. You can change this default using the GRMERGE parameter:

- ☐ With GRMERGE OFF (the default), if a request contains two BY fields, there will be as many graphs as there are values in the first BY field. The second BY field will determine the X-axis. For example, if you have selected a BY field with two values, two graphs will be generated. If you have selected a field with ten values, ten graphs will be generated. If there is one BY phrase and one ACROSS phrase, as many graphs will display as there are values in the BY field. The ACROSS field will determine the X-axis. You can select the second horizontal category by including multiple BY phrases or an ACROSS and BY phrase in the same request.

- ☐ With GRMERGE ON, WebFOCUS creates one merged graph.
- ☐ With GRMERGE ADVANCED, WebFOCUS uses three parameters to determine:
  - ☐ How many graphs to generate (GRMULTIGRAPH).
  - ☐ How many sort fields should be placed on the graph legend (GRLEGEND).
  - ☐ How many sort fields should be placed on the X-axis (GRXAXIS).

Multiple graphs can be displayed in either merged format or in columns. For details, see [Merging Multiple Graphs](#) on page 1686 and [Displaying Multiple Graphs in Columns](#) on page 1690.

### Merging Multiple Graphs

By default, when you create a graph that has multiple BY fields, or a BY and ACROSS field, multiple graphs are generated. You can merge these graphs into a single graph or into multiple merged graphs.

To do this, use the SET command GRMERGE.

#### **Syntax:** How to Merge Multiple Graphs

`SET GRMERGE={ON | OFF | ADVANCED}`

where:

[ON](#)

Turns on the merge graph option.

[OFF](#)

Turns off the merge graph option. This is the default.

[ADVANCED](#)

Turns on the advanced merge option. This option uses three parameters to determine how to merge the graphs:

- ☐ GRMULTIGRAPH, which specifies how many sort fields to use to create multiple graphs.
- ☐ GRLEGEND, which specifies how many sort fields to place on the graph legend.
- ☐ GRXAXIS, which specifies how many sort fields to display on the X-axis. GRXAXIS must be at least 1 in order to plot the graph. A value greater than one creates nested X-axes.

**Note:** The sum of the sort fields used by GRMULTIGRAPH, GRLEGEND, and GRXAXIS must equal the number of sort fields in the graph request.

The syntax for the GRMULTIGRAPH, GRLEGEND, and GRXAXIS parameters is:

`ON GRAPHSET GRMULTIGRAPH n`

Specifies how many sort fields (0 through 2) to use to break the output into multiple graphs. The outermost sort fields are used to separate the graphs. When *n* is greater than zero, this is similar to GRMERGE=OFF, but allows an additional sort field.

`ON GRAPH SET GRLEGEND n`

Specifies how many of the remaining outermost sort fields (0 through 2), after the ones used for GRMULTIGRAPH, to add to the graph legend. When *n* is greater than zero, this is similar to GRMERGE=ON, but allows an additional sort field.

`ON GRAPH SET GRXAXIS n`

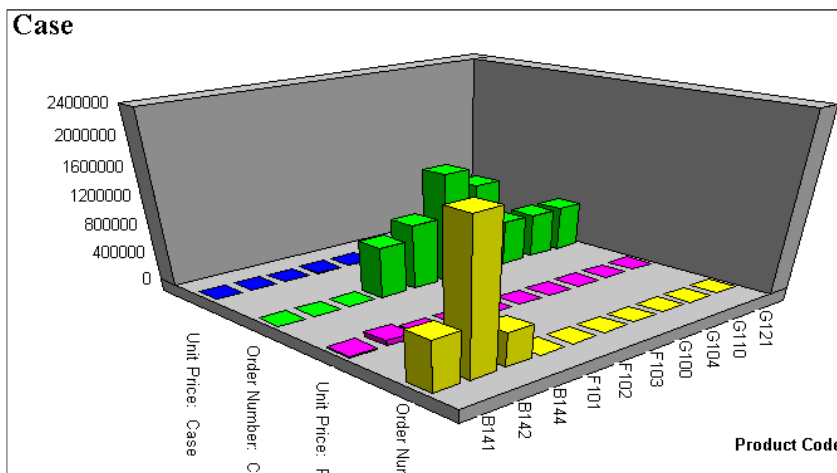
Specifies how many of the remaining sort fields (1 through 3) to display on the X-axis. When *n* is greater than 1, this creates nested X-axes.

### **Example:** Merging Multiple Graphs With GRMERGE ON

The following illustrates a graph with two horizontal, or X-axes, categories (PRODUCT\_ID and PACKAGE\_TYPE) that have been merged.

```
SET GRMERGE=ON
GRAPH FILE GGORDER
SUM UNIT_PRICE ORDER_NUMBER
ACROSS PRODUCT_ID
BY PACKAGE_TYPE
END
```

The output is:

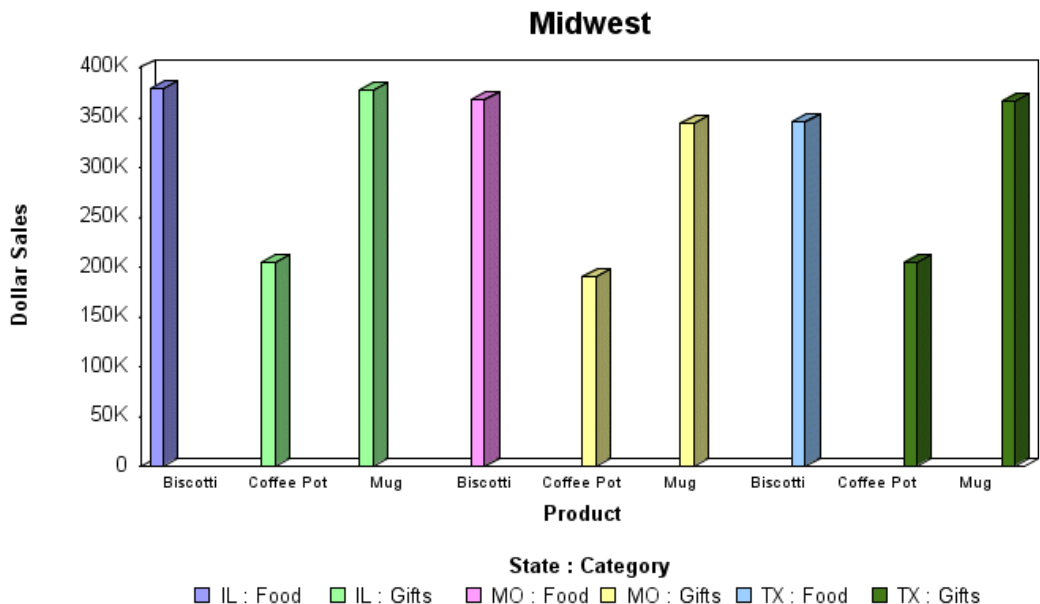


### Example: Merging Multiple Graphs With GRMERGE ADVANCED

The following example generates a vertical bar graph that separates the outermost sort field (REGION) onto separate graphs, distinguishes the next two sort fields (ST and CATEGORY) by combining them on the graph legend, and places the CATEGORY sort field on the X-axis:

```
GRAPH FILE GGSales
SUM DOLLARS
BY REGION BY ST BY CATEGORY BY PRODUCT
WHERE CATEGORY EQ 'Food' OR 'Gifts'
WHERE PRODUCT EQ 'Coffee Pot' OR 'Biscotti' OR 'Mug'
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 2
ON GRAPH SET GRXAXIS 1
ON GRAPH SET LOOKGRAPH VBAR
END
```

The first graph is for region Midwest. The legend distinguishes State-Category combinations by color, and the PRODUCT sort field is repeated on the X-axis for each State-Category combination:



### Merging Multiple OLAP Graphs

When you create an OLAP graph that has multiple BY fields, or a BY and ACROSS field, multiple graphs are generated. You can merge these graphs into a single graph.

To do this, use the SET command OLAPGRMERGE.

**Syntax:**      **How to Merge Multiple OLAP Graphs**

```
SET OLAPGRMERGE={ON | OFF}
```

where:

ON

Turns on the merge graph option. With this setting AUTODRILL is disabled for the graph.

OFF

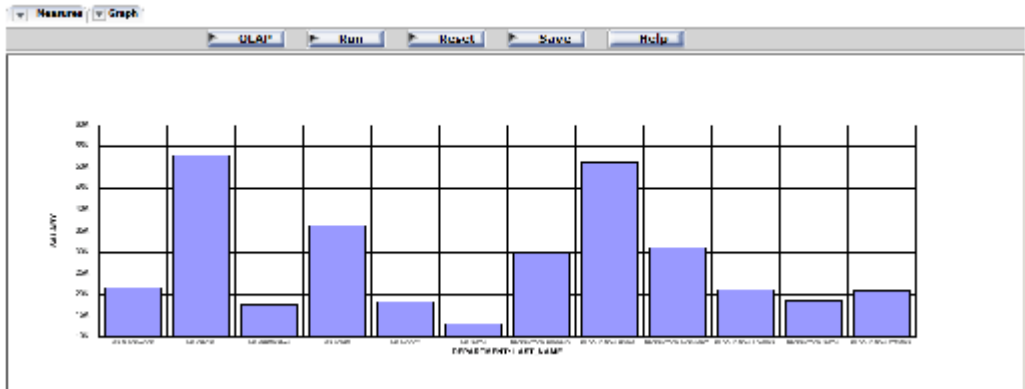
Turns off the merge graph option and creates a separate graph for every value of the outer sort field. OFF is the default value.

**Example:**      **Merging OLAP-Enabled Graphs**

The following OLAP request against the EMPLOYEE data source has two BY fields. To merge the graphs, the SET OLAPGRMERGE=ON command is issued:

```
-OLAP ON
SET GRAPHEDIT=SERVER
SET OLAPGRMERGE=ON
TABLE FILE EMPLOYEE
SUM SALARY
BY DEP
BY LAST_NAME
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLCSS ON
ON GRAPH SET HAXIS 300
ON GRAPH SET VAXIS 100
ON TABLE SET AUTODRILL ALL
ON TABLE SET OLAPPANE TABBED
ON TABLE SET STYLE *
    INCLUDE = endeflt,
$
    LEFTMARGIN=0.500000,
    RIGHTMARGIN=0.500000,
    TOPMARGIN=0.500000,
    BOTTOMMARGIN=0.500000,
$
TYPE=REPORT,
    TOPGAP=0.000000,
    BOTTOMGAP=0.013889,
$
ENDSTYLE
END
```

The output is:



### Displaying Multiple Graphs in Columns

When you create a graph that has multiple BY fields, or a BY and ACROSS field, multiple graphs are generated. You can display these graphs in columns.

To do this, use the SET command GRWIDTH. GRWIDTH may be set to any value between 0-512. The default is 0.

#### **Syntax:** How to Display Multiple Graphs in Columns

`SET GRWIDTH=nn`

where:

*nn*

Is the number of columns in which to display multiple graphs. This may be any value from 0-512. The default is 0.

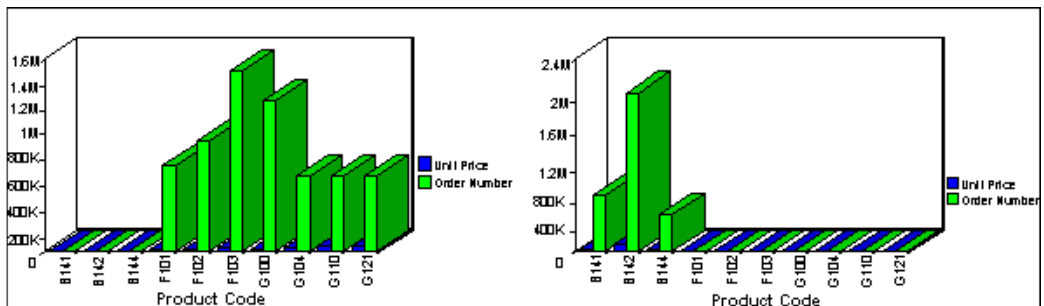
All values from 1-512 will display graphs in an HTML table with the corresponding number of columns. The default value of 0 will display the graphs one under the other in a Java applet.

**Example: Displaying Multiple Graphs in Columns**

The following illustrates how to set the number of columns in which you wish to display multiple graphs. In this example, the graphs are set to display in two columns.

```
SET GRWIDTH=2
GRAPH FILE GGORDER
SUM UNIT_PRICE ORDER_NUMBER
ACROSS PRODUCT_ID
BY PACKAGE_TYPE
END
```

The output is:

**Plotting Dates in Graphs**

Numeric fields containing dates are recognized by the field formats specified in the Master File. Such fields can be used in ACROSS or BY phrases in GRAPH requests. To review the various format types, refer to the *Describing Data With WebFOCUS Language* manual.

Plotted dates are handled in the following manner:

- ❑ If the date field named has a month-first format, it is plotted in ascending time order (even if the file is not sorted in ascending date order). Hence, month/year values of 01/76, 03/76, 09/75 will be plotted by month within year: 09/75, 01/76, 03/76.
- ❑ Axis scaling is performed on the basis of days in the month and months in the year. When the date format includes the day, the scale usually starts at the first day of the month, at the "zero" axis point.

You can selectively combine groups of date point plots to reduce the number of separate points on the horizontal axis. You do this with the IN-GROUPS-OF option. For example, if the date field format is I6YMD, you can display the data by month rather than by day by grouping it in 30-day increments:

```
ACROSS DATE IN-GROUPS-OF 30
```

This eliminates plot points for individual days. If your date format is YMD, you can redefine the format and divide the field contents by 100 to eliminate the days:

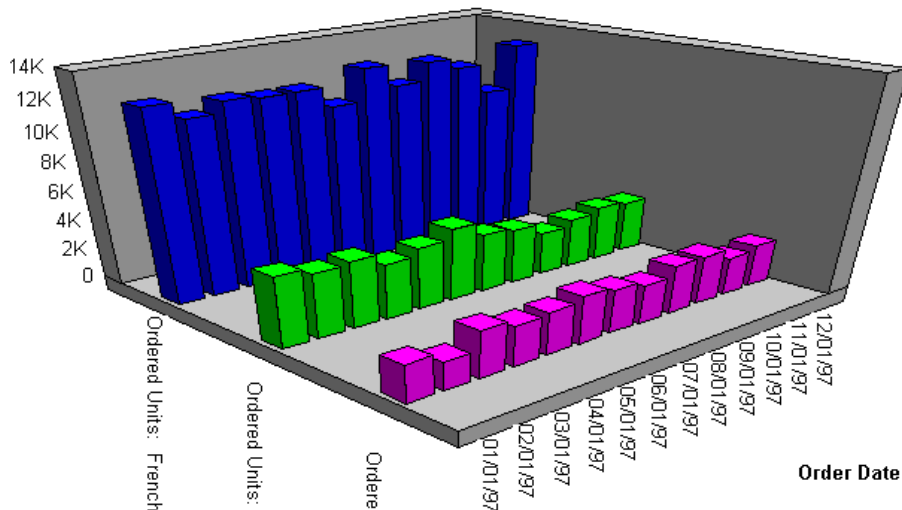
```
DATE/I4YM=DATE/100
```

### **Example:** Including Date Fields in a Graph

The following example illustrates how month-first formatted date fields are displayed in a graph.

```
SET GRMERGE = ON
GRAPH FILE GGORDER
SUM QUANTITY
ACROSS ORDER_DATE
BY PRODUCT_DESC
WHERE PRODUCT_DESC EQ 'French Roast' OR 'Hazelnut' OR 'Kona' AND
ORDER_DATE GE '010197'
END
```

The output is:



### Basic Date Support for X and Y Axes

OLDDATES can be manipulated accordingly with the usage of YRTHRESH and DEFCENT SET parameters. If you do not specify the YRTHRESH and DEFCENT commands for dates with the Y format (for example, YMD, MDY, DMY, YM, etc.), the code will assume the format 19XX.



**Reference: Date Support Limitations**

- ❑ The following date formats are supported:
  - ❑ SHORT (18) is completely numeric, such as 12/13/52 or 3:30pm.
  - ❑ MEDIUM (19) is longer, such as Jan 12, 1952.
  - ❑ LONG (20) is longer, such as January 12, 1952 or 3:30:32pm.
  - ❑ FULL (21) is almost completely specified, such as Tuesday, April 12, 1952 AD or 3:30:42pm PST.

The default date format for the X and Y axes is MEDIUM. You can overwrite the default by using one of the following API calls (where xx is one of the numbers listed above):

```
setTextFormatPreset(getXlLabel(),xx); // for X Axis
```

```
setTextFormatPreset(getYlLabel(),xx); // for Y Axis
```

The default date format for Data Text is LONG. This only applies to graphs with dates on the Y axis. Currently this format is not supported on the X axis. You can overwrite the default by using the following API call:

```
setDataTextFormat(xx);
```

For more information, see [Customizing Graphs Using the Graph API and HTML5 JSON Properties](#) on page 1719.

- ❑ In a graph with dates on the X axis and numeric fields on the Y axis, the tool tip displays the data format of the graph by default. This means that a date value will display its raw GMT value in milliseconds. This does not occur for dates on the Y axis and strings on the X axis, because the data format is already in date format.
- ❑ DATETIME is not fully supported.
- ❑ The ability to set the start and end dates for the appropriate axis is not supported.
- ❑ The ability to set the step for dates is not supported.

Formatting Dates for Y-Axis Values

You can display date-formatted numbers for Y-axis fields and on tool tips. The following date formats are supported:

Display Format	Corresponding WebFOCUS Format
yy/mm/dd	YMD
yy/mm	YM
mm/dd/yyyy	MDYY
mm/dd/yy	MDY
mm/dd	MD

For complete details on date formats, see the *Describing Data With WebFOCUS Language* manual.

Refining the Data Set For Your Graph

After selecting field values for the X and Y axes, you may wish to limit the data that displays in your graph. You can do this by creating WHERE statements. A WHERE statement limits data by creating parameters the data must satisfy before it is included in the data set.

For details on WHERE, WHERE TOTAL, and IF phrases, see [Selecting Records for Your Report](#) on page 219.

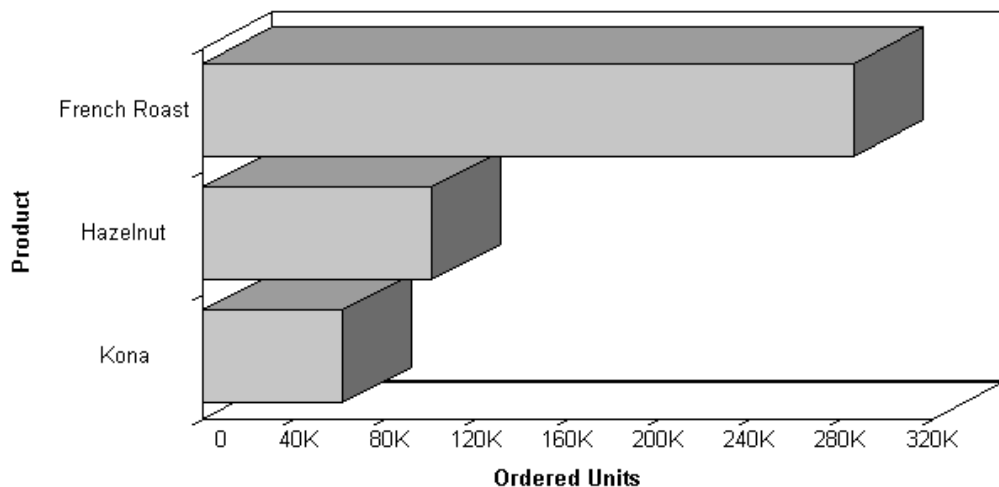
**Example: Specifying WHERE Criteria in a Graph Request**

The syntax for WHERE, WHERE TOTAL, and IF phrases in a GRAPH request is identical to that used in a TABLE request.

The following graph request shows data for specific product descriptions, namely French Roast, Hazelnut, or Kona.

```
GRAPH FILE GGORDER
SUM QUANTITY
BY PRODUCT_DESC
WHERE PRODUCT_DESC EQ 'French Roast' OR 'Hazelnut' OR 'Kona'
END
```

The output is:

**Displaying Missing Data Values in a Graph**

You can display missing data values (in a bar graph, line graph, area graph, or any variation of these graph types) in one of the following formats:

- ☐ **Graph as zero.** In bar graphs, a bar appears on the zero line. In line graphs, a solid line connects the missing value with the succeeding value. In area graphs, the area appears on the zero line.
- ☐ **Graph as gap.** In all graph types (bar, line, or area), missing values appear as a gap in the graph.

- ❑ **Dotted line to zero.** In line graphs, a dotted line connects the missing value with the succeeding value. In 3D bar graphs, solid lines outline the flat bar corresponding to the missing value. In 2D bar graphs, a gap appears in the graph. In area graphs, a transparent area extends down to the zero line and then up to the succeeding value.
- ❑ **Interpolated dotted line.** In a line graph, missing values appear as an interpolated dotted line that connects the plot points immediately preceding and succeeding the missing value. In bar and area graphs, missing values display as an interpolated (transparent) bar or area.

**Note:** You can specify a default value (other than the default value of zero) to represent missing data. To do this use a DEFINE command. For details, see [Handling Records With Missing Field Values](#) on page 971.

### **Syntax:** How to Display Missing Values in a Graph

```
GRAPH FILE filename
.
.
.
SET VZERO={ON|OFF}
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
API call
*END
ENDSTYLE
END
```

where:

**ON**

Displays missing values as zero. An API call is not necessary when VZERO is set to ON. Alternatively, you can add ON GRAPH SET VZERO ON.

**OFF**

Displays missing values as a gap, a dotted line to zero, or, an interpolated dotted line, depending on the API call that is added. Alternatively, you can add ON GRAPH SET VZERO OFF.

*API call*

Determines how missing values display in the graph when VZERO is set to OFF. Possible values are:

`setFillMissingData(0);` displays missing values as a gap.

`setFillMissingData(1);` displays missing values as a dotted line to zero.

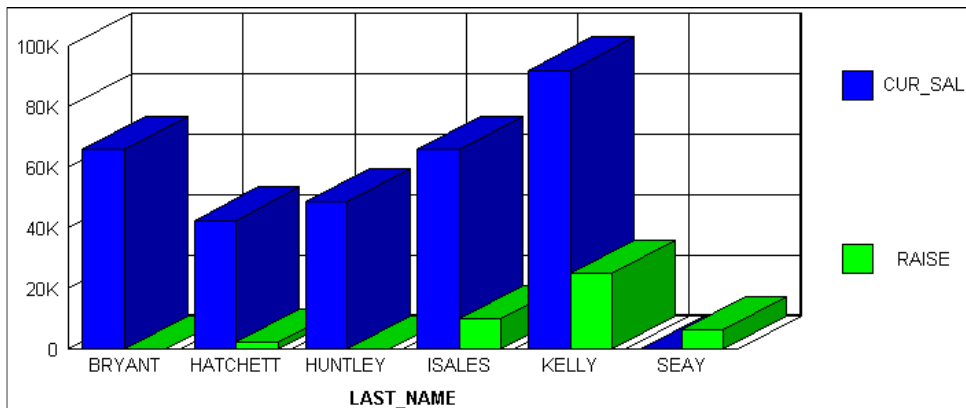
`setFillMissingData(2);` displays missing values as an interpolated dotted line.

**Example:** **Displaying Missing Values as Zero In a Graph**

The following illustrates how missing values are represented in a bar graph when designated to appear as zero. The CURR\_SAL value for Seay is missing, as well as the RAISE value for Bryant and Huntley.

```
SET LOOKGRAPH=BAR
SET GRAPHEDIT=SERVER
SET GRID=ON
SET VZERO=ON
GRAPH FILE MSFATIA
SUM CUR_SAL
RAISE
ACROSS LAST_NAME
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
setTextRotation(getOlLabel(),0);
*END
ENDSTYLE
END
```

The output is:

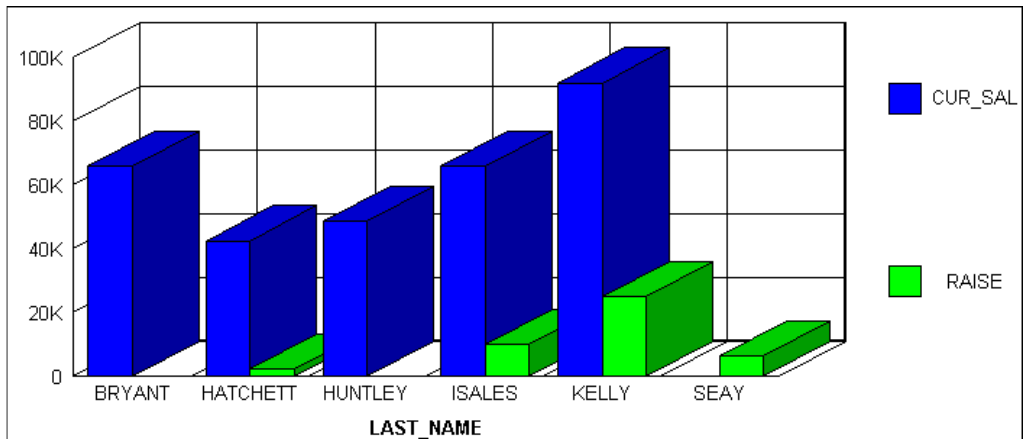


**Example:**    **Displaying Missing Values as a Gap**

The following illustrates how missing values are represented in a bar graph when designated to appear as a gap. The CURR\_SAL value for Seay is missing, as well as the RAISE value for Bryant and Huntley.

```
SET LOOKGRAPH=BAR
SET GRAPHEDIT=SERVER
SET GRID=ON
SET VZERO=OFF
GRAPH FILE MSFATIA
SUM CUR_SAL
RAISE
ACROSS LAST_NAME
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
setFillMissingData(0);
setTextRotation(getOlLabel(),0);
*END
ENDSTYLE
END
```

The output is:

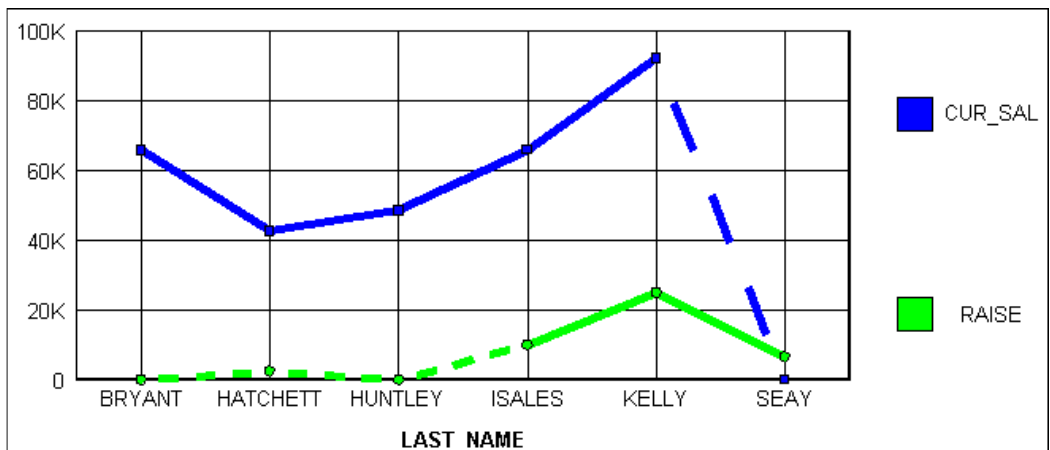


**Example: Displaying Missing Values as a Dotted Line to Zero**

The following illustrates how missing values are represented in a line graph when designated to appear as a dotted line to zero. The CURR\_SAL value for Seay is missing, as well as the RAISE value for Bryant and Huntley.

```
SET LOOKGRAPH=LINE
SET GRAPHEDIT=SERVER
SET GRID=ON
SET VZERO=OFF
GRAPH FILE MSFATIA
SUM CUR_SAL
RAISE
ACROSS LAST_NAME
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
setFillMissingData(1);
setTextRotation(getOlLabel(),0);
*END
ENDSTYLE
END
```

The output is:

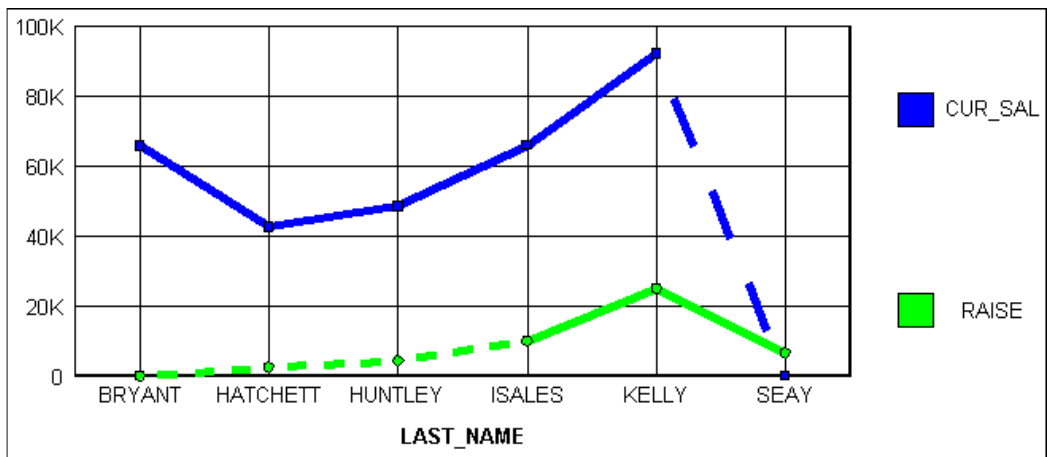


**Example: Displaying Missing Values as an Interpolated Dotted Line**

The following illustrates how missing values are represented in a line graph when designated to appear as an interpolated dotted line. The CURR\_SAL value for Seay is missing, as well as the RAISE value for Bryant and Huntley.

```
SET LOOKGRAPH=LINE
SET GRAPHEDIT=SERVER
SET GRID=ON
SET VZERO=OFF
GRAPH FILE MSFATIA
SUM CUR_SAL
RAISE
ACROSS LAST_NAME
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
setFillMissingData(2);
setTextRotation(getOlLabel(),0);
*END
ENDSTYLE
END
```

The output is:



**Applying Conditional Styling to a Graph**

You can add further value to your graph by using conditional styling to highlight certain defined data with specific styles and colors.



For example, you can apply the color red to all departments that did not reach their sales quotas and apply the color black to all departments that did reach their sales quotas. In this example, the user can view quickly which departments did or did not reach their quotas. To examine how the results of one department may impact the results of a second department, you may want to provide a drill-down to a report that examines this possibility.

You can apply color to the following graph types:

- ☐ Bar graphs.
- ☐ Three-dimensional graphs with noncontinuous plot points.
- ☐ Pie graphs.
- ☐ Stack charts.

You can apply conditional styling using StyleSheets.

**Note:** Conditional styling is only supported for Y-axis values.

### **Syntax:** How to Apply Conditional Styling to a Graph

```
TYPE=DATA, [ COLUMN|ACROSSCOLUMN=Nn, ]COLOR=color, [WHEN=expression, ]$
```

where:

**DATA**

Identifies data as the graph component to which color is being applied. This value must appear at the beginning of the declaration.

**COLUMN|ACROSSCOLUMN**

Is the graph subcomponent to which you want to apply color. Valid graph subcomponents are COLUMN and ACROSSCOLUMN.

**Nn**

Identifies a column by its position in the report. To determine this value, count BY fields, display fields, and ROW-TOTAL fields, from left to right, including NOPRINT fields. For details, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

*color*

Identifies the color that you want to apply to the graph component or subcomponent. For a list of valid colors, see [Formatting Report Data](#) on page 1611.

### *expression*

Is any Boolean expression that specifies conditions for applying the specified color to the graph component. The expression must be valid on the right side of a COMPUTE command. For details, see [Using Expressions](#) on page 431.

**Note:** IF... THEN ... ELSE logic is not necessary in a WHEN clause and is not supported.

All non-numeric literals in a WHEN expression must be specified within single quotation marks.

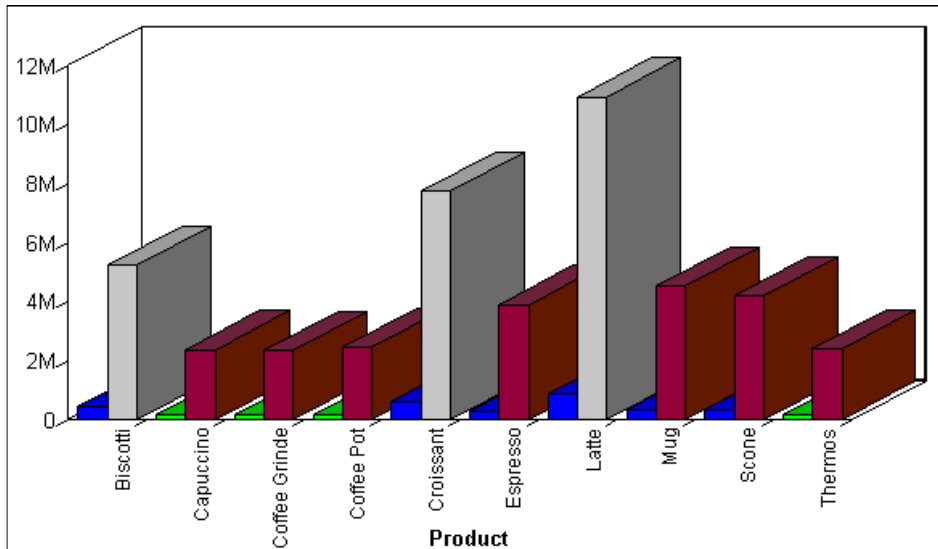
### **Example:** Applying Conditional Styling to a Graph

The following illustrates how you can apply conditional styling to a graph.

```
GRAPH FILE GGSales
SUM UNITS DOLLARS ACROSS PRODUCT
ON GRAPH SET STYLE *
1. TYPE=DATA,COLOR=BLUE,$
2. TYPE=DATA,ACROSSCOLUMN=N2,COLOR=FIREBRICK,$
3. TYPE=DATA,ACROSSCOLUMN=N2,COLOR=SILVER,WHEN=N2 GT 5000000,$
4. TYPE=DATA,ACROSSCOLUMN=N1,COLOR=LIME,WHEN=N1 LT 200000,$
ENDSTYLE
END
```

1. This line specifies blue as the default data color.
2. This line specifies firebrick as the default color for the DOLLARS column.
3. This line specifies silver as the color for the DOLLARS column when the value for DOLLARS is greater than five million.
4. This line specifies lime as the UNITS column color when the number of UNITS is less than two hundred thousand.

The output is:



## Linking Graphs to Other Resources

To drill down to a more detailed level of information in a graph, you can link a procedure (FOCEXEC) or a URL to one or more values in your graph. When you run your graph, the selected values become "hot spots" that invoke the underlying procedure, JavaScript function, or URL.

The JSURLS parameter includes JavaScript or VBScript files in an HTML graph. This allows you to customize the display of WebFOCUS HTML graphs with any JavaScript or VBScript functions. The JavaScript and VBScript files are the last files loaded, and are loaded in the order they are listed, allowing complete customization of the HTML page.

This feature works with any graph format that outputs an HTML document, for example JSCHART and PCHOLD FORMAT PNG.

In addition, when a WebFOCUS graph is run, a set of pre-defined JavaScript functions is invoked. Using JSURLS, you can disable or modify these default functions. To view the full set of pre-defined JavaScript functions, see [/ibi/WebFOCUSxx/ibi\\_apps/ibi\\_html/javaassist/ibi/html/js/ibigl.js](/ibi/WebFOCUSxx/ibi_apps/ibi_html/javaassist/ibi/html/js/ibigl.js).

The syntax is:

```
SET JSURLS='/file1 [/file2] [/file3]...'
```

where:

*/file1 [/file2] [/file3]...*

Are the files that contain JavaScript or VBScript. If there is more than one js file, the delimiter is a blank and the values must be enclosed in single quotes. Files must be in a location that is accessible by the web server. The total length of the value is limited to 256 bytes.

You can reference files with a URL.

### **Syntax:** How to Link a Graph to Another Request

```
TYPE=DATA, [ COLUMN|ACROSSCOLUMN=Nn, ] COLOR=color, [ WHEN=expression, ]  
FOCEXEC=fex(parameters ...), ]$
```

where:

**DATA**

Identifies Data as the graph component to which the user is applying the color. The TYPE attribute and its value must appear at the beginning of the declaration.

**COLUMN|ACROSSCOLUMN**

Is the graph subcomponent to which you want to apply color. Valid graph subcomponents are COLUMN and ACROSSCOLUMN.

*color*

Identifies the color that you want to apply to the graph component or subcomponent. For a list of valid colors, see [Formatting Report Data](#) on page 1611.

**Nn**

Identifies a column by its position in the report. To determine this value, count BY fields, display fields, and ROW-TOTAL fields, from left to right, including NOPRINT fields. For more information, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

**FOCEXEC=*fex***

Identifies the file name of the linked procedure to run when a user selects the report object.

*parameters*

Are values to be passed to the procedure. You can pass one or more values, using any combination of the following methods:

- ❑ You can specify a constant value, enclosed in single quotation marks.

- ❑ You can specify the name or the position of a graph column.
- ❑ You can specify the name of a Dialogue Manager amper variable to pass its value.
- ❑ You can use amper variables only in inline StyleSheets.

**Note:** The usual use of an amper variable is to pass a constant value, in which case, it would have to be embedded in single quotation marks. For example:

```
'&ABC' .
```

The method you can use to pass values can vary, depending on the method you use to execute the hyperlink. You can pass one or more values. The entire string of values must be enclosed in parentheses, and separated from each other by a blank space.

#### *expression*

Is any Boolean expression that specifies conditions for applying the specified color to the graph component. The expression must be valid on the right side of a COMPUTE command. For details, see [Using Expressions](#) on page 431.

**Note:** IF ... THEN ... ELSE logic is not necessary in a WHEN clause and is not supported.

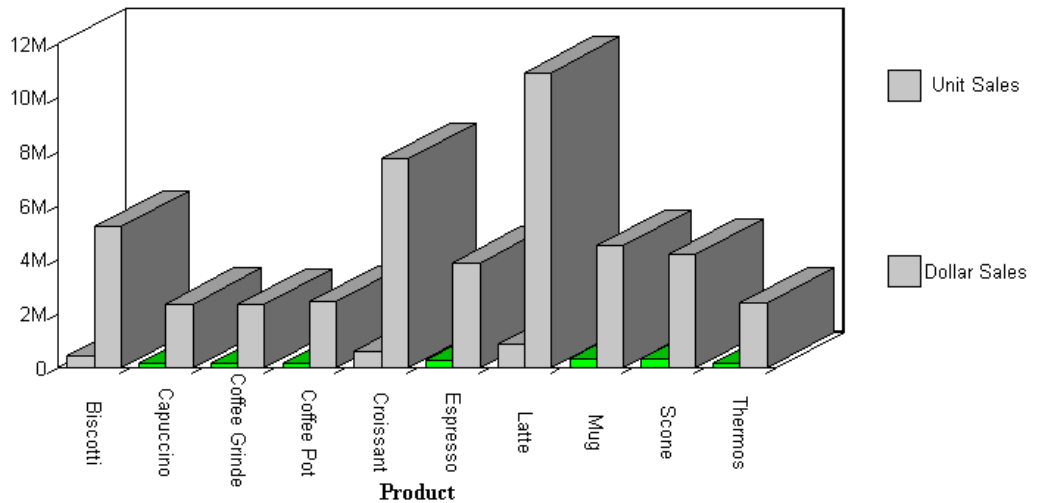
All non-numeric literals in a WHEN expression must be specified within single quotation marks.

### **Example:** Linking to Additional Reports or Graphs

In this example, when the value for UNITS is less than four hundred thousand, the color is lime and you can drill-down to a detail report.

```
GRAPH FILE GGSales
SUM UNITS DOLLARS ACROSS PRODUCT
ON GRAPH SET STYLE *
TYPE=DATA,COLOR=SILVER,$
TYPE=DATA,ACROSSCOLUMN=N1,COLOR=LIME,WHEN=N1 LT 400000,FOCEXEC=GRAPH2,$
ENDSTYLE
END
```

The output is:



### **Syntax:** How to Link to a URL

You can define a link from any component to any URL including webpages, websites, Servlet programs, or non-World Wide Web resources, such as an email application. After you have defined a link, you can select the component to access the URL.

The links you create can be dynamic. With a dynamic link, your selection passes the value of the selected component to the URL. The resource uses the passed value to dynamically determine the results that are returned. You can pass one or more parameters. For details, see [Creating Parameters](#) on page 797.

```
TYPE=type, [subtype], URL=url[(parameters ...)], [TARGET=frame,] [ALT = 'description',] $
```

where:

#### *type*

Identifies the report or graph component that you select in the web browser to execute the link. The TYPE attribute and its value must appear at the beginning of the declaration.

#### *subtype*

Are any additional attributes, such as COLUMN, LINE, or ITEM, that are needed to identify the report component that you are formatting. For information on identifying components, see [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167.

*url*

Identifies any valid URL, including a URL that specifies a WebFOCUS Servlet program, or the name of a report column enclosed in parentheses whose value is a valid URL to which the link will jump.

**Note:**

- ❑ The maximum length of a URL=*url* argument, including any associated variable=object parameters, is limited by the maximum number of characters allowed by the browser. For information about this limit for your browser, search on your browser vendor's support site. The URL argument can span more than one line, as described in [Creating and Managing a WebFOCUS StyleSheet](#) on page 1117.

Note that the length of the URL is limited by the maximum number of characters allowed by the browser. For information about this limit for your browser, search on your browser vendor's support site.

- ❑ If the URL refers to a WebFOCUS Servlet program that takes parameters, the URL must end with a question mark (?).

*parameters*

Values that are passed to the URL. For details, see [Creating Parameters](#) on page 797.

*frame*

Identifies the target frame in the webpage in which the output from the drill-down link is displayed. For details, see [Specifying a Target Frame](#) on page 814.

*description*

Is a textual description of the link supported in an HTML report for compliance with Section 508 accessibility. Enclose the description in single quotation marks.

The description also displays as a pop-up description when your mouse or cursor hovers over the link in the report output.

**Syntax:****How to Link to a JavaScript Function**

You can use a StyleSheet to define a link to a JavaScript function from any report or graph component. After you have defined the link, you can select the component to execute the JavaScript function.

Just as with drill-down links to procedures and URLs, you can specify optional parameters that allow values of a component to be passed to the JavaScript function. The function will use the passed value to dynamically determine the results that are returned to the browser. For details, see [Creating Parameters](#) on page 797.

**Note:**

- ❑ JavaScript functions can, in turn, call other JavaScript functions.
- ❑ You cannot specify a target frame if you are executing a JavaScript function. However, the JavaScript function itself can specify a target frame for its results.

`TYPE=type, [subtype], JAVASCRIPT=function(parameters ...), $`

where:

*type*

Identifies the report component that you select in the web browser to execute the link. The TYPE attribute and its value must appear at the beginning of the declaration.

*subtype*

Are any additional attributes, such as COLUMN, LINE, or ITEM, that are needed to identify the report component that you are formatting. See [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167 for details.

*function*

Identifies the JavaScript function to run when you select the report component.

The maximum length of a JAVASCRIPT=function argument, including any associated parameters, is 2400 characters and can span more than one line. If you split a single argument across a line, you need to use the \ character at the end of the first line, as continuation syntax. If you split an argument at a point where a space is required as a delimiter, the space must be before the \ character or be the first character on the next line. The \ character does not act as the delimiter.

In this example,

```
JAVASCRIPT=myfunc(COUNTRY \
CAR MODEL 'ABC'),$
```

the argument correctly spans two lines.

**Note:**

- ❑ You can use the SET JSURLS command or the Dialogue Manager -HTMLFORM command to embed the chart into an HTML document in which the function is defined.
- ❑ When you have an HTML document called by -HTMLFORM, ensure that the file extension is .HTM (not .HTML).

For more information about the -HTMLFORM command, see the *Developing Reporting Applications* manual.



*parameters*

Values that are passed to the JavaScript function. For details, see [Creating Parameters](#) on page 797.

**Syntax:**     **How to Create Multiple Drill-Down Links**

```
TYPE=type, [subtype], DRILLMENUITEM='description'|'DrillDown n',  
          type_of_link
```

where:

*type*

Identifies the component that you select in the web browser to execute the link. The TYPE attribute and its value must appear at the beginning of the declaration.

*subtype*

Are any additional attributes, such as COLUMN, LINE, or ITEM, that are needed to identify the component that you are formatting. See [Identifying a Report Component in a WebFOCUS StyleSheet](#) on page 1167 for information on identifying report components.

*description*

Is the text that appears on the pop-up menu of drill-down options on the output. The default value is DrillDown *n*, where *n* is a consecutive integer, such as DrillDown 1, DrillDown 2, and so on.

*type\_of\_link*

Is the type of link, for example, a link to a detail report or URL. For a summary of valid values, see [Summary of Drill-Down Links](#) on page 790.

**Reference:**     **Syntax Guidelines**

You can create multiple drill-down links from a component on a summary request, chosen from any combination of supported actions. For example, you can create links to a detail report, chart, or Maintain procedure, a JavaScript function, and a URL. When you click a component in the summary request, a pop-up menu appears listing the drill-down options.

This feature does not apply to headings or footings,

When you create multiple drill-down links, you cannot specify a single drill-down action (for example, FOCEXEC or URL) before the first DRILLMENUITEM.

The menu created by the DRILLMENUITEM keyword is styled using a Cascading Stylesheet file. The file is /ibi/WebFOCUSxx/ibi\_apps/ibi\_html/javaassist/ibi/html/js/multidrill.css, where xx is the version and major release number of WebFOCUS.

**Tip:** You can make changes to this file to affect the font, size, and color of the DRILLMENUITEM menu. Make a backup of this file before modifying it.

### **Reference: Summary of Drill-Down Links**

You can link to:

- ❑ Another request. The StyleSheet attribute is FOCEXEC.

- ❑ A URL. The StyleSheet attribute is URL. You pass a valid URL.

Note that the length of the URL is limited by the maximum number of characters allowed by the browser. For information about this limit for your browser, search on the support site for your browser vendor.

- ❑ A URL from a field. The StyleSheet attribute is URL. You pass the name of a report column whose value is a valid URL to which the link will jump.

- ❑ A JavaScript function. The StyleSheet attribute is JAVASCRIPT.

- ❑ A WebFOCUS Maintain procedure. The StyleSheet attribute is URL with the keyword MNTCON EX. For details on the syntax, see [Linking to a Maintain Data Procedure](#) on page 779.

- ❑ A WebFOCUS compiled Maintain procedure. The StyleSheet attribute is URL with the keyword MNTCON RUN. For details on the syntax, see [Linking to a Maintain Data Procedure](#) on page 779.

### **Creating Parameters**

Parameters allow you to specify criteria and conditions for the linked (drill down) report. By defining parameters, you can control the amount and type of information to retrieve when you click on a hot spot.

For complete details, see [Linking a Report to Other Resources](#) on page 763.

### **Adding Labels to a Graph**

Adding labels to your graph helps provide important information about what the data in your graph represents. You may choose to add headings and/or footings to your graph, as well as horizontal (X) and vertical (Y) axis labels.

You can add a heading or footing to your graph using the HEADING or FOOTING phrase. The syntax is the same as that used for headings and footings in TABLE requests. You can also embed field values in graph headings and footings. This capability is particularly useful for annotating graphs that contain multiple sort fields.

For details on headings, footings, and embedded fields, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.

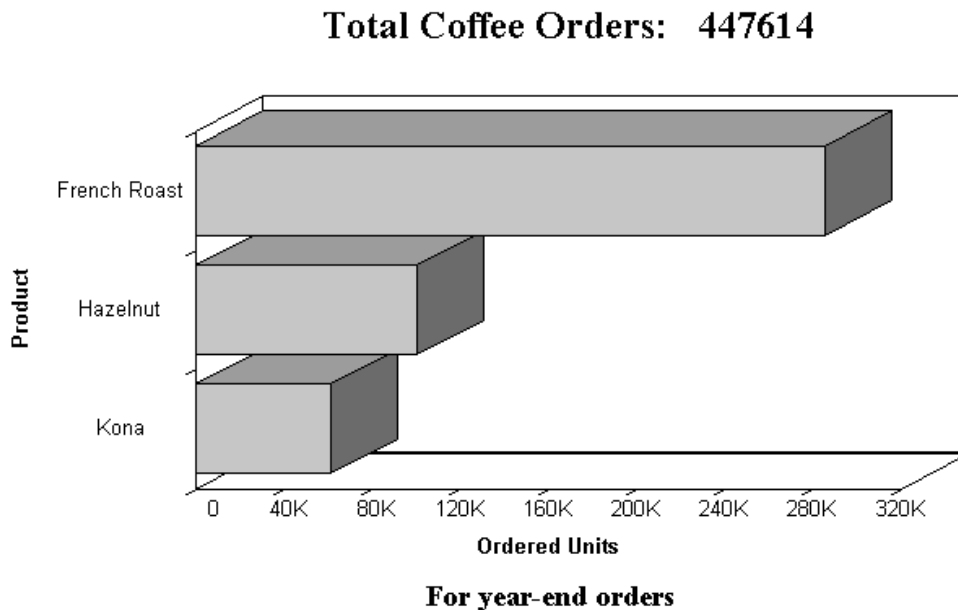
**Note:** If your graph labels or legends are not appearing correctly when you run your graph, see [How to Change Color Settings](#) on page 1727 for details on correcting this.

**Example: Adding a Heading and Footing to a Graph**

The following illustrates how to add a heading with an embedded field value to your graph. In this example, the heading is "Total Coffee Orders" with the embedded field TOT.QUANTITY.

```
GRAPH FILE GGORDER
HEADING CENTER
"Total Coffee Orders: <TOT.QUANTITY >"
SUM QUANTITY
BY PRODUCT_DESC
WHERE PRODUCT_DESC EQ 'French Roast' OR 'Hazelnut' OR 'Kona'
FOOTING CENTER
"For year-end orders"
END
```

The output is:



## Adding Vertical (Y-axis) and Horizontal (X-axis) Labels to a Graph

Vertical (Y-axis) and horizontal (X-axis) graph labels are placed on the graph according to the display fields and sort fields specified in the request. The titles that appear on the graph are the titles that appear in the Master File for that particular field.

The vertical (Y-axis) title of the graph is determined by the display field. Note that when the number of Y-axis labels is greater than one, the labels do not appear along the Y-axis. Instead, the labels appear in a legend that provides the names of the fields being graphed.

The horizontal (X-axis) title of the graph is determined by the sort field.

You can replace a title by using an AS phrase in the GRAPH request.

## Applying Custom Styling to a Graph

You can customize your graph using StyleSheets and SET commands. You can set the graph width and height, set fixed scales for the X and Y axes, enable the Graph Editor, and use Graph API calls to further customize your graph.

For details on customizing graph headings and footings, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.

## Setting the Graph Height and Width

The width (or horizontal axis) of each graph, which includes any surrounding text, is automatically set to 760 pixels. When setting the graph width, you should allow for the inclusion of any text required for the vertical axis and its labels along the left margin.

To maximize display space, you can limit the size of your labels through the use of either AS phrases or DECODE expressions.

The height (or vertical axis) of your graph is automatically set to 405 pixels.

The vertical axis is automatically set (VAUTO=ON) to cover the total range of plotted values. The height of the axis is set as high as possible (taking into consideration the presence of any headings or footings and the need to provide suitably rounded vertical class markers). The range is divided into intervals called "classes." The scale is normalized to provide class values rounded to the appropriate multiples and powers of 10 for the intervals plotted on the axis.

**Syntax:**      **How to Set the Graph Width**

```
SET HAXIS={nn|760}
```

where:

```
{nn|760}
```

Is a positive numeric value. The default is 760 pixels.

**Note:** The maximum HAXIS size for SVG graphs is 40 inches.

**Syntax:**      **How to Set the Graph Height**

```
SET VAXIS={nn|400}
```

where:

```
{nn|400}
```

Is a positive numeric value. The default is 400 pixels.

**Note:** The maximum VAXIS size for SVG graphs is 40 inches.

**Customizing Graphs Using SET Parameters**

The GRAPH environment includes a set of parameters that control the appearance of the graph and offer some additional control when you run the request.

For example, the BSTACK parameter enables you to specify that the bars on a bar graph are to be stacked rather than placed side by side.

**Syntax:**      **How to Use SET Parameters With GRAPH Requests**

To set the parameters that control the GRAPH environment, use the appropriate variation of the SET parameter.

```
SET parameter=value,parameter=value...
```

For a list of supported GRAPH parameters, see [Values and Functions of SET Parameters for Graphs](#) on page 1714.

**Note:**

- ☐ Repeat the command SET on each new line.
- ☐ When entering more than one parameter on a line, separate them with commas.

- ☐ You can use unique truncations of parameter names. You must make sure that they are unique.

**Example:**    **Using SET Parameters With GRAPH Requests**

The following shows how to set the height (Y-axis) and width (X-axis) for a graph.

```
SET HAXIS=75,VAXIS=40
GRAPH FILE filename
.
.
.
END
```

**Reference:**    **Values and Functions of SET Parameters for Graphs**

Graph SET Parameter	Values	Parameter Function
3D	<a href="#">ON</a>   <a href="#">OFF</a>	When ON, a three-dimensional chart is produced. When OFF, a two-dimensional chart is produced. ON is the default.
AUTOTICK	<a href="#">ON</a>   <a href="#">OFF</a>	When ON, tick mark intervals are automatically set. ON is the default. (See also HTICK and VTICK.)
BARNUMB	<a href="#">ON</a>   <a href="#">OFF</a>	When ON, places the summary values at the ends of the bars on bar graphs or slices on pie graphs. OFF is the default.
BSTACK	<a href="#">ON</a>   <a href="#">OFF</a>	When ON, specifies that the bars on a bar graph are to be stacked rather than placed side by side. OFF is the default.

Graph SET Parameter	Values	Parameter Function
GRAPHEDIT	<i>graphedit</i>	As of WebFOCUS 8.0, this parameter has been deprecated. For information about editing charts, see the <i>WebFOCUS InfoAssist User's Manual</i> .
GRID	ON   OFF	When ON, specifies that a grid is to be drawn on the graph at the horizontal and vertical class marks (see also VGRID). OFF is the default.
HAUTO	ON   OFF	When ON, specifies automatic scaling of the horizontal axis unless overridden by the user. If OFF, user must supply values for HMAX and HMIN. ON is the default.
HAXIS		Specifies the width in characters of the horizontal axis. This parameter can be adjusted for graphs generated offline. HAXIS is ignored for online displays since the width of the graph is automatically adjusted to the width of the display area.
HCLASS	<i>nnn</i>	Specifies the horizontal interval mark when AUTOTICK=OFF (see also HTICK).
HISTOGRAM	ON   OFF	When ON, a histogram is drawn instead of a curve when values on the horizontal axis are not numeric. ON is the default.

Graph SET Parameter	Values	Parameter Function
HMAX	<i>nnn</i>	Specifies the maximum value on the horizontal axis when the automatic scaling is not used (HAUTO=OFF).
HMIN	<i>nnn</i>	Specifies the minimum value on the horizontal axis when the automatic scaling is not used (HAUTO=OFF).
HSTACK	ON   <u>OFF</u>	When ON, specifies that the bars on a histogram are to be stacked rather than placed side by side. OFF is the default.
HTICK	<i>nnn</i>	Specifies the horizontal axis tick mark interval when AUTOTICK is OFF (see also HCLASS).
LOOKGRAPH	<i>option</i>	Specifies the graph type. For more information, see <a href="#">Determining Graph Styles Using LOOKGRAPH</a> on page 1672.
PIE	ON   <u>OFF</u>	When ON, specifies that a pie graph is desired. OFF is the default.
VAUTO	<u>ON</u>   OFF	When ON, specifies automatic scaling of the vertical axis unless overridden by the user. If OFF, the user must supply values for VMAX and VMIN. ON is the default.



Graph SET Parameter	Values	Parameter Function
VAXIS		Specifies page length in lines. This parameter can be adjusted for graphs generated offline. VAXIS is ignored for online displays since the height of the graph is automatically adjusted to the display area.
VCLASS	<i>nnn</i>	Specifies the vertical interval mark when AUTOTICK=OFF (see also VTICK).
VGRID	ON   <u>OFF</u>	When ON, specifies that a grid is to be drawn on the graph at the horizontal and vertical class marks (see also GRID). OFF is the default.
VMAX	<i>nnn</i>	Specifies the maximum value on the vertical axis when the automatic scaling is not used (VAUTO=OFF).
VMIN	<i>nnn</i>	Specifies the minimum value on the vertical axis when the automatic scaling is not used (VAUTO=OFF).
VTICK	<i>nnn</i>	Specifies the vertical axis tick mark interval when AUTOTICK is OFF (see also VCLASS).

Graph SET Parameter	Values	Parameter Function
VZERO	ON   OFF	Determines whether values along the Y-axis are stored or ignored. If ON, missing data along the Y-axis is treated as zero. If OFF, missing data along the Y-axis is ignored and values are not stored in the plot matrix. OFF is the default.

Setting Fixed Scales for the X-Axis

The horizontal scale is automatically set to cover the total range of values to be plotted (HAUTO=ON). The range is divided into intervals called "classes." The scale is normalized to provide class values rounded to the appropriate multiples of 10 for the intervals plotted on the axis.

To assign fixed upper and lower limits (useful when producing a series of graphs where consistent scales are needed), you can turn off the automatic scaling mechanism and set new limit values by setting HAUTO=OFF.

Syntax: How to Set Fixed Scales for the X-Axis

```
SET HAUTO=OFF, HMAX=nn, HMIN=nn
```

where:

HAUTO

Is the automatic scaling facility. If HAUTO is ON, any values for HMAX and HMIN are overridden.

HMAX=nn

Sets the upper limit on the horizontal axis. The default is 0.

HMIN=nn

Controls the lower limit on the horizontal axis when HAUTO is OFF. The default is 0.

Note:

- ❑ When entering several SET parameters on one line, separate them with commas.

- ❑ If you define limits that do not incorporate all of the data values, OVER or UNDER will be displayed to indicate that some of the data extracted is not reflected on the graph.

## Setting Fixed Scales for the Y-Axis

To give the vertical scale fixed upper and lower limits (useful when producing a series of graphs where consistent scales are needed), you can turn off the automatic scaling mechanism and set fixed limits using the SET VAUTO=OFF command.

### **Syntax:** How to Set Fixed Scales for the Y-Axis

```
SET VAUTO=OFF, VMAX=nn, VMIN=nn
```

where:

**VAUTO**

Is the automatic scaling facility. If VAUTO is ON, any values for VMAX and VMIN are overridden.

**VMAX=nn**

Sets the upper limit on the vertical axis when VAUTO is OFF. The default is 0.

**VMIN=nn**

Controls the lower limit on the vertical axis when VAUTO is OFF. The default is 0.

#### **Note:**

- ❑ When entering several SET parameters on one line, separate them with commas.
- ❑ If you define limits that do not incorporate all of the data values, OVER or UNDER will be displayed to indicate that some of the data extracted is not reflected on the graph.

## Customizing Graphs Using the Graph API and HTML5 JSON Properties

You can further enhance your graph output by manually adding API calls inside the ON GRAPH SET STYLE \* and ENDSTYLE commands in the GRAPH request. If you are creating an HTML5 graph, you can also include JavaScript Object Notation (JSON) methods and properties. When you save changes the corresponding API calls and properties will be written to the graph StyleSheet.

For reference information about the Graph API, see the *WebFOCUS Graphics* manual.

When you include both JSON and API calls in the StyleSheet section of the request, the API calls are parsed first, then the JSON. Therefore, if the JSON sets the same property as an API call, the JSON will take precedence. In general, the later declarations overwrite the properties set in earlier declarations.

### **Syntax:** How to Customize a Graph Using the Graph API

```
GRAPH FILE filename
graph commands
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
API calls
*END

*GRAPH_JS
JSON
*END

WEBFOCUS StyleSheet commands
ENDSTYLE
END
```

where:

*filename*

Specifies a data source for the graph.

*API calls*

Are API calls. They must be included in a GRAPH\_SCRIPT block within \*GRAPH\_SCRIPT and \*END declarations. A request can contain multiple GRAPH\_SCRIPT blocks anywhere within the style section. For reference information about the Graph API, see the *WebFOCUS Graphics* manual.

*JSON*

Are JSON methods and properties that apply to HTML5 graph output. They must be included in a GRAPH\_JS block within \*GRAPH\_JS \*END declarations. A request can contain multiple GRAPH\_JS blocks anywhere within the style section. For reference information about the JSON methods and properties, see the *Creating HTML5 Charts With WebFOCUS Language* manual.

*WEBFOCUS StyleSheet commands*

For details on StyleSheet commands, see [Creating and Managing a WebFOCUS StyleSheet](#) on page 1117.

**Example: Customizing Graphs Using the Graph API**

The following annotated example illustrates how to customize a graph using ON GRAPH SET STYLE \*. The Graph API code is highlighted in the request.

```

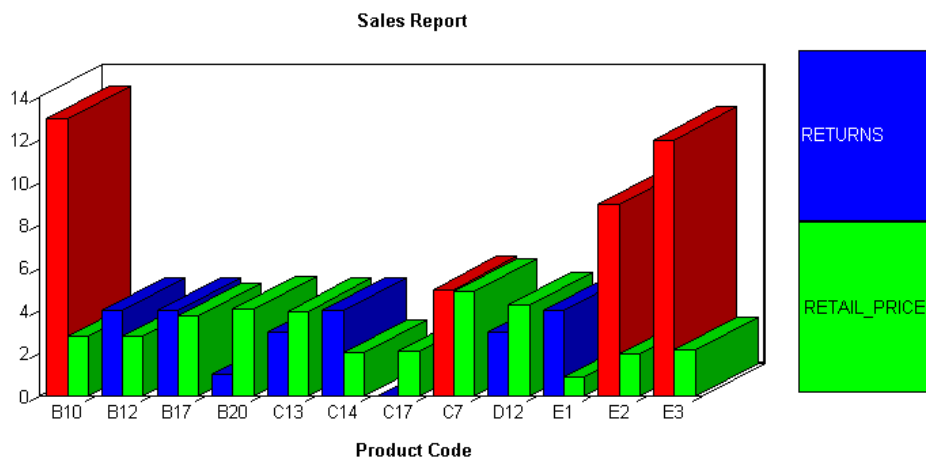
GRAPH FILE SALES
SUM RETURNS
RETAIL_PRICE
ACROSS PROD_CODE AS 'Product Code'
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
1. setLegendMarkerPosition(4);
2. setO1LabelRotate(0);
3. setTitleString("Sales Report");
4. setTextJustHoriz(getTitle(),1);
*END
DEFMACRO=COND0001, MACTYPE=RULE, WHEN=RETURNS GT 4,$
TYPE=DATA,MACRO=COND0001,ACROSSCOLUMN=RETURNS,COLOR=RED,$
ENDSTYLE
END

```

where:

1. Displays legend text inside the legend marker.
2. Displays the X-axis labels horizontally.
3. Displays the title (Sales Report) without quotes.
4. Centers the title.

The output is:



## Saving a Graph as an Image File

You can save graph output to an image file using the GRAPHSEVURL parameter or the JSCOM3 configuration on the WebFOCUS Reporting Server. Saving graph output as an image file is useful when you want to create a single PDF or HTML report that contains multiple outputs, such as output from a TABLE request and a GRAPH request. You can distribute this type of report using ReportCaster.

For details, see [Saving a Graph as an Image File Using GRAPHSEVURL](#) on page 1722.

### Saving a Graph as an Image File Using GRAPHSEVURL

The GRAPHSEVURL parameter enables users who are running against a server environment where the WebFOCUS Reporting Server is installed on a z/OS, Windows, or UNIX machine to save graph output as a GIF file. GIF images can be embedded in a PDF or HTML report.

The GRAPHSEVURL parameter sends an http request to the machine that has the WebFOCUS Graph Servlet. The graph image is created by the WebFOCUS Graph Servlet, and the image is sent back to a temporary location on the WebFOCUS Reporting Server (if an Allocation has not been specified), or to the location specified in a FILEDEF command. You may use the Allocation Wizard to create a FILEDEF command.

#### **Procedure:** How to Save a Graph as an Image File Using GRAPHSEVURL

1. Install JDK as per the requirements in the *WebFOCUS & ReportCaster Installation & Configuration* manual for your platform.
2. Create a procedure that produces the image, and set GRAPHSEVURL in the procedure to the URL that invokes the WebFOCUS Graph Servlet. For example,

```
SET GRAPHSEVURL=http://hostname/ibi_apps/IBIGraphServlet
```

where:

*hostname*

Is the name of the machine where WebFOCUS is installed.

For more details, see [How to Save a Graph as an Image File](#) on page 1723.

3. Run the procedure directly on the WebFOCUS Reporting Server, using a browser.

**Syntax:**      **How to Save a Graph as an Image File**

```
[FILEDEF filename DISK drive:\...\filename.fmt]
SET GRAPHSEVRURL= graph_servlet_URL
GRAPH FILE file
graph commands
ON GRAPH HOLD AS filename FORMAT fmt
END
```

where:

**FILEDEF**

Saves the image file to the location you specify.

*filename*

Is the name you give the image file, which must match the FILEDEF command filename. If you want to prompt for a filename, include an amper variable such as &FILENAME in the procedure.

*fmt*

Is the type of image file in which to store the graph. Acceptable values are: PNG, SVG, GIF, or JPG.

*graph\_servlet\_URL*

Is the URL to invoke the WebFOCUS Graph Servlet. The maximum number of characters is 256.

*file*

Is the name of the data source you wish to report against.

**Note:** To insert an image that resides in a permanent location, you must provide the fully qualified path to the image file. For example, to insert a GIF file,

```
TYPE=REPORT, IMAGE=drive:\...\filename.gif
```

where:

*drive:\...\filename.gif*

Is the path where the GIF file is located. The WebFOCUS Reporting server must be installed on that drive.

**Example:**    **Inserting a GIF Image Into a PDF Report**

The following illustrates how you can create a GIF file from a graph request, and then embed the GIF image into a PDF report.

1. Create the remote procedure in a location accessible to the EDA path or application path. For example, if you are running against your local server it may look like this:

```
SET GRAPHSEVRURL= http://localhost/ibi_apps/IBIGraphServlet
GRAPH FILE CENTORD
SUM LINEPRICE
ACROSS PLANTLNG AS 'Plant'
ON GRAPH HOLD AS PLANT FORMAT GIF
END

TABLE FILE CENTORD
SUM LINEPRICE
BY PLANTLNG AS 'Plant'
ON TABLE SET STYLE *
TYPE=REPORT, IMAGE=plant.gif, POSITION=(4 0), SIZE=(5 3), $
ENDSTYLE
ON TABLE PCHOLD FORMAT PDF
END
```

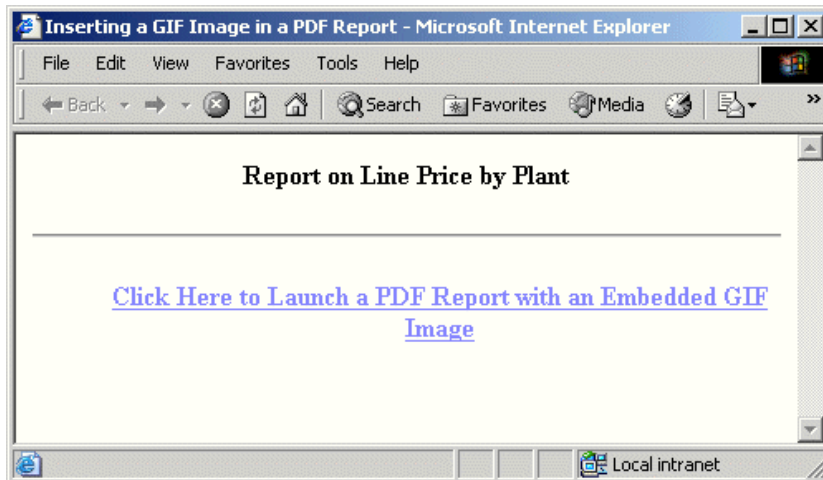
**Note:** If you are using JSCOM3, you can eliminate the SET GRAPHSEVRURL parameter from the procedure.

2. Save the procedure as HOLDGIF.
3. Create an HTML file that calls the HOLDGIF WebFOCUS procedure from a hyperlink. For example,

```
<HTML>
<HEAD>
<TITLE> Inserting a GIF Image in a PDF Report </TITLE>
</HEAD>
<BODY>
<H4 ALIGN=CENTER>Report on Line Price by Plant</H4>
<HR>
<P><FONT SIZE=+2></FONT></P>
<UL TYPE=SQUARE>
<LI><A HREF="http://localhost/ibi_apps/WFServlet?IBIF_ex=holdgif">
<H4 ALIGN=CENTER>Click Here to Launch a PDF Report with an Embedded
    GIF Image</H4></A>
</UL>
</BODY>
</HTML>
```

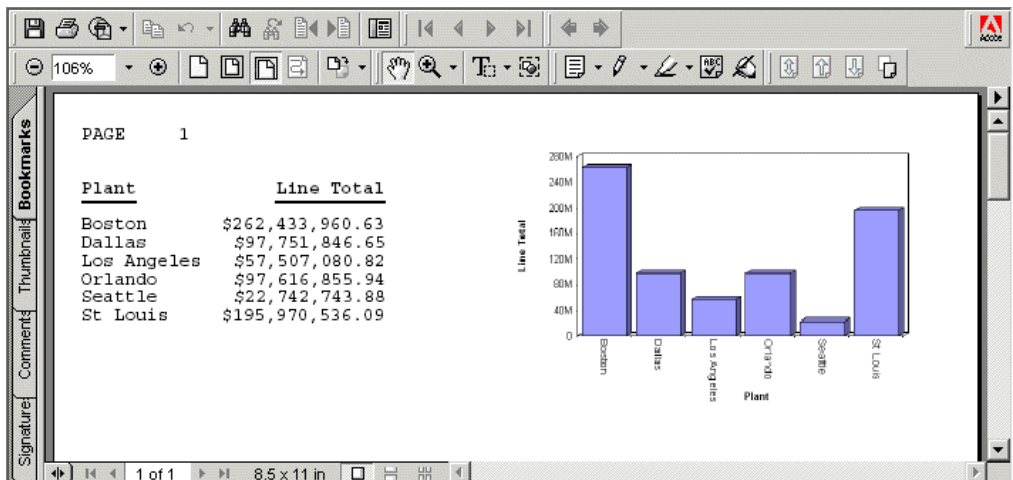


The resulting launch page looks like this:



You can now run the report from a browser. To distribute the report using ReportCaster, you would schedule the actual procedure, in this case HOLDGIF, to distribute the report.

4. Click the link to run the report. The report looks like this:



**Note:** To run this procedure as a Managed Reporting Standard Report, add the -MRNOEDIT command to the beginning of the StyleSheet declaration containing the IMAGE attribute. This prevents Managed Reporting from looking for the GIF file in the Managed Reporting Repository. The image that is specified must reside on the WebFOCUS Reporting Server.

The -MRNOEDIT syntax is not case-sensitive and it can be used on a single line or a block of lines. For example:

### Single line

```
-MRNOEDIT TYPE=REPORT, IMAGE=PLANT.gif, POSITION=(4 0), SIZE=(5 3), $
```

or

```
-MRNOEDIT TYPE=REPORT, IMAGE=PLANT.gif,  
-MRNOEDIT POSITION=(4 0), SIZE=(5 3), $
```

### Multiple lines

```
-MRNOEDIT BEGIN  
TYPE=REPORT, IMAGE=PLANT.gif, POSITION=(4 0), SIZE=(5 3), $  
-MRNOEDIT END
```

## **Reference:** Usage Notes for Saving a Graph

If you selected the *Save Report* check box in the configuration pane of the WebFOCUS Administration Console (under Redirection Settings), you will be prompted whether to save or open the output file. If the procedure contained a PCHOLD command that specified an AS name for the output file, the name is retained if you choose to save the file. If no AS name was specified, a random filename is generated.

**Important:** You must do the following in the WebFOCUS Administration Console to utilize *Save Report* functionality for WebFOCUS GRAPH requests (specified with a PNG, SVG, GIF, or JPG format in the procedure):

- ☐ Set *Save Report* to yes for the .htm Extension.

Running a server-side GRAPH request creates an HTM file that contains a link to the actual graph output, which is stored as a temporary image file with a .jpg, .gif, .svg, or .png extension.

When you execute a GRAPH request, if you select the *Save* option when prompted to *Open* or *Save* the output, the output is saved to an HTM file using only a reference to the graph image, which will eventually expire and be deleted from the server (according to the temporary file expiration settings in the WebFOCUS Client Configuration).

To preserve the output of the GRAPH request, open the saved HTM file, right-click the graph image, and select *Save Picture As* to save it to disk permanently. You can then substitute an absolute reference to the saved image file in the returned HTM output file.

- ☐ Click *Save* to save your changes in the Redirection Settings panel.

## Printing a Graph

When you run your graph, you may print the output directly from the browser.

**Note:** If your graph labels or legends are not appearing correctly when you run your graph, see [How to Change Color Settings](#) on page 1727 for details on correcting this.

### **Procedure:** How to Print Your Graph

1. Run your graph.
2. From the browser, select *Print* from the File menu.

### **Syntax:** How to Send Graph Output Directly to a Printer

Add the following syntax to your GRAPH request:

```
ON GRAPH SET PRINT OFFLINE
```

### **Procedure:** How to Change Color Settings

1. From the Windows Control Panel, select *Display*.
2. Click the *Settings* tab.
3. In the Color palette box, click the drop-down arrow and select True Color or High Color.

**Note:** If your version of Windows does not have these options, select 65536 or a higher color count.

4. Click *OK*.

If you use different color settings from this recommended value, your graphs may appear in grayscale format.



The Financial Modeling Language (FML) is designed for the special needs associated with creating, calculating, and presenting financially oriented data, such as balance sheets, consolidations, or budgets. These reports are distinguished from other reports because calculations are inter-row, as well as inter-column, and each row or line represents a unique entry or series of entries that can be aggregated directly from the input data or calculated as a function of the data.

**In this chapter:**

- |  |  |
|--|--|
| <input type="checkbox"/> Reporting With FML                            | <input type="checkbox"/> Inserting Rows of Free Text                       |
| <input type="checkbox"/> Creating Rows From Data                       | <input type="checkbox"/> Adding a Column to an FML Report                  |
| <input type="checkbox"/> Supplying Data Directly in a Request          | <input type="checkbox"/> Creating a Recursive Model                        |
| <input type="checkbox"/> Performing Inter-Row Calculations             | <input type="checkbox"/> Reporting Dynamically From a Hierarchy            |
| <input type="checkbox"/> Referring to Rows in Calculations             | <input type="checkbox"/> Customizing a Row Title                           |
| <input type="checkbox"/> Referring to Columns in Calculations          | <input type="checkbox"/> Formatting an FML Report                          |
| <input type="checkbox"/> Referring to Rows and Columns in Calculations | <input type="checkbox"/> Suppressing the Display of Rows                   |
| <input type="checkbox"/> Referring to Cells in Calculations            | <input type="checkbox"/> Saving and Retrieving Intermediate Report Results |
| <input type="checkbox"/> Using Functions in RECAP Calculations         | <input type="checkbox"/> Creating HOLD Files From FML Reports              |
- 

### Reporting With FML

FML is an integrated extension of the TABLE command. By adding the FOR phrase and the RECAP command, you can handle an expanded range of applications.

**Note:** MORE is not supported in FML requests.

In conjunction with Dialogue Manager, FML can evaluate "what if" scenarios and develop complete decision support systems. These systems can take advantage of business intelligence features, such as statistical analysis and graphics, in addition to standard financial statements.

Procedures using FML are not hard-wired to the data. As in any other report request, they can easily be changed. FML includes the following facilities:

- ❑ **Row and column formatting.** You can specify results in a row-by-row, column-by-column fashion. For more information, see [Performing Inter-Row Calculations](#) on page 1743.
- ❑ **Intermediate results.** You can post FML results to an external file and pick them up at a later time for analysis. This is useful when intermediate results are developed and a final procedure consolidates the results later. For more information, see [Saving and Retrieving Intermediate Report Results](#) on page 1804.
- ❑ **Inline data entry.** FML enables you to specify constants from within the procedure, in addition to the data values retrieved from your data source. For more information, see [Supplying Data Directly in a Request](#) on page 1741.
- ❑ **Recursive reporting.** You can produce reports where the results from the end of one time period or column become the starting balance in the next. For example, you can use recursive reports to produce a cash flow projection. For more information, see [Creating a Recursive Model](#) on page 1763.
- ❑ **Dynamic reporting from a chart of accounts or a similar hierarchy of information.** You can create a report that changes as the organization of information changes, ensuring that you automatically retrieve information that reflects the latest structure and its values. There is no need to alter either the Master File or the report request. For more information, see [Reporting Dynamically From a Hierarchy](#) on page 1764.

### ***Example:*** Sample FML Request

This example produces a simple asset sheet, contrasting the results of two years. It illustrates many key features of the Financial Modeling Language (FML). Numbers to the left of the procedure lines correspond to explanations that follow the request.

```

TABLE FILE FINANCE
HEADING CENTER
"COMPARATIVE ASSET SHEET </2"
SUM AMOUNT ACROSS HIGHEST YEAR
WHERE YEAR EQ '1983' OR '1982'
1. FOR ACCOUNT
2. 1000          AS 'UTILITY PLANT'          LABEL  UTP    OVER
2. 1010 TO 1050  AS 'LESS ACCUMULATED DEPRECIATION' LABEL  UTPAD   OVER
3. BAR
4. RECAP UTPNET = UTP-UTPAD; AS 'TOTAL PLANT-NET'
   BAR
   2000 TO 3999  AS 'INVESTMENTS'          LABEL  INV    OVER
5. "CURRENT ASSETS"
   4000          AS 'CASH'                  LABEL  CASH   OVER
   5000 TO 5999  AS 'ACCOUNTS RECEIVABLE-NET' LABEL  ACR    OVER
   6000          AS 'INTEREST RECEIVABLE'    LABEL  ACI    OVER
   6500          AS 'FUEL INVENTORY'         LABEL  FUEL   OVER
   6600          AS 'MATERIALS AND SUPPLIES'  LABEL  MAT    OVER
   6900          AS 'OTHER'                 LABEL  MISC   OVER
   BAR
   RECAP TOTCAS=CASH+ACR+ACI+FUEL+MAT+MISC;AS 'TOTAL CURRENT ASSETS' OVER
   BAR
   7000          AS 'DEFERRED DEBITS'        LABEL  DEFDB  OVER

   BAR
6. RECAP TOTAL = UTPNET+INV+TOTCAS+DEFDB; AS 'TOTAL ASSETS' OVER
   BAR AS '='
   FOOTING
   "</2 *** PRELIMINARY ASSET SHEET BASED ON UNAUDITED FIGURES ***"
END

```

1. FOR and OVER are FML phrases that enable you to structure the report on a row-by-row basis.
2. LABEL assigns a variable name to a row item for use in a RECAP calculation.  
1000 and 1010 TO 1050 are tags that identify the data values of the FOR field, ACCOUNT in the FINANCE data source. A report row can be associated with a tag that represents a single data value (like 1000), multiple data values, or a range of values (like 1010 TO 1050).
3. BAR enables you to underline a column of numbers before performing a RECAP calculation.
4. The RECAP command creates a new value based on values already identified in the report with LABEL. In this case, the value UTPNET is derived from UTP and UTPAD and is renamed TOTAL PLANT-NET with an AS phrase to provide it with greater meaning in the report.
5. Free text can be incorporated at any point in an FML report, similar to underlines.
6. Notice that this RECAP command derives a total (TOTAL ASSETS) from values retrieved directly from the data source, and from values derived from previous RECAP computations (UTPNET and TOTCAS).

The output is shown as follows.

#### COMPARATIVE ASSET SHEET

	YEAR	
	1983	1982
UTILITY PLANT	1,430,903	1,294,611
LESS ACCUMULATED DEPRECIATION	249,504	213,225
TOTAL PLANT-NET	1,181,399	1,081,386
INVESTMENTS	818	5,639
CURRENT ASSETS		
CASH	4,938	4,200
ACCOUNTS RECEIVABLE-NET	28,052	23,758
INTEREST RECEIVABLE	15,945	10,206
FUEL INVENTORY	35,158	45,643
MATERIALS AND SUPPLIES	16,099	12,909
OTHER	1,264	1,743
TOTAL CURRENT ASSETS	101,456	98,459
DEFERRED DEBITS	30,294	17,459
TOTAL ASSETS	1,313,967	1,202,943

\*\*\* PRELIMINARY ASSET SHEET BASED ON UNAUDITED FIGURES \*\*\*

## Creating Rows From Data

A normal TABLE request sorts rows of a report according to the BY phrase you use. The data retrieved is sorted from either low-to-high or high-to-low, as requested. The rows may be limited by a screening phrase to a specific subset, but:

- ☐ They appear in a sort order.
- ☐ Rows appear only for values that are retrieved from the file.



- ❑ You can only insert free text between rows when a sort field changes value, such as:

```
ON DIVISION SUBFOOT
```

- ❑ You can only insert calculations between rows when a sort field changes value, such as:

```
ON DIVISION RECAP
```

In contrast, the FML FOR phrase creates a matrix in which you can structure your report row-by-row. This organization gives you greater control over the data that is incorporated into a report, and its presentation. You can:

- ❑ Report on specific data values for a field in a data source and combine particular data values under a common label, for use in calculations.
- ❑ Type data directly into the request to supplement data retrieved from the data source.
- ❑ Include text, underlines, and calculations at points in the report that are not related to sort breaks.
- ❑ Perform recursive processing, in which the result of an interim calculation is saved and then used as the starting point for a subsequent calculation.
- ❑ Suppress the display of rows for which no data is retrieved.
- ❑ Identify rows by labels and columns by numbers, addresses, and values so that you can point to the individual cells formed at each intersection (as on a spreadsheet).

### **Syntax:** How to Retrieve FOR Field Values From a Data Source

The syntax for specifying rows is:

```
FOR fieldname [AS 'coltitle'] value [OR value OR...] [AS 'text']
[LABEL label] OVER
.
.
.
[value [OR value ...]] [AS 'text'] [LABEL label]
END
```

where:

*fieldname*

Is the FOR field for the FML report.

*coltitle*

Is the column title for the FOR field on the report output.

value

Is the value (also known as a tag value) describing the data that is retrieved for this row of the report.

AS 'text'

Enables you to assign a name to a tag value, which replaces the tag value in the output. Enclose the text in single quotation marks.

label

Assigns a label to the row for reference in a RECAP expression. The label can be up to 66 characters and cannot have blanks or special characters. Each explicit label you assign must be unique.

Even if you assign an explicit label, the positional label (R1, R2, and so on) is retained internally.

By default, a tag value for a FOR field (like 1010) may be added only once to the FML matrix. However, if you wish to add the same value of a FOR field to the matrix more than once, you can turn on the FORMULTIPLE parameter (the default setting is OFF). For more information, see [How to Use the Same FOR Field Value in Multiple Rows](#) on page 1738.

For more information about the FMLFOR, FMLLIST, and FMLINFO functions that return the tag values used in an FML request, see the *Using Functions* manual.

Example: Creating Rows From Values in a Data Source

Assume you have a simple data source with financial data for each corporate account, as follows:

CHART OF ACCOUNTS	
ACCOUNT	DESCRIPTION
1010	CASH ON HAND
1020	DEMAND DEPOSITS
1030	TIME DEPOSITS
1100	ACCOUNTS RECEIVABLE
1200	INVENTORY
.	.
.	.
.	.

Using the FOR phrase in FML, you can issue the following TABLE request in which each value of ACCOUNT is represented by a tag (1010, 1020, and so on), and displays as a separate row:

```
TABLE FILE LEDGER
SUM AMOUNT
FOR ACCOUNT
1010 OVER
1020 OVER
1030 OVER
1100 OVER
1200
END
```

The output is shown as follows.

	AMOUNT
	-----
1010	8,784
1020	4,494
1030	7,961
1100	18,829
1200	27,307

Creating Rows From Multiple Records

There are different ways to combine multiple values from your data sources into an FML report row. You can use:

- ❑ The OR phrase to sum the values of two or more tags in a single expression. For more information, see [How to Sum Values in Rows With the OR Phrase](#) on page 1736.
- ❑ The TO phrase to identify a range of tag values on which to report. For more information, see [How to Identify a Range of Values With the TO Phrase](#) on page 1737.
- ❑ A mask to specify a group of tag values without having to name each one. For more information, see [How to Use Masking Characters to Retrieve Tag Values](#) on page 1738.

By default, a FOR field value can only be included in a single row of an FML matrix. However, by turning on the FORMULTIPLE parameter, you can include the same data value in multiple rows in the FML matrix. For example, the same value can exist as a solitary value in one row, be part of a range in another row, and be used in a calculation in a third row. For more information, see [How to Use the Same FOR Field Value in Multiple Rows](#) on page 1738.

In addition to these methods, you can extract multiple tags for a row from an external file.

**Syntax:**      **How to Sum Values in Rows With the OR Phrase**

To sum the values of two or more tags in a single report row, use the OR phrase in the FOR phrase. The syntax is:

```
FOR fieldname
value1 OR value2 [OR valuen...] [AS 'text'] [LABEL label] [OVER]
.
.
.
```

where:

*fieldname*

Is a field name in the data source.

*value1*, *value2*, *valuen*

Are the tag values to be retrieved and summed.

AS '*text*'

Assigns a title to the combined tag values. Enclose the text in single quotation marks (').

*label*

Assigns a label to the row for reference in a RECAP expression. The label can be up to 66 characters and cannot have blanks or special characters. Each explicit label you assign must be unique.

Even if you assign an explicit label, the positional label (R1, R2, and so on) is retained internally.

**Example:**      **Summing Values in Rows**

The following model sums the values of three tags (1010, 1020, 1030) as CASH.

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 OR 1020 OR 1030 AS 'CASH' OVER
1100 AS 'ACCOUNTS RECEIVABLE' OVER
1200 AS 'INVENTORY'
END
```

The output is shown as follows.

	AMOUNT
	-----
CASH	21,239
ACCOUNTS RECEIVABLE	18,829
INVENTORY	27,307

**Syntax:**      **How to Identify a Range of Values With the TO Phrase**

To sum the values of a range of tags in a single report row, use the TO phrase in the FOR phrase. The syntax is:

```
FOR fieldname
value1 TO value2 [AS 'text'] [LABEL label] [OVER]
```

where:

*fieldname*

Is a field name in the data source.

*value1*

Is the tag value at the lower limit of the range.

TO

Is the required phrase.

*value2*

Is the tag value at the upper limit of the range.

AS '*text*'

Assigns a title to the combined tag values. Enclose the text in single quotation marks (').

*label*

Assigns a label to the row for reference in a RECAP expression. The label can be up to 66 characters and cannot have blanks or special characters. Each explicit label you assign must be unique.

Even if you assign an explicit label, the positional label (R1, R2, and so on) is retained internally.

**Example:**      **Identifying a Range of Values**

Since CASH accounts in the LEDGER system are identified by the tags 1010, 1020, and 1030, you can specify the range 1010 to 1030:

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 TO 1030 AS 'CASH'
END
```

**Syntax:**      **How to Use Masking Characters to Retrieve Tag Values**

If the tag field has a character (alphanumeric) format, you can perform a masked match. Use the dollar sign character (\$) as the mask. For instance,

```
A$$D
```

matches any four-character value beginning with A and ending with D. The two middle places can be any character. This is useful for specifying a whole group of tag values without having to name each one.

**Example:**      **Using Masking Characters to Match a Group of Tags**

In this example, the amounts associated with all four-character accounts that begin with 10, expressed with a mask as 10\$\$, are used to produce the CASH row of the report.

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
10$$ AS 'CASH' OVER
1100 AS 'ACCOUNTS RECEIVABLE' OVER
1200 AS 'INVENTORY'
END
```

The output is shown as follows.

	AMOUNT
	-----
CASH	21,239
ACCOUNTS RECEIVABLE	18,829
INVENTORY	27,307

**Syntax:**      **How to Use the Same FOR Field Value in Multiple Rows**

You can use the same value of a FOR field in many separate rows (whether alone, as part of a range, or in a calculation) by including the following syntax before or within an FML request.

```
SET FORMULTIPLE={ON|OFF}
```

or

```
ON TABLE SET FORMULTIPLE {ON|OFF}
```

where:

```
ON
```

Enables you to reference the same value of a FOR field in more than one row in an FML request.

With FORMULTIPLE set to ON, a value retrieved from the data source is included on every line in the report output for which it matches the tag references.

OFF

Does not enable you to include the same value in multiple rows. OFF is the default value.

With FORMULTIPLE set to OFF, multiple tags referenced in any of these ways (OR, TO, \*) are evaluated first for an exact reference or for the end points of a range, then for a mask, and finally within a range. For example, if a value is specified as an exact reference and then as part of a range, the exact reference is displayed. Note that the result is unpredictable if a value fits into more than one row whose tags have the same priority (for example, an exact reference and the end point of a range).

For more information, see [Reporting Dynamically From a Hierarchy](#) on page 1764.

**Example: Referencing the Same Value in More Than One Row**

This request retrieves the tag values for accounts 1010, 1020, and 1030, and lists corresponding values individually. It then aggregates the same values and displays the sum as TOTAL CASH. Similarly, the tag values for accounts 1100 and 1200 displays as detail items, and then summarized as TOTAL NON-CASH ASSETS.

```
SET FORMULTIPLE=ON
TABLE FILE LEDGER
SUM AMOUNT
FOR ACCOUNT
1010 AS 'CASH ON HAND' OVER
1020 AS 'DEMAND DEPOSITS' OVER
1030 AS 'TIME DEPOSITS' OVER
BAR OVER
1010 OR 1020 OR 1030 AS 'TOTAL CASH' OVER
" " OVER
1100 AS 'ACCOUNTS RECEIVABLE' OVER
1200 AS 'INVENTORY' OVER
BAR OVER
1100 TO 1200 AS 'TOTAL NON-CASH ASSETS'
END
```

The output is shown as follows.

	AMOUNT
	-----
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
	-----
TOTAL CASH	21,239
ACCOUNTS RECEIVABLE	18,829
INVENTORY	27,307
	-----
TOTAL NON-CASH ASSETS	46,136

**Example: Using Tags From External Files**

In this example, the values for a row of the FML report come from an external file called CASHSTUF, which contains the following tags.

```
1010
1020
1030
```

The following TABLE request uses the tag values from the external file, summing the amounts in accounts 1010, 1020, and 1030 into the CASH row of the FML report.

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
(CASHSTUF) AS 'CASH' OVER
1100 AS 'ACCOUNTS RECEIVABLE'
END
```

Notice that the file name must be enclosed in parentheses.

**Using the BY Phrase in FML Requests**

Only one FOR phrase is permitted in a TABLE request. It substitutes in part for a BY phrase, which controls the sort sequence. However, the request can also include up to 32 BY phrases. In general, BY phrases specify the major (outer) sort fields in FML reports, and the FOR phrase specifies the minor (inner) sort field. Note that the BY ROWS OVER phrase is not supported in a request that uses the FOR phrase.



Combining BY and FOR Phrases in an FML Request

In this example, the report results for ACCOUNT (the inner sort field) are sorted by REGION (the outer sort field).

```
DEFINE FILE REGION
CUR_YR=E_ACTUAL;
LAST_YR=.831*CUR_YR;
REGION/A4=IF E_ACTUAL NE 0 OR E_BUDGET NE 0 THEN 'EAST' ELSE 'WEST';
END

TABLE FILE REGION
HEADING CENTER
"CURRENT ASSETS FOR REGION <REGION"
" "
SUM CUR_YR LAST_YR
BY REGION NOPRINT
FOR ACCOUNT
10$$ AS 'CASH' OVER
1100 AS 'ACCOUNTS RECEIVABLE' OVER
1200 AS 'INVENTORY' OVER
BAR OVER
RECAP CUR_ASSET/I5C = R1 + R2 + R3;
END
```

The output is shown as follows.

CURRENT ASSETS FOR REGION EAST		
	CUR_YR	LAST_YR
	-----	-----
CASH	9,511.00	7,903.64
ACCOUNTS RECEIVABLE	.	.
INVENTORY	.	.
	-----	-----
CUR_ASSET	9,511	7,903

A sort field value can be used in a RECAP command to allow the model to take different actions within each major sort break. For instance, the following calculation computes a non-zero value only for the EAST region.

```
RECAP X=IF REGION EQ 'EAST' THEN .25*CASH ELSE 0;
AS 'AVAILABLE FOR DIVIDENDS'
```

For more information, see [Performing Inter-Row Calculations](#) on page 1743.

Supplying Data Directly in a Request

In certain cases, you may need to include additional constants (such as exchange rates or inflation rates) in your model. Not all data values for the model have to be retrieved from the data source. Using FML, you can supply data directly in the request.

### **Syntax:** How to Supply Data Directly in a Request

```
DATA value,[..., value],$ [AS 'text'] [LABEL label] OVER
```

where:

*value*

Specifies the values that you are supplying. Values in a list must be separated by commas. The list must end with a comma and a dollar sign (,\$).

*AS 'text'*

Enables you to assign a title to the data row. Enclose the text in single quotation marks.

Without this entry, the row title is blank on the report.

*label*

Assigns a name to the data row for use in RECAP calculations. The label can be up to 66 characters and cannot have blanks or special characters. Each explicit label you assign must be unique.

### **Example:** Supplying Data Directly in a Request

In this example, two values (.87 and 1.67) are provided for the exchange rates of euros and pounds, respectively.

```
DEFINE FILE LEDGER
EUROS/I5C=AMOUNT;
POUNDS/I5C=3.2*AMOUNT;
END

TABLE FILE LEDGER
SUM EUROS AS 'EUROPE,DIVISION'
POUNDS AS 'ENGLISH,DIVISION'
FOR ACCOUNT
1010 AS 'CASH--LOCAL CURRENCY' LABEL CASH OVER
DATA .87, 1.67 , $ AS 'EXCHANGE RATE' LABEL EXCH OVER
RECAP US_DOLLARS/I5C = CASH * EXCH;
END
```

The values supplied are taken one column at a time for as many columns as the report originally specified.

The output is shown in the following image.

	EUROPE	ENGLISH
	<u>DIVISION</u>	<u>DIVISION</u>
CASH--LOCAL CURRENCY	8,784	28,108
EXCHANGE RATE	.87	1.67
US_DOLLARS	7,642	46,940

### Performing Inter-Row Calculations

The RECAP command enables you to perform calculations on data in the rows of the report to produce new rows. You must supply the name and format of the value that results from the calculation, and an expression that defines the calculation you wish to perform. Since RECAP calculations are performed among rows, each row in the calculation must be uniquely identified. FML supplies default row labels for this purpose (R1, R2, and so on). However, you may assign more meaningful labels. For more information, see [Referring to Rows in Calculations](#) on page 1744.

**Syntax:**      **How to Define Inter-Row Calculations**

```
RECAP calcname[/format]=expression; [AS 'text']
```

where:

*RECAP*

Is the required command name. It should begin on a line by itself.

*calcname*

Is the name you assign to the calculated value. The name can be up to 66 characters long, and must start with an alphabetic character. This name also serves as an explicit label. For more information, see [Referring to Rows in Calculations](#) on page 1744.

*format*

Is the USAGE format of the calculated value. It cannot exceed the column width. The default is the format of the column in which the calculated value is displayed.

*expression*

Can be any calculation available with the DEFINE command (including IF ... THEN ... ELSE syntax, functions, excluding DECODE and EDIT, and fields in date format). The expression may extend to as many lines as it requires. A semicolon is required at the end of the expression. For more information, see [Using Functions in RECAP Calculations](#) on page 1757 and the *Using Functions* manual.

The expression can include references to specific rows using the default FML positional labels (R1, R2, and so on), or it can refer to rows, columns, and cells using a variety of flexible notation techniques. Note that Rn references can only be used for rows previously evaluated within the model. For more information, see [Referring to Rows in Calculations](#) on page 1744, [Referring to Columns in Calculations](#) on page 1747, and [Referring to Cells in Calculations](#) on page 1755.

**AS** 'text'

Changes the default title of the row. By default, the name of the RECAP value is displayed as the row title in output. The AS phrase replaces the default. Enclose the text in single quotation marks.

### **Reference:** Usage Notes for RECAP

- ☐ RECAP expressions refer to other rows in the model by their labels (either explicit or default). Labels referred to in a RECAP expression must also be specified in the report request.
- ☐ The format specified for the RECAP result overrides the format of the column. In the following example,

```
RECAP TOTVAL/D6.2S=IF R1 GT R4 THEN R4 ELSE R1;  
AS 'REDUCED VALUE'
```

TOTVAL/D6.2S displays the result as six positions with two decimal places (and displays blanks if the value was zero) in each column of the report, regardless of the format of the data in the column. This feature can be used to display percentages in a column of whole numbers.

- ☐ Subtotals are not supported in FML.
- ☐ In environments that support the RETYPE command, note that RETYPE does not recognize labels in FML with field format redefinition.
- ☐ Rn references (default positional row labels) can only be used for rows previously evaluated within the model.

## Referring to Rows in Calculations

FML assigns a default positional label to each TAG, DATA, RECAP, and PICKUP row. These positional labels are automatically prefixed with the letter R, so that the first such row in the model is R1, the second is R2, and so on. You can use these labels to refer to rows in RECAP expressions.

**Note:** Default labels are not assigned to rows that contain underlines, blank lines, or free text, since these row types need not be referenced in expressions.

When you refer to rows in a RECAP expression, you can:

- ☐ Use the positional row label assigned by FML.
- ☐ Create an explicit row label of your own.

**Note:** You should not create an explicit label with a name of the form  $R_n$ , as that type of name is used for default positional row labels assigned by FML and may cause problems with subsequent RECAPs.

- ☐ Mix positional and explicit row labels.

If you assign an explicit label, the positional label (R1, R2, and so on) is retained internally.

Note that an explicit label is not needed for a RECAP row, because the name of the calculated value on the left of the equal sign can be used as a label.

In addition to their role in RECAP calculations, you can use labels to format rows in an FML report. For more information, see [Formatting an FML Report](#) on page 1782.

### **Syntax:** How to Assign an Explicit Row Label

```
rowtype [AS 'text'] LABEL label [OVER]
```

where:

*rowtype*

Can be a TAG, DATA, or PICKUP row.

AS '*text*'

Assigns a different name to the row for the report. Enclose the text in single quotation marks (').

*label*

Assigns a label to a row for reference in a RECAP expression or a StyleSheet declaration. The label can be up to 66 characters and cannot have blanks or special characters. Each explicit label you assign must be unique.

**Note:** You should not create an explicit label with a name of the form  $R_n$ , as that type of name is used for default positional row labels assigned by FML and may cause problems with subsequent RECAPs.

Even if you assign an explicit label, the positional label (R1, R2, and so on) is retained internally.

**Example:** Referring to Default Row Labels in RECAP Expressions

In this example, FML assigns account 1010 the implicit label R1, account 1020, the implicit label R2, and account 1030, the implicit label R3. Since no label is assigned to a BAR row, the RECAP row is assigned the implicit label R4.

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 AS 'CASH ON HAND'      OVER
1020 AS 'DEMAND DEPOSITS'   OVER
1030 AS 'TIME DEPOSITS'     OVER
BAR                          OVER
RECAP TOTCASH = R1 + R2 + R3; AS 'TOTAL CASH'
END
```

The output is shown as follows.

	AMOUNT
	-----
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
	-----
TOTAL CASH	21,239

**Referring to Explicit Row Labels in RECAP Expressions**

The following request assigns the labels CA, AR, and INV to three tag rows, which are referenced in the RECAP expression.

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
10$$ AS 'CASH'              LABEL CA    OVER
1100 AS 'ACCOUNTS RECEIVABLE' LABEL AR    OVER
1200 AS 'INVENTORY'         LABEL INV   OVER
BAR                          OVER
RECAP CURASST/I5C = CA + AR + INV;
END
```

The output is shown as follows.

	AMOUNT
	-----
CASH	21,239
ACCOUNTS RECEIVABLE	18,829
INVENTORY	27,307
	-----
CURASST	67,375

Note that the RECAP value could subsequently be referred to by the name CURASST, which functions as an explicit label.

### Using Labels to Repeat Rows

In certain cases, you may wish to repeat an entire row later in your report. For example, the CASH account can appear in the Asset statement and Cash Flow statement of a financial analysis, as shown below.

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
"ASSETS"                                OVER
10$$$ AS 'CASH' LABEL TOTCASH          OVER
.
.
"CASH FLOW"                             OVER
RECAP SAMECASH/I5C = TOTCASH; AS 'CASH'
END
```

When you refer to the CASH row the second time, you can use a RECAP calculation (with a new name) and refer to the label, either explicitly (TOTCASH) or implicitly (R1), in the row where CASH was first used.

**Tip:** If you set the FORMULTIPLE parameter ON, you can repeat the row without giving it another name. For more information, see [Creating Rows From Multiple Records](#) on page 1735.

### Referring to Columns in Calculations

An FML report can refer to explicit columns, as well as explicit rows. You can refer to columns using:

- ☐ Column numbers.
- ☐ Contiguous column notation in RECAP expressions. For example, (2,5) represents columns 2 through 5.
- ☐ Column addressing.
- ☐ A factor to represent every other column, or every third column, and so on.
- ☐ Column notation to control the creation of column reference numbers.
- ☐ Column values.

#### **Example:** Applying Column Declarations in RECAP Expressions

The following request generates an FML matrix with four rows and three columns of data.

```
DEFINE FILE LEDGER
CUR_YR/I5C=AMOUNT;
LAST_YR/I5C=.87*CUR_YR - 142;
END

TABLE FILE LEDGER
SUM CUR_YR AS 'CURRENT,YEAR'
    LAST_YR AS 'LAST,YEAR'
COMPUTE CHANGE/I5C = CUR_YR - LAST_YR;
FOR ACCOUNT
1010 AS 'CASH ON HAND'                                OVER
1020 AS 'DEMAND DEPOSITS'                              OVER
1030 AS 'TIME DEPOSITS'                                OVER
BAR                                                    OVER
RECAP TOTCASH/I5C = R1 + R2 + R3; AS 'TOTAL CASH'
END
```

Both the columns of the report, as well as the cells of the matrix, can be referenced in another FML report.

The output is shown in the following image.

	CURRENT	LAST	
	<u>YEAR</u>	<u>YEAR</u>	<u>CHANGE</u>
CASH ON HAND	8,784	7,216	1,568
DEMAND DEPOSITS	4,494	3,483	1,011
TIME DEPOSITS	7,961	6,499	1,462
TOTAL CASH	<u>21,239</u>	<u>17,198</u>	<u>4,041</u>

For example, you could use the value 6,499 in another FML report by referring to column 2, row 3. For more information, see [Referring to Cells in Calculations](#) on page 1755.

Referring to Column Numbers in Calculations

You can perform a calculation for one column or for a specific set of columns. To identify the columns, place the column number in parentheses after the label name.



**Example:** Referring to Column Numbers in a RECAP Expression

```
DEFINE FILE LEDGER
CUR_YR/I5C=AMOUNT;
LAST_YR/I5C=.87*CUR_YR - 142;
END

TABLE FILE LEDGER
SUM CUR_YR AS 'CURRENT,YEAR'
LAST_YR AS 'LAST,YEAR'
FOR ACCOUNT
1010 AS 'CASH ON HAND' OVER
1020 AS 'DEMAND DEPOSITS' OVER
1030 AS 'TIME DEPOSITS' OVER
BAR OVER
RECAP TOTCASH/I5C = R1 + R2 + R3; AS 'TOTAL CASH' OVER
" " OVER
RECAP GROCASH(2)/F5.2 = 100*TOTCASH(1)/TOTCASH(2) - 100;
AS 'CASH GROWTH(%)'
END
```

In the second RECAP expression, note that:

- ❑ TOTCASH(1) refers to total cash in column 1.
- ❑ TOTCASH(2) refers to total cash in column 2.
- ❑ The resulting calculation is displayed in column 2 of the row labeled CASH GROWTH(%).

The RECAP value is only calculated for the column specified.

The output is shown in the following image.

	CURRENT	LAST
	<u>YEAR</u>	<u>YEAR</u>
CASH ON HAND	8,784	7,216
DEMAND DEPOSITS	4,494	3,483
TIME DEPOSITS	7,961	6,499
TOTAL CASH	21,239	17,198
CASH GROWTH(%)		23.50

After data retrieval is completed, a single column is calculated all at once, and multiple columns one by one.

**Referring to Contiguous Columns in Calculations**

When a set of contiguous columns is needed within a RECAP, you can separate the first and last column numbers with commas. For example, DIFFERENCE (2,5) indicates that you want to compute the results for columns 2 through 5.

**Example:**    **Recapping Over Contiguous Columns**

In this example, the RECAP calculation for ATOT occurs only for columns 2 and 3, as specified in the request. No calculation is performed for column 1.

```
DEFINE FILE LEDGER
CUR_YR/I5C=AMOUNT;
LAST_YR/I5C=.87*CUR_YR - 142;
NEXT_YR/I5C=1.13*CUR_YR + 222;
END

TABLE FILE LEDGER
SUM NEXT_YR CUR_YR LAST_YR
FOR ACCOUNT
10$$ AS 'CASH'                                OVER
1100 AS 'ACCOUNTS RECEIVABLE'                 OVER
1200 AS 'INVENTORY'                           OVER
BAR                                            OVER
RECAP ATOT(2,3)/I5C = R1 + R2 + R3;
AS 'ASSETS--ACTUAL'
END
```

The output is shown in the following image.

	<u>NEXT_YR</u>	<u>CUR_YR</u>	<u>LAST_YR</u>
CASH	25,992	21,239	17,198
ACCOUNTS RECEIVABLE	21,941	18,829	15,954
INVENTORY	31,522	27,307	23,329
ASSETS--ACTUAL		67,375	56,481

**Referring to Column Addresses in Calculations**

When you need a calculation for every other or every third column instead of every column, you can supply a factor, or column address, to do this. Column addressing is useful when several data fields are displayed within each value of a column sort.

**Syntax:**    **How to Use Column Addressing in a RECAP Expression**

The left-hand side of the expression has the form:

```
value(s,e,i)[/format]=
```

where:

*value*

Is the name you assign to the result of the RECAP calculation.

*s*

Is the starting column.

*e*

Is the ending column (it may be \* to denote all columns).

*i*

Is the increment factor.

*format*

Is the USAGE format of the calculated value. The default value is the format of the original column.

### **Example: Applying Column Addressing in a RECAP Expression**

In the following statement, there are two columns for each month:

```
SUM ACTUAL AND FORECAST ACROSS MONTH
```

If you want to perform a calculation only for the ACTUAL data, control the placement of the results with a RECAP in the form:

```
RECAP calcname(1,*,2)=expression;
```

The asterisk means to continue the RECAP for all odd-numbered columns (beginning in column 1, with an increment of 2, for all columns).

### **Referring to Relative Column Addresses in Calculations**

A calculation can use a specific column as a base, and refer to all other columns by their displacement from that column. The column to the left of the base column has a displacement of -1 relative to the base column. The column to the right has a displacement of +1. For example,

```
COMP=FIX(*)-FIX(*-1);
```

can refer to the change in fixed assets from one period to the next. The reference to COMP=FIX(\*) is equivalent to COMP=FIX.

When referring to a prior column, the column must already have been retrieved, or its value is zero.

### Applying Relative Column Addressing in a RECAP Expression

This example computes the change in cash (CHGCASH) for columns 1 and 2.

```
DEFINE FILE LEDGER
CUR_YR/I5C=AMOUNT;
LAST_YR/I5C=.87*CUR_YR - 142;
NEXT_YR/I5C=1.13*CUR_YR + 222;
END

TABLE FILE LEDGER
SUM NEXT_YR CUR_YR LAST_YR
FOR ACCOUNT
10$$$ AS 'TOTAL CASH' LABEL TOTCASH          OVER
" "                                           OVER
RECAP CHGCASH(1,2)/I5C = TOTCASH(*) - TOTCASH(**+1); AS 'CHANGE IN CASH'
END
```

The output is shown in the following image.

	<u>NEXT_YR</u>	<u>CUR_YR</u>	<u>LAST_YR</u>
TOTAL CASH	25,992	21,239	17,198
CHANGE IN CASH	4,753	4,041	

### Controlling the Creation of Column Reference Numbers

Column notation assigns a sequential column number to each column in the internal matrix created for a report request. If you want to control the creation of column reference numbers for the columns that are used in your report, use the CNOTATION column notation command.

Because column numbers refer to columns in the internal matrix, they are assigned after retrieval and aggregation of data are completed. Columns created and displayed in a report are stored in the internal matrix, and columns that are not displayed in a report may also be generated and stored in the internal matrix. Columns stored in the internal matrix include calculated values, reformatted field values, BY fields, fields with the NOPRINT option, and certain RECAP calculations such as FORECAST and REGRESS. Every other column in the internal matrix is assigned a column number by default, which means you have to account for all internally generated columns, if you want to refer to the appropriate column value in your request.

You can change the default assignment of column reference numbers by using the SET CNOTATION=PRINTONLY command which assigns column numbers only to columns that display in the report output. You can use column notation in COMPUTE and RECAP commands to refer to these columns in your request.

**Syntax:**      **How to Control the Creation of Column Reference Numbers**

```
SET CNOTATION={ALL | PRINTONLY | EXPLICIT}
```

where:

ALL

Assigns column reference numbers to every column in the internal matrix. ALL is the default value.

PRINTONLY

Assigns column reference numbers only to columns that display in the report output.

EXPLICIT

Assigns column reference numbers to all fields referenced in the request, whether displayed or not.

**Note:** CNOTATION is not supported in an ON TABLE phrase.

**Referring to Column Values in Calculations**

When a report is sorted using the ACROSS phrase, all of the retrieved values are aligned under their appropriate columns. Each column has a title consisting of one value of the ACROSS field. The entire column of data can be addressed by this value in a RECAP calculation.

**Example:**      **Referring to a Column by Its Value in a RECAP Expression**

The following request uses a factor that depends on the value of the ACROSS field (YEAR) to calculate the inventory cost for each year. It then calculates the profit by summing the assets and subtracting the inventory cost for each year.

```
TABLE FILE LEDGER
SUM AMOUNT ACROSS YEAR
FOR ACCOUNT
10$$ AS 'CASH' LABEL CASH OVER
1100 AS 'ACCOUNTS RECEIVABLE' LABEL RECEIVE OVER
BAR OVER
1200 AS 'INVENTORY VALUE' LABEL INVENT OVER
RECAP INVENTORY_FACTOR/F5.2 = IF YEAR LT '1986'
THEN 1.1 ELSE 1.25; AS 'INVENTORY COST FACTOR' OVER
RECAP INVENTORY_COST = INVENTORY_FACTOR * INVENT;
AS 'INVENTORY COST' OVER
BAR OVER
RECAP PROFIT = CASH + RECEIVE - INVENTORY_COST;
END
```

The output is shown in the following image.

	YEAR		
	1985	1986	1987
CASH	5,663	7,001	8,575
ACCOUNTS RECEIVABLE	5,295	6,250	7,284
INVENTORY VALUE	7,754	9,076	10,477
INVENTORY COST FACTOR	1.10	1.25	1.25
INVENTORY COST	8,529	11,345	13,096
PROFIT	2,429	1,906	2,763

Referring to Rows and Columns in Calculations

You can style multiple RECAP commands in a matrix when the RECAP statements are placed after the last ACROSS value.

**Example: Styling Multiple RECAP Statements in a Matrix**

```
TABLE FILE GGSALES
SUM UNITS
BY PRODUCT
ACROSS REGION
RECAP
TTL1/I8 = C1+C2+C3+C4;
TTL2/D12.2 = TTL1*1.25;ON TABLE SET STYLE *
TYPE=DATA, COLUMN=TTL1 (*), COLOR=BLUE, BACKCOLOR=SILVER, STYLE=BOLD, $
TYPE=DATA, COLUMN=TTL2 (*), COLOR=RED, BACKCOLOR=AQUA, STYLE=BOLD, $
ENDSTYLE
END
```

The output is shown in the following image.

Product	Region				TTL1	TTL2
	Midwest	Northeast	Southeast	West		
Biscotti	86105	145242	119594	70436	421377	526,721.25
Capuccino	.	44785	73264	71168	189217	236,521.25
Coffee Grinder	50393	40977	47083	48081	186534	233,167.50
Coffee Pot	47156	46185	49922	47432	190695	238,368.75
Croissant	139182	137394	156456	197022	630054	787,567.50
Espresso	101154	68127	68030	71675	308986	386,232.50
Latte	231623	222866	209654	213920	878063	1,097,578.75
Mug	86718	91497	88474	93881	360570	450,712.50
Scone	116127	70732	73779	72776	333414	416,767.50
Thermos	46587	48870	48976	45648	190081	237,601.25

**Referring to Cells in Calculations**

You can refer to columns and rows using a form of cell notation that identifies the intersection of a row and a column as (r, c).

**Syntax:**      **How to Use Cell Notation for Rows and Columns in a RECAP Expression**

A row and column can be addressed in an expression by the notation:

`E(r, c)`

where:

`E`

Is a required constant.

`r`

Is the row number.

`c`

Is the column number. Use an asterisk (\*) to indicate the current column.

**Example:**      **Referring to Columns Using Cell Notation in a RECAP Expression**

In this request, two RECAP expressions derive VARIANCEs (EVAR and WVAR) by subtracting values in four columns (1, 2, 3, 4) in row three (PROFIT). These values are identified using cell notation (r,c).

```
TABLE FILE REGION
SUM  E_ACTUAL E_BUDGET W_ACTUAL W_BUDGET
FOR  ACCOUNT
3000 AS 'SALES'                                OVER
3100 AS 'COST'                                OVER
BAR                                             OVER
RECAP PROFIT/I5C = R1 - R2;                    OVER
" "                                             OVER
RECAP EVAR(1)/I5C = E(3,1) - E(3,2);
AS 'EAST--VARIANCE'                        OVER
RECAP WVAR(3)/I5C = E(3,3) - E(3,4);
AS 'WEST--VARIANCE'
END
```

The output is shown as follows.

	E_ACTUAL	E_BUDGET	W_ACTUAL	W_BUDGET
	-----	-----	-----	-----
SALES	6,000	4,934	7,222	7,056
COST	4,650	3,760	5,697	5,410
	-----	-----	-----	-----
PROFIT	1,350	1,174	1,525	1,646
EAST--VARIANCE	176			
WEST--VARIANCE			-121	



**Note:** In addition to illustrating cell notation, this example demonstrates the use of column numbering. Notice that the display of the EAST and WEST VARIANCES in columns 1 and 3, respectively, are controlled by the numbers in parentheses in the request: EVAR (1) and WVAR (3).

## Using Functions in RECAP Calculations

You may provide your own calculation routines in RECAP rows to perform special-purpose calculations, a useful feature when these calculations are mathematically complex or require extensive look-up tables.

User-written functions are coded as subroutines in any language that supports a call process, such as FORTRAN, COBOL, PL/1, and BAL. For information about creating your own functions, see the *Using Functions* manual.

### **Syntax:** How to Call a Function in a RECAP Command

```
RECAP calcname[(s,e,i)][/format]=function
(input1,...,inputn, 'format2');
```

where:

*calcname*

Is the name you assign to the calculated value.

(*s,e,i*)

Specify a start (s), end (e), and increment (i) value for the column where you want the value displayed. If omitted, the value appears in all columns.

*format*

The format for the calculation is optional. The default is the format of the column. If the calculation consists of only the subroutine, make sure that the format of the subroutine output value (*format2*) agrees with the calculation format. If the calculation format is larger than the column width, the value displays in that column as asterisks (\*).

*function*

Is the name of the function, up to eight characters long. It must be different from any row label and cannot contain any of the following special characters:

= - , / ( )

*input1, inputn*

Are the input arguments for the call to the function. They may include numeric constants, alphanumeric literals, row and column references notation, E notation, or labels, or names of other RECAP calculations.

Make sure that the values being passed to the function agree in number and type with the arguments as coded in the function.

*format2*

Is the format of the return value, which must be enclosed in single quotation marks.

### ***Example:* Calling a Function in a RECAP Command**

Suppose you have a function named INVEST in your private collection of functions (INVEST is not available in the supplied library), and it calculates an amount on the basis of cash on hand, total assets, and the current date. In order to create a report that prints an account of company assets and calculates how much money the company has available to invest, you must create a report request that invokes the INVEST function.

The current date is obtained from the &YMD system variable. The NOPRINT option beside it prevents the date from appearing in the report. The date is solely used as input for the next RECAP statement.

The request is:

```
TABLE FILE LEDGER
HEADING CENTER
"ASSETS AND MONEY AVAILABLE FOR INVESTMENT </2"
SUM AMOUNT ACROSS HIGHEST YEAR
IF YEAR EQ 1985 OR 1986
FOR ACCOUNT
1010 AS 'CASH'                                LABEL CASH          OVER
1020 AS 'ACCOUNTS RECEIVABLE'                LABEL ACR           OVER
1030 AS 'INTEREST RECEIVABLE'                LABEL ACI           OVER
1100 AS 'FUEL INVENTORY'                     LABEL FUEL          OVER
1200 AS 'MATERIALS AND SUPPLIES'              LABEL MAT           OVER
BAR                                           OVER
RECAP TOTCAS = CASH+ACR+ACI+FUEL+MAT; AS 'TOTAL ASSETS' OVER
BAR                                           OVER
RECAP THISDATE/A8 = &YMD; NOPRINT            OVER
RECAP INVAIL = INVEST(CASH,TOTCAS,THISDATE,'D12.2'); AS OVER
          'AVAIL. FOR INVESTMENT'           OVER
BAR AS '='
END
```

The output is shown in the following image.

ASSETS AND MONEY AVAILABLE FOR INVESTMENT		
	YEAR	
	1986	1985
CASH	2,100	1,684
ACCOUNTS RECEIVABLE	875	619
INTEREST RECEIVABLE	4,026	3,360
FUEL INVENTORY	6,250	5,295
MATERIALS AND SUPPLIES	9,076	7,754
TOTAL ASSETS	22,327	18,712
AVAIL. FOR INVESTMENT	3,481	2,994

Inserting Rows of Free Text

Insert text anywhere in your FML report by typing it on a line by itself and enclosing it within double quotation marks. You can also add blank lines, designated as text, to improve the appearance of the report.

In addition, you can include data developed in your FML report in a row of free text by including the label for the data variable in the text row.

**Example:**     Inserting Free Text

In this example, three rows of free text are inserted, one blank and two text rows.

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
  ▮ --- CASH ACCOUNTS --- ▮          OVER
1010 AS 'CASH ON HAND'              OVER
1020 AS 'DEMAND DEPOSITS'           OVER
1030 AS 'TIME DEPOSITS'             OVER
" "                                  OVER
  ▮ --- OTHER CURRENT ASSETS --- ▮   OVER
1100 AS 'ACCOUNTS RECEIVABLE'       OVER
1200 AS 'INVENTORY'
END
```

The output is shown as follows.

	AMOUNT
---	-----
--- CASH ACCOUNTS ---	
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
---	
--- OTHER CURRENT ASSETS ---	
ACCOUNTS RECEIVABLE	18,829
INVENTORY	27,307

Notice that the blank row was created by enclosing a blank within double quotation marks on a separate line of the report request.

**Syntax:**     How to Insert Data Variables in Text Rows

```
"text <label[(c)][>]"
```

where:

<

Is a required left caret to bracket the label.

label

Is the explicit or implicit row label. (In a RECAP, the calculated value functions as the label.)

c

Is an optional cell identifier that indicates the column number of the cell. However, this identifier is required whenever there is more than one column in the report. If you use it, enclose it in parentheses.

>

Is an optional right caret that can be used to make the positioning clearer.

**Example:**     **Inserting a Data Variable in a Text Row**

In this example, the RECAP value CURASST is suppressed by the NOPRINT command, and inserted instead as a data variable in the text row.

```
SET PAGE-NUM=OFF
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
10$$ AS 'Cash'                                LABEL CA      OVER
1100 AS 'Accounts Receivable'                 LABEL AR      OVER
1200 AS 'Inventory'                           LABEL INV     OVER
RECAP CURASST/I5C = CA + AR + INV; NOPRINT    OVER
"Current Assets: <CURASST"
END
```

The output is shown in the following image.

	AMOUNT
	-----
Cash	21,239
Accounts Receivable	18,829
Inventory	27,307
Current Assets:	67,375

**Adding a Column to an FML Report**

The request controls the number of columns in any report. For instance, if a request contains the display command SUM AMOUNT AND FORECAST, the report contains two columns: AMOUNT and FORECAST.

Add columns in an FML request, just as in a TABLE request, using the COMPUTE command to calculate a value or simply to allocate the space, column title, and format for a column.

**Example:**     **Adding a Column to an FML Report**

This example uses a COMPUTE command to generate the calculated value CHANGE and display it as a new column in the FML report. The following request generates an FML matrix with four rows and three columns of data.

```

DEFINE FILE LEDGER
CUR_YR/I5C=AMOUNT;
LAST_YR/I5C=.87*CUR_YR - 142;
END

TABLE FILE LEDGER
SUM CUR_YR AS 'CURRENT,YEAR'
    LAST_YR AS 'LAST,YEAR'
COMPUTE CHANGE/I5C = CUR_YR - LAST_YR;
FOR ACCOUNT
1010 AS 'CASH ON HAND' OVER
1020 AS 'DEMAND DEPOSITS' OVER
1030 AS 'TIME DEPOSITS' OVER
BAR OVER
RECAP TOTCASH/I5C = R1 + R2 + R3; AS 'TOTAL CASH'
END

```

The output is shown in the following image.

	CURRENT	LAST	
	<u>YEAR</u>	<u>YEAR</u>	<u>CHANGE</u>
CASH ON HAND	8,784	7,216	1,568
DEMAND DEPOSITS	4,494	3,483	1,011
TIME DEPOSITS	7,961	6,499	1,462
<b>TOTAL CASH</b>	<b>21,239</b>	<b>17,198</b>	<b>4,041</b>

**Note:** The designated calculation is performed on each tag or RECAP row of the report. The RECAP rows, however, may change the calculation.

### Adding a New Time Period as a Column

The following request adds a future time period to a report.

```

DEFINE FILE LEDGER
CUR_YR/P5C=AMOUNT;
LAST_YR/P5C=.87*AMOUNT - 142;
END

TABLE FILE LEDGER
SUM AMOUNT
ACROSS YEAR AND COMPUTE 1999/P5C = 2.5*AMOUNT;
FOR ACCOUNT
1010 AS 'CASH ON HAND' OVER
1020 AS 'DEMAND DEPOSITS' OVER
1030 AS 'TIME DEPOSITS' OVER
BAR OVER
RECAP TOTCASH/P5C = R1 + R2 + R3; AS 'TOTAL CASH' OVER
RECAP CHANGE(2,*) = TOTCASH(*) - TOTCASH(*-1);
END

```

The output is shown as follows.

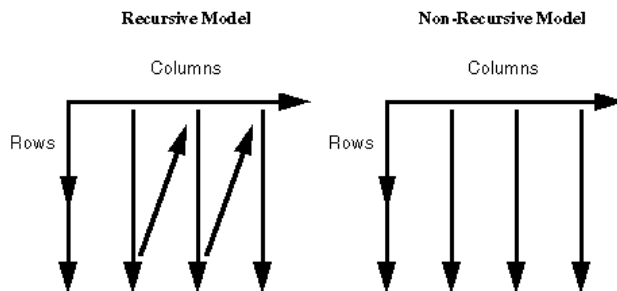
	YEAR			
	1985	1986	1987	1999
CASH ON HAND	1,684	2,100	5,000	4,210
DEMAND DEPOSITS	619	875	3,000	1,548
TIME DEPOSITS	3,360	4,026	575	8,400
TOTAL CASH	5,663	7,001	8,575	14,158
CHANGE		1,338	1,574	5,583

## Creating a Recursive Model

Models involving different time periods often require using the ending value of one time period as the starting value for the next. The calculations describing these situations have two characteristics:

- ❑ The labels on one or more RECAP rows are duplicates of other rows. They are used repeatedly to recompute certain values.
- ❑ A calculation may refer to a label not yet described, but provided later in the model. If, at the end of the model, a label that is needed is missing, a message is displayed.

Recursive models require that the columns are produced in sequential order, one by one. In nonrecursive models, all of the columns can be produced simultaneously. Schematically, these patterns are shown as follows.



FML automatically switches to sequential order as soon as either of the two modeling conditions requiring the switch is recognized (either reuse of labels by different rows, or forward reference to a label in a calculation).

**Example:    Creating a Recursive Model**

The following example illustrates recursive models. Note that one year of ENDCASH becomes the next year of STARTING CASH.

```
DEFINE FILE REGION
CUR_YR=E_ACTUAL;
LAST_YR=.831*CUR_YR;
NEXT_YR=1.2297*CUR_YR;
END

TABLE FILE REGION
SUM LAST_YR CUR_YR NEXT_YR
FOR ACCOUNT
10$$ AS 'STARTING CASH' LABEL STCASH          OVER
RECAP STCASH(2,*) = ENDCASH(*-1);             OVER
" "                                           OVER
3000 AS 'SALES' LABEL SLS                     OVER
3100 AS 'COST' LABEL COST                     OVER
BAR                                           OVER
RECAP PROFIT/I5C = SLS - COST;                 OVER
" "                                           OVER
RECAP ENDCASH/I5C = STCASH + PROFIT;
END
```

The output is shown as follows.

	<u>LAST_YR</u>	<u>CUR_YR</u>	<u>NEXT_YR</u>
STARTING CASH	7,903.64	9,024.00	10,374.00
SALES	4,986.00	6,000.00	7,378.20
COST	3,864.15	4,650.00	5,718.11
PROFIT	<u>1,121</u>	<u>1,350</u>	<u>1,660</u>
ENDCASH	9,024	10,374	12,034

**Reporting Dynamically From a Hierarchy**

Hierarchical relationships between fields can be defined in a Master File, and automatically displayed using the Financial Modeling Language (FML). The parent and child fields must share data values, and their relationship should be hierarchical. The formats of the parent and child fields must both be either numeric or alphanumeric.



For example, suppose that:

- ❑ Employee and manager IDs are contained within an employee data source.

or

- ❑ A general ledger data source contains both an account number field and an account parent field.

By examining these fields, it is possible to construct the entire organization chart or chart of accounts structure. However, to print the chart in a traditional FML report, you need to list the employee IDs or account numbers in the request syntax in the order in which they should appear on the report. If an employee or account is added, removed, or transferred, you have to change the report request to reflect this change in organizational structure. For example:

```
TABLE FILE EMPLOYEE
PRINT DEPARTMENT CURR_JOBCODE
FOR EMP_ID
999999999 OVER
222222222 OVER
.
.
.
```

In contrast, with FML hierarchies you can define the hierarchical relationship between two fields in the Master File and load this information into memory. The FML request can then dynamically construct the rows that represent this relationship and display them in the report, starting at any point in the hierarchy. In the example shown, EMP\_ID is called the hierarchy field.

## Requirements for FML Hierarchies

1. In the Master File, use the PROPERTY=PARENT\_OF and REFERENCE=*hierarchyfld* attributes to define the hierarchical relationship between two fields. For more information, see the *Describing Data With WebFOCUS Language* manual.

The hierarchy must be loaded into memory. This loaded hierarchy is called a chart. If the hierarchy is defined in the Master File and referenced by the FML request, it is loaded automatically. If you want to use a hierarchy defined in a Master File that is not either referenced in the FML request or joined to the Master File referenced in the FML request, issue the LOAD CHART command before issuing the FML request.

The number of charts that can be loaded is 16. Charts are automatically unloaded when the session ends.

2. In the FOR phrase of the FML request. Use the GET/WITH CHILDREN or ADD phrase to retrieve the hierarchical data starting at a specific point in the hierarchy.

To use FML hierarchies, the FOR field must either be:

- ☐ The hierarchy field.

or

- ☐ Used as the join field to a unique segment that has the hierarchy field. In this case, the hierarchy field must be the join field. Note that the condition that the join be unique only applies if the hierarchy is defined in the cross-referenced segment.

In other words, the FOR field must be in a parent-child hierarchy, or linked to one. The latter case allows transaction data that contains the hierarchy field to be joined to a separate data source that contains the hierarchy definition.

As with any FML request, a tagged row is displayed even if no data is found in the file for the tag values, with a period (.) representing the missing data. You can override this convention by adding the phrase WHEN EXISTS to the definition of a tagged row. This makes displaying a row dependent upon the existence of data for the tag.

**Note:** In order for the hierarchical indentations to be retained in HTML output, the setting SHOWBLANKS=ON must be in effect.

### **Example:** Defining a Hierarchy in a Master File

The CENTGL Master File contains a charts of accounts hierarchy. The field GL\_ACCOUNT\_PARENT is the parent field in the hierarchy. The field GL\_ACCOUNT is the hierarchy field. The field GL\_ACCOUNT\_CAPTION can be used as the descriptive caption for the hierarchy field.

```
FILE=CENTGL          , SUFFIX=FOC
SEGNAME=ACCOUNTS, SEGTYPE=S01
FIELDNAME=GL_ACCOUNT,          ALIAS=GLACCT,   FORMAT=A7,
                                TITLE='Ledger,Account', FIELDTYPE=I, $
FIELDNAME=GL_ACCOUNT_PARENT,   ALIAS=GLPAR,    FORMAT=A7,
                                TITLE=Parent,
                                PROPERTY=PARENT_OF, REFERENCE=GL_ACCOUNT, $
FIELDNAME=GL_ACCOUNT_TYPE,     ALIAS=GLTYPE,   FORMAT=A1,
                                TITLE=Type,$
FIELDNAME=GL_ROLLUP_OP,        ALIAS=GLROLL,   FORMAT=A1,
                                TITLE=Op, $
FIELDNAME=GL_ACCOUNT_LEVEL,    ALIAS=GLLEVEL,  FORMAT=I3,
                                TITLE=Lev, $
FIELDNAME=GL_ACCOUNT_CAPTION,  ALIAS=GLCAP,    FORMAT=A30,
                                TITLE=Caption,
                                PROPERTY=CAPTION, REFERENCE=GL_ACCOUNT, $
FIELDNAME=SYS_ACCOUNT,         ALIAS=ALINE,    FORMAT=A6,
                                TITLE='System,Account,Line', MISSING=ON, $
```

The CENTSYSF data source contains detail-level financial data. This is unconsolidated financial data for a fictional corporation, CenturyCorp. It is designed to be separate from the CENTGL database as if it came from an external accounting system. It uses a different account line system (SYS\_ACCOUNT) which can be joined to the SYS\_ACCOUNT field in CENTGL. Data uses *natural* signs (expenses are positive, revenue negative).

```
FILE=CENTSYSF      ,SUFFIX=FOC
SEGNAME=RAWDATA    ,SEGTYPE=S2
FIELDNAME=SYS_ACCOUNT , ,A6      , FIELDTYPE=I,
                    TITLE='System,Account,Line', $
FIELDNAME=PERIOD    , ,YYM      , FIELDTYPE=I, $
FIELDNAME=NAT_AMOUNT , ,D10.0   , TITLE='Month,Actual', $
FIELDNAME=NAT_BUDGET , ,D10.0   , TITLE='Month,Budget', $
FIELDNAME=NAT_YTDAMT , ,D12.0   , TITLE='YTD,Actual', $
```

## Displaying an FML Hierarchy

The GET CHILDREN and WITH CHILDREN commands dynamically retrieve and display hierarchical data on the FML report. GET CHILDREN displays only the children, not the parent value referenced in the command. WITH CHILDREN displays the parent and then the children.

### **Syntax:** How to Display an FML Hierarchy

```
TABLE FILE filename{PRINT|SUM} ...
FOR hierarchyfld
parentvalue {GET|WITH} CHILD[REN] [n|ALL]
  [AS CAPTION|'text'] [LABEL label]
.
.
.
END
```

where:

*filename*

Is the name of the file to be used in the FML request. If the hierarchy for this request cannot be loaded automatically, it must have been loaded previously by issuing the LOAD CHART command.

*hierarchyfld*

Is the hierarchy field name. If the request references a joined structure, the name must be the field name from the host file. The alias name is not supported.

*parentvalue*

Is the parent value for which the children are to be retrieved.

#### GET CHILDREN

Displays the hierarchy starting from the first child of the specified *parentvalue*. It does not include the parent in the display. (This corresponds to the FML syntax CHILD1 OVER CHILD2 OVER ...)

#### WITH CHILDREN

Displays the hierarchy starting from the specified *parentvalue*. It includes the parent in the display. (This corresponds to the FML syntax *parentvalue* OVER CHILD1 OVER CHILD2 OVER ...).

#### *n* | ALL

Is a positive integer from 1 to 99, specifying the number of levels of the hierarchy to display. If a number greater than 99 is specified, a warning message is displayed and *n* is set to 99. The default value is 1. Therefore, if *n* is omitted, only direct children are displayed. GET or WITH CHILDREN 2 displays direct children and grandchildren. GET or WITH CHILDREN 99 displays children to 99 levels. ALL is a synonym for 99. Each child instance is printed over the one that follows. Successive levels of the hierarchy field are indented two spaces from the previous level.

#### CAPTION

Indicates that the caption values to display should be taken from the field defined as the CAPTION in the Master File.

Note that the AS CAPTION phrase is supported for tagged rows, including those that do not use the GET/WITH CHILDREN or ADD syntax. However, the hierarchy must be defined (by specifying the PARENT\_OF attribute) in order to load and display the caption values. If the hierarchy is not defined, the AS CAPTION phrase is ignored.

#### 'text'

Is a text string to use as the row title for the hierarchy field values. The CAPTION field defined in the Master File is not used as the caption on the report output.

#### label

Is an explicit row label. Each generated row is labeled with the specified label text.

**Note:** The hierarchy is displayed sorted by the parent field and, within parent, sorted by the hierarchy field.

For information about the FMLFOR, FMLLIST, FMLCAP, and FMLINFO functions that return the tag values and captions used in an FML request, see the Using Functions manual.

**Example:**    **Displaying an FML Hierarchy**

The following request displays two levels of account numbers, starting from account 3000:

```
SET SHOWBLANKS=ON
TABLE FILE CENTGL
PRINT GL_ACCOUNT_PARENT
FOR GL_ACCOUNT
3000 WITH CHILDREN 2
END
```

The output is shown as follows.

	Parent
	-----
3000	1000
3100	3000
3110	3100
3120	3100
3130	3100
3140	3100
3200	3000
3300	3200
3400	3200
3500	3200
3600	3200
3700	3200
3800	3200
3900	3200

**Note:**

- ❑ If the request specifies GET CHILDREN instead of WITH CHILDREN, the line for the parent value (3000) does not display on the report output.
- ❑ In order to retain the indentations in HTML output, the SET SHOWBLANKS=ON command must be in effect.

**Displaying an FML Hierarchy With Captions**

The following request displays two levels of a chart of accounts hierarchy, starting with account 1000 (the top of the hierarchy), and displays the caption field values instead of the account numbers.

```
SET SHOWBLANKS=ON
TABLE FILE CENTGL
PRINT GL_ACCOUNT_PARENT
FOR GL_ACCOUNT
1000 WITH CHILDREN 2 AS CAPTION
END
```

The output is shown as follows.

	Parent
	-----
Profit Before Tax	
Gross Margin	1000
Sales Revenue	2000
Cost Of Goods Sold	2000
Total Operating Expenses	1000
Selling Expenses	3000
General + Admin Expenses	3000
Total R+D Costs	1000
Salaries	5000
Misc. Equipment	5000

**Note:** If the request specifies GET CHILDREN instead of WITH CHILDREN, the line for the parent value (1000, Profit Before Tax) does not display on the report output.

Consolidating an FML Hierarchy

The ADD command consolidates multiple levels of the hierarchy on one line of the FML report output. You can use ADD alone or in conjunction with GET CHILDREN or WITH CHILDREN. Note that ADD is designed to work with requests that use the SUM command. It is also designed to be used with detail-level data, not data that is consolidated.

When used alone, ADD aggregates the parent and children on one line of the report output, summing the numeric data values included on the line. This corresponds to the FML syntax *parentvalue* or CHILD1 OR CHILD2 OR ...

When used in conjunction with GET CHILDREN, ADD displays one line for each child of the specified parent value. Each line is a summation of that child and all of its children. You can specify the number of levels of children to display (which determines the number of lines generated on the report output) and the depth of summation under each child. By default, only direct children have a line in the report output, and the summary for each child includes all of its children.

When used in conjunction with WITH CHILDREN, ADD first displays a line in the report output that consists of the summation of the parent value and all of its children. Then it displays additional lines identical to those displayed by GET CHILDREN ADD.

In order to use a data record in more than one line of an FML report (for example, to display both detail and summary lines or to consolidate detail data at multiple levels), the following setting is required:

```
SET FORMULTIPLE=ON
```

### **Syntax:** How to Create One Summary Row for an FML Hierarchy

```
TABLE FILE filename SUM ...
FOR hierarchyfld
parentvalue ADD [ n | ALL ]
  [ AS CAPTION ' text ' ] [ LABEL label ]
.
.
.
END
```

where:

*filename*

Is the name of the file to be used in the FML request. If the hierarchy for this request cannot be loaded automatically, it must have been loaded previously by issuing the LOAD CHART command.

*hierarchyfld*

Is the hierarchy field name. If the request references a joined structure, the name must be the field name from the host file. The alias name is not supported.

*parentvalue*

Is the parent value that determines the starting point in the hierarchy for the aggregation.

ADD

Displays the parent and *n* levels of its children on one row, summing the numeric data values displayed on the row. This corresponds to the FML syntax *parentvalue* or CHILD1 OR CHILD2 OR CHILD3 and more, if applicable.

To display the sum of just the children, you must display the parent row, display the summary row, and use a RECAP to subtract the parent row from the sum. For example:

```
FOR ...  
parentvalue                                OVER  
parentvalue ADD 1                          OVER  
RECAP CHILDSUM = R2-R1;
```

*n*|ALL

Is a positive integer from 1 to 99, specifying the number of levels of the hierarchy to aggregate. ALL is the default value. Therefore, if *n* is omitted, all children are included in the sum. If *n* is 1, only direct children are included. If *n* is 2, direct children and grandchildren are included. ADD 99 includes up to 99 levels of children. ALL is a synonym for 99.

CAPTION

Indicates that the caption of the parent value displays for the total row.

Note that the AS CAPTION phrase is supported for tagged rows, including those that do not use the GET CHILDREN or ADD syntax. However, the hierarchy must be defined (by specifying the PARENT\_OF attribute) in order to load and display the caption values. If the hierarchy is not defined, the AS CAPTION phrase is ignored.

'*text*'

Is a text string to use as the row title for the aggregate row. The CAPTION field defined in the Master File is not used as the caption on the report output.

*label*

Is an explicit row label. Each generated row is labeled with the specified label text.



**Example: Displaying One Summary Line for an FML Hierarchy**

The CENTSYSF data source contains detail-level financial data. To use the account hierarchy in the CENTGL data source with this financial data, the two data sources are joined. The data in CENTSYSF is stored with natural signs, which means, in financial terms, that revenues and liabilities are stored as negative numbers. The portion of the hierarchy used in this request contains only positive data.

Note that the join is not required to be unique, because the hierarchy is defined in the host segment.

First the WITH CHILDREN command displays the lines of the hierarchy starting with account Selling Expenses (3100). Note that only accounts with no children are populated in this detail-level data source. The ADD command then creates one line that is the sum of account 3100 and all of its children.

```
SET SHOWBLANKS=ON
SET FORMULTIPLE=ON
JOIN SYS_ACCOUNT IN CENTGL TO ALL SYS_ACCOUNT IN CENTSYSF
TABLE FILE CENTGL
SUM NAT_AMOUNT/D10.0 NAT_YTDAMT/D10.0
FOR GL_ACCOUNT
3100 WITH CHILDREN ALL AS CAPTION OVER
BAR
3100 ADD AS CAPTION
IF PERIOD EQ '2002/03'
END
```

The output is shown as follows.

	Month	YTD
	<u>Actual</u>	<u>Actual</u>
Selling Expenses	.	.
Advertising	.	.
TV/Radio	1,049,146.	2,954,342.
Print Media	244,589.	721,448.
Internet Advertising	9,542.	29,578.
Promotional Expenses	53,719.	151,732.
Joint Marketing	97,135.	289,799.
Bonuses/Commissions	100,188.	304,199.
	<hr/>	<hr/>
Selling Expenses	1,554,319.	4,451,098.

**Syntax:**      **How to Consolidate FML Hierarchy Data to Any Level and Depth**

```
TABLE FILE filename
SUM ...
FOR hierarchyfld
parentvalue {GET|WITH} CHILD[REN] [n|ALL] ADD [m|ALL]
  [AS CAPTION 'text' ] [LABEL label]
.
.
.
END
```

where:

*filename*

Is the name of the file used in the FML request. If the hierarchy for this request cannot load automatically, it previously loaded by issuing the LOAD CHART command.

*hierarchyfld*

Is the hierarchy field name. If the request references a joined structure, the name must be the field name from the host file. The alias name is not supported.

*parentvalue*

Is the parent value that determines the starting point in the hierarchy for the aggregation.

GET|WITH

GET specifies that the first line generated on the report is the consolidated line for the first child of the parent value. WITH specifies that the first line generated on the report is the consolidated line for the parent value, followed by the consolidated lines for each of its children, to the level specified by *n*.

*n*|ALL

Is a positive integer from 1 to 99, specifying the number of levels of children to display. The line of output for each child has the sum of that child and its children to the depth specified for the ADD option. The default value is 1. Therefore, if *n* is omitted, each direct child has a line on the report. If *n* is 2, direct children and grandchildren each have a line on the report output. ALL is a synonym for 99.

ADD

Sums the hierarchy to the depth specified by *m* for each line generated by the GET or WITH CHILDREN command.

*m*|ALL

Is a positive integer from 1 to 99, specifying the number of levels of children to consolidate on each line of the report output. If a number greater than 99 is specified, a warning message is displayed and *m* is set to 99. The default value is ALL. Therefore, if *m* is omitted, the consolidated line sums all children. If *m* is 2, only direct children and grandchildren are consolidated for each line on the report output. ADD 99 aggregates children to 99 levels. ALL is a synonym for 99.

CAPTION

Indicates that the caption of the parent value displays for the total row.

Note that the AS CAPTION phrase is supported for tagged rows, including those that do not use the GET CHILDREN or ADD syntax. However, the hierarchy must be defined (by specifying the PARENT\_OF attribute) in order to load and display the caption values. If the hierarchy is not defined, the AS CAPTION phrase is ignored.

'text'

Is a text string to use as the row title for the aggregate row. The CAPTION field defined in the Master File is not used as the caption on the report output.

*label*

Is an explicit row label. Each generated row is labeled with the specified label text.

### **Example: Consolidating FML Hierarchy Data**

In the following request, the first WITH CHILD command displays the detail data for the hierarchy starting with account 3100. The next WITH CHILD command creates a consolidated line for the parent account (3100) and each direct child.

```
SET SHOWBLANKS=ON
SET FORMULTIPLE=ON
JOIN SYS_ACCOUNT IN CENTGL TO ALL SYS_ACCOUNT IN CENTSYSF
TABLE FILE CENTGL
SUM NAT_AMOUNT/D10.0 NAT_YTDAMT/D10.0
FOR GL_ACCOUNT
3100 WITH CHILDREN ALL AS CAPTION      OVER
" "                                     OVER
BAR AS =                               OVER
" "                                     OVER
3100 WITH CHILDREN ADD AS CAPTION
IF PERIOD EQ '2002/03'
END
```

Note that the join is not required to be unique, because the hierarchy is defined in the host segment.

In the following output, the top portion shows the detail-level data. The bottom portion shows the consolidated data. In the consolidated portion of the report:

- ❑ There is one line for the parent that is the sum of itself plus all of its children to all levels.
- ❑ There is one line for each direct child of account Selling Expenses (3100): Advertising, Promotional Expenses, Joint Marketing, and Bonuses/Commissions.
- ❑ The line for Advertising is the sum of itself plus all of its children. If it has multiple levels of children, they are all added into the sum. The other direct children of 3100 do not themselves have children, so the sum on each of those lines consists of only the parent value.

	Month	YTD
	<u>Actual</u>	<u>Actual</u>
Selling Expenses	.	.
Advertising	.	.
TV/Radio	1,049,146.	2,954,342.
Print Media	244,589.	721,448.
Internet Advertising	9,542.	29,578.
Promotional Expenses	53,719.	151,732.
Joint Marketing	97,135.	289,799.
Bonuses/Commissions	100,188.	304,199.
	<hr/>	<hr/>
Selling Expenses	1,554,319.	4,451,098.
Advertising	1,303,277.	3,705,368.
Promotional Expenses	53,719.	151,732.
Joint Marketing	97,135.	289,799.
Bonuses/Commissions	100,188.	304,199.

Using GET CHILDREN instead of WITH CHILDREN eliminates the top line from each portion of the output. The remaining lines are the same.

	Month	YTD
	<u>Actual</u>	<u>Actual</u>
Advertising	.	.
TV/Radio	1,049,146.	2,954,342.
Print Media	244,589.	721,448.
Internet Advertising	9,542.	29,578.
Promotional Expenses	53,719.	151,732.
Joint Marketing	97,135.	289,799.
Bonuses/Commissions	100,188.	304,199.
	-----	-----
Advertising	1,303,277.	3,705,368.
Promotional Expenses	53,719.	151,732.
Joint Marketing	97,135.	289,799.
Bonuses/Commissions	100,188.	304,199.

The following request displays a consolidated line for account 2000 and each of its direct children and grandchildren.

```
SET SHOWBLANKS=ON
SET FORMULTIPLE=ON
JOIN SYS_ACCOUNT IN CENTGL TO ALL SYS_ACCOUNT IN CENTSYSF
TABLE FILE CENTGL
SUM NAT_AMOUNT/D10.0 NAT_YTDAMT/D10.0
FOR GL_ACCOUNT
2000 WITH CHILDREN 2 ADD AS CAPTION
IF PERIOD EQ '2002/03'
END
```

The output is shown as follows.

	Month	YTD
	<u>Actual</u>	<u>Actual</u>
Gross Margin	-4,513,659.	-13,080,549.
Sales Revenue	-10,398,305.	-30,877,546.
Retail Sales	-8,237,253.	-24,539,197.
Mail Order Sales	-1,138,414.	-3,403,387.
Internet Sales	-1,022,638.	-2,934,962.
Cost Of Goods Sold	5,884,646.	17,796,997.
Variable Material Costs	4,415,560.	13,410,629.
Direct Labor	961,143.	2,920,449.
Fixed Costs	507,943.	1,465,919.

Loading a Hierarchy Manually

In most cases, a hierarchy loads automatically as a result of the request syntax. However, if you need to use a hierarchy defined in one Master File against a data source that is not joined to the hierarchy file (but that contains the same hierarchy field), you can manually load the hierarchy data using the LOAD CHART command.

The number of charts that can load is limited by available memory. Charts automatically unload when the session ends.

The chart loads by running a TABLE request that produces a list of parent values and their associated children.

```
TABLE FILE chartfile
BY parentfield BY hierarchyfield
[SUM captionfield]
END
```

The resulting chart contains the following information. It may also contain the associated captions, depending on whether the AS CAPTION phrase was used in the request.

<i>parentfield</i>	<i>hierarchyfield</i>
-----	-----
<i>parentvalue1</i>	<i>child1</i>
<i>parentvalue1</i>	<i>child2</i>
<i>parentvalue2</i>	<i>child3</i>
.	
.	
.	

### **Syntax:** How to Load a Hierarchy From One Master File for Use With a Separate Master File

You can manually load the hierarchy data, if you need to use a hierarchy defined in one Master File, against a data source that is not joined to the hierarchy file but that contains the same hierarchy field.

Available memory dictates the number of charts that can load. Charts automatically unload when WebFOCUS terminates.

```
LOAD CHART chartfile[.sega].hierarchyfld
  [FOR requestfile[.segb].fieldb]
```

where:

*chartfile*

Is the name of the Master File that contains the hierarchy information.

*sega*

Is the name of the segment that contains the hierarchy field. The segment name is only required if a field in another segment in the structure has the same field name as the hierarchy field.

*hierarchyfld*

Is the hierarchy field. It is required because a Master File can define multiple hierarchies.

FOR

Loads a hierarchy defined in a Master File that is not used in the FML report request. For example, if Master File B contains the hierarchy information but Master File A is used in the request (without a join between Master Files A and B), issue the following LOAD CHART command prior to the FML request:

```
LOAD CHART B.FLDB FOR A.FLDA
TABLE FILE A ...
```

### *requestfile*

Is the name of the Master File used in the FML request.

### *segb*

Is the name of the segment that contains the hierarchy field values in the Master File used in the FML request. It is not required if it has the same name as *sega*.

### *fieldb*

Is the field in the Master File specified in the FML request that contains the values of the hierarchy field. It is not required if it has the same name as the hierarchy field.

### **Note:**

- ☐ If you issue the LOAD CHART command multiple times for the same hierarchy, the new hierarchy overlays the previous version in memory.
- ☐ If you issue the LOAD CHART command for a data source that is dynamically joined to the hierarchy file, you must issue the JOIN command prior to issuing the LOAD CHART command.

## **Reference:** Usage Notes for FML Hierarchies

- ☐ PROPERTY and REFERENCE are propagated to HOLD Master Files when HOLDATTR is set to ON.
- ☐ The following setting is required in order to use a data record in more than one row of an FML request (for example, both a detail and summary row):  
`SET FORMULTIPLE=ON`
- ☐ When reporting against a rolled-up data source such as ESSBASE, the data values stored for the parent instance are an aggregate of all of its children. Do not use the ADD feature on consolidated data.
- ☐ When reporting against a data source with shared members (such as ESSBASE), in which the same data can be defined multiple times with different hierarchy field values, data shared by two different parents is counted twice in an aggregation operation. To avoid this double aggregation, use the FST operator in the SUM command for the shared fields.
- ☐ When the report output is in HTML format, the setting SHOWBLANKS=ON must be in effect in order to retain the hierarchical indentations.



## Customizing a Row Title

You can customize a row title in an FML report for accurate data identification. Using the AS phrase, you can provide new titles for TAG, DATA, RECAP, and PICKUP rows.

### **Syntax:** How to Customize a Row Title in FML

For a TAG row, use the syntax:

```
value AS {'title'|CAPTION}
```

For a DATA or PICKUP row, use the syntax:

```
value AS 'title'
```

For a RECAP row, use the syntax:

```
RECAP calcname[/format]=expression; AS 'title'
```

where:

*value*

Is the value on which you are reporting, whether retrieved from a data source or external file (represented by a tag), supplied directly by a user in the request, or picked up from a work file.

*title*

Is the customized row title, enclosed in single quotation marks if it contains embedded blanks.

In a TAG, DATA, or PICKUP row, the default row title is *value*.

In a RECAP row, the default title is *calcname*.

CAPTION

In the Master File of a hierarchical data source, CAPTION identifies a TAG row using a caption field. Note that the hierarchy in the Master File defines the PARENT-OF the FOR field.

*calcname*

Is the value that is derived by the RECAP calculation.

### **Example:** Changing the Titles of Tag Rows

In the following example, the row titles CASH ON HAND and DEMAND DEPOSITS provide meaningful identifications for the corresponding tags.

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 AS 'CASH ON HAND' OVER
1020 AS 'DEMAND DEPOSITS'
END
```

Note that single quotation marks are necessary since the row title being assigned has embedded blanks.

The output is shown as follows.

	AMOUNT
	-----
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494

If no AS phrase is included, the tag values are displayed in the report.

**Customizing a Row Title for a RECAP Value**

This request creates the title TOTAL CASH for the RECAP value TOTCASH.

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 AS 'CASH ON HAND' OVER
1020 AS 'DEMAND DEPOSITS' OVER
1030 AS 'TIME DEPOSITS' OVER
RECAP TOTCASH = R1 + R2 + R3; AS 'TOTAL CASH'
END
```

The output is shown as follows.

	AMOUNT
	-----
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
TOTAL CASH	21,239

If no AS phrases are included, the name of the RECAP value (TOTCASH) displays in the report.

**Formatting an FML Report**

Improve the readability and presentation of your FML report by:

- ☐ **Underlining numeric columns.** Reports with columns of numbers frequently need to display underlines before some RECAP calculations. You can specify an underline character, introduced by the word BAR, in place of the tag value.
- ☐ **Adding page breaks.** You can request a new page at any point in a report by placing the word PAGE-BREAK in place of the tag value.

- ❑ **Formatting rows, columns, and cells.** You can apply StyleSheet attributes, such as FONT, SIZE, STYLE, and COLOR, to individual rows and columns, or to cells within those rows.
- ❑ **Adding borders around rows, columns, and cells.** You can use BORDER attributes in a StyleSheet to specify the weight, style, and color of border lines around a row or cell. You can specify formatting variations for the top, bottom, left, and right borders.
- ❑ **Indenting text or numbers.** You can indent a tag value, label text, or caption text a specified number of spaces for an FML tag row, hierarchy, or RECAP row. If you apply the indent to rows in an FML hierarchy, the parent line of the hierarchy is indented the number of spaces specified as the indent.

**Note:** For an HTML, PDF, or PostScript report, you can use the BLANKINDENT setting to specify an indentation between levels of an FML hierarchy. See [Indenting Row Titles in an FML Hierarchy](#) on page 1799.

**Syntax:**      **How to Add an Underline Character for Columns**

```
BAR [AS 'character'] OVER
```

where:

*character*  
Is either the hyphen character (-) or the equal character (=). Enclose the character in single quotation marks. The default character is the hyphen (-).

**Example:**      **Underlining Columns**

This example uses the default underscore character (\_).

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 AS 'CASH ON HAND'          OVER
1020 AS 'DEMAND DEPOSITS'      OVER
1030 AS 'TIME DEPOSITS'        OVER
BAR                             OVER
RECAP TOTCASH = R1 + R2 + R3;
END
```

The output is shown as follows.

	AMOUNT
	-----
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
	-----
TOTCASH	21,239

Notice that the BAR ... OVER phrase underlines only the column containing the display field.

**Syntax:**      **How to Specify a Page Break in an FML Report**

Include the following syntax in the FML request in place of a tag value.

PAGE-BREAK OVER

**Example:**      **Specifying a Page Break in an FML Report**

In this example, a page break is inserted after the first two RECAP commands to highlight each calculation.

TABLE FILE LEDGER	
SUM AMOUNT FOR ACCOUNT	
1010 AS 'CASH ON HAND'	OVER
1020 AS 'DEMAND DEPOSITS'	OVER
1030 AS 'TIME DEPOSITS'	OVER
BAR	OVER
RECAP TOTCASH = R1 + R2 + R3; AS 'TOTAL CASH'	OVER
PAGE-BREAK	OVER
1100 AS 'ACCOUNTS RECEIVABLE' LABEL RECEIVE	OVER
1200 AS 'INVENTORY' LABEL INVENT	OVER
BAR	OVER
RECAP TOTASSET = RECEIVE + INVENT; AS 'TOTAL ASSETS'	OVER
PAGE-BREAK	OVER
RECAP TOTAL = TOTCASH + TOTASSET;	
END	

The output is shown as follows.

PAGE 1	
	<u>AMOUNT</u>
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
<hr/>	
TOTAL CASH	21,239
PAGE 2	
	<u>AMOUNT</u>
ACCOUNTS RECEIVABLE	18,829
INVENTORY	27,307
<hr/>	
TOTAL ASSETS	46,136
PAGE 3	
	<u>AMOUNT</u>
TOTAL	67,375

**Syntax:**      **How to Format a Row, Column, or Cell in an FML Report**

`TYPE=type, [COLUMN=column] [LABEL={Rn|label}], format_def, $`

where:

*type*

Identifies the component you wish to format. Valid values are:

**REPORT** denotes a row with the specified label.

**DATA** denotes a row with the specified label, which contains user-supplied data values.

**FREETEXT** denotes a free text or a blank row with the specified label.

**UNDERLINE** denotes underlines generated by BAR. Formatting of an underline is supported for PDF and PS, but not for HTML reports.

### *column*

Identifies a specific column. You can identify the column by its name or position in a row.

### **LABEL**

Is the controlling factor in identifying and formatting an FML row.

Note that the label is used to identify a row for calculation or formatting. The label for a TAG or DATA row never appears in the report output. It is used only to identify rows within the FML code. For a RECAP row, the name of the calculated value serves as a label. It appears in the report unless an alternate title is specified.

*label* is an explicit row label that you can assign to identify a row more clearly.

### *format\_def*

Is the formatting definition, such as FONT, SIZE, STYLE, and COLOR. See [Formatting an FML Report](#) on page 1782.

**Note:** To format a cell, identify the cell as the intersection of a column and a row using COLUMN= ... plus LABEL= ... in the same StyleSheet declaration.

## **Example: Formatting Rows in an FML Report**

The following illustrates how to identify and format an entire FML row, consisting of the row label and the row data. The LABEL attribute in the StyleSheet identifies the three TAG rows, which are styled here as italic.

```
SET PAGE-NUM=OFF
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 AS 'CASH ON HAND' LABEL COH OVER
1020 AS 'DEMAND DEPOSITS' LABEL DD OVER
1030 AS 'TIME DEPOSITS' LABEL TD OVER
BAR OVER
RECAP TOTCASH = R1 + R2 + R3; AS 'TOTAL CASH'
ON TABLE SET STYLESHEET *
TYPE = REPORT, GRID = OFF, $
TYPE = REPORT, LABEL = COH, STYLE = ITALIC, $
TYPE = REPORT, LABEL = DD, STYLE = ITALIC, $
TYPE = REPORT, LABEL = TD, STYLE = ITALIC, $
ENDSTYLE
END
```

The output is shown in the following image.

	<u>AMOUNT</u>
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
<hr/>	
TOTAL CASH	21,239

### Applying Boldface to a TAG Row in an FML Report

This request applies boldface to the customized row title, CASH, and to the related data in the AMOUNT column. The StyleSheet uses the explicit label CA to identify the component to format.

```
SET PAGE-NUM=OFF
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
10$$ AS 'CASH' LABEL CA OVER
1100 AS 'ACCOUNTS RECEIVABLE' LABEL AR OVER
1200 AS 'INVENTORY' LABEL INV OVER
RECAP CURASST/I5C = CA + AR + INV;
ON TABLE SET STYLE SHEET *
TYPE = REPORT, GRID = OFF, $
TYPE = REPORT, LABEL = CA, STYLE = BOLD, $
ENDSTYLE
END
```

The output is shown in the following image.

	<u>AMOUNT</u>
<b>CASH</b>	<b>21,239</b>
ACCOUNTS RECEIVABLE	18,829
INVENTORY	27,307
CURASST	67,375

### Applying Boldface to a Cell in an FML Matrix

This request generates a report in which the data value for AMOUNT is bold in the row titled CASH. However, the row title CASH is not bold. This is accomplished by pinpointing the cell in the StyleSheet declaration. In this case, the column (N2) within the row (CA).

```

SET PAGE-NUM=OFF
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
10$$ AS 'CASH' LABEL CA OVER
1100 AS 'ACCOUNTS RECEIVABLE' LABEL AR OVER
1200 AS 'INVENTORY' LABEL INV OVER
RECAP CURASST/I5C = CA + AR + INV;
ON TABLE SET STYLESHEET *
TYPE = REPORT, GRID = OFF, $
TYPE = REPORT, COLUMN = N2, LABEL = CA, STYLE = BOLD, $
ENDSTYLE
END

```

The output is shown in the following image.

	<u>AMOUNT</u>
<b>CASH</b>	<b>21,239</b>
ACCOUNTS RECEIVABLE	18,829
INVENTORY	27,307
CURASST	67,375

### Applying Boldface to a Column in an FML Report

This request identifies the AMOUNT column by name and formats its title and data in bold. The same result is achieved if the column is identified as N2.

```

SET PAGE-NUM=OFF
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
"---CASH ACCOUNTS---" OVER
1010 AS 'CASH ON HAND' OVER
1020 AS 'DEMAND DEPOSITS' OVER
1030 AS 'TIME DEPOSITS' OVER
" " OVER
"---OTHER CURRENT ASSETS---" OVER
1100 AS 'ACCOUNTS RECEIVABLE' OVER
1200 AS 'INVENTORY' OVER
ON TABLE SET STYLESHEET *
TYPE = REPORT, GRID = OFF, $
TYPE = REPORT, COLUMN = AMOUNT, STYLE = BOLD, $
ENDSTYLE
END

```



The output is shown in the following image.

	<u>AMOUNT</u>
---CASH ACCOUNTS---	
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
---OTHER CURRENT ASSETS---	
ACCOUNTS RECEIVABLE	18,829
INVENTORY	27,307

**Applying Boldface to a Free Text Row**

This request styles the free text as bold. Since in this example the same styling applies to both free text rows, labels are not required to distinguish between them.

```
SET PAGE-NUM=OFF
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
"---CASH ACCOUNTS---" LABEL CA OVER
1010 AS 'CASH ON HAND' OVER
1020 AS 'DEMAND DEPOSITS' OVER
1030 AS 'TIME DEPOSITS' OVER
" " OVER
"---OTHER CURRENT ASSETS---" LABEL OCA OVER
1100 AS 'ACCOUNTS RECEIVABLE' OVER
1200 AS 'INVENTORY'
ON TABLE SET STYLE SHEET *
TYPE = REPORT, GRID = OFF, $
TYPE = FREETEXT, STYLE = BOLD, $
ENDSTYLE
END
```

The output is shown in the following image.

	<u>AMOUNT</u>
---CASH ACCOUNTS---	
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
---OTHER CURRENT ASSETS---	
ACCOUNTS RECEIVABLE	18,829
INVENTORY	27,307

### Formatting Free Text Rows Separately in an FML Report

This request uses the SIZE attribute to distinguish two lines of free text: CASH ACCOUNTS and OTHER CURRENT ASSETS. The labels CA and OCA are used to identify and format each row separately.

```

SET PAGE-NUM=OFF
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
  ■ --- CASH ACCOUNTS ---■ LABEL CA          OVER
1010 AS 'CASH ON HAND'                        OVER
1020 AS 'DEMAND DEPOSITS'                      OVER
1030 AS 'TIME DEPOSITS'                        OVER
" "                                           OVER
  ■ --- OTHER CURRENT ASSETS ---■ LABEL OCA    OVER
1100 AS 'ACCOUNTS RECEIVABLE'                  OVER
1200 AS 'INVENTORY'
ON TABLE SET STYLESHEET *
TYPE = REPORT, GRID = OFF, $
TYPE = FREETEXT, LABEL = CA, STYLE = BOLD, SIZE = 12, $
TYPE = FREETEXT, LABEL = OCA, STYLE = BOLD, SIZE = 10, $
ENDSTYLE
END

```

The output is shown in the following image.

	<u>AMOUNT</u>
<b>---CASH ACCOUNTS---</b>	
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
<b>---OTHER CURRENT ASSETS---</b>	
ACCOUNTS RECEIVABLE	18,829
INVENTORY	27,307

### Styling Text and a Variable in a Free Text Row

In this example, the text and variable components of the free text row are styled separately. The text, Current Assets, is italic and the value derived from the RECAP calculation is bold.

```
SET PAGE-NUM=OFF
TABLE FILE LEDGER
SUM AMOUNT AS 'Amount' FOR ACCOUNT
10$$ AS 'Cash' LABEL CA OVER
1100 AS 'Accounts Receivable' LABEL AR OVER
1200 AS 'Inventory' LABEL INV OVER
RECAP CURASST/I5C = CA + AR + INV; NOPRINT OVER
"Current Assets: <CURASST"
ON TABLE SET STYLE SHEET *
TYPE = REPORT, GRID=OFF, $
TYPE = FREETEXT, OBJECT = TEXT, ITEM = 1, SIZE = 12, STYLE = ITALIC, $
TYPE = FREETEXT, OBJECT = FIELD, ITEM = 1, STYLE = BOLD, $
ENDSTYLE
END
```

The output is shown in the following image.

	<u>AMOUNT</u>
<i>CASH ON HAND</i>	8,784
<i>DEMAND DEPOSITS</i>	4,494
<i>TIME DEPOSITS</i>	7,961
<hr/>	
TOTAL CASH	21,239

### Applying Boldface to an FML RECAP Row

This request applies boldface to the row title and calculated value in a RECAP row. Notice that the RECAP label in the StyleSheet is TOTCASH. In a RECAP, the name assigned to the calculated value serves as the explicit label.

```
SET PAGE-NUM=OFF
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 AS 'CASH ON HAND'      LABEL CASH      OVER
1020 AS 'DEMAND DEPOSITS'  LABEL DD        OVER
1030 AS 'TIME DEPOSITS'    LABEL TD        OVER
RECAP TOTCASH = R1 + R2 + R3; AS 'TOTAL CASH'
ON TABLE SET STYLESHEET *
TYPE = REPORT, GRID = OFF, $
TYPE = REPORT, LABEL = TOTCASH, STYLE = BOLD, $
TYPE = REPORT, LABEL = CASH, COLUMN = N1, STYLE = ITALIC, $
TYPE = REPORT, LABEL = DD, COLUMN = N1, STYLE = ITALIC, $
TYPE = REPORT, LABEL = TD, COLUMN = N1, STYLE = ITALIC, $
ENDSTYLE
END
```

The output is shown in the following image.

	<u>AMOUNT</u>
<i>CASH ON HAND</i>	8,784
<i>DEMAND DEPOSITS</i>	4,494
<i>TIME DEPOSITS</i>	7,961
<b>TOTAL CASH</b>	<b>21,239</b>

**Syntax:**      **How to Add and Format Row and Cell Borders**

To request a uniform border around a row or cell, use this syntax:

```
TYPE=REPORT, LABEL=row_label, [COLUMN=column,] BORDER=option,
[BORDER-STYLE=line_style,] [BORDER-COLOR={color|RGB® g b)},,] $
```

To specify different characteristics for the top, bottom, left, and/or right borders, use this syntax:

```
TYPE=REPORT, LABEL=row_label, [COLUMN=column,] BORDER-position=option,
[BORDER-[position-]STYLE=line_style,]
[BORDER-[position-]COLOR={color|RGB(r g b)},,] $
```

To specify different characteristics for the top, bottom, left, and/or right borders, use this syntax:

```
TYPE=REPORT, LABEL=row_label, [COLUMN=column,] BORDER-position=option,
[BORDER-[position-]STYLE=line_style,]
[BORDER-[position-]COLOR={color|RGB(r g b)},,] $
```

where:

*row\_label*

Is the row to which the specified border characteristics are applied.

*column*

Used in conjunction with row label. Designates a cell (at the point of intersection of the row and the column) to which the specified border characteristics are applied.

*option*

Can be one of the following values:

**ON** turns borders on for the entire heading or footing. ON generates the same line as MEDIUM.

**OFF** turns borders off for the entire heading or footing. OFF is the default.

**LIGHT** specifies a thin line. You can specify a light line for the entire heading or footing, or for one or more border positions.

**MEDIUM** identifies a medium line (ON sets the line as MEDIUM). You can specify a light line for the entire heading or footing, or for one or more border positions. Note that the medium line setting ensures consistency with lines created with GRID attributes.

**HEAVY** identifies a thick line. You can specify a heavy line for the entire heading or footing, or for one or more border positions.

*width* specifies the line width in points (where 72 pts=1 inch). You can specify a line width in points for the entire heading or footing or for one or more border positions. Line width specified in points is displayed differently in HTML and PDF output. For uniform appearance, regardless of display format, use LIGHT, MEDIUM, or HEAVY.

*position*

Specifies which border line to format. Valid values are TOP, BOTTOM, LEFT, RIGHT.

You can specify a position qualifier for any of the BORDER keywords. This enables you to format line width, line style, and line color individually, for any side of the border.

*line\_style*

Sets the style of the border line. WebFOCUS StyleSheets support all of the standard Cascading Style Sheets line styles. Several 3-dimensional styles are only available in HTML, as noted by asterisks. Valid values are:

NONE  
SOLID  
DOTTED  
DASHED  
DOUBLE\*  
GROOVE\*  
RIDGE\*  
INSET\*  
OUTSET\*

### *color*

Is one of the preset color values. The default value is BLACK.

If the display or output device does not support colors, it substitutes shades of gray.

### RGB

Specifies the font color using a mixture of red, green, and blue.

### *( r g b )*

Is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense. Note that using the three color components in equal intensities results in shades of gray.

**Note:** For HTML reports, the BORDERS feature requires that cascading style sheets be turned ON. This code is not required for PDF and PS reports.

### **Example:** Emphasizing a Row Using Uniform Border Lines

This example places a dashed border of medium thickness around the RECAP row identified by the label TOTCASH. For HTML reports, the BORDERS feature requires that cascading style sheets be turned ON.

```
SET PAGE-NUM=OFF
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 AS 'CASH ON HAND'      LABEL CASH      OVER
1020 AS 'DEMAND DEPOSITS'  LABEL DD      OVER
1030 AS 'TIME DEPOSITS'    LABEL TD      OVER
RECAP TOTCASH = R1 + R2 + R3; AS 'TOTAL CASH'
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLESHEET *
TYPE = REPORT, GRID = OFF, $
TYPE = REPORT, LABEL = TOTCASH, BORDER = MEDIUM,
BORDER-STYLE = DASHED, $
ENDSTYLE
END
```

The output is shown in the following image.

	AMOUNT
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
TOTAL CASH	21,239

**Example: Emphasizing a Row Using Different Top/Bottom and Left/Right Borders**

This example places a heavy black border line above and below the RECAP row identified by the label TOTCASH, and a thin silver dotted line to the left and right of each column in the row.

For HTML reports, the BORDERS feature requires that cascading style sheets be turned ON.

```

SET PAGE-NUM=OFF
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 AS 'CASH ON HAND' LABEL CASH OVER
1020 AS 'DEMAND DEPOSITS' LABEL DD OVER
1030 AS 'TIME DEPOSITS' LABEL TD OVER
RECAP TOTCASH = R1 + R2 + R3; AS 'TOTAL CASH'
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE SHEET *
TYPE = REPORT, GRID = OFF,$
TYPE = REPORT, LABEL = TOTCASH,
    BORDER-TOP = HEAVY,
    BORDER-BOTTOM = HEAVY,
    BORDER-LEFT = LIGHT,
    BORDER-RIGHT = LIGHT,
    BORDER-TOP-STYLE = SOLID,
    BORDER-BOTTOM-STYLE = SOLID,
    BORDER-LEFT-STYLE = DOTTED,
    BORDER-RIGHT-STYLE = DOTTED,
    BORDER-LEFT-COLOR = 'SILVER',
    BORDER-RIGHT-COLOR = 'SILVER', $
ENDSTYLE
END

```

The output is shown in the following image.

	<u>AMOUNT</u>
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
<b>TOTAL CASH</b>	<b>21,239</b>

**Example: Adding Uniform Border Lines Around a Cell**

This example places a border of medium thickness around the cell in the second column of the row identified by the label TOTCASH. The combined LABEL and COLUMN specifications are identified in the cell. The BORDERS feature requires that cascading style sheets be turned ON.

```

SET PAGE-NUM=OFF
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 AS 'CASH ON HAND'    LABEL CASH    OVER
1020 AS 'DEMAND DEPOSITS' LABEL DD     OVER
1030 AS 'TIME DEPOSITS'   LABEL TD     OVER
RECAP TOTCASH = R1 + R2 + R3; AS 'TOTAL CASH'
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE SHEET *
TYPE = REPORT, GRID = OFF,$
TYPE = REPORT, LABEL = TOTCASH, COLUMN = N2, BORDER = MEDIUM, $
ENDSTYLE
END
    
```

The output is shown in the following image.

	<u>AMOUNT</u>
CASH ON HAND	8,784
DEMAND DEPOSITS	4,494
TIME DEPOSITS	7,961
<b>TOTAL CASH</b>	<b>21,239</b>



**Syntax:**      **How to Specify an Indent for an FML Label, Tag, or Caption**

```
FOR forfield [IN k]
tag [[GET CHILDREN|WITH CHILDREN] n] INDENT m [AS ['text'|CAPTION]]
[OVER]
```

or

```
RECAP fieldname[/format]=expression; INDENT m [AS 'text']
```

where:

*forfield*

Is a field in the data source whose values are included in the report.

*k*

Is the starting column for the FOR value in an FML report.

*tag*

Is a value of *forfield* to be displayed on a row of the FML report.

*n*

Is the number of levels of an FML hierarchy to display on the FML report.

*m*

Is a positive integer (zero is not supported) specifying the number of spaces to indent the tag value, label, or caption of an FML row or hierarchy. The indentation starts from column one if there is no IN phrase specified in the FOR command. If there is an IN phrase, indentation starts from the column specified in the IN phrase. The maximum indentation is the same as the maximum length of an AS name.

If you indent an FML hierarchy, the parent line of the hierarchy is indented the number of spaces specified as the indent. The hierarchy levels are indented two spaces from each other. If the GET CHILDREN phrase is used, the first line of the hierarchy is indented an additional two spaces because the hierarchy output begins with the first child rather than the parent. For more information about the use of GET CHILDREN, see [Displaying an FML Hierarchy](#) on page 1767.

'*text*'

Is a label to be displayed on a row of the FML report.

CAPTION

Indicates that a caption field has been defined in the Master File.

OVER

Indicates that this row is not the last row to be displayed.

fieldname

Is a name you assign to the value calculated by the RECAP command.

format

Is the USAGE format for RECAP field. It cannot exceed the column width. The default is the format of the column in which the calculated value is displayed.

expression

Is the expression that describes how to calculate the field value for RECAP.

Example: Indenting a Tag Row in an FML Hierarchy

In the following request, the label of the second row for tag value 3000 is indented five spaces. Because the GET CHILDREN phrase is used, the first line of the FML hierarchy, in the third row for tag value 3000, is indented seven spaces (five + two).

```
SET FORMULTIPLE=ON
TABLE FILE CENTGL
PRINT GL_ACCOUNT_PARENT
FOR GL_ACCOUNT
3000                                AS 'Not Indented'          OVER
3000                                AS 'Indented 5'            OVER
3000 GET CHILDREN 2 INDENT 5 AS 'Hierarchy Indented 5'
END
```

The output is shown as follows.

	Parent
	-----
Not Indented	3000
Indented 5	3000
Hierarchy Indented 5	3000
Hierarchy Indented 5	3100
Hierarchy Indented 5	3100
Hierarchy Indented 5	3100
Hierarchy Indented 5	3100
Hierarchy Indented 5	3000
Hierarchy Indented 5	3200
Hierarchy Indented 5	3200
Hierarchy Indented 5	3200
Hierarchy Indented 5	3200
Hierarchy Indented 5	3200
Hierarchy Indented 5	3200
Hierarchy Indented 5	3200

Indenting FML RECAP Rows

The following request sums price, cost, and quantity in stock for digital and analog product types. The first RECAP command calculates the total for each column, and indents the label five spaces. The second RECAP command calculates the profit, and indents the label 10 spaces.

```

SET FORMULTIPLE=ON
TABLE FILE CENTINV
SUM PRICE COST QTY_IN_STOCK
FOR PRODTYPE
Digital
Analog
BAR
RECAP TOTAL = R1 + R2; INDENT 5 AS 'Total:'
BAR
RECAP PROFIT(2) = TOTAL(1) - TOTAL(2); AS 'Profit:' INDENT 10
END
OVER
OVER
OVER
OVER
OVER
OVER

```

The output is shown as follows.

	Price:	Our Cost:	Quantity In Stock:
	-----	-----	-----
Digital	4,080.00	3,052.00	119143
Analog	1,883.00	1,371.00	139345
	-----	-----	-----
Total:	5,963.00	4,423.00	258488
	-----	-----	-----
Profit:		1,540.00	

### Indenting Row Titles in an FML Hierarchy

To clarify relationships within an FML hierarchy, the captions (titles) of values are indented at each level. Use the BLANKINDENT parameter in an HTML, PDF, or PostScript report to specify the indentation between each level in the hierarchy. You can use the default indentation for each hierarchy level or choose your own indentation value. To print indented captions in an HTML report, you must set the BLANKINDENT parameter to ON or to a number.

SET BLANKINDENT does not redefine the width of the indented column, if it is not wide enough to accommodate the indented fields. Columns in table-based formats will automatically size themselves as needed, while columns in position-based formats, such as PDF, PostScript, or PPTX, shift out of alignment. You can use StyleSheet syntax to make the column wide enough for the indented values, and move the columns that follow it. Change the width of a column using the StyleSheet SQUEEZE = *n* attribute to supply the required space.

A related feature enables you to change the number of blank spaces before the parent line of a hierarchy or before any FML tag or RECAP row in any FML request. For more information, see [Formatting an FML Report](#) on page 1782.

**Syntax:**      **How to Indent FML Hierarchy Captions in an HTML Report**

```
SET BLANKINDENT={ON|OFF|n}
ON TABLE SET BLANKINDENT {ON|OFF|n}
```

where:

ON

Indents FML hierarchy captions 0.125 units for each space that normally displays before the caption. For child levels in an FML hierarchy, it indents 0.125 units for each space that normally displays between this line and the line above it.

OFF

Turns off indentations for FML hierarchy captions in an HTML report. OFF is the default value. For other formats, uses the default indentation of two spaces.

n

Is an explicit measurement in the unit of measurement defined by the UNITS parameter. This measurement is multiplied by the number of spaces that normally displays before the caption. For child levels in an FML hierarchy, it indents *n* units for each space that normally displays between this line and the line above it. The default number of spaces is two. Zero (0) produces the same report output as OFF. Negative values for *n* are not supported.

**Example:**      **Using the Default Indentation for FML Hierarchy Captions**

The following request creates an HTML report with the default indentation:

```
SET PAGE-NUM=NOPAGE
SET BLANKINDENT=ON
SET FORMULTIPLE=ON
TABLE FILE CENTGL
PRINT GL_ACCOUNT_PARENT
FOR GL_ACCOUNT
3000          AS CAPTION      OVER
3000 GET  CHILDREN 2  AS CAPTION

ON TABLE SET ONLINE-FMT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE = REPORT, GRID = OFF, $
ENDSTYLE
END
```

The output is shown in the following image.

	<u>Parent</u>
Total Operating Expenses	1000
Selling Expenses	3000
Advertising	3100
Promotional Expenses	3100
Joint Marketing	3100
Bonuses/Commissions	3100
General + Admin Expenses	3000
Salaries-Corporate	3200
Company Benefits	3200
Depreciation Expenses	3200
Gain/(Loss) Sale of Equipment	3200
Leasehold Expenses	3200
Interest Expenses	3200
Utilities	3200

**Example: Specifying an Indentation Value for FML Hierarchy Captions**

The following request specifies an indentation of .25 for each level of an FML hierarchy. This number is expressed in the default unit of measurement, which is inches:

```

SET PAGE-NUM=NOPAGE
SET BLANKINDENT=.25
SET FORMULTIPLE=ON
TABLE FILE CENTGL
PRINT GL_ACCOUNT_PARENT
FOR GL_ACCOUNT
3000                                AS CAPTION                OVER
3000 GET  CHILDREN 2 AS CAPTION

ON TABLE SET STYLE *
TYPE = REPORT, GRID = OFF, $
ENDSTYLE
END

```

The output is shown in the following image.

	<u>Parent</u>
Total Operating Expenses	1000
Selling Expenses	3000
Advertising	3100
Promotional Expenses	3100
Joint Marketing	3100
Bonuses/Commissions	3100
General + Admin Expenses	3000
Salaries-Corporate	3200
Company Benefits	3200
Depreciation Expenses	3200
Gain/(Loss) Sale of Equipment	3200
Leasehold Expenses	3200
Interest Expenses	3200
Utilities	3200

## Suppressing the Display of Rows

You may sometimes wish to retrieve data in a TAG row solely for use in a calculation, without displaying the row in a report. To suppress the display of a tag row, add the word NOPRINT to the row declaration, as in a TABLE request.

You may also wish to suppress the display of a TAG row if no data is found for the values. For more information, see [Suppressing Rows With No Data](#) on page 1803.

In addition, you can suppress the display of RECAP rows by adding the word NOPRINT to the RECAP command, following the semicolon. This technique is useful to suppress the display of an intermediate RECAP value, which is intended for use as input to other calculations.

### **Example:** Suppressing the Display of a TAG Row

This example uses the value of COST in its computation, but does not display COST as a row in the report.

```

DEFINE FILE REGION
AMOUNT/I5C=E_ACTUAL;
END

TABLE FILE REGION
SUM AMOUNT FOR ACCOUNT
3000 AS 'SALES' LABEL SLS                OVER
3100 AS 'COST' LABEL COST NOPRINT        OVER
RECAP PROFIT/I5C = SLS - COST;           OVER
" "                                       OVER
RECAP ROS/F6.2 = 100*PROFIT/SLS;
AS 'RETURN ON SALES'
END

```

The output is shown in the following image.

	<u>AMOUNT</u>
SALES	6,000
PROFIT	1,350
RETURN ON SALES	22.50

## Suppressing Rows With No Data

The text for a tag row is displayed even if no data is found in the file for the tag values, with a period (.) representing the missing data. You can override this convention by adding the phrase WHEN EXISTS to the definition of a TAG row. This makes displaying a row dependent upon the existence of data for the tag. This feature is useful, for example, when the same model is applied to different divisions in a company.

### *Example:* Suppressing Rows With No Data

In this example, assume that the variable DIVISION contains Division 1, a real estate syndicate, and Division 2, a bank. The following request describes their balance sheets in one FML report. Rows that are irrelevant for each division are not displayed.

```

TABLE FILE LEDGER
HEADING CENTER
"BALANCE SHEET FOR DIVISION <DIVISION>"
" "
SUM AMOUNT
BY DIVISION NOPRINT
ON DIVISION PAGE-BREAK
FOR ACCOUNT
2000 AS 'LAND' WHEN EXISTS LABEL LD      OVER
2100 AS 'CAR LOANS' WHEN EXISTS LABEL LOAN OVER
.
.
.

```

## Saving and Retrieving Intermediate Report Results

Many reports require results developed in prior reports. This can be accomplished only if a place is provided for storing intermediate values. An example is the need to compute net profit in an Income Statement prior to calculating equity in a Balance Sheet. FML can save selected rows from one or more models by posting them to a work file. The posted rows can then be picked up from the work file and reused.

The default work file is FOCPOST. This is a comma-delimited file from which you can report directly if a FOCPOST Master File is available. In order to use the work file in a request, you must assign a physical name to the FOCPOST ddname before running the report that posts the data, and again before running the report that picks up the data.

You can assign the physical name to the file by issuing a FILEDEF command on Windows and UNIX, or a TSO ALLOCATE or DYNAM ALLOCATE command on z/OS, before the request is run. You may create a FILEDEF command by using the Allocation Wizard.

While you cannot prepare an FML report entirely from data that you supply directly in your request, you can prepare a report entirely from data that is stored in a comma-delimited work file.

## Posting Data

You can save any TAG, RECAP, or DATA row by posting the output to a file. You can use these rows as though they were provided in a DATA row.

The row is processed in the usual manner in the report, depending on its other options, and then posted. The label of the row is written first, followed by the numeric values of the columns, each comma-separated, and ending with the terminator character (\$). For more information, see [Posting Rows to a Work File](#) on page 1805.

**Note:** Only fields that are actually displayed on the report output are posted. Fields that are not printed (for example, fields specified with the NOPRINT option, extra fields that are created when you reformat fields in the request, or fields implied by use in a calculation) are not posted.



**Syntax:**      **How to Post Data to a File**

The syntax for saving any TAG, RECAP, or DATA row is:

```
POST [TO ddname]
```

where:

*ddname*

Is the logical name you assign to the work file in which you are posting data.

Add this syntax to any row you wish to post to the work file.

**Example:**      **Posting Rows to a Work File**

The following request creates an FML report, and posts two tag rows to the LEDGEOUT work file.

```
FILEDEF LEDGEOUT DISK [APP]\LEDGEOUT.DAT
```

```
DEFINE FILE LEDGER
CUR_YR/I5C=AMOUNT;
LAST_YR/I5C=.87*CUR_YR - 142;
END
```

```
TABLE FILE LEDGER
SUM CUR_YR LAST_YR
FOR ACCOUNT
1100 LABEL AR POST TO LEDGEOUT OVER
1200 LABEL INV POST TO LEDGEOUT
END
```

The output is shown in the following image.

	<u>CUR_YR</u>	<u>LAST_YR</u>
1100	18,829	15,954
1200	27,307	23,329

**Syntax:**      **How to Pick Up Data From a Work File**

You can retrieve posted rows from any work file and use them as if they were provided in a DATA row by adding the following phrase to an FML request.

```
DATA PICKUP [FROM ddname] id1 [OR id2 ...] [LABEL label] [AS 'text']
```

where:

*ddname*

Is the logical name of the work file from which you are retrieving data.

### *id*

Is the label that was assigned in the work file to the posted row of data that is now being picked up.

### *label*

Is the label you wish to assign to the data you are picking up.

The label you assign to the picked data can, but is not required to, match the label (id) of the posted data.

You can include LABEL and AS phrases, but WHEN EXISTS is not supported.

**Note:** The retrieved fields are mapped to all fields (printed or not) in the memory repository (internal matrix) of the report. If the matrix contains columns that do not correspond to the fields in the posted file, the retrieved values may be misaligned. For example, if you reformat a field in the PICKUP request, that field will be represented by two columns in the internal matrix. However, the posted file will have only one value representing that field, and the retrieved values will not be mapped properly to the associated columns in the matrix.

### **Example:** Picking Up Data From a Work File

In the following example, the data in the LEDGER data source and in the LEDGEOUT work file are used in the RECAP calculation. To see how this file was created, refer to [Posting Rows to a Work File](#) on page 1805.

**Tip:** You must assign a logical name to the file by issuing a FILEDEF command on Windows and UNIX, or a DYNAM ALLOCATE command on z/OS, before the request is run. You may create a FILEDEF command by using the Allocation Wizard.

```
DEFINE FILE LEDGER
CUR_YR/I5C=AMOUNT;
LAST_YR/I5C=.87*CUR_YR - 142;
END

TABLE FILE LEDGER
SUM CUR_YR LAST_YR
FOR ACCOUNT
1010 TO 1030 AS 'CASH' LABEL CASH OVER
DATA PICKUP FROM LEDGEOUT AR AS 'ACCOUNTS RECEIVABLE' LABEL AR OVER
DATA PICKUP FROM LEDGEOUT INV AS 'INVENTORY' LABEL INV OVER
BAR OVER
RECAP CUR_ASSET/I5C = CASH + AR + INV;
END
```

The output is shown in the following image.

	<u>CUR_YR</u>	<u>LAST_YR</u>
CASH	21,239	17,198
ACCOUNTS RECEIVABLE	18,829	15,954
INVENTORY	27,307	23,329
CUR_ASSET	<u>67,375</u>	<u>56,481</u>

The following line can be used to pick up the sum of the two accounts from LEDGEOUT.

```
DATA PICKUP FROM LEDGEOUT AR OR INV
AS 'ACCTS REC AND INVENTORY'
```

**Note:** Since the rows in a PICKUP file are stored in standard comma-delimited format, they can be provided either from a prior posting, or directly by a user.

### Creating HOLD Files From FML Reports

A report created with FML can be extracted to a HOLD file in the same way as all other reports created with the TABLE language.

In this case, you identify the set of tag values specified for each row by the description field (the AS text supplied in the model). When no text is given for a row, the first tag value is used automatically. Therefore, in simple models with only one tag per row and no text, the lines in the HOLD file contain the single tag value. The rows derived from the RECAP calculation form part of the HOLD file. Pure text rows (including BAR rows) are omitted.

For HOLD to be supported with RECAP, the format of the RECAP field must be the same as the format of the original column.

This feature enables you to create new rows in the HOLD file that are the result of calculations. The augmented HOLD file may then be used in a variety of TABLE requests.

**Note:** You cannot reformat RECAP rows when creating HOLD files.

**Example:**     **Creating a Hold File From an FML Report**

The following request creates a HOLD file that contains records for CASH, ACCOUNTS RECEIVABLE, INVENTORY, and the RECAP row CURRENT ASSETS.

```
TABLE FILE LEDGER
SUM AMOUNT FOR ACCOUNT
1010 TO 1030 AS 'CASH'                                OVER
1100 AS 'ACCOUNTS RECEIVABLE'                          OVER
1200 AS 'INVENTORY'                                    OVER
RECAP CA = R1 + R2 + R3; AS 'CURRENT ASSETS'
ON TABLE HOLD
END
```

Query the HOLD file:

```
>  
? HOLD
```

```
DEFINITION OF HOLD FILE: HOLD
```

FIELDNAME	ALIAS	FORMAT
	E01	A 19
AMOUNT	E02	I5C

Then report from the HOLD file as:

```
TABLE FILE HOLD  
PRINT E01 E02  
END
```

The output is shown in the following image.

	AMOUNT
	-----
CASH	21,239
ACCOUNTS RECEIVABLE	18,829
INVENTORY	27,307
CURRENT ASSETS	67,375

## Creating a Free-Form Report

---

You can present data in an unrestricted or free-form format using a layout of your own design.

Whereas tabular and matrix reports present data in columns and rows for the purpose of comparison across records, and graphic reports present data visually using charts and graphs, free-form reports reflect your chosen positioning of data on a page. Free-form reporting meets your needs when your goal is to present a customized picture of a data source record on each page of a report.

**Note:** You can create free-form reports with PDF, HTML, and Styled formats. HTML output has all report pages on one HTML page. Page breaks are retained in PDF output.

### In this chapter:

- ☐ [Creating a Free-Form Report](#)
  - ☐ [Designing a Free-Form Report](#)
- 

## Creating a Free-Form Report

You can create a free-form report from a TABLE request that omits the display commands that control columnar and matrix formatting (PRINT, LIST, SUM, and COUNT). Instead, the request includes the following report features:

<b>Heading</b>	Contains the body of the report. It displays the text characters, graphic characters, and data fields that make up the report.
<b>Footing</b>	Contains the footing of the report. This is the text that appears at the bottom of each page of the report. The footing may display the same characters and data fields as the heading.
<b>Prefix operators</b>	Indicates field calculations and manipulation.
<b>Temporary fields</b>	Derives new values from existing fields in a data source.

<b>BY phrases</b>	Specifies the report sort order, and determines how many records are included on each page.
<b>WHERE criteria</b>	Selects records for the report.

When creating a free-form report, you can:

- ☐ Design your report to include text, data fields, and graphic characters. See [Designing a Free-Form Report](#) on page 1813.
- ☐ Customize the layout of your report. See [Laying Out a Free-Form Report](#) on page 1815.
- ☐ Select the sort order and the records that are included in your report. See [Sorting and Selecting Records in a Free-Form Report](#) on page 1816.

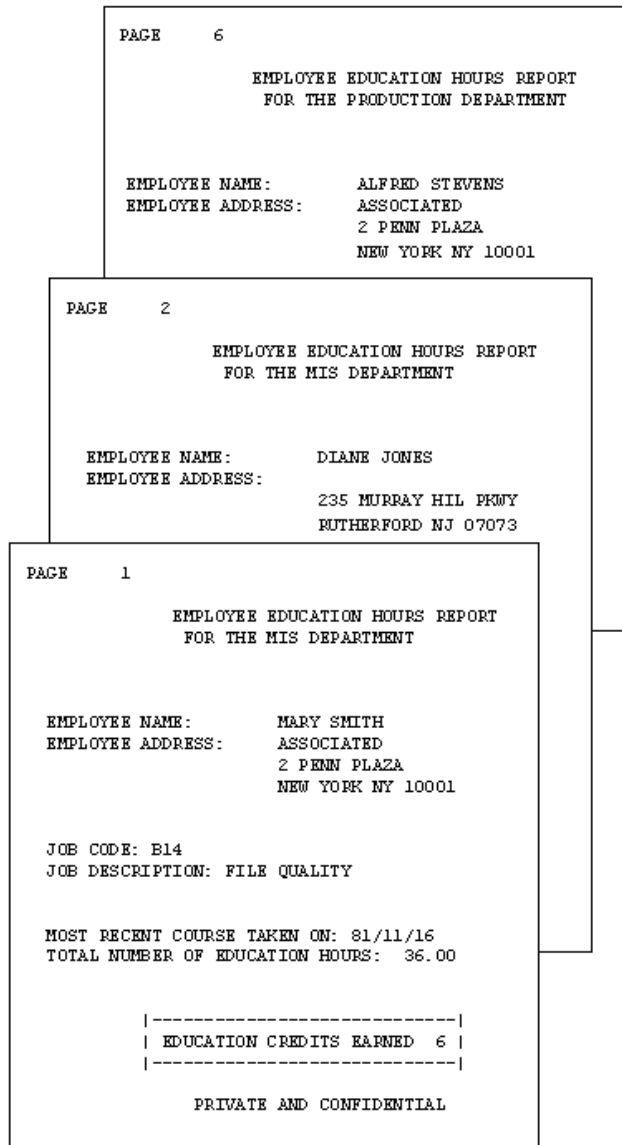
### ***Example:*** Creating a Free-Form Report

Suppose that you are a Personnel Manager and it is your responsibility to administer your company education policies. This education policy states that the number of hours of outside education that an employee may take at the company expense is determined by the number of hours of in-house education completed by the employee.

To do your job efficiently, you want a report that shows the in-house education history of each employee. Each employee information should display on a separate page so that it can be placed in the employee personnel file and referenced when an employee requests approval to take outside courses.

To meet this requirement, you create the EMPLOYEE EDUCATION HOURS REPORT, which displays a separate page for each employee. Notice that pages 1 and 2 of the report provide information about employees in the MIS department, while page 6 provides information for an employee in the Production department.

The following diagram simulates the output you would see if you ran the procedure in the example named *Request for EMPLOYEE EDUCATION HOURS REPORT* on page 1812.



### **Example: Request for EMPLOYEE EDUCATION HOURS REPORT**

The following request produces the EMPLOYEE EDUCATION HOURS REPORT. Numbers to the left of the request correspond to numbers in the following annotations:

```

1. SET STYLE = OFF
   SET STYLEMODE=FIXED
   SET ONLINE-FMT = PDF
2. DEFINE FILE EMPLOYEE
   CR_EARNED/I2 = IF ED_HRS GE 50 THEN 9
   ELSE IF ED_HRS GE 30 THEN 6
   ELSE 3;
   END
3. TABLE FILE EMPLOYEE
   BY DEPARTMENT
4. HEADING
   " "
   "<13>EMPLOYEE EDUCATION HOURS REPORT"
5. "<14>FOR THE <DEPARTMENT DEPARTMENT"
6. "</2>"
   "EMPLOYEE NAME:      <23><FIRST_NAME <LAST_NAME>"
   "EMPLOYEE ADDRESS: <23><ADDRESS_LN1>"
   "<23><ADDRESS_LN2>"
   "<23><ADDRESS_LN3>"
   "</1>"
   "JOB CODE: <JOBCODE>"
   "JOB DESCRIPTION: <JOB_DESC>"
   "</1>"
7. "MOST RECENT COURSE TAKEN ON: <MAX.DATE_ATTEND>"
   "TOTAL NUMBER OF EDUCATION HOURS: <ED_HRS>"
   "</1>"
8. "<10>|-----| "
9. "<10>|  EDUCATION CREDITS EARNED <CR_EARNED>| "
   "<10>|-----| "
10. BY EMP_ID NOPRINT PAGE-BREAK
11. WHERE ED_HRS GT 0
12. FOOTING
   "<15>PRIVATE AND CONFIDENTIAL"
   END

```

The following explains the role of each line of the request in producing the sample report:

1. Two SET commands are required to view the desired display in a browser. The SET STYLE = OFF command enables a free-form design by ignoring default StyleSheet parameters. SET STYLEMODE = FIXED turns off HTML formatting and allows the report designer to determine where items in the report are placed, using spot markers and skip-line commands.
2. The DEFINE command creates a virtual field for the report. The calculation reflects the company policy for earning outside education credits. The result is stored in CR\_EARNED and appears later in the report.
3. A free-form report begins with a standard TABLE FILE command. The sample report uses the EMPLOYEE data source.



4. The heading section, initiated by the `HEADING` command, defines the body of the report. Most of the text and data fields that display in the report are specified in the heading section. In this request, the heading section continues until the second `BY` phrase `BY EMP_ID NOPRINT PAGE-BREAK`.
5. This line illustrates the following:
  - ☐ The second line of the text in the page heading.
  - ☐ A data field embedded in the text: `<DEPARTMENT`.
  - ☐ The start position of the line, column 14: `<14>`.
6. You can enhance the readability of a report using skip-line commands. The command `</2`, when coded on a line by itself, generates two blank lines, as seen between the page heading and employee name.
7. This line illustrates how to perform a field calculation in a free-form report using a prefix operator. In this case, we requested the date on which the most recent course was taken—that is, the maximum value for the `DATE_ATTEND` field.
8. The next three lines illustrate the use of special characters to create a graphic in the report. The box around `EDUCATION CREDITS EARNED` may need adjustment for output displayed in a proportional font.
9. The value of the field created by the `DEFINE` command displays in the box, highlighting the number of education credits an employee has earned. This line demonstrates that you can display a virtual field in the body of your report.
10. This line illustrates the use of sorting in a free-form report. The report specifications require that information for only one employee displays per page. This is achieved by using the `BY` and `PAGE-BREAK` commands. Note that in order to produce a report with page breaks, the report output must be PDF.
11. You can specify record selection in a free-form report. As a result of the `WHERE` criterion, the report includes only employees who have accumulated in-house education credits.
12. Since we have designed a personnel report, it is important to have the words `PRIVATE` AND `CONFIDENTIAL` at the end of each report page. The `FOOTING` command accomplishes this.

## Designing a Free-Form Report

To design the body of a free-form report, use the `HEADING` and `FOOTING` commands. They enable you to:

- ☐ Incorporate text, data fields, and graphic characters in your report.
- ☐ Lay out your report by positioning text and data in exact column locations and skipping lines for readability.

Use the **HEADING** command to define the body of a free-form report, and the **FOOTING** command to define what appears at the bottom of each page of a report. A footing is optional. You can define an entire report using just a heading.

### Incorporating Text in a Free-Form Report

You can specify text anywhere in a free-form report, for a variety of purposes. In the sample request (see the example named [Request for EMPLOYEE EDUCATION HOURS REPORT](#) on page 1812) text is used:

- ❑ As a report title:

```
"<13>EMPLOYEE EDUCATION HOURS REPORT"
```

- ❑ As a label for data fields:

```
"EMPLOYEE NAME: <FIRST_NAME <LAST_NAME>"
```

- ❑ With a data field and graphic characters:

```
"<10>| EDUCATION CREDITS EARNED <CR_EARNED>| "
```

- ❑ As a page footing:

```
"<15>PRIVATE AND CONFIDENTIAL"
```

### Incorporating Data Fields in a Free-Form Report

The crucial element in any report, free-form or otherwise, is the data. The data fields available in a request include data fields in the Master File, cross-referenced fields, and virtual fields created with the **DEFINE** command.

The sample request (see [Request for EMPLOYEE EDUCATION HOURS REPORT](#) on page 1812) references all three types of data fields:

- ❑ **ED\_HRS** is found in the **EMPLOYEE** Master File:

```
"TOTAL NUMBER OF EDUCATION HOURS: <ED_HRS>"
```

- ❑ **DATE\_ATTEND** is found in the **EDUCFILE** Master File, which is cross-referenced in the **EMPLOYEE** Master File:

```
"MOST RECENT COURSE TAKEN ON: <MAX.DATE_ATTEND>"
```

- ❑ **CR\_EARNED** is created with the **DEFINE** command before the **TABLE FILE** command, and is referenced as follows:

```
"<10>| EDUCATION CREDITS EARNED <CR_EARNED>| "
```

You can also apply a prefix operator to a data field to select a particular value (for example, the maximum value within a sort group) or to perform a calculation (for example, to compute the average value of a field). You can use any available prefix operator in a free-form report.

In the sample request, the MAX prefix operator selects the most recent completion date of an in-house course:

```
"MOST RECENT COURSE TAKEN ON: <MAX.DATE_ATTEND>"
```

As is true with all types of reports, you must understand the structure of the data source to use the prefix operators correctly.

## Incorporating Graphic Characters in a Free-Form Report

Graphics in a report can be as creative as your imagination. The sample report (see [Creating a Free-Form Report](#) on page 1809) uses special characters to enclose text and a virtual field in a box. Some other ideas include:

- ☐ Highlighting key data fields using asterisks or other special characters available directly from your keyboard, or using the HEXBYT function. See the *Using Functions* manual for details on HEXBYT.
- ☐ Enclosing the entire report in a box to give it a form-like appearance.
- ☐ Using double lines to separate the body of the report from its page heading and page footing.

The use of special characters to create graphics is limited by what can be entered and viewed from your workstation and what can be printed on your printer. If you have difficulty producing the graphics that you want, be sure to check with someone in your organization who knows what is available.

## Laying Out a Free-Form Report

To provide spacing in a report and position text and data fields, use the spot marker feature of the HEADING and FOOTING commands.

**Note:** To take advantage of this feature in an HTML report, include the SET STYLEMODE=FIXED command in your request.

The sample request (see the example named *Request for EMPLOYEE EDUCATION HOURS REPORT* on page 1812) illustrates this feature. The first two examples show how to position text and data fields on your report, while the third example shows how to skip lines:

- ❑ The spot marker <13> positions the specified text in column 13 of the report:

```
"<13>EMPLOYEE EDUCATION HOURS REPORT"
```

- ❑ The spot marker <23> positions the specified data field in column 23 of the report:

```
"<23><ADDRESS_LN2>"
```

- ❑ The spot marker </1> on a line by itself skips two lines after displaying the job description:

```
"JOB DESCRIPTION: <JOB_DESC>"</1>"MOST RECENT COURSE TAKEN ON:  
<MAX.DATE_ATTEND>"
```

When designing a free-form report, take advantage of sort field options, such as NOPRINT, PAGE-BREAK (PDF output only), and UNDER-LINE. The sample request uses PAGE-BREAK to place each employee information on a separate page:

```
BY EMP_ID NOPRINT PAGE-BREAK
```

## Sorting and Selecting Records in a Free-Form Report

As with tabular and matrix reports, you can both sort a report and conditionally select records for it. Use the same commands as for tabular and matrix reports. For example, use the BY phrase to sort a report and define WHERE criteria to select records from the data source.



# Chapter 27

## Using SQL to Create Reports

---

SQL users can issue report requests that combine SQL statements with TABLE formatting phrases to take advantage of a wide range of report preparation options.

These combined requests are supported through the SQL Translator, which converts ANSI Level 2 SQL statements into executable FOCUS requests.

You can use the SQL Translator to retrieve and analyze FOCUS and DBMS data.

### In this chapter:

- ☐ [Supported and Unsupported SQL Statements](#)
  - ☐ [Using SQL Translator Commands](#)
  - ☐ [SQL Translator Support for Date, Time, and Timestamp Fields](#)
  - ☐ [Index Optimized Retrieval](#)
  - ☐ [TABLEF Optimization](#)
  - ☐ [SQL INSERT, UPDATE, and DELETE Commands](#)
- 

### Supported and Unsupported SQL Statements

SQL Translation Services is compliant with ANSI Level 2. This facility supports many, but not all, SQL statements. The Reporting Server and specific RDBMS engines may also support the *alpha1* CONCAT *alpha2* syntax. See [Supported SQL Statements](#) on page 1818 and [Unsupported SQL Statements](#) on page 1819.

Many of the supported SQL statements are candidates for Dialect Translation. This feature enables a server to route inbound SQL requests to SQL-capable subservers and data adapters where possible. Dialect Translation avoids translation to the Reporting Server Data Manipulation Language (DML), while maintaining data location transparency. It transforms a standard SQL statement into one that can be processed by the destination SQL engine, while preserving the semantic meaning of the statement.

**Note:** Because the SQL Translator is ANSI Level 2 compliant, some requests that worked in prior releases may no longer work.

**Reference: Supported SQL Statements**

SQL Translation Services supports the following:

- ☐ SELECT, including SELECT ALL and SELECT DISTINCT.
- ☐ CREATE TABLE. The following data types are supported for CREATE TABLE: REAL, DOUBLE PRECISION, FLOAT, INTEGER, DECIMAL, CHARACTER, SMALLINT, DATE, TIME, and TIMESTAMP.
- ☐ INSERT, UPDATE, and DELETE for relational, IMS, and FOCUS data sources.
- ☐ Equijoins and non-equijoins.
- ☐ Outer joins, subject to certain restrictions. See [SQL Joins](#) on page 1823.
- ☐ CREATE VIEW and DROP VIEW.
- ☐ PREPARE and EXECUTE.
- ☐ Delimited identifiers of table names and column names. Table and column names containing embedded blanks or other special characters in the SELECT list should be enclosed in double quotation marks.
- ☐ Column names qualified by table names or by table tags.
- ☐ The UNION [ALL], INTERSECT [ALL], and EXCEPT [ALL] operators.
- ☐ Non-correlated subqueries for all requests in the WHERE predicate and in the FROM list.
- ☐ Correlated subqueries for requests that are candidates for Dialect Translation to an RDBMS that supports this feature. Note that correlated subqueries are not supported for FOCUS and other non-relational data sources.
- ☐ Numeric constants, literals, and expressions in the SELECT list.
- ☐ Scalar functions for queries that are candidates for Dialect Translation if the RDBMS engine supports the scalar function type. These include: ABS, CHAR, CHAR\_LENGTH, CONCAT, COUNTBY, DATE, DAY, DAYS, DECIMAL, EDIT, EXTRACT, FLOAT, HOUR, IF, INT, INTEGER, LCASE, LENGTH, LOG, LTRIM, MICROSECOND, MILLISECOND, MINUTE, MONTH, POSITION, RTRIM, SECOND, SQRT, SUBSTR (or SUBSTRING), TIME, TIMESTAMP, TRIM, VALUE, UCASE, and YEAR.
- ☐ The concatenation operator, '||', used with literals or alphanumeric columns.
- ☐ The following aggregate functions: COUNT, MIN, MAX, SUM, and AVG.

- ❑ The following expressions can appear in conditions: CASE, NULLIF, and COALESCE.
- ❑ Date, time, and timestamp literals of several different formats. See [SQL Translator Support for Date, Time, and Timestamp Fields](#) on page 1830.
- ❑ All requests that contain ANY, SOME, and ALL that do not contain =ALL, <>ANY, and <>SOME.
- ❑ =ALL, <>ANY, and <>SOME for requests that are candidates for Dialect Translation if the RDBMS engine supports quantified subqueries.
- ❑ The special registers USER, CURRENT\_DATE, CURRENT\_TIME, CURRENT\_TIMESTAMP, CURRENT\_EDASQLVERSION, and CURRENT\_TIMEZONE.
- ❑ NULL and NOT NULL predicates.
- ❑ LIKE and NOT LIKE predicates.
- ❑ IN and NOT IN predicates.
- ❑ Date and time arithmetic.
- ❑ EXISTS and NOT EXISTS predicates.
- ❑ GROUP BY clauses expressed using explicit column names.
- ❑ ORDER BY clauses expressed using explicit column names or column numbers.
- ❑ FOR FETCH ONLY feature to circumvent record locking.
- ❑ Continental Decimal Notation (CDN) when the CDN variable is set.
- ❑ National Language Support (NLS).

**Reference: Unsupported SQL Statements**

SQL Translation Services does not support the following:

- ❑ More than 15 joins per SELECT. This limit is set by SQL. FOCUS supports up to 16 joins.
- ❑ ALIAS names in Master Files and the use of formatting options to format output.
- ❑ Unique truncations of column names.
- ❑ Temporary defined columns. Permanent defined columns, defined in the Reporting Server Dynamic Catalog or in the Master File, are supported.

- ❑ Correlated subqueries for DML Generation.

### **Reference:** SQL Translator Reserved Words

The following words may not be used as field names in a Master File that is used with the SQL Translator:

- ❑ ALL
- ❑ COUNT
- ❑ SUM
- ❑ MAX
- ❑ MIN
- ❑ AVG
- ❑ CURRENT
- ❑ DISTINCT
- ❑ USER

## Using SQL Translator Commands

The SQL command may be used to report from any supported data source or set of data sources. Standard TABLE phrases for formatting reports can be appended to the SQL statements to take advantage of a wide range of report preparation options.

**Note:** If you need to join data sources for your request, you have two options: use the JOIN command before you issue any SQL statements, or use the WHERE predicate in the SQL SELECT statement to join the required files dynamically. See [SQL Joins](#) on page 1823.

### **Syntax:** How to Use SQL Translator Commands

```
SQL
sql statement;
[ECHO|FILE]
[TABLE phrases]
END
```

where:

SQL

Is the SQL command identifier, which invokes the SQL Translator.



**Note:** The SQL command components must appear in the order represented above.

#### *sql statement*

Is a supported SQL statement. The statement must be terminated by a semicolon (;). It can continue for more than one line. See [Supported SQL Statements](#) on page 1818.

Within the SQL statement, field names are limited to 48 characters (an ANSI standard Level 2 limitation). View names generated through the SQL CREATE VIEW statement are limited to 18 characters and subqueries can be nested up to 15 levels deep. Correlated subqueries are not supported by FOCUS and other non-relational data sources.

#### *ECHO*

Are optional debugging phrases that capture the generated TABLE request. These options are placed after the SQL statement.

#### *FILE [name]*

Writes the translated TABLE phrases to the named procedure. If you do not supply a file name, a default name is assigned when the request runs. The file is then deleted.

#### *TABLE phrases*

Are optional TABLE formatting phrases. See [TABLE Formatting Phrases in SQL Requests](#) on page 1821.

#### *END or QUIT*

Is required to terminate the procedure.

### **Example:** Using SQL Translator Commands

The following request contains an SQL statement and TABLE formatting commands:

```
SQL
SELECT BODYTYPE, AVG(MPG), SUM(SALES)
FROM CAR
WHERE RETAIL_COST > 5000
GROUP BY BODYTYPE;
TABLE HEADING CENTER
"AVERAGE MPG AND TOTAL SALES PER BODYTYPE"
END
```

### **Reference:** TABLE Formatting Phrases in SQL Requests

You can include TABLE formatting phrases in an SQL request, subject to the following rules:

- ☐ Use TABLE formatting phrases with SELECT and UNION only.
- ☐ Introduce the formatting phrases with the word TABLE.

- ❑ You may specify headings and footings, describe actions with an ON phrase, or use the ON TABLE SET command. Additionally, you can use ON TABLE HOLD or ON TABLE PCHOLD to create an extract file. You can also specify READLIMIT and RECORDLIMIT tests.

For details on headings and footings, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.

For details on ON TABLE HOLD or ON TABLE PCHOLD, see [Saving and Reusing Your Report Output](#) on page 471.

- ❑ You cannot specify additional display fields, ACROSS fields, WHERE or IF criteria (other than READLIMIT or RECORDLIMIT tests), or calculated values. BY phrases are ignored.

## The SQL SELECT Statement

The SQL SELECT statement translates into one or more TABLE PRINT or TABLE SUM commands, depending on whether individual field display or aggregation is applied in the request. See [Displaying Report Data](#) on page 43.

The SQL statement SELECT \* translates to a PRINT of every field in the Master File, and uses all of the fields of the Cartesian product. This is a quick way to display a file, provided it fits in a reasonable number of screens for display, or provided you use ON TABLE HOLD or ON TABLE PCHOLD to retain retrieved data in a file for reuse. See [Saving and Reusing Your Report Output](#) on page 471.

SQL functions (such as COUNT, SUM, MAX, MIN, AVG) are supported in SELECT lists and HAVING conditions. Expressions may be used as function arguments.

The function COUNT (\*) translates to a count of the number of records produced by printing all fields in the Master File. This is the same as counting all rows in the Cartesian product that results from a SELECT on all fields.

Whenever possible, expressions in the SQL WHERE predicate are translated into corresponding WHERE criteria in the TABLE request. Expressions in SELECT lists generate virtual fields. The SQL HAVING clauses also translate into corresponding WHERE TOTAL criteria in the TABLE request. The SQL LIKE operator is translated directly into the corresponding LIKE operator in the WHERE criteria of the TABLE request. For details on record selection in TABLE requests, see [Selecting Records for Your Report](#) on page 219.

Only subqueries based on equality, when the WHERE expression is compared to a subquery by using an equal (=) sign, are supported. For example: WHERE field = (SELECT ...).

The SQL UNION operator translates to a TABLE request that creates a HOLD file for each data source specified, followed by a MATCH command with option HOLD OLD-OR-NEW, which combines records from both the first (old) data source and the second (new) data source. See [Merging Data Sources](#) on page 1075.

For related information, see [Supported SQL Statements](#) on page 1818 and [How to Use SQL Translator Commands](#) on page 1820.

## SQL Joins

When performing SQL joins, the formats of the joined fields must be the same. Join fields need not be indexed, and non-equi joins are supported.

Recursive, outer, and inner joins are supported. Inner join is the default.

### **Syntax:** How to Create an Inner Join

Two syntax variations are supported for inner joins.

#### **Variation 1**

```
SQL
SELECT fieldlist FROM file1 [alias1], file2 [alias2]
[WHERE where_condition];
END
```

#### **Variation 2**

```
SQL
SELECT fieldlist FROM file1 [alias1] INNER JOIN file2 [alias2]
ON join_condition [INNER JOIN ...]
[WHERE where_condition];
END
```

where:

*fieldlist*

Identifies which fields are retrieved from which data sources.

Joined fields in the SQL WHERE predicate must be qualified if the names are not unique. Specify them with their corresponding file names or file aliases. For example:

```
{file1|alias1}.field1, {file2|alias2}.field2
```

FROM

Introduces the data sources to be joined.

*file1, file2*

Are the data sources to be joined.

*alias1, alias2*

Are optional alternate names for the data sources to be joined.

*where\_condition*

Is an optional selection condition for the joined answer set. Joined rows that do not satisfy this condition are eliminated from the returned answer set. If omitted in Variation 1, the answer set is the Cartesian product of the two data sources.

*join\_condition*

Is the join condition.

### **Syntax:**      **How to Create an Outer Join**

```
SQL
SELECT fieldlist FROM file1 {LEFT|RIGHT|FULL} JOIN file2
ON join_condition [{LEFT|RIGHT|FULL} JOIN ...]
WHERE where_condition
END
```

where:

*fieldlist*

Identifies which fields are to be retrieved from which data sources.

Joined fields in the SQL WHERE predicate must be qualified if the names are not unique. Specify them with their corresponding file names or file aliases. For example:

```
{file1|alias1}.field1, {file2|
alias2}.field2
```

FROM

Introduces the data sources to be joined.

*file1, file2*

Are the data sources to be joined.

*alias1, alias2*

Are optional alternate names for the data sources to be joined.

*join\_condition*

Is the join condition. The condition must specify equality. For example, T1.A=T2.B.

*where\_condition*

Is an optional selection condition for the joined answer set. Joined rows that do not satisfy this condition are eliminated from the returned answer set.

**Reference: Join Name Assignments From the SQL Translator**

Joins issued by the SQL Translator are assigned names in the format:

*SQLJNMnn*

where:

*SQLJNM*

Is the SQL Translator join prefix.

*nn*

Is a number between 01 and 16 assigned in the order in which the joins are created (FOCUS supports a maximum of 16 joins). The first join has the AS name SQLJNM01, the second join is named SQLJNM02, and so on, up to SQLJNM16.

All joins are automatically created and cleared by the SQL Translator. No user-specified joins are affected.

**Example: Using Qualified Field Names in SQL Joins**

In the following statement, T.A and U.B are qualified field names:

```
SQL
SELECT T.A, T.B
FROM T, U
WHERE T.A = U.B;
END
```

**Example: Using Recursive SQL Joins**

In the following statement, A and B are aliases for the same data source, CAR. The output from CAR is pairs of B values that have the same A values:

```
SQL
SELECT A.SEATS, B.SEATS
FROM CAR A, CAR B
WHERE A.MODEL = B.MODEL;
END
```

Note that all field names in the SELECT clause must be unique or qualified.

**Example:**    **Using SQL Full Outer Joins**

In the following statement, B, C, and D are aliases for different data sources:

```
SQL
SELECT
    B.FIELD1 AS B_FIELD1, B.FIELD2 AS B_FIELD2,
    D.FIELD1 AS D_FIELD1, D.FIELD2 AS D_FIELD2
FROM
    ((FILE1 B FULL OUTER JOIN FILE2 C ON B.FIELD2 = C.FIELD2 )
     FULL OUTER JOIN FILE3 D ON C.FIELD2 = D.FIELD2 )
WHERE B.FIELD1 < 2
END
```

Multiple FULL OUTER JOINS are supported. However, they generate from a few to many temporary HOLD files.

**Reference:**    **SQL Join Considerations**

- ☐ In standard SQL, WHERE field='a' selects records where the field has the value 'a' or 'A'. The SQL Translator is case-sensitive and returns the exact value requested (in this case, 'a' only).
- ☐ The SQL comparison operators ANY, SOME, and ALL are supported, with the exception of =ALL, <>ANY, and <>SOME.
- ☐ Sub-selects are not supported in HAVING conditions.
- ☐ In a multi-segment structure, parent segments are omitted from reports if no instances of their descendant segments exist. This is an inner join.
- ☐ The SQL Translator applies optimization techniques when constructing joins. See [Index Optimized Retrieval](#) on page 1835.

For related information about index optimization and optimized join statements, see your Server documentation.

**SQL CREATE TABLE and INSERT INTO Commands**

SQL Translator supports the commands CREATE TABLE and INSERT INTO table:

- ☐ CREATE TABLE creates a new data source table. It only generates single-segment Master Files.
- ☐ INSERT INTO inserts a row or block of rows into a table or view. Single-record insert with actual data values is supported.

These commands enable you to create tables to enhance reporting efficiency.

**Note:** When applications are enabled, the Master File and data source are written to the APPHOLD directory. When applications are disabled, the Master File and data source are written to the TEMP directory.

**Reference:** Usage Notes for CREATE TABLE and INSERT INTO Commands

- ❑ According to normal SQL data definition syntax, each CREATE TABLE or INSERT INTO statement must terminate with a semicolon.
- ❑ The CREATE TABLE command supports the INTEGER, SMALLINT, FLOAT, CHARACTER, DATE, TIME, TIMESTAMP, DECIMAL, DOUBLE PRECISION and REAL data types. Decimals are rounded in the DOUBLE PRECISION and REAL data types.
- ❑ When using the CREATE TABLE and INSERT INTO commands, the data type FLOAT should be declared with a precision and used in an INSERT INTO command without the 'E' designation. This requires the entire value to be specified without an exponent.
- ❑ The CHECK and DEFAULT options are not supported with the CREATE TABLE command.

**Example:** Creating a Table With Single-Record Insert

The following shows a single-record insert, creating the table U with one record:

```

-* Single-record insert example.
-*
SQL
CREATE TABLE U (A INT, B CHAR(6), C CHAR(6), X INT, Y INT);
END
SQL
INSERT INTO U (A,B,C,X,Y) VALUES (10, '123456','654321', 10, 15);
END

```

## SQL CREATE VIEW and DROP VIEW Commands

A view is a transient object that inherits most of the characteristics of a table. Like a table, it is composed of rows and columns:

- ❑ CREATE VIEW creates views. Note that it does not put the view in the system catalog.
- ❑ DROP VIEW explicitly removes transient tables and views from the environment.

**Tip:** To use a view, issue a SELECT from it. You cannot issue a TABLE request against the view because the view is not extracted as a physical FOCUS data source. To create a HOLD file for extracted data, specify ON TABLE HOLD after the SQL statements. For details on creating HOLD files, see [Saving and Reusing Your Report Output](#) on page 471.

**Syntax:**      **How to Create a View**

The SQL Translator supports the following SQL statement:

```
CREATE VIEW viewname AS subquery ;
```

where:

*viewname*

Is the name of the view.

*subquery*

Is a SELECT statement that nests inside:

- ☐ A WHERE, HAVING, or SELECT clause of another SELECT.
- ☐ An UPDATE, DELETE, or INSERT statement.
- ☐ Another subquery.

**Example:**      **Creating and Reporting From an SQL View**

The following example creates a view named XYZ:

```
SQL  
CREATE VIEW XYZ  
  AS SELECT CAR, MODEL  
  FROM CAR;  
END
```

To report from the view, issue:

```
SQL  
  SELECT CAR, MODEL  
  FROM XYZ;  
END
```

According to normal SQL data definition syntax, each CREATE VIEW statement must terminate with a semicolon.

**Example:**      **Dropping an SQL View**

The following request removes the XYZ view:

```
SQL  
  DROP VIEW XYZ;  
END
```



## Cartesian Product Style Answer Sets

The SQL Translator automatically generates Cartesian product style answer sets unless you explicitly turn this feature off. However, it is advisable to leave the CARTESIAN setting on, since turning it off does not comply with ANSI standards. For details on the SET CARTESIAN command, see [Merging Data Sources](#) on page 1075.

## Continental Decimal Notation (CDN)

Continental Decimal Notation displays numbers using a comma to mark the decimal position and periods for separating significant digits into groups of three. This notation is available for SQL Translator requests.

### *Example:* Using CDN to Separate Digits

The following example creates a column defined as 1.2 + SEATS:

```
SET CDN=ON
SQL
    SELECT SEATS + 1,2
    FROM CAR;
END
```

## Specifying Field Names in SQL Requests

Specify fields in an SQL request using:

- ☐ **Delimited identifiers.** A field name may contain (but not begin with) the symbols ., #, @, \_, and \$. You must enclose such field names in double quotation marks when referring to them.
- ☐ **Qualified field names.** Qualify a field name with file and file alias names. File alias names are described in the discussion of joins in [SQL Joins](#) on page 1823. See the *Describing Data With WebFOCUS Language* manual for more information.
- ☐ **Field names with embedded blanks and special characters.** A SELECT list can specify field names with embedded blanks or other special characters. You must enclose such field names in double quotation marks. Special characters are any characters not listed as delimited identifiers, and not contained in the national character set of the installed FOCUS environment.

### *Example:* Specifying a Field Name With a Delimited Identifier

The following field identifier can be included in a request:

```
"COUNTRY . NAME "
```

### **Example: Qualifying a Delimited Field Name**

To qualify the delimited field name COUNTRY.NAME with its file name, use:

```
CAR . "COUNTRY . NAME "
```

### **SQL UNION, INTERSECT, and EXCEPT Operators**

The SQL UNION, INTERSECT, and EXCEPT operators generate MATCH logic. The number of files that can participate is determined by the MATCH limit. UNION with parentheses is supported.

- ❑ SELECT A UNION SELECT B retrieves rows in A or B or both. (This is equivalent to the MATCH phrase OLD-OR-NEW.)
- ❑ INTERSECT retrieves rows in both A and B. (This is equivalent to the MATCH phrase OLD-AND-NEW.)
- ❑ EXCEPT retrieves rows in A, but not B. (This is equivalent to the MATCH phrase OLD-NOT-NEW.)

Match logic merges the contents of your data sources. See [Merging Data Sources](#) on page 1075.

### **Numeric Constants, Literals, Expressions, and Functions**

The SQL SELECT list, WHERE predicate, and HAVING clause can include numeric constants, literals enclosed in single quotation marks, expressions, and any scalar functions. Internally, a virtual field is created for each of these in the SELECT list. The value of the virtual field is provided in the answer set.

### **SQL Translator Support for Date, Time, and Timestamp Fields**

Several new data types have been defined for the SQL Translator to support date-time fields in the WHERE predicate or field list of a SELECT statement.

In addition, time or timestamp columns can be defined in relational or FOCUS data sources, and are accessible to the translator. Values can be entered using INSERT and UPDATE statements, and displayed in SELECT statements.

Time or timestamp data items (columns or literals) can be compared in conditions. Time values or timestamp values can be added to or subtracted from each other, with the result being the difference in number of seconds. Expressions of the form T + 2 HOURS or TS + 5 YEARS are allowed. These expressions are translated to calls to the date-time functions described in the *Using Functions* manual.

All date formats for actual and virtual fields in the Master File are converted to the form YYYYMMDD. If you specify a format that lacks any component, the SQL Translator supplies a default value for the missing component. To specify a portion of a date, such as the month, use a virtual field with an alphanumeric format.

**Reference: SQL Translator Support for Date, Time, and Timestamp Fields**

In the following chart, fff represents the second to three decimal places (milliseconds) and ffffff represents the second to six decimal places (microseconds).

The following formats are allowed as input to the Translator:

Format	USAGE Attribute in Master File	Date Components
Date	YYMD	YYYY-MM-DD
Hour	HH	HH
Hour through minute	HHI	HH.MM
Hour through second	HHIS	HH.MM.SS
Hour through millisecond	HHISs	HH.MM.SS.fff
Hour through microsecond	HHISsm	HH.MM.SS.ffffff
Year through hour	HYYMDH	YYYY-MM-DD HH
Year through minute	HYYMDI	YYYY-MM-DD HH.MM
Year through second	HYYMDS	YYYY-MM-DD HH.MM.SS
Year through millisecond	HYYMDs	YYYY-MM-DD HH.MM.SS.fff
Year through microsecond	HYYMDm	YYYY-MM-DD HH.MM.SS.ffffff

**Note:**

- ☐ Time information may be given to the hour, minute, second, or fraction of a second.

- ❑ The separator within date information may be either a hyphen or a slash.
- ❑ The separator within time information must be a colon.
- ❑ The separator between date and time information must be a space.

Extracting Date-Time Components Using the SQL Translator

The SQL Translator supports several functions that return components from date-time values. Use the EXTRACT statement to extract components.

Use the TRIM function to remove leading and/or trailing patterns from date, time, and timestamp values. See the *Using Functions* manual.

**Syntax:**      **How to Use Date, Time, and Timestamp Functions Accepted by the SQL Translator**

The following functions return date-time components as integer values. Assume x is a date-time value:

Function	Return Value
<code>YEAR ( x )</code>	year
<code>MONTH ( x )</code>	month number
<code>DAY ( x )</code>	day number
<code>HOURL ( x )</code>	hour
<code>MINUTE ( x )</code>	minute
<code>SECOND ( x )</code>	second
<code>MILLISECOND ( x )</code>	millisecond
<code>MICROSECOND ( x )</code>	microsecond

**Example: Using SQL Translator Date, Time, and Timestamp Functions**

Using the timestamp column TS whose value is '1999-11-23 07:32:16.123456':

```
YEAR(TS) = 1999
MONTH(TS) = 11
DAY(TS) = 23
HOUR(TS) = 7
MINUTE(TS) = 32
SECOND(TS) = 16
MILLISECOND(TS) = 123
MICROSECOND(TS) = 123456
```

**Example: Using SQL Translator Date, Time, and Timestamp Functions in a SELECT Statement**

Assume that a FOCUS data source called VIDEOTR2 includes a date-time field named TRANSDATE.

```
SQL
SELECT TRANSDATE,
YEAR(TRANSDATE), MONTH(TRANSDATE),
MINUTE(TRANSDATE)
FROM VIDEOTR2;
FILE VIDSQ
END
```

The SQL Translator produces the following virtual fields for functions, followed by a TABLE request to display the output:

```
SET COUNTWIDTH=ON
-SET SQLERRNUM = 0;
DEFINE FILE
VIDEOTR2 TEMP
SQLDEF01/I4 MISSING ON NEEDS ALL = HPART(TRANSDATE,'YEAR','I4');
SQLDEF02/I2 MISSING ON NEEDS ALL = HPART(TRANSDATE,'MONTH','I2');
SQLDEF03/I2 MISSING ON NEEDS ALL = HPART(TRANSDATE,'MINUTE','I2');
END
TABLEF FILE VIDEOTR2
PRINT TRANSDATE SQLDEF01 AS 'SQLDEF01' SQLDEF02 AS 'SQLDEF02' SQLDEF03 AS
'SQLDEF03'
ON TABLE SET CARTESIAN ON
ON TABLE SET ASNAMES ON
ON TABLE SET HOLDLIST PRINTONLY
END
```

The output is:

TRANSDATE	SQLDEF02	SQLDEF04	SQLDEF05
1999/06/20 04:14	1999	6	14
1991/06/27 02:45	1991	6	45
1996/06/21 01:16	1996	6	16
1991/06/21 07:11	1991	6	11
1991/06/20 05:15	1991	6	15
1999/06/26 12:34	1999	6	34
1919/06/26 05:45	1919	6	45
1991/06/21 01:10	1991	6	10
1991/06/19 07:18	1991	6	18
1991/06/19 04:11	1991	6	11
1998/10/03 02:41	1998	10	41
1991/06/25 01:19	1991	6	19
1986/02/05 03:30	1986	2	30
1991/06/24 04:43	1991	6	43
1991/06/24 02:08	1991	6	8
1999/10/06 02:51	1999	10	51
1991/06/25 01:17	1991	6	17

**Syntax:**      **How to Use the SQL Translator EXTRACT Function to Extract Date-Time Components**

Use the following ANSI standard function to extract date-time components as integer values:

```
EXTRACT(component FROM value)
```

where:

*component*

Is one of the following: YEAR, MONTH, QUARTER, DAY, WEEKDAY, HOUR, MINUTE, SECOND, MILLISECOND, or MICROSECOND.

*value*

Is a date-time, DATE, TIME, or TIMESTAMP field, constant or expression.

For example, the following are equivalent:

```
EXTRACT(YEAR FROM TS)
YEAR(TS)
```

**Example:**      **Using the EXTRACT Function**

```
SELECT D. EXTRACT(YEAR FROM D), EXTRACT(MONTH FROM D),
EXTRACT(DAY FROM D) FROM T
```

This request produces rows similar to the following:

1999-01-01	1999	1	1
2000-03-03	2000	3	3

## Index Optimized Retrieval

The SQL Translator improves query performance by generating optimized code that enables the underlying retrieval engine to access the selected records directly, without scanning all segment instances.

For more information about index optimization and optimized join statements, see your Server documentation for your platform.

## Optimized Joins

The SQL Translator accepts joins in SQL syntax. SQL language joins have no implied direction. The concepts of host and cross-referenced files do not exist in SQL.

The SQL Translator analyzes each join to identify efficient implementation. First, it assigns costs to the candidate joins in the query:

- ❑ Cost = 1 for an equijoin to a field that can participate as a cross-referenced field according to FOCUS join rules. This is common in queries against relational tables with equijoin predicates in the WHERE clause.
- ❑ Cost = 16 for an equijoin to a field that cannot participate as a cross-referenced field according to FOCUS join rules.
- ❑ Cost = 256 for a non-equijoin or an unrestricted Cartesian product.

The Translator then uses these costs to build a join structure for the query. The order of the tables in the FROM clause of the query influences the first two phases of the join analysis:

1. If there are cost=1 joins from the first table referenced in the FROM clause to the second, from the second table to the third, and so on, the Translator joins the tables in the order specified in the query. If not, it goes on to Phase 2.
2. If Phase 1 fails to generate an acceptable join structure, the Translator attempts to generate a join structure without joining any table to a table that precedes it in the FROM clause. Therefore, this phase always makes the first table referenced in the query the host table. If there is no cost=1 join between two tables, or if using one requires changing the table order, the Translator abandons Phase 2 and implements Phase 3.
3. The Translator generates the join structure from the lowest-cost joins first, and then from the more expensive joins as necessary. This sorting process may change the order in which tables are joined. The efficiency of the join that this procedure generates depends on the relative sizes of the tables being joined.

If the analysis results in joining to a table that cannot participate as a cross-referenced file according to FOCUS rules (because it lacks an index, for example), the Translator generates code to build an indexed HOLD file, and implements the join with this file. However, the HOLD file does not participate in the analysis of join order.

## TABLEF Optimization

To improve performance, the SQL Translator can be set to generate FOCUS TABLEF commands instead of TABLE commands. Take advantage of this optimization using the SET SQLTOPTTF command (SQL Translator OPTimization TableF). See [Improving Report Processing](#) on page 1839.

### **Syntax:** How to Improve Performance Using SQLTOPTTF

```
SET SQLTOPTTF = {ON|OFF}
```

where:

[ON](#)

Causes TABLEF commands to be generated when possible (for example, if there is no join or GROUP BY phrase). ON is the default value.

[OFF](#)

Causes TABLE commands to be generated.

## SQL INSERT, UPDATE, and DELETE Commands

The SQL INSERT, UPDATE, and DELETE commands enable SQL users to manipulate and modify data:

- ☐ The INSERT statement introduces new rows into an existing table.
- ☐ The DELETE statement removes a row or combination of rows from a table.
- ☐ The UPDATE statement enables users to update a row or group of rows in a table.

You can issue an SQL INSERT, UPDATE, or DELETE command against one segment instance (row) at a time. When you issue one of these commands against a multi-segment Master File:

- ☐ All fields referenced in the command must be on a single path through the file structure.
- ☐ The command must explicitly specify (in the WHERE predicate) every key value from the root to the target segment instance, and this combination of key values must uniquely identify one segment instance (row) to be affected by the command.



If you are modifying every field in the row, you can omit the list of field names from the command.

- ❑ The SQL Translator supports subqueries, such as:

```
INSERT...INTO...SELECT...FROM...
```

Although each INSERT, UPDATE, or DELETE command can specify only one row, referential integrity constraints may produce the following modifications to the data source:

- ❑ If you delete a segment instance that has descendant segment instances (children), the children are automatically deleted.
- ❑ If you insert a segment for which parent segments are missing, the parent segments are automatically created.





# Chapter 28

## Improving Report Processing

---

The following high-performance methods optimize data retrieval and report processing:

- ☐ Temporary rotation of network and hierarchical data sources to create an alternate view of the data.
- ☐ Automatic alternate file views with the AUTOPATH feature.
- ☐ Automatic indexed retrieval (AUTOINDEX).
- ☐ Retrieval of pre-sorted data using the TABLEF command.
- ☐ Preserving the internal matrix of a report using the SAVEMATRIX parameter.
- ☐ Compiling expressions into machine code to provide faster processing.

**Note:** These techniques may not be available for all data sources. See your data adapter documentation to determine if a technique is valid for your data source.

**In this chapter:**

- ☐ [Rotating a Data Structure for Enhanced Retrieval](#)
  - ☐ [Optimizing Retrieval Speed for FOCUS Data Sources](#)
  - ☐ [Automatic Indexed Retrieval](#)
  - ☐ [Data Retrieval Using TABLEF](#)
  - ☐ [Compiling Expressions](#)
- 

### Rotating a Data Structure for Enhanced Retrieval

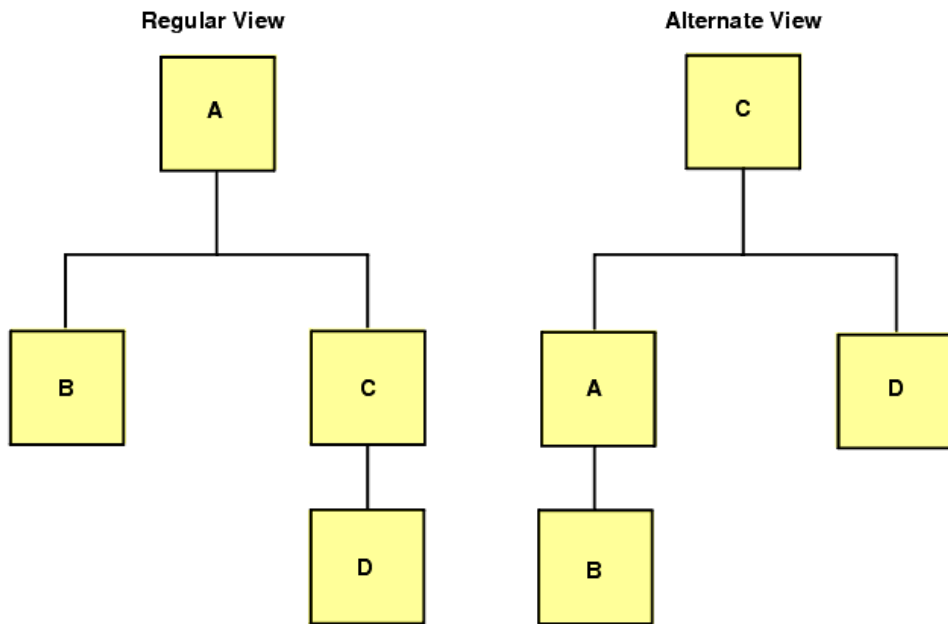
If you are using certain network or hierarchical data sources such as IMS, CA-IDMS/DB, or FOCUS, you can rotate the data source, creating an alternate view which changes some of the segment relationships and enables you to access the segments in a different order. By reporting from an alternate view, you can do the following:

- ☐ Change the access path. For example, you can access data in a lower segment more quickly by promoting that segment to a higher level.

- ❑ Change the path structure of a data source. This option is especially helpful if you wish to create a report using several sort fields that are on different paths in the file. By changing the view of the file hierarchy, all the desired sort fields can be on the same path.

It should be noted that retrieval is controlled by the minimum referenced subtree. For more information, see *Understanding the Efficiency of the Minimum Referenced Subtree* in the *Describing a Group of Fields* chapter in the *Describing Data With WebFOCUS Language* manual.

For example, consider the regular and alternate views below:



Since C is the root segment in the alternate view, particular instances of C can be selected faster.

### **Syntax:** How to Request an Alternate View

To request an alternate view, add the name of a field found in the alternate root segment to the file name in the TABLE command, separated by a period (.):

```
TABLE FILE filename.fieldname
```

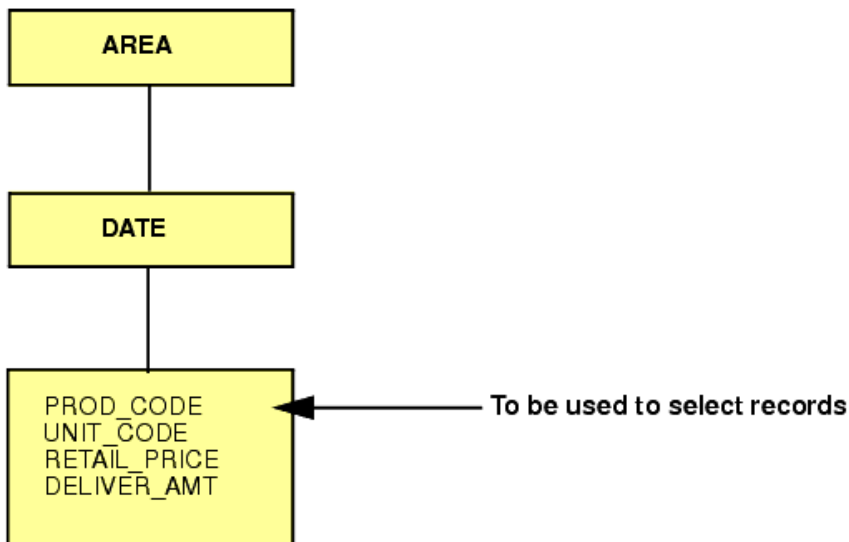
### **Reference:** Usage Notes for Restructuring Data

- ❑ If you use a non-indexed field, each segment instance is retrieved until the specified record is found. Therefore, this process is less efficient than using an indexed field.

- ❑ When you use the alternate view feature on a particular child segment, the data retrieved from that segment is retrieved in physical order, not logical order. This is because the child becomes a root segment for the report request, and there are no logical pointers between the child segments of different parents.
- ❑ Alternate view on an indexed field is a special case that uses the index for retrieval. When you perform an alternate view on an indexed field, you enhance the speed of retrieval. However, you must include an equality test on the indexed field, for example WHERE (MONTH EQ 1) OR (MONTH EQ 2), in order to benefit from the performance improvement.
- ❑ A field name specified in an alternate file view may not be qualified or exceed 12 characters.
- ❑ Automatic Indexed Retrieval (AUTOINDEX) is never invoked in a TABLE request against an alternate file view.

**Example: Restructuring Data**

Consider the following data structure, in which PROD\_CODE is an indexed field:



You could issue the following request to promote the segment containing PROD\_CODE to the top of the hierarchy, thereby enabling quicker access to the data in that segment.

```
TABLE FILE SALES.PROD_CODE
"SALES OF B10 DISTRIBUTED BY AREA"
SUM UNIT_SOLD AND RETAIL_PRICE
BY AREA
WHERE PROD_CODE EQ 'B10'
ON TABLE COLUMN-TOTAL
END
```

## Optimizing Retrieval Speed for FOCUS Data Sources

When the AUTOPATH parameter is set ON, an optimized retrieval path—that is, one in which the lowest retrieved segment is the entry point—is selected dynamically. It is equivalent to the alternate view syntax

```
TABLE FILE filename.fieldname
```

where:

*fieldname*

Is not indexed. Retrieval starts at the segment in which *fieldname* resides.

The system determines whether optimized retrieval is appropriate by analyzing the fields referenced in a request and the data source structure. For more information on the AUTOPATH parameter, see the *Developing Reporting Applications* manual.

## Automatic Indexed Retrieval

Automatic indexed retrieval (AUTOINDEX) optimizes the speed of data retrieval in FOCUS data sources. To take advantage of automatic indexed retrieval, a TABLE request must contain an equality or range test on an indexed field in the highest segment referenced in the request.

This method is not supported if a:

- ☐ Range test applies to a packed data value.
- ☐ Request specifies an alternate view (that is, TABLE FILE *filename.fieldname*).
- ☐ Request contains the code BY HIGHEST or BY LOWEST.

For related information on AUTOINDEX, see the *Developing Reporting Applications* manual.

**Syntax:**      **How to Use Indexed Retrieval**

```
SET AUTOINDEX = {ON|OFF}
```

where:

ON

Uses indexed data retrieval for optimized speed when possible. The request must contain an equality or range test on an indexed field in the highest segment referenced in the request. ON is the default value.

OFF

Uses sequential data retrieval unless a request specifies an indexed view (TABLE FILE *filename.indexed\_fieldname*) and contains an equality test on *indexed\_fieldname*. In that case, indexed data retrieval is automatically performed.

**Reference:**      **Usage Notes for Indexed Retrieval**

- ☐ AUTOINDEX is never invoked when the TABLE request contains an alternate file view (that is, TABLE FILE *filename.fieldname*).
- ☐ Even if AUTOINDEX is ON, indexed retrieval is not performed when the TABLE request contains BY HIGHEST or BY LOWEST phrases.
- ☐ When a request specifies an indexed view (as in TABLE FILE *filename.indexed\_fieldname*), indexed retrieval is implemented under the following circumstances:
  - ☐ AUTOINDEX is OFF and the request contains an equality test on the indexed field.
  - ☐ AUTOINDEX is ON and the request contains either an equality or a range (FROM ... TO) test against the indexed field.

**Example: Using Indexed Retrieval**

The following Master File is referenced in the examples that follow:

```
FILENAME=SALES,SUFFIX=FOC,
  SEGNAME=STOR_SEG,SEGTYPE=S1,
    FIELDNAME=AREA,ALIAS=LOC,FORMAT=A1,$
  SEGNAME=DATE_SEG,PARENT=STOR_SEG,SEGTYPE=SH1,
    FIELDNAME=DATE,ALIAS=DTE,FORMAT=A4MD,$
  SEGNAME=DEPT,PARENT=DATE_SEG,SEGTYPE=S1,
    FIELDNAME=DEPARTMENT,ALIAS=DEPT,FORMAT=A5,FIELDTYPE=I,$
    FIELDNAME=DEPT_CODE,ALIAS=DCODE,FORMAT=A3,FIELDTYPE=I,$
    FIELDNAME=PROD_TYPE,ALIAS=PTYPE,FORMAT=A10,FIELDTYPE=I,$
  SEGNAME=INVENTORY,PARENT=DEPT,SEGTYPE=S1,$
    FIELDNAME=PROD_CODE,ALIAS=PCODE,FORMAT=A3,FIELDTYPE=I,$
    FIELDNAME=UNIT_SOLD,ALIAS=SOLD,FORMAT=I5,$
    FIELDNAME=RETAIL_PRICE,ALIAS=RP,FORMAT=D5.2M,$
    FIELDNAME=DELIVER_AMT,ALIAS=SHIP,FORMAT=I5,$
```

The following procedure contains an equality test on DEPT\_CODE and PROD\_CODE.

DEPT\_CODE is used for indexed retrieval since it is in the higher of the referenced segments.

```
SET AUTOINDEX=ON
TABLE FILE SALES
SUM UNIT_SOLD RETAIL_PRICE
IF DEPT_CODE EQ 'H01'
IF PROD_CODE EQ 'B10'
END
```

If your TABLE request contains an equality or range test against more than one indexed field in the same segment, AUTOINDEX uses the first index referenced in that segment for retrieval.

The following stored procedure contains an equality test against two indexed fields. Since DEPT\_CODE appears before PROD\_TYPE in the Master File, AUTOINDEX uses DEPT\_CODE for retrieval.

```
SET AUTOINDEX=ON
TABLE FILE SALES
SUM UNIT_SOLD AND RETAIL_PRICE
IF PROD_TYPE EQ 'STEREO'
IF DEPT_CODE EQ 'H01'
END
```



Indexed retrieval is not invoked if the equality or range test is run against an indexed field that does not reside in the highest referenced segment. In the following example, indexed retrieval is not performed, because the request contains a reference to AREA, a field in the STOR\_SEG segment:

```
SET AUTOINDEX=ON
TABLE FILE SALES
SUM UNIT_SOLD AND RETAIL_PRICE
BY AREA
IF PROD_CODE EQ 'B10'
IF PROD_TYPE EQ 'STEREO'
END
```

## Data Retrieval Using TABLEF

TABLEF is a variation of the TABLE command that provides a fast method of retrieving data that is already stored in the order required for printing and requires no additional sorting.

Using TABLEF, records are retrieved in the logical sequence from the data source. The standard report request syntax applies, subject to the following rules:

- ☐ Any BY phrases must be compatible with the logical sequence of the data source. BY phrases are used only to establish control breaks, not to change the order of the records.
- ☐ ACROSS phrases are not permitted.
- ☐ Multiple display commands are not permitted. Only one display command may be used.
- ☐ After the report is executed, RETYPE, HOLD, and SAVE are not available. However, you can produce an extract file if you include ON TABLE HOLD or ON TABLE SAVE as part of the request.
- ☐ NOSPLIT is not compatible with the TABLEF command, and produces a FOC037 error message.
- ☐ TABLEF can be used with HOLD files and other non-FOCUS data sources when the natural sort sequence of both the request and the data are the same.
- ☐ TABLEF is not supported with SET EMPTYREPORT. When a TABLEF request retrieves zero records, EMPTYREPORT behaves as if it were set to ON.
- ☐ The DST. prefix operator is not permitted.
- ☐ BORDER styling is not supported with TABLEF.
- ☐ TABLEF is not supported with SQUEEZE.

### **Example:** Printing Using Fast Table Retrieval

If you previously created a HOLD file from the EMPLOYEE data source, sorted by the CURR\_SAL, LAST\_NAME, and FIRST\_NAME fields, you can issue the following TABLEF request:

```
TABLEF FILE HOLD  
PRINT CURR_SAL AND LAST_NAME AND FIRST_NAME  
END
```

## Compiling Expressions

Compiling expressions into machine code provides faster processing.

### Compiling Expressions Using the DEFINES Parameter

The SET DEFINES, SET COMPUTE, and SET MODCOMPUTE commands have been deprecated. Expressions are compiled unless environmental conditions prevent compilation.

Among the benefits of the compiling expressions are:

- ☐ Compilation of only those expressions that are actually used in the TABLE request.
- ☐ Much faster execution of expressions containing complex calculations on long packed fields.
- ☐ Compilation of date expressions.

### **Reference:** Usage Notes for Compiled Expressions

- ☐ Any expression that cannot be compiled runs without compilation. This does not affect compilation of other expressions. The following elements in an expression disable compilation:
  - ☐ Functions. However, expressions that use the following functions can be compiled: YMD, DMY, INT, and DECODE.
  - ☐ CONTAINS, OMITS, LAST.

If compilation is not possible because of environmental conditions, the processing is handled without compilation. No message is generated indicating that compilation did not take place. To determine whether it did take place, issue the ? COMPILE command.

## Master Files and Diagrams

---

This appendix contains descriptions and structure diagrams for the sample data sources used throughout the documentation.

**In this appendix:**

- |   |   |
|---|---|
| <input type="checkbox"/> <a href="#">EMPLOYEE Data Source</a> | <input type="checkbox"/> <a href="#">COURSE Data Source</a>                       |
| <input type="checkbox"/> <a href="#">JOBFILE Data Source</a>  | <input type="checkbox"/> <a href="#">JOBHIST Data Source</a>                      |
| <input type="checkbox"/> <a href="#">EDUCFILE Data Source</a> | <input type="checkbox"/> <a href="#">JOBLIST Data Source</a>                      |
| <input type="checkbox"/> <a href="#">SALES Data Source</a>    | <input type="checkbox"/> <a href="#">LOCATOR Data Source</a>                      |
| <input type="checkbox"/> <a href="#">CAR Data Source</a>      | <input type="checkbox"/> <a href="#">PERSINFO Data Source</a>                     |
| <input type="checkbox"/> <a href="#">LEDGER Data Source</a>   | <input type="checkbox"/> <a href="#">SALHIST Data Source</a>                      |
| <input type="checkbox"/> <a href="#">FINANCE Data Source</a>  | <input type="checkbox"/> <a href="#">VIDEOTRK, MOVIES, and ITEMS Data Sources</a> |
| <input type="checkbox"/> <a href="#">REGION Data Source</a>   | <input type="checkbox"/> <a href="#">VIDEOTR2 Data Source</a>                     |
| <input type="checkbox"/> <a href="#">EMPDATA Data Source</a>  | <input type="checkbox"/> <a href="#">Gotham Grinds Data Sources</a>               |
| <input type="checkbox"/> <a href="#">TRAINING Data Source</a> | <input type="checkbox"/> <a href="#">Century Corp Data Sources</a>                |
- 

### EMPLOYEE Data Source

EMPLOYEE contains sample data about company employees. Its segments are:

[EMPINFO](#)

Contains employee IDs, names, and positions.

[FUNDTRAN](#)

Specifies employee direct deposit accounts. This segment is unique.

### PAYINFO

Contains the employee salary history.

### ADDRESS

Contains employee home and bank addresses.

### SALINFO

Contains data on employee monthly pay.

### DEDUCT

Contains data on monthly pay deductions.

EMPLOYEE also contains cross-referenced segments belonging to the JOBFIL and EDUCFIL files, also described in this appendix. The segments are:

### JOBSEG (from JOBFIL)

Describes the job positions held by each employee.

### SKILLSEG (from JOBFIL)

Lists the skills required by each position.

### SECSEG (from JOBFIL)

Specifies the security clearance needed for each job position.

### ATTNDSEG (from EDUCFIL)

Lists the dates that employees attended in-house courses.

COURSEG (from EDUCFILE)

Lists the courses that the employees attended.

## EMPLOYEE Master File

```

FILENAME=EMPLOYEE, SUFFIX=FOC
SEGNAME=EMPINFO, SEGTYPE=S1
  FIELDNAME=EMP_ID, ALIAS=EID, FORMAT=A9, $
  FIELDNAME=LAST_NAME, ALIAS=LN, FORMAT=A15, $
  FIELDNAME=FIRST_NAME, ALIAS=FN, FORMAT=A10, $
  FIELDNAME=HIRE_DATE, ALIAS=HDT, FORMAT=I6YMD, $
  FIELDNAME=DEPARTMENT, ALIAS=DPT, FORMAT=A10, $
  FIELDNAME=CURR_SAL, ALIAS=CSAL, FORMAT=D12.2M, $
  FIELDNAME=CURR_JOBCODE, ALIAS=CJC, FORMAT=A3, $
  FIELDNAME=ED_HRS, ALIAS=OJT, FORMAT=F6.2, $
SEGNAME=FUNDTRAN, SEGTYPE=U, PARENT=EMPINFO
  FIELDNAME=BANK_NAME, ALIAS=BN, FORMAT=A20, $
  FIELDNAME=BANK_CODE, ALIAS=BC, FORMAT=I6S, $
  FIELDNAME=BANK_ACCT, ALIAS=BA, FORMAT=I9S, $
  FIELDNAME=EFFECT_DATE, ALIAS=EDATE, FORMAT=I6YMD, $
SEGNAME=PAYINFO, SEGTYPE=SH1, PARENT=EMPINFO
  FIELDNAME=DAT_INC, ALIAS=DI, FORMAT=I6YMD, $
  FIELDNAME=PCT_INC, ALIAS=PI, FORMAT=F6.2, $
  FIELDNAME=SALARY, ALIAS=SAL, FORMAT=D12.2M, $
  FIELDNAME=JOBCODE, ALIAS=JBC, FORMAT=A3, $
SEGNAME=ADDRESS, SEGTYPE=S1, PARENT=EMPINFO
  FIELDNAME=TYPE, ALIAS=AT, FORMAT=A4, $
  FIELDNAME=ADDRESS_LN1, ALIAS=LN1, FORMAT=A20, $
  FIELDNAME=ADDRESS_LN2, ALIAS=LN2, FORMAT=A20, $
  FIELDNAME=ADDRESS_LN3, ALIAS=LN3, FORMAT=A20, $
  FIELDNAME=ACCTNUMBER, ALIAS=ANO, FORMAT=I9L, $
SEGNAME=SALINFO, SEGTYPE=SH1, PARENT=EMPINFO
  FIELDNAME=PAY_DATE, ALIAS=PD, FORMAT=I6YMD, $
  FIELDNAME=GROSS, ALIAS=MO_PAY, FORMAT=D12.2M, $
SEGNAME=DEDUCT, SEGTYPE=S1, PARENT=SALINFO
  FIELDNAME=DED_CODE, ALIAS=DC, FORMAT=A4, $
  FIELDNAME=DED_AMT, ALIAS=DA, FORMAT=D12.2M, $
SEGNAME=JOBSEG, SEGTYPE=KU, PARENT=PAYINFO, CRFILE=JOBFILE,
  CRKEY=JOBCODE,$
SEGNAME=SECSEG, SEGTYPE=KLU, PARENT=JOBSEG, CRFILE=JOBFILE, $
SEGNAME=SKILLSEG, SEGTYPE=KL, PARENT=JOBSEG, CRFILE=JOBFILE, $
SEGNAME=ATTNDSEG, SEGTYPE=KM, PARENT=EMPINFO, CRFILE=EDUCFILE,
  CRKEY=EMP_ID,$
SEGNAME=COURSEG, SEGTYPE=KLU, PARENT=ATTNDSEG, CRFILE=EDUCFILE,$

```

```

SECTION 01
STRUCTURE OF FOCUS FILE EMPLOYEE ON 05/15/03 AT 10.16.27

EMPINFO
01 S1
*****
*EMP_ID **
*LAST_NAME **
*FIRST_NAME **
*HIRE_DATE **
*
*****
I
I-----I
I I I I I I I
I I FUNDTRAN I PAYINFO I ADDRESS I SALINFO I ATTNDEG
02 I U 03 I SH1 07 I S1 08 I SH1 10 I KM
*****
*BANK_NAME * *DAT_INC ** *TYPE ** *PAY_DATE ** :DATE_ATTEND :
*BANK_CODE * *PCT_INC ** *ADDRESS_LN1 ** *GROSS ** :EMP_ID :K
*BANK_ACCT * *SALARY ** *ADDRESS_LN2 ** : :
*EFFECT_DATE * *JOBCODE ** *ADDRESS_LN3 ** * : :
* * * ** * ** : :
*****
I I I I I
I I I I I
I I I I I
04 I JOBSEC 09 I DEDUCT 11 I ELU
I I KU
*****
:JOBCODE :K :DED_CODE ** :COURSE_CODE :
:JOB_DESC : :DED_AMT ** :COURSE_NAME :
: : : * ** : :
: : : * ** : :
: : : * ** : :
: : : : :
*****
I JOBFILE ***** EDUCFILE
I
I-----I
I I I
I I SECSEC I SKILLSEC
05 I KLU 06 I KL
*****
:SEC_CLEAR : :SKILLS :
: : :SKILL_DESC :
: : : :
: : : :
: : : :
: : : :
*****
JOBFILE *****
JOBFILE

```

**SKILLSEG**

Lists the skills required by each position.

**SECSEG**

Specifies the security clearance needed, if any. This segment is unique.

**JOBFILE Master File**

```

FILENAME=JOBFILE,  SUFFIX=FOC
SEGNAME=JOBSEG,   SEGTYPE=S1
  FIELDNAME=JOBCODE,    ALIAS=JC,  FORMAT=A3,    INDEX=I,$
  FIELDNAME=JOB_DESC,   ALIAS=JD,  FORMAT=A25     ,,$
SEGNAME=SKILLSEG,  SEGTYPE=S1,   PARENT=JOBSEG
  FIELDNAME=SKILLS,     ALIAS=,    FORMAT=A4      ,,$
  FIELDNAME=SKILL_DESC, ALIAS=SD,  FORMAT=A30     ,,$
SEGNAME=SECSEG,   SEGTYPE=U,    PARENT=JOBSEG
  FIELDNAME=SEC_CLEAR, ALIAS=SC,  FORMAT=A6       ,,$

```

**JOBFILE Structure Diagram**

```

SECTION 01
STRUCTURE OF FOCUS      FILE JOBFILE ON 05/15/03 AT 14.40.06

```

```

      JOBSEG
01      S1
*****
*JOBCODE      **I
*JOB_DESC     **
*              **
*              **
*              **
*****
      I
      +-----+
      I              I
      I SECSEG      I SKILLSEG
02      I U          03      I S1
*****
*SEC_CLEAR    *      *SKILLS      **
*              *      *SKILL_DESC **
*              *      *              **
*              *      *              **
*              *      *              **
*****
                        *****
                        *****

```

**EDUCFILE Data Source**

EDUCFILE contains sample data about company in-house courses. Its segments are:

**COURSEG**

Contains data on each course.

**ATTNDSEG**

Specifies which employees attended the courses. Both fields in the segment are key fields. The field EMP\_ID in this segment is indexed.

**EDUCFILE Master File**

```
FILENAME=EDUCFILE, SUFFIX=FOC
SEGNAME=COURSEG, SEGTYPE=S1
  FIELDNAME=COURSE_CODE, ALIAS=CC, FORMAT=A6, $
  FIELDNAME=COURSE_NAME, ALIAS=CD, FORMAT=A30, $
SEGNAME=ATTNDSEG, SEGTYPE=SH2, PARENT=COURSEG
  FIELDNAME=DATE_ATTEND, ALIAS=DA, FORMAT=I6YMD, $
  FIELDNAME=EMP_ID, ALIAS=EID, FORMAT=A9, INDEX=I, $
```



## EDUCFILE Structure Diagram

```

SECTION 01                                STRUCTURE OF FOCUS    FILE EDUCFILE ON 05/15/03 AT 14.45.44

                                COURSEG
01                                S1
*****
*COURSE_CODE **
*COURSE_NAME **
*              **
*              **
*              **
*****
*****
                                I
                                I
                                I
                                I ATTNDSEG
02                                I SH2
*****
*DATE_ATTEND **
*EMP_ID       **I
*              **
*              **
*              **
*****
*****

```

## SALES Data Source

SALES contains sample data about a dairy company with an affiliated store chain. Its segments are:

### STOR\_SEG

Lists the stores buying the products.

### DAT\_SEG

Contains the dates of inventory.

### PRODUCT

Contains sales data for each product on each date. The PROD\_CODE field is indexed. The RETURNS and DAMAGED fields have the MISSING=ON attribute.

**SALES Master File**

```
FILENAME=KSALES,    SUFFIX=FOC
SEGNAME=STOR_SEG,  SEGTYPE=S1
  FIELDNAME=STORE_CODE,  ALIAS=SNO,    FORMAT=A3,    $
  FIELDNAME=CITY,        ALIAS=CTY,    FORMAT=A15,   $
  FIELDNAME=AREA,        ALIAS=LOC,    FORMAT=A1,    $
SEGNAME=DATE_SEG,  PARENT=STOR_SEG,  SEGTYPE=SH1,
  FIELDNAME=DATE,        ALIAS=DTE,    FORMAT=A4MD,  $
SEGNAME=PRODUCT,  PARENT=DATE_SEG,  SEGTYPE=S1,
  FIELDNAME=PROD_CODE,   ALIAS=PCODE,  FORMAT=A3,    FIELDTYPE=I,$
  FIELDNAME=UNIT_SOLD,   ALIAS=SOLD,   FORMAT=I5,    $
  FIELDNAME=RETAIL_PRICE,ALIAS=RP,     FORMAT=D5.2M,$
  FIELDNAME=DELIVER_AMT, ALIAS=SHIP,   FORMAT=I5,    $
  FIELDNAME=OPENING_AMT, ALIAS=INV,    FORMAT=I5,    $
  FIELDNAME=RETURNS,     ALIAS=RTN,    FORMAT=I3,    MISSING=ON,$
  FIELDNAME=DAMAGED,     ALIAS=BAD,    FORMAT=I3,    MISSING=ON,$
```

## SALES Structure Diagram

```

SECTION 01
      STRUCTURE OF FOCUS      FILE SALES ON 05/15/03 AT 14.50.28

      STOR_SEG
01      S1
*****
*STORE_CODE **
*CITY      **
*AREA      **
*          **
*          **
*****
      I
      I
      I
      I DATE_SEG
02      I SH1
*****
*DATE      **
*          **
*          **
*          **
*          **
*****
      I
      I
      I
      I PRODUCT
03      I S1
*****
*PROD_CODE **I
*UNIT_SOLD **
*RETAIL_PRICE**
*DELIVER_AMT **
*          **
*****
*****

```

## CAR Data Source

CAR contains sample data about specifications and sales information for rare cars. Its segments are:

### ORIGIN

Lists the country that manufactures the car. The field COUNTRY is indexed.

### COMP

Contains the car name.

### CARREC

Contains the car model.

### BODY

Lists the body type, seats, dealer and retail costs, and units sold.

### SPECS

Lists car specifications. This segment is unique.

### WARANT

Lists the type of warranty.

### EQUIP

Lists standard equipment.

The aliases in the CAR Master File are specified without the ALIAS keyword.

## CAR Master File

```

FILENAME=CAR,SUFFIX=FOC
SEGNAME=ORIGIN,SEGTYPE=S1
  FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
  FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
  FIELDNAME=MODEL,MODEL,A24,$
SEGNAME=BODY,SEGTYPE=S1,PARENT=CARREC
  FIELDNAME=BODYTYPE,TYPE,A12,$
  FIELDNAME=SEATS,SEAT,I3,$
  FIELDNAME=DEALER_COST,DCOST,D7,$
  FIELDNAME=RETAIL_COST,RCOST,D7,$
  FIELDNAME=SALES,UNITS,I6,$
SEGNAME=SPECS,SEGTYPE=U,PARENT=BODY
  FIELDNAME=LENGTH,LEN,D5,$
  FIELDNAME=WIDTH,WIDTH,D5,$
  FIELDNAME=HEIGHT,HEIGHT,D5,$
  FIELDNAME=WEIGHT,WEIGHT,D6,$
  FIELDNAME=WHEELBASE,BASE,D6.1,$
  FIELDNAME=FUEL_CAP,FUEL,D6.1,$
  FIELDNAME=BHP,POWER,D6,$
  FIELDNAME=RPM,RPM,I5,$
  FIELDNAME=MPG,MILES,D6,$
  FIELDNAME=ACCEL,SECONDS,D6,$
SEGNAME=WARRANT,SEGTYPE=S1,PARENT=COMP
  FIELDNAME=WARRANTY,WARR,A40,$
SEGNAME=EQUIP,SEGTYPE=S1,PARENT=COMP
  FIELDNAME=STANDARD,EQUIP,A40,$

```

## CAR Structure Diagram

```

SECTION 01          STRUCTURE OF FOCUS      FILE CAR      ON 04/06/07 AT 11.13.56

          ORIGIN
01          S1
*****
* COUNTRY      **I
*              **
*              **
*              **
*****
          I
          I
          I
          I COMP
02          I S1
*****
* CAR          **
*              **
*              **
*              **
*****
          I
          I -----+-----+-----+-----+
          I CARBDC      I WARRNT      I EQUIP
          I S1          I S1          I S1
03          S1          06          07
*****          *****          *****
* MODEL        **      * WARRANTY   **      * STANDARD    **
*              **      *              **      *              **
*              **      *              **      *              **
*              **      *              **      *              **
*****          *****          *****
          I
          I
          I
          I BODY
04          I S1
*****
* BODYTYPE     **
* SEATS        **
* DEALER_COST  **
* RETAIL_COST  **
*              **
*****
          I
          I
          I
          I SPECS
05          I U
*****
* LENGTH       **
* WIDTH        **
* HEIGHT       **
* WEIGHT       **
*              **
*****

```

## LEDGER Data Source

LEDGER contains sample accounting data. It consists of one segment, TOP. This data source is specified primarily for FML examples. Aliases do not exist for the fields in this Master File, and the commas act as placeholders.

### LEDGER Master File

```
FILENAME=LEDGER, SUFFIX=FOC,$
SEGNAME=TOP, SEGTYPE=S2,$
FIELDNAME=YEAR , , FORMAT=A4, $
FIELDNAME=ACCOUNT, , FORMAT=A4, $
FIELDNAME=AMOUNT , , FORMAT=I5C,$
```

## LEDGER Structure Diagram

```
SECTION 01
STRUCTURE OF FOCUS FILE LEDGER ON 05/15/03 AT 15.17.08

TOP
01 S2
*****
*YEAR **
*ACCOUNT **
*AMOUNT **
* **
* **
*****
*****
```

## FINANCE Data Source

FINANCE contains sample financial data for balance sheets. It consists of one segment, TOP. This data source is specified primarily for FML examples. Aliases do not exist for the fields in this Master File, and the commas act as placeholders.

### FINANCE Master File

```
FILENAME=FINANCE, SUFFIX=FOC,$
SEGNAME=TOP, SEGTYPE=S2,$
FIELDNAME=YEAR , , FORMAT=A4, $
FIELDNAME=ACCOUNT, , FORMAT=A4, $
FIELDNAME=AMOUNT , , FORMAT=D12C,$
```

FINANCE Structure Diagram

```
SECTION 01
STRUCTURE OF FOCUS      FILE FINANCE  ON 05/15/03 AT 15.17.08

      TOP
01      S2
*****
*YEAR          **
*ACCOUNT        **
*AMOUNT         **
*              **
*              **
*****
*****
```

REGION Data Source

REGION contains sample account data for the eastern and western regions of the country. It consists of one segment, TOP. This data source is specified primarily for FML examples. Aliases do not exist for the fields in this Master File, and the commas act as placeholders.

REGION Master File

```
FILENAME=REGION, SUFFIX=FOC,$
SEGNAME=TOP,      SEGTYPE=S1,$
  FIELDNAME=ACCOUNT, , FORMAT=A4, $
  FIELDNAME=E_ACTUAL, , FORMAT=I5C,$
  FIELDNAME=E_BUDGET, , FORMAT=I5C,$
  FIELDNAME=W_ACTUAL, , FORMAT=I5C,$
  FIELDNAME=W_BUDGET, , FORMAT=I5C,$
```

REGION Structure Diagram

```
SECTION 01
STRUCTURE OF FOCUS      FILE REGION   ON 05/15/03 AT 15.18.48

      TOP
01      S1
*****
*ACCOUNT      **
*E_ACTUAL     **
*E_BUDGET     **
*W_ACTUAL     **
*            **
*****
*****
```



## EMPDATA Data Source

EMPDATA contains sample data about company employees. It consists of one segment, EMPDATA. The PIN field is indexed. The AREA field is a temporary field.

## EMPDATA Master File

```

FILENAME=EMPDATA, SUFFIX=FOC
SEGNAME=EMPDATA, SEGTYPE=S1
  FIELDNAME=PIN,          ALIAS=ID,          FORMAT=A9,   INDEX=I,   $
  FIELDNAME=LASTNAME,     ALIAS=LN,          FORMAT=A15,   $
  FIELDNAME=FIRSTNAME,    ALIAS=FN,          FORMAT=A10,   $
  FIELDNAME=MIDINITIAL,   ALIAS=MI,          FORMAT=A1,    $
  FIELDNAME=DIV,           ALIAS=CDIV,         FORMAT=A4,    $
  FIELDNAME=DEPT,          ALIAS=CDEPT,        FORMAT=A20,   $
  FIELDNAME=JOBCLASS,      ALIAS=CJCLAS,       FORMAT=A8,    $
  FIELDNAME=TITLE,         ALIAS=CFUNC,        FORMAT=A20,   $
  FIELDNAME=SALARY,        ALIAS=CSAL,         FORMAT=D12.2M, $
  FIELDNAME=HIREDATE,      ALIAS=HDAT,         FORMAT=YMD,   $
$
DEFINE AREA/A13=DECODE DIV (NE 'NORTH EASTERN' SE 'SOUTH EASTERN'
CE 'CENTRAL' WE 'WESTERN' CORP 'CORPORATE' ELSE 'INVALID AREA');$

```

## EMPDATA Structure Diagram

```

SECTION 01
      STRUCTURE OF FOCUS      FILE EMPDATA ON 05/15/03 AT 14.49.09

      EMPDATA
01      S1
*****
*PIN          **I
*LASTNAME     **
*FIRSTNAME    **
*MIDINITIAL   **
*             **
*****
*****

```

## TRAINING Data Source

TRAINING contains sample data about training courses for employees. It consists of one segment, TRAINING. The PIN field is indexed. The EXPENSES, GRADE, and LOCATION fields have the MISSING=ON attribute.

## TRAINING Master File

```

FILENAME=TRAINING, SUFFIX=FOC
SEGNAME=TRAINING, SEGTYPE=SH3
  FIELDNAME=PIN,          ALIAS=ID,          FORMAT=A9,      INDEX=I,      $
  FIELDNAME=COURSESTART,  ALIAS=CSTART,  FORMAT=YMD,      $
  FIELDNAME=COURSECODE,   ALIAS=CCOD,    FORMAT=A7,      $
  FIELDNAME=EXPENSES,     ALIAS=COST,    FORMAT=D8.2,    MISSING=ON,   $
  FIELDNAME=GRADE,        ALIAS=GRA,     FORMAT=A2,      MISSING=ON,   $
  FIELDNAME=LOCATION,       ALIAS=LOC,     FORMAT=A6,      MISSING=ON,   $

```

## TRAINING Structure Diagram

```

SECTION 01
      STRUCTURE OF FOCUS      FILE TRAINING ON 05/15/03 AT 14.51.28

      TRAINING
01      SH3
*****
*PIN          **I
*COURSESTART **
*COURSECODE   **
*EXPENSES     **
*             **
*****
*****

```

## COURSE Data Source

COURSE contains sample data about education courses. It consists of one segment, CRSELIST.

## COURSE Master File

```

FILENAME=COURSE, SUFFIX=FOC
SEGNAME=CRSELIST, SEGTYPE=S1
  FIELDNAME=COURSECODE, ALIAS=CCOD, FORMAT=A7, INDEX=I, $
  FIELDNAME=CTITLE,     ALIAS=COURSE, FORMAT=A35, $
  FIELDNAME=SOURCE,     ALIAS=ORG,   FORMAT=A35, $
  FIELDNAME=CLASSIF,    ALIAS=CLASS, FORMAT=A10, $
  FIELDNAME=TUITION,    ALIAS=FEE,   FORMAT=D8.2, MISSING=ON, $
  FIELDNAME=DURATION,   ALIAS=DAYS,  FORMAT=A3,   MISSING=ON, $
  FIELDNAME=DESCRIPTN1, ALIAS=DESC1, FORMAT=A40, $
  FIELDNAME=DESCRIPTN2, ALIAS=DESC2, FORMAT=A40, $
  FIELDNAME=DESCRIPTN2, ALIAS=DESC3, FORMAT=A40, $

```

## COURSE Structure Diagram

```
SECTION 01
      STRUCTURE OF FOCUS      FILE COURSE      ON 05/15/03 AT 12.26.05

      CRSELIST
01      S1
*****
*COURSECODE      **I
*CTITLE          **
*SOURCE          **
*CLASSIF         **
*                **
*****
*****
```

## JOBHIST Data Source

JOBHIST contains information about employee jobs. Both the PIN and JOBSTART fields are keys. The PIN field is indexed.

## JOBHIST Master File

```
FILENAME=JOBHIST, SUFFIX=FOC
SEGNAME=JOBHIST, SEGTYPE=SH2
FIELDNAME=PIN,      ALIAS=ID,      FORMAT=A9,      INDEX=I , $
FIELDNAME=JOBSTART, ALIAS=SDAT,    FORMAT=YMD,      $
FIELDNAME=JOBCLASS, ALIAS=JCLASS,  FORMAT=A8,      $
FIELDNAME=FUNCTITLE, ALIAS=FUNC,    FORMAT=A20,      $
```

## JOBHIST Structure Diagram

```
SECTION 01
      STRUCTURE OF FOCUS      FILE JOBHIST      ON 01/22/08 AT 16.23.46
      JOBHIST
01      SH2
*****
*PIN      **I
*JOBSTART **
*JOBCLASS **
*FUNCTITLE **
*         **
*****
*****
```

## JOBLIST Data Source

JOBLIST contains information about jobs. The JOBCLASS field is indexed.

JOBLIST Master File

```
FILENAME=JOBLIST, SUFFIX=FOC
SEGMNAME=JOBSEG, SEGTYPE=S1
FIELDNAME=JOBCLASS, ALIAS=JCLASS, FORMAT=A8, INDEX=I, $
FIELDNAME=CATEGORY, ALIAS=JGROUP, FORMAT=A25, $
FIELDNAME=JOBDESC, ALIAS=JDESC, FORMAT=A40, $
FIELDNAME=LOWSAL, ALIAS=LSAL, FORMAT=D12.2M, $
FIELDNAME=HIGHSAL, ALIAS=HSAL, FORMAT=D12.2M, $
DEFINE GRADE/A2=EDIT (JCLASS,'$$$99');$
DEFINE LEVEL/A25=DECODE GRADE (08 'GRADE 8' 09 'GRADE 9' 10
'GRADE 10' 11 'GRADE 11' 12 'GRADE 12' 13 'GRADE 13' 14 'GRADE 14');$
```

JOBLIST Structure Diagram

```
SECTION 01
STRUCTURE OF FOCUS FILE JOBLIST ON 01/22/08 AT 16.24.52
JOBSEG
01 S1
*****
*JOBCLASS **I
*CATEGORY **
*JOBDESC **
*LOWSAL **
* **
*****
*****
```

LOCATOR Data Source

JOBHIST contains information about employee location and phone number. The PIN field is indexed.

LOCATOR Master File

```
FILENAME=LOCATOR, SUFFIX=FOC
SEGMNAME=LOCATOR, SEGTYPE=S1,
FIELDNAME=PIN, ALIAS=ID_NO, FORMAT=A9, INDEX=I, $
FIELDNAME=SITE, ALIAS=SITE, FORMAT=A25, $
FIELDNAME=FLOOR, ALIAS=FL, FORMAT=A3, $
FIELDNAME=ZONE, ALIAS=ZONE, FORMAT=A2, $
FIELDNAME=BUS_PHONE, ALIAS=BTCL, FORMAT=A5, $
```

## LOCATOR Structure Diagram

```

SECTION 01
      STRUCTURE OF FOCUS      FILE LOCATOR  ON 01/22/08 AT 16.26.55
      LOCATOR
01      S1
*****
*PIN          **I
*SITE         **
*FLOOR        **
*ZONE         **
*             **
*****
*****

```

## PERSINFO Data Source

PERSINFO contains employee personal information. The PIN field is indexed.

## PERSINFO Master File

```

FILENAME=PERSINFO, SUFFIX=FOC
SEGNAME=PERSONAL, SEGTYPE=S1
FIELDNAME=PIN,          ALIAS=ID,          FORMAT=A9,      INDEX=I,      $
FIELDNAME=INCAREOF,     ALIAS=ICO,          FORMAT=A35,      $
FIELDNAME=STREETNO,     ALIAS=STR,          FORMAT=A20,      $
FIELDNAME=APT,          ALIAS=APT,          FORMAT=A4,        $
FIELDNAME=CITY,         ALIAS=CITY,         FORMAT=A20,      $
FIELDNAME=STATE,        ALIAS=PROV,         FORMAT=A4,        $
FIELDNAME=POSTALCODE,   ALIAS=ZIP,          FORMAT=A10,      $
FIELDNAME=COUNTRY,     ALIAS=CTRY,         FORMAT=A15,      $
FIELDNAME=HOMEPHONE,    ALIAS=TEL,          FORMAT=A10,      $
FIELDNAME=EMERGENCYNO,  ALIAS=ENO,          FORMAT=A10,      $
FIELDNAME=EMERGCONTACT, ALIAS=ENAME,        FORMAT=A35,      $
FIELDNAME=RELATIONSHIP, ALIAS=REL,          FORMAT=A8,        $
FIELDNAME=BIRTHDATE,    ALIAS=BDAT,         FORMAT=YMD,      $

```

## PERSINFO Structure Diagram

```

SECTION 01
      STRUCTURE OF FOCUS      FILE PERSINFO ON 01/22/08 AT 16.27.24
      PERSONAL
01      S1
*****
*PIN          **I
*INCAREOF     **
*STREETNO     **
*APT          **
*             **
*****
*****

```

## SALHIST Data Source

SALHIST contains information about employee salary history. The PIN field is indexed. Both the PIN and EFFECTDATE fields are keys.

### SALHIST Master File

```
FILENAME=SALHIST,  SUFFIX=FOC
SEGNAME=SLHISTORY, SEGTYPE=SH2
  FIELDNAME=PIN,      ALIAS=ID,      FORMAT=A9,      INDEX=I,      $
  FIELDNAME=EFFECTDATE, ALIAS=EDAT,   FORMAT=YMD,      $
  FIELDNAME=OLDSALARY, ALIAS=OSAL,   FORMAT=D12.2,    $
```

### SALHIST Structure Diagram

```
SECTION 01
  STRUCTURE OF FOCUS      FILE SALHIST  ON 01/22/08 AT 16.28.02
  SLHISTORY
    01      SH2
  *****
  *PIN      **I
  *EFFECTDATE **
  *OLDSALARY **
  *          **
  *          **
  *****
  *****
```

## VIDEOTRK, MOVIES, and ITEMS Data Sources

VIDEOTRK contains sample data about customer, rental, and purchase information for a video rental business. It can be joined to the MOVIES or ITEMS data source. VIDEOTRK and MOVIES are used in examples that illustrate the use of the Maintain Data facility.

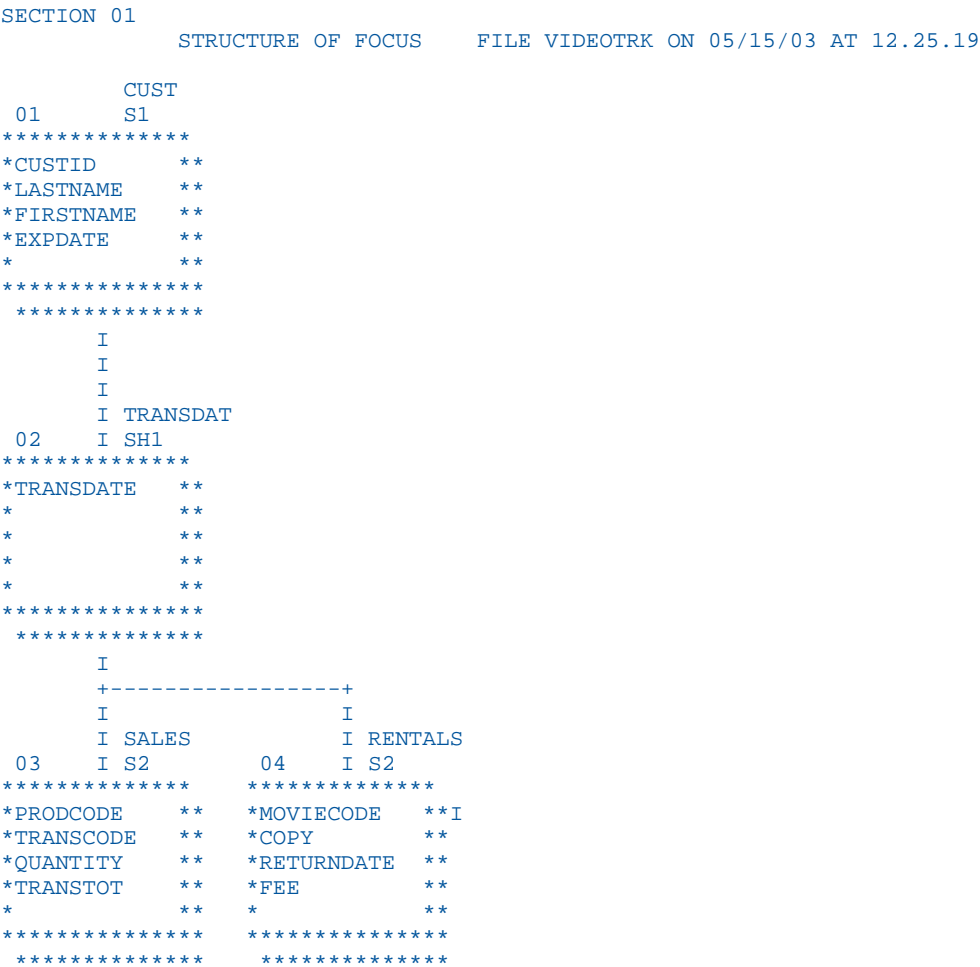
## VIDEOTRK Master File

```

FILENAME=VIDEOTRK, SUFFIX=FOC
SEGNAME=CUST, SEGTYPE=S1
  FIELDNAME=CUSTID, ALIAS=CIN, FORMAT=A4, $
  FIELDNAME=LASTNAME, ALIAS=LN, FORMAT=A15, $
  FIELDNAME=FIRSTNAME, ALIAS=FN, FORMAT=A10, $
  FIELDNAME=EXPDATE, ALIAS=EXDAT, FORMAT=YMD, $
  FIELDNAME=PHONE, ALIAS=TEL, FORMAT=A10, $
  FIELDNAME=STREET, ALIAS=STR, FORMAT=A20, $
  FIELDNAME=CITY, ALIAS=CITY, FORMAT=A20, $
  FIELDNAME=STATE, ALIAS=PROV, FORMAT=A4, $
  FIELDNAME=ZIP, ALIAS=POSTAL_CODE, FORMAT=A9, $
SEGNAME=TRANSDAT, SEGTYPE=SH1, PARENT=CUST
  FIELDNAME=TRANSDATE, ALIAS=OUTDATE, FORMAT=YMD, $
SEGNAME=SALES, SEGTYPE=S2, PARENT=TRANSDAT
  FIELDNAME=PRODCODE, ALIAS=PCOD, FORMAT=A6, $
  FIELDNAME=TRANSCODE, ALIAS=TCOD, FORMAT=I3, $
  FIELDNAME=QUANTITY, ALIAS=NO, FORMAT=I3S, $
  FIELDNAME=TRANSTOT, ALIAS=TTOT, FORMAT=F7.2S, $
SEGNAME=RENTALS, SEGTYPE=S2, PARENT=TRANSDAT
  FIELDNAME=MOVIECODE, ALIAS=MCOD, FORMAT=A6, INDEX=I, $
  FIELDNAME=COPY, ALIAS=COPY, FORMAT=I2, $
  FIELDNAME=RETURNDATE, ALIAS=INDATE, FORMAT=YMD, $
  FIELDNAME=FEE, ALIAS=FEE, FORMAT=F5.2S, $

```

VIDEOTRK Structure Diagram





## MOVIES Master File

```

FILENAME=MOVIES,      SUFFIX=FOC
SEGNAME=MOVINFO,     SEGTYPE=S1
  FIELDNAME=MOVIECODE,  ALIAS=MCOD,   FORMAT=A6,  INDEX=I,  $
  FIELDNAME=TITLE,     ALIAS=MTL,    FORMAT=A39,  $
  FIELDNAME=CATEGORY,  ALIAS=CLASS,  FORMAT=A8,   $
  FIELDNAME=DIRECTOR,  ALIAS=DIR,    FORMAT=A17,  $
  FIELDNAME=RATING,    ALIAS=RTG,    FORMAT=A4,   $
  FIELDNAME=RELDATE,   ALIAS=RDAT,   FORMAT=YMD,  $
  FIELDNAME=WHOLESALEPR, ALIAS=WPRC,  FORMAT=F6.2, $
  FIELDNAME=LISTPR,    ALIAS=LPRC,   FORMAT=F6.2, $
  FIELDNAME=COPIES,    ALIAS=NOC,    FORMAT=I3,   $

```

## MOVIES Structure Diagram

```

SECTION 01
          STRUCTURE OF FOCUS      FILE MOVIES      ON 05/15/03 AT 12.26.05

          MOVINFO
01         S1
*****
*MOVIECODE  **I
*TITLE      **
*CATEGORY   **
*DIRECTOR   **
*           **
*****
*****

```

## ITEMS Master File

```

FILENAME=ITEMS,      SUFFIX=FOC
SEGNAME=ITMINFO,     SEGTYPE=S1
  FIELDNAME=PRODCODE,  ALIAS=PCOD,   FORMAT=A6,  INDEX=I,  $
  FIELDNAME=PRODNAME,  ALIAS=PROD,   FORMAT=A20,  $
  FIELDNAME=OURCOST,   ALIAS=WCost,   FORMAT=F6.2,  $
  FIELDNAME=RETAILPR,  ALIAS=PRICE,   FORMAT=F6.2,  $
  FIELDNAME=ON_HAND,   ALIAS=NUM,     FORMAT=I5,   $

```

ITEMS Structure Diagram

```
SECTION 01
STRUCTURE OF FOCUS      FILE ITEMS      ON 05/15/03 AT 12.26.05

      ITMINFO
01      S1
*****
*PRODCODE      **I
*PRODNAME      **
*OURCOST      **
*RETAILPR      **
*              **
*****
*****
```

VIDEOTR2 Data Source

VIDEOTR2 contains sample data about customer, rental, and purchase information for a video rental business. It consists of four segments.

VIDEOTR2 Master File

```
FILENAME=VIDEOTR2, SUFFIX=FOC
SEGNAME=CUST, SEGTYPE=S1
FIELDNAME=CUSTID, ALIAS=CIN, FORMAT=A4, $
FIELDNAME=LASTNAME, ALIAS=LN, FORMAT=A15, $
FIELDNAME=FIRSTNAME, ALIAS=FN, FORMAT=A10, $
FIELDNAME=EXPDATE, ALIAS=EXDAT, FORMAT=YMD, $
FIELDNAME=PHONE, ALIAS=TEL, FORMAT=A10, $
FIELDNAME=STREET, ALIAS=STR, FORMAT=A20, $
FIELDNAME=CITY, ALIAS=CITY, FORMAT=A20, $
FIELDNAME=STATE, ALIAS=PROV, FORMAT=A4, $
FIELDNAME=ZIP, ALIAS=POSTAL_CODE, FORMAT=A9, $
FIELDNAME=EMAIL, ALIAS=EMAIL, FORMAT=A18, $
SEGNAME=TRANSDAT, SEGTYPE=SH1, PARENT=CUST
FIELDNAME=TRANSDATE, ALIAS=OUTDATE, FORMAT=HYMDI, $
SEGNAME=SALES, SEGTYPE=S2, PARENT=TRANSDAT
FIELDNAME=TRANSCODE, ALIAS=TCOD, FORMAT=I3, $
FIELDNAME=QUANTITY, ALIAS=NO, FORMAT=I3S, $
FIELDNAME=TRANSTOT, ALIAS=TTOT, FORMAT=F7.2S, $
SEGNAME=RENTALS, SEGTYPE=S2, PARENT=TRANSDAT
FIELDNAME=MOVIECODE, ALIAS=MCOD, FORMAT=A6, INDEX=I, $
FIELDNAME=COPY, ALIAS=COPY, FORMAT=I2, $
FIELDNAME=RETURNDATE, ALIAS=INDATE, FORMAT=YMD, $
FIELDNAME=FEE, ALIAS=FEE, FORMAT=F5.2S, $
```

## VIDEOTR2 Structure Diagram

```

SECTION 01
  STRUCTURE OF FOCUS      FILE VIDEOTR2 ON 05/15/03 AT 16.45.48

      CUST
01      S1
*****
*CUSTID      **
*LASTNAME    **
*FIRSTNAME   **
*EXPDATE     **
*            **
*****
      I
      I
      I
      I TRANSDAT
02      I SH1
*****
*TRANSDATE   **
*            **
*            **
*            **
*            **
*****
      I
      +-----+
      I                      I
      I SALES                I RENTALS
03      I S2                04      I S2
*****                    *****
*TRANSCODE    **      *MOVIECODE    **I
*QUANTITY     **      *COPY          **
*TRANSTOT     **      *RETURNDATE   **
*            **      *FEE           **
*            **      *            **
*****                    *****
*****                    *****

```

## Gotham Grinds Data Sources

Gotham Grinds is a group of data sources that contain sample data about a specialty items company.

- ☐ GGDEMOG contains demographic information about the customers of Gotham Grinds, a company that sells specialty items like coffee, gourmet snacks, and gifts. It consists of one segment, DEMOG01.
- ☐ GGORDER contains order information for Gotham Grinds. It consists of two segments, ORDER01 and ORDER02.

- ❑ GGPRODS contains product information for Gotham Grinds. It consists of one segment, PRODS01.
- ❑ GGSales contains sales information for Gotham Grinds. It consists of one segment, SALES01.
- ❑ GGSTORES contains information for each of Gotham Grinds 12 stores in the United States. It consists of one segment, STORES01.

## GGDEMOG Master File

```
FILENAME=GGDEMOG, SUFFIX=FOC
SEGNAME=DEMOG01, SEGTYPE=S1
  FIELD=ST,          ALIAS=E02, FORMAT=A02, INDEX=I, TITLE='State',
    DESC='State', $
  FIELD=HH,          ALIAS=E03, FORMAT=I09, TITLE='Number of Households',
    DESC='Number of Households', $
  FIELD=AVGHHSZ98, ALIAS=E04, FORMAT=I09, TITLE='Average Household Size',
    DESC='Average Household Size', $
  FIELD=MEDHHI98, ALIAS=E05, FORMAT=I09, TITLE='Median Household Income',
    DESC='Median Household Income', $
  FIELD=AVGHHI98, ALIAS=E06, FORMAT=I09, TITLE='Average Household Income',
    DESC='Average Household Income', $
  FIELD=MALEPOP98, ALIAS=E07, FORMAT=I09, TITLE='Male Population',
    DESC='Male Population', $
  FIELD=FEMPOP98, ALIAS=E08, FORMAT=I09, TITLE='Female Population',
    DESC='Female Population', $
  FIELD=P15TO1998, ALIAS=E09, FORMAT=I09, TITLE='15 to 19',
    DESC='Population 15 to 19 years old', $
  FIELD=P20TO2998, ALIAS=E10, FORMAT=I09, TITLE='20 to 29',
    DESC='Population 20 to 29 years old', $
  FIELD=P30TO4998, ALIAS=E11, FORMAT=I09, TITLE='30 to 49',
    DESC='Population 30 to 49 years old', $
  FIELD=P50TO6498, ALIAS=E12, FORMAT=I09, TITLE='50 to 64',
    DESC='Population 50 to 64 years old', $
  FIELD=P65OVR98, ALIAS=E13, FORMAT=I09, TITLE='65 and over',
    DESC='Population 65 and over', $
```

## GGDEMOG Structure Diagram

```

SECTION 01
  STRUCTURE OF FOCUS      FILE GGDEMOG   ON 05/15/03 AT 12.26.05

      GGDEMOG
01      S1
*****
*ST          **I
*HH          **
*AVGHHSZ98   **
*MEDDHI98    **
*            **
*****
*****

```

## GGORDER Master File

```

FILENAME=GGORDER, SUFFIX=FOC,$
SEGNAME=ORDER01, SEGTYPE=S1,$
  FIELD=ORDER_NUMBER, ALIAS=ORDN01,  FORMAT=I6,  TITLE='Order,Number',
  DESC='Order Identification Number', $
  FIELD=ORDER_DATE,   ALIAS=DATE,    FORMAT=MDY, TITLE='Order,Date',
  DESC='Date order was placed', $
  FIELD=STORE_CODE,   ALIAS=STCD,    FORMAT=A5,  TITLE='Store,Code',
  DESC='Store Identification Code (for order)', $
  FIELD=PRODUCT_CODE, ALIAS=PCD,     FORMAT=A4,  TITLE='Product,Code',
  DESC='Product Identification Code (for order)', $
  FIELD=QUANTITY,     ALIAS=ORDUNITS, FORMAT=I8,  TITLE='Ordered,Units',
  DESC='Quantity Ordered', $
SEGNAME=ORDER02, SEGTYPE=KU, PARENT=ORDER01, CRFILE=GGPRODS, CRKEY=PCD,
CRSEG=PRODS01  , $

```

## GGORDER Structure Diagram

```

SECTION 01
  STRUCTURE OF FOCUS      FILE GGORDER  ON 05/15/03 AT 16.45.48

      GGORDER
01      S1
*****
*ORDER_NUMBER**
*ORDER_DATE   **
*STORE_CODE   **
*PRODUCT_CODE**
*              **
*****
*****
      I
      I
      I
      I ORDER02
02      I KU
.....
:PRODUCT_ID   :K
:PRODUCT_DESC:
:VENDOR_CODE  :
:VENDOR_NAME  :
:              :
:.....:

```

## GGPRODS Master File

```

FILENAME=GGPRODS, SUFFIX=FOC
SEGNAME=PRODS01, SEGTYPE=S1
  FIELD=PRODUCT_ID, ALIAS=PCD, FORMAT=A4, INDEX=I, TITLE='Product,Code',
    DESC='Product Identification Code',$
  FIELD=PRODUCT_DESCRIPTION, ALIAS=PRODUCT, FORMAT=A16, TITLE='Product',
    DESC='Product Name',$
  FIELD=VENDOR_CODE, ALIAS=VCD, FORMAT=A4, INDEX=I, TITLE='Vendor ID',
    DESC='Vendor Identification Code',$
  FIELD=VENDOR_NAME, ALIAS=VENDOR, FORMAT=A23, TITLE='Vendor Name',
    DESC='Vendor Name',$
  FIELD=PACKAGE_TYPE, ALIAS=PACK, FORMAT=A7, TITLE='Package',
    DESC='Packaging Style',$
  FIELD=SIZE, ALIAS=SZ, FORMAT=I2, TITLE='Size',
    DESC='Package Size',$
  FIELD=UNIT_PRICE, ALIAS=UNITPR, FORMAT=D7.2, TITLE='Unit,Price',
    DESC='Price for one unit',$

```

## GGPRODS Structure Diagram

```
SECTION 01
  STRUCTURE OF FOCUS      FILE GGPRODS   ON 05/15/03 AT 12.26.05

      GGPRODS
01      S1
*****
*PRODUCT_ID  **I
*PRODUCT_DESC**I
*VENDOR_CODE **
*VENDOR_NAME **
*              **
*****
*****
```

## GGSALES Master File

```
FILENAME=GGSALES, SUFFIX=FOC
SEGNAME=SALES01, SEGTYPE=S1
  FIELD=SEQ_NO, ALIAS=SEQ, FORMAT=I5, TITLE='Sequence#',
  DESC='Sequence number in database',$
  FIELD=CATEGORY, ALIAS=E02, FORMAT=A11, INDEX=I, TITLE='Category',
  DESC='Product category',$
  FIELD=PCD, ALIAS=E03, FORMAT=A04, INDEX=I, TITLE='Product ID',
  DESC='Product Identification code (for sale)',$
  FIELD=PRODUCT, ALIAS=E04, FORMAT=A16, TITLE='Product',
  DESC='Product name',$
  FIELD=REGION, ALIAS=E05, FORMAT=A11, INDEX=I, TITLE='Region',
  DESC='Region code',$
  FIELD=ST, ALIAS=E06, FORMAT=A02, INDEX=I, TITLE='State',
  DESC='State',$
  FIELD=CITY, ALIAS=E07, FORMAT=A20, TITLE='City',
  DESC='City',$
  FIELD=STCD, ALIAS=E08, FORMAT=A05, INDEX=I, TITLE='Store ID',
  DESC='Store identification code (for sale)',$
  FIELD=DATE, ALIAS=E09, FORMAT=I8YYMD, TITLE='Date',
  DESC='Date of sales report',$
  FIELD=UNITS, ALIAS=E10, FORMAT=I08, TITLE='Unit Sales',
  DESC='Number of units sold',$
  FIELD=DOLLARS, ALIAS=E11, FORMAT=I08, TITLE='Dollar Sales',
  DESC='Total dollar amount of reported sales',$
  FIELD=BUDUNITS, ALIAS=E12, FORMAT=I08, TITLE='Budget Units',
  DESC='Number of units budgeted',$
  FIELD=BUDDOLLARS, ALIAS=E13, FORMAT=I08, TITLE='Budget Dollars',
  DESC='Total sales quota in dollars',$
```

## GGSALES Structure Diagram

```
SECTION 01
  STRUCTURE OF FOCUS      FILE GGSALES  ON 05/15/03 AT 12.26.05

      GGSALES
01      S1
*****
*SEQ_NO      **
*CATEGORY    **I
*PCD         **I
*PRODUCT     **I
*            **
*****
*****
```

## GGSTORES Master File

```
FILENAME=GGSTORES, SUFFIX=FOC
SEGNAME=STORES01, SEGTYPE=S1
  FIELD=STORE_CODE, ALIAS=E02, FORMAT=A05, INDEX=I, TITLE='Store ID',
  DESC='Franchisee ID Code',$
  FIELD=STORE_NAME, ALIAS=E03, FORMAT=A23, TITLE='Store Name',
  DESC='Store Name',$
  FIELD=ADDRESS1, ALIAS=E04, FORMAT=A19, TITLE='Contact',
  DESC='Franchisee Owner',$
  FIELD=ADDRESS2, ALIAS=E05, FORMAT=A31, TITLE='Address',
  DESC='Street Address',$
  FIELD=CITY, ALIAS=E06, FORMAT=A22, TITLE='City',
  DESC='City',$
  FIELD=STATE, ALIAS=E07, FORMAT=A02, INDEX=I, TITLE='State',
  DESC='State',$
  FIELD=ZIP, ALIAS=E08, FORMAT=A06, TITLE='Zip Code',
  DESC='Postal Code',$
```

## GGSTORES Structure Diagram

```
SECTION 01
  STRUCTURE OF FOCUS      FILE GGSTORES ON 05/15/03 AT 12.26.05

      GGSTORES
01      S1
*****
*STORE_CODE  **I
*STORE_NAME  **
*ADDRESS1    **
*ADDRESS2    **
*            **
*****
*****
```



## Century Corp Data Sources

Century Corp is a consumer electronics manufacturer that distributes products through retailers around the world. Century Corp has thousands of employees in plants, warehouses, and offices worldwide. Their mission is to provide quality products and services to their customers.

Century Corp is a group of data sources that contain financial, human resources, inventory, and order information. The last three data sources are designed to be used with chart of accounts data.

- ☐ CENTCOMP Master File contains location information for stores. It consists of one segment, COMPINFO.
- ☐ CENTFIN Master File contains financial information. It consists of one segment, ROOT\_SEG.
- ☐ CENTHR Master File contains human resources information. It consists of one segment, EMPSEG.
- ☐ CENTINV Master File contains inventory information. It consists of one segment, INVINFO.
- ☐ CENTORD Master File contains order information. It consists of four segments, OINFO, STOSEG, PINFO, and INVSEG.
- ☐ CENTQA Master File contains problem information. It consists of three segments, PROD\_SEG, INVSEG, and PROB\_SEG.
- ☐ CENTGL Master File contains a chart of accounts hierarchy. The field GL\_ACCOUNT\_PARENT is the parent field in the hierarchy. The field GL\_ACCOUNT is the hierarchy field. The field GL\_ACCOUNT\_CAPTION can be used as the descriptive caption for the hierarchy field.
- ☐ CENTSYSF Master File contains detail-level financial data. CENTSYSF uses a different account line system (SYS\_ACCOUNT), which can be joined to the SYS\_ACCOUNT field in CENTGL. Data uses "natural" signs (expenses are positive, revenue negative).
- ☐ CENTSTMT Master File contains detail-level financial data and a cross-reference to the CENTGL data source.
- ☐ CENTGLL Master File contains a chart of accounts hierarchy. The field GL\_ACCOUNT\_PARENT is the parent field in the hierarchy. The field GL\_ACCOUNT is the hierarchy field. The field GL\_ACCOUNT\_CAPTION can be used as the descriptive caption for the hierarchy field.

CENTCOMP Master File

```
FILE=CENTCOMP, SUFFIX=FOC, FDFC=19, FYRT=00
  SEGNAME=COMPINFO, SEGTYPE=S1, $
  FIELD=STORE_CODE, ALIAS=SNUM, FORMAT=A6, INDEX=I,
    TITLE='Store Id#:',
    DESCRIPTION='Store Id#', $
  FIELD=STORENAME, ALIAS=SNAME, FORMAT=A20,
    WITHIN=STATE,
    TITLE='Store,Name:',
    DESCRIPTION='Store Name', $
  FIELD=STATE, ALIAS=STATE, FORMAT=A2,
    WITHIN=PLANT,
    TITLE='State:',
    DESCRIPTION=State, $
  DEFINE REGION/A5=DECODE STATE ('AL' 'SOUTH' 'AK' 'WEST' 'AR' 'SOUTH'
    'AZ' 'WEST' 'CA' 'WEST' 'CO' 'WEST' 'CT' 'EAST'
    'DE' 'EAST' 'DC' 'EAST' 'FL' 'SOUTH' 'GA' 'SOUTH' 'HI' 'WEST'
    'ID' 'WEST' 'IL' 'NORTH' 'IN' 'NORTH' 'IA' 'NORTH'
    'KS' 'NORTH' 'KY' 'SOUTH' 'LA' 'SOUTH' 'ME' 'EAST' 'MD' 'EAST'
    'MA' 'EAST' 'MI' 'NORTH' 'MN' 'NORTH' 'MS' 'SOUTH' 'MT' 'WEST'
    'MO' 'SOUTH' 'NE' 'WEST' 'NV' 'WEST' 'NH' 'EAST' 'NJ' 'EAST'
    'NM' 'WEST' 'NY' 'EAST' 'NC' 'SOUTH' 'ND' 'NORTH' 'OH' 'NORTH'
    'OK' 'SOUTH' 'OR' 'WEST' 'PA' 'EAST' 'RI' 'EAST' 'SC' 'SOUTH'
    'SD' 'NORTH' 'TN' 'SOUTH' 'TX' 'SOUTH' 'UT' 'WEST' 'VT' 'EAST'
    'VA' 'SOUTH' 'WA' 'WEST' 'WV' 'SOUTH' 'WI' 'NORTH' 'WY' 'WEST'
    'NA' 'NORTH' 'ON' 'NORTH' ELSE ' ');
  TITLE='Region:',
  DESCRIPTION=Region, $
```

CENTCOMP Structure Diagram

```
SECTION 01
  STRUCTURE OF FOCUS      FILE CENTCOMP ON 05/15/03 AT 10.20.49

      COMPINFO
01      S1
*****
*STORE_CODE  **I
*STORENAME   **
*STATE       **
*            **
*            **
*****
*****
```

## CENTFIN Master File

```

FILE=CENTFIN, SUFFIX=FOC, FDFC=19, FYRT=00
  SEGNAME=ROOT_SEG, SEGTYPE=S4, $
  FIELD=YEAR, ALIAS=YEAR, FORMAT=YY,
    WITHIN='*Time Period', $
  FIELD=QUARTER, ALIAS=QTR, FORMAT=Q,
    WITHIN=YEAR,
    TITLE=Quarter,
    DESCRIPTION=Quarter, $
  FIELD=MONTH, ALIAS=MONTH, FORMAT=M,
    TITLE=Month,
    DESCRIPTION=Month, $
  FIELD=ITEM, ALIAS=ITEM, FORMAT=A20,
    TITLE=Item,
    DESCRIPTION=Item, $
  FIELD=VALUE, ALIAS=VALUE, FORMAT=D12.2,
    TITLE=Value,
    DESCRIPTION=Value, $
  DEFINE ITYPE/A12=IF EDIT(ITEM,'9$$$$$$$$$$$$$$$$') EQ 'E'
    THEN 'Expense' ELSE IF EDIT(ITEM,'9$$$$$$$$$$$$$$$$') EQ 'R'
    THEN 'Revenue' ELSE 'Asset';,
    TITLE=Type,
    DESCRIPTION='Type of Financial Line Item', $
  DEFINE MOTEXT/MT=MONTH;,$

```

## CENTFIN Structure Diagram

```

SECTION 01
  STRUCTURE OF FOCUS      FILE CENTFIN  ON 05/15/03 AT 10.25.52

      ROOT_SEG
01      S4
*****
*YEAR          **
*QUARTER        **
*MONTH          **
*ITEM          **
*              **
*****
*****

```

## CENTHR Master File

```

FILE=CENTHR, SUFFIX=FOC
  SEGNAME=EMPSEG, SEGTYPE=S1, $
  FIELD=ID_NUM, ALIAS=ID#, FORMAT=I9,
    TITLE='Employee, ID#',
    DESCRIPTION='Employee Identification Number', $
  FIELD=LNAME, ALIAS=LN, FORMAT=A14,
    TITLE='Last, Name',
    DESCRIPTION='Employee Last Name', $
  FIELD=FNAME, ALIAS=FN, FORMAT=A12,
    TITLE='First, Name',
    DESCRIPTION='Employee First Name', $
  FIELD=PLANT, ALIAS=PLT, FORMAT=A3,
    TITLE='Plant, Location',
    DESCRIPTION='Location of the manufacturing plant',
    WITHIN='*Location', $
  FIELD=START_DATE, ALIAS=SDATE, FORMAT=YYMD,
    TITLE='Starting, Date',
    DESCRIPTION='Date of employment', $
  FIELD=TERM_DATE, ALIAS=TERM_DATE, FORMAT=YYMD,
    TITLE='Termination, Date',
    DESCRIPTION='Termination Date', $
  FIELD=STATUS, ALIAS=STATUS, FORMAT=A10,
    TITLE='Current, Status',
    DESCRIPTION='Job Status', $
  FIELD=POSITION, ALIAS=JOB, FORMAT=A2,
    TITLE=Position,
    DESCRIPTION='Job Position', $
  FIELD=PAYSCALE, ALIAS=PAYLEVEL, FORMAT=I2,
    TITLE='Pay, Level',
    DESCRIPTION='Pay Level',
    WITHIN='*Wages', $
  DEFINE POSITION_DESC/A17=IF POSITION EQ 'BM' THEN
    'Plant Manager' ELSE
    IF POSITION EQ 'MR' THEN 'Line Worker' ELSE
    IF POSITION EQ 'TM' THEN 'Line Manager' ELSE
    'Technician';
    TITLE='Position, Description',
    DESCRIPTION='Position Description',
    WITHIN='PLANT', $
  DEFINE BYEAR/YY=START_DATE;
    TITLE='Beginning, Year',
    DESCRIPTION='Beginning Year',
    WITHIN='*Starting Time Period', $

```

```

DEFINE BQUARTER/Q=START_DATE;
  TITLE='Beginning,Quarter',
  DESCRIPTION='Beginning Quarter',
  WITHIN='BYEAR',
DEFINE BMONTH/M=START_DATE;
  TITLE='Beginning,Month',
  DESCRIPTION='Beginning Month',
  WITHIN='BQUARTER', $
DEFINE EYEAR/YY=TERM_DATE;
  TITLE='Ending,Year',
  DESCRIPTION='Ending Year',
  WITHIN='*Termination Time Period', $
DEFINE EQUARTER/Q=TERM_DATE;
  TITLE='Ending,Quarter',
  DESCRIPTION='Ending Quarter',
  WITHIN='EYEAR', $
DEFINE EMONTH/M=TERM_DATE;
  TITLE='Ending,Month',
  DESCRIPTION='Ending Month',
  WITHIN='EQUARTER', $
DEFINE RESIGN_COUNT/I3=IF STATUS EQ 'RESIGNED' THEN 1
  ELSE 0;
  TITLE='Resigned,Count',
  DESCRIPTION='Resigned Count', $
DEFINE FIRE_COUNT/I3=IF STATUS EQ 'TERMINAT' THEN 1
  ELSE 0;
  TITLE='Terminated,Count',
  DESCRIPTION='Terminated Count', $
DEFINE DECLINE_COUNT/I3=IF STATUS EQ 'DECLINED' THEN 1
  ELSE 0;
  TITLE='Declined,Count',
  DESCRIPTION='Declined Count', $
DEFINE EMP_COUNT/I3=IF STATUS EQ 'EMPLOYED' THEN 1
  ELSE 0;
  TITLE='Employed,Count',
  DESCRIPTION='Employed Count', $
DEFINE PEND_COUNT/I3=IF STATUS EQ 'PENDING' THEN 1
  ELSE 0;
  TITLE='Pending,Count',
  DESCRIPTION='Pending Count', $
DEFINE REJECT_COUNT/I3=IF STATUS EQ 'REJECTED' THEN 1
  ELSE 0;
  TITLE='Rejected,Count',
  DESCRIPTION='Rejected Count', $
DEFINE FULLNAME/A28=LNAME|'|','|FNAME;
  TITLE='Full Name',
  DESCRIPTION='Full Name: Last, First', WITHIN='POSITION_DESC', $

```

```
DEFINE SALARY/D12.2=IF BMONTH LT 4 THEN PAYLEVEL * 12321
ELSE IF BMONTH GE 4 AND BMONTH LT 8 THEN PAYLEVEL * 13827
ELSE PAYLEVEL * 14400;,
TITLE='Salary',
DESCRIPTION='Salary',
DEFINE PLANTLNG/A11=DECODE PLANT (BOS 'Boston' DAL 'Dallas'
LA 'Los Angeles' ORL 'Orlando' SEA 'Seattle' STL 'St Louis'
ELSE 'n/a');$
```

CENTHR Structure Diagram

```
SECTION 01
STRUCTURE OF FOCUS      FILE CENTHR      ON 05/15/03 AT 10.40.34

      EMPSEG
01      S1
*****
*ID_NUM      **
*LNAME      **
*FNAME      **
*PLANT      **
*           **
*****
*****
```

## CENTINV Master File

```

FILE=CENTINV, SUFFIX=FOC, FDFC=19, FYRT=00
SEGNAME=INVINFO, SEGTYPE=S1, $
  FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4, INDEX=I,
  TITLE='Product,Number:', $
  DESCRIPTION='Product Number', $
  FIELD=PRODNAME, ALIAS=PNAME, FORMAT=A30,
  WITHIN=PRODCAT,
  TITLE='Product,Name:', $
  DESCRIPTION='Product Name', $
  FIELD=QTY_IN_STOCK, ALIAS=QIS, FORMAT=I7,
  TITLE='Quantity,In Stock:', $
  DESCRIPTION='Quantity In Stock', $
  FIELD=PRICE, ALIAS=RETAIL, FORMAT=D10.2,
  TITLE='Price:', $
  DESCRIPTION=Price, $
  FIELD=COST, ALIAS=OUR_COST, FORMAT=D10.2,
  TITLE='Our,Cost:', $
  DESCRIPTION='Our Cost:', $
  DEFINE PRODCAT/A22 = IF PRODNAME CONTAINS 'LCD'
  THEN 'VCRs' ELSE IF PRODNAME
  CONTAINS 'DVD' THEN 'DVD' ELSE IF PRODNAME CONTAINS 'Camcor'
  THEN 'Camcorders'
  ELSE IF PRODNAME CONTAINS 'Camera' THEN 'Cameras' ELSE IF PRODNAME
  CONTAINS 'CD' THEN 'CD Players'
  ELSE IF PRODNAME CONTAINS 'Tape' THEN 'Digital Tape Recorders'
  ELSE IF PRODNAME CONTAINS 'Combo' THEN 'Combo Players'
  ELSE 'PDA Devices'; WITHIN=PRODTYPE, TITLE='Product Category:', $
  DEFINE PRODTYPE/A19 = IF PRODNAME CONTAINS 'Digital' OR 'DVD' OR 'QX'
  THEN 'Digital' ELSE 'Analog'; WITHIN='*Product Dimension',
  TITLE='Product Type:', $

```

## CENTINV Structure Diagram

```

SECTION 01
  STRUCTURE OF FOCUS      FILE CENTINV   ON 05/15/03 AT 10.43.35

      INVINFO
01      S1
*****
*PROD_NUM      **I
*PRODNAME      **
*QTY_IN_STOCK**
*PRICE         **
*              **
*****
*****

```

**CENTORD Master File**

```

FILE=CENTORD, SUFFIX=FOC
SEGNAME=OINFO, SEGTYPE=S1, $
  FIELD=ORDER_NUM, ALIAS=ONUM, FORMAT=A5, INDEX=I,
  TITLE='Order,Number:', $
  DESCRIPTION='Order Number', $
  FIELD=ORDER_DATE, ALIAS=ODATE, FORMAT=YYMD,
  TITLE='Date,Of Order:', $
  DESCRIPTION='Date Of Order', $
  FIELD=STORE_CODE, ALIAS=SNUM, FORMAT=A6, INDEX=I,
  TITLE='Company ID#:', $
  DESCRIPTION='Company ID#', $
  FIELD=PLANT, ALIAS=PLNT, FORMAT=A3, INDEX=I,
  TITLE='Manufacturing,Plant', $
  DESCRIPTION='Location Of Manufacturing Plant',
  WITHIN='*Location', $
  DEFINE YEAR/YY=ORDER_DATE;,
  WITHIN='*Time Period', $
  DEFINE QUARTER/Q=ORDER_DATE;,
  WITHIN='YEAR', $
  DEFINE MONTH/M=ORDER_DATE;,
  WITHIN='QUARTER', $
SEGNAME=PINFO, SEGTYPE=S1, PARENT=OINFO, $
  FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4, INDEX=I,
  TITLE='Product,Number#:', $
  DESCRIPTION='Product Number#', $
  FIELD=QUANTITY, ALIAS=QTY, FORMAT=I8C,
  TITLE='Quantity:', $
  DESCRIPTION=Quantity, $
  FIELD=LINEPRICE, ALIAS=LINETOTAL, FORMAT=D12.2MC,
  TITLE='Line,Total', $
  DESCRIPTION='Line Total', $
  DEFINE LINE_COGS/D12.2=QUANTITY*COST;,
  TITLE='Line,Cost Of,Goods Sold', $
  DESCRIPTION='Line cost of goods sold', $
  DEFINE PLANTLNG/All=DECODE PLANT (BOS 'Boston' DAL 'Dallas'
  LA 'Los Angeles' ORL 'Orlando' SEA 'Seattle' STL 'St Louis'
  ELSE 'n/a');
SEGNAME=INVSEG, SEGTYPE=DKU, PARENT=PINFO, CRFILE=CENTINV,
CRKEY=PROD_NUM, CRSEG=INVINFO, $
SEGNAME=STOSEG, SEGTYPE=DKU, PARENT=OINFO, CRFILE=CENTCOMP,
CRKEY=STORE_CODE, CRSEG=COMPINFO, $

```



## CENTORD Structure Diagram

```

SECTION 01
      STRUCTURE OF FOCUS      FILE CENTORD  ON 05/15/03 AT 10.17.52

      OINFO
01      S1
*****
*ORDER_NUM      **I
*STORE_CODE     **I
*PLANT          **I
*ORDER_DATE     **
*               **
*****
      I
      +-----+
      I               I
      I STOSEG       I PINFO
02      I KU          03      I S1
.....
:STORE_CODE :K *PROD_NUM      **I
:STORENAME  : *QUANTITY      **
:STATE      : *LINEPRICE     **
:           : *              **
:           : *              **
:           : *              **
:.....: *****
JOINED  CENTCOMP *****
      I
      I
      I
      I INVSEG
      04      I KU
.....
:PROD_NUM      :K
:PRODNAME      :
:QTY_IN_STOCK:
:PRICE         :
:           :
:.....:
JOINED  CENTINV

```

**CENTQA Master File**

```

FILE=CENTQA, SUFFIX=FOC, FDFC=19, FYRT=00
SEGNAME=PROD_SEG, SEGTYPE=S1, $
  FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4, INDEX=I,
  TITLE='Product,Number',
  DESCRIPTION='Product Number', $
SEGNAME=PROB_SEG, PARENT=PROD_SEG, SEGTYPE=S1, $
  FIELD=PROBNUM, ALIAS=PROBNO, FORMAT=I5,
  TITLE='Problem,Number',
  DESCRIPTION='Problem Number',
  WITHIN=PLANT,$
  FIELD=PLANT, ALIAS=PLT, FORMAT=A3, INDEX=I,
  TITLE=Plant,
  DESCRIPTION=Plant,
  WITHIN=PROBLEM_LOCATION,$
  FIELD=PROBLEM_DATE, ALIAS=PDATE, FORMAT=YYMD,
  TITLE='Date,Problem,Reported',
  DESCRIPTION='Date Problem Was Reported', $
  FIELD=PROBLEM_CATEGORY, ALIAS=PROBCAT, FORMAT=A20, $
  TITLE='Problem,Category',
  DESCRIPTION='Problem Category',
  WITHIN=*Problem,$
  FIELD=PROBLEM_LOCATION, ALIAS=PROBLOC, FORMAT=A10,
  TITLE='Location,Problem,Occurred',
  DESCRIPTION='Location Where Problem Occurred',
  WITHIN=PROBLEM_CATEGORY,$
  DEFINE PROB_YEAR/YY=PROBLEM_DATE;
  TITLE='Year,Problem,Occurred',
  DESCRIPTION='Year Problem Occurred',
  WITHIN=*Time Period,$
  DEFINE PROB_QUARTER/Q=PROBLEM_DATE;
  TITLE='Quarter,Problem,Occurred',
  DESCRIPTION='Quarter Problem Occurred',
  WITHIN=PROB_YEAR,$
  DEFINE PROB_MONTH/M=PROBLEM_DATE;
  TITLE='Month,Problem,Occurred',
  DESCRIPTION='Month Problem Occurred',
  WITHIN=PROB_QUARTER,$
  DEFINE PROBLEM_OCCUR/I5 WITH PROBNUM=1;
  TITLE='Problem,Occurrence'
  DESCRIPTION='# of times a problem occurs',$
  DEFINE PLANTLNG/All=DECODE PLANT (BOS 'Boston' DAL 'Dallas'
  LA 'Los Angeles' ORL 'Orlando' SEA 'Seattle' STL 'St Louis'
  ELSE 'n/a');$
SEGNAME=INVSEG, SEGTYPE=DKU, PARENT=PROD_SEG, CRFILE=CENTINV,
CRKEY=PROD_NUM, CRSEG=INVINFO,$

```

## CENTQA Structure Diagram

```

SECTION 01
      STRUCTURE OF FOCUS      FILE CENTQA      ON 05/15/03 AT 10.46.43

      PROD_SEG
01      S1
*****
*PROD_NUM      **I
*
*              **
*              **
*              **
*              **
*****
*****
      I
      +-----+
      I              I
      I INVSEG      I PROB_SEG
02      I KU          03      I S1
.....          *****
:PROD_NUM      :K      *PROBNUM      **
:PRODNAME      :      *PLANT      **I
:QTY_IN_STOCK:      *PROBLEM_DATE**
:PRICE      :      *PROBLEM_CAT>**
:              :      *              **
:.....:      *****
JOINED CENTINV      *****

```

## CENTGL Master File

```

FILE=CENTGL ,SUFFIX=FOC
SEGNAME=ACCOUNTS, SEGTYPE=S1
FIELDNAME=GL_ACCOUNT, ALIAS=GLACCT, FORMAT=A7,
  TITLE='Ledger,Account', FIELDTYPE=I, $
FIELDNAME=GL_ACCOUNT_PARENT, ALIAS=GLPAR, FORMAT=A7,
  TITLE=Parent,
  PROPERTY=PARENT_OF, REFERENCE=GL_ACCOUNT, $
FIELDNAME=GL_ACCOUNT_TYPE, ALIAS=GLTYPE, FORMAT=A1,
  TITLE=Type,$
FIELDNAME=GL_ROLLUP_OP, ALIAS=GLROLL, FORMAT=A1,
  TITLE=Op, $
FIELDNAME=GL_ACCOUNT_LEVEL, ALIAS=GLLEVEL, FORMAT=I3,
  TITLE=Lev, $
FIELDNAME=GL_ACCOUNT_CAPTION, ALIAS=GLCAP, FORMAT=A30,
  TITLE=Caption,
  PROPERTY=CAPTION, REFERENCE=GL_ACCOUNT, $
FIELDNAME=SYS_ACCOUNT, ALIAS=ALINE, FORMAT=A6,
  TITLE='System,Account,Line', MISSING=ON, $

```

CENTGL Structure Diagram

```
SECTION 01
STRUCTURE OF FOCUS      FILE CENTGL      ON 05/15/03 AT 15.18.48

      ACCOUNTS
01      S1
*****
*GL_ACCOUNT  **I
*GL_ACCOUNT_> **
*GL_ACCOUNT_> **
*GL_ROLLUP_OP **
*              **
*****
*****
```

CENTSYSF Master File

```
FILE=CENTSYSF ,SUFFIX=FOC
SEGNAME=RAWDATA ,SEGTYPE=S2
FIELDNAME = SYS_ACCOUNT , ,A6      , FIELDTYPE=I,
  TITLE='System,Account,Line', $
FIELDNAME = PERIOD      , ,YYM      , FIELDTYPE=I,$
FIELDNAME = NAT_AMOUNT  , ,D10.0    , TITLE='Month,Actual', $
FIELDNAME = NAT_BUDGET  , ,D10.0    , TITLE='Month,Budget', $
FIELDNAME = NAT_YTDAMT  , ,D12.0    , TITLE='YTD,Actual', $
FIELDNAME = NAT_YTDBUD  , ,D12.0    , TITLE='YTD,Budget', $
```

CENTSYSF Structure Diagram

```
SECTION 01
STRUCTURE OF FOCUS      FILE CENTSYSF      ON 05/15/03 AT 15.19.27

      RAWDATA
01      S2
*****
*SYS_ACCOUNT  **I
*PERIOD       **I
*NAT_AMOUNT   **
*NAT_BUDGET    **
*              **
*****
*****
```

**CENTSTMT Master File**

```

FILE=CENTSTMT, SUFFIX=FOC
SEGNAME=ACCOUNTS, SEGTYPE=S1
  FIELD=GL_ACCOUNT, ALIAS=GLACCT, FORMAT=A7,
    TITLE='Ledger,Account', FIELDTYPE=I, $
  FIELD=GL_ACCOUNT_PARENT, ALIAS=GLPAR, FORMAT=A7,
    TITLE=Parent,
    PROPERTY=PARENT_OF, REFERENCE=GL_ACCOUNT, $
  FIELD=GL_ACCOUNT_TYPE, ALIAS=GLTYPE, FORMAT=A1,
    TITLE=Type,$
  FIELD=GL_ROLLUP_OP, ALIAS=GLROLL, FORMAT=A1,
    TITLE=Op, $
  FIELD=GL_ACCOUNT_LEVEL, ALIAS=GLLEVEL, FORMAT=I3,
    TITLE=Lev, $
  FIELD=GL_ACCOUNT_CAPTION, ALIAS=GLCAP, FORMAT=A30,
    TITLE=Caption,
    PROPERTY=CAPTION, REFERENCE=GL_ACCOUNT, $
SEGNAME=CONSOL, SEGTYPE=S1, PARENT=ACCOUNTS, $
  FIELD=PERIOD, ALIAS=MONTH, FORMAT=YYM, $
  FIELD=ACTUAL_AMT, ALIAS=AA, FORMAT=D10.0, MISSING=ON,
    TITLE='Actual', $
  FIELD=BUDGET_AMT, ALIAS=BA, FORMAT=D10.0, MISSING=ON,
    TITLE='Budget', $
  FIELD=ACTUAL_YTD, ALIAS=AYTD, FORMAT=D12.0, MISSING=ON,
    TITLE='YTD,Actual', $
  FIELD=BUDGET_YTD, ALIAS=BYTD, FORMAT=D12.0, MISSING=ON,
    TITLE='YTD,Budget', $

```

## CENTSTMT Structure Diagram

```

SECTION 01
  STRUCTURE OF FOCUS      FILE CENTSTMT ON 05/15/03 AT 14.45.44

      ACCOUNTS
01      S1
*****
*GL_ACCOUNT  **I
*GL_ACCOUNT_> **
*GL_ACCOUNT_> **
*GL_ROLLUP_OP **
*          **
*****
*****
      I
      I
      I
      I CONSOL
02      I S1
*****
*PERIOD      **
*ACTUAL_AMT  **
*BUDGET_AMT  **
*ACTUAL_YTD  **
*          **
*****
*****

```

## CENTGLL Master File

```

FILE=CENTGLL      , SUFFIX=FOC
SEGNAME=ACCOUNTS  , SEGTYPE=S01
FIELDNAME=GL_ACCOUNT, ALIAS=GLACCT,  FORMAT=A7,
      TITLE='Ledger,Account', FIELDTYPE=I, $
FIELDNAME=GL_ACCOUNT_PARENT, ALIAS=GLPAR,  FORMAT=A7,
      TITLE=Parent,
      PROPERTY=PARENT_OF, REFERENCE=GL_ACCOUNT, $
FIELDNAME=GL_ACCOUNT_TYPE, ALIAS=GLTYPE,  FORMAT=A1,
      TITLE=Type,$
FIELDNAME=GL_ROLLUP_OP, ALIAS=GLROLL,  FORMAT=A1,
      TITLE=Op, $
FIELDNAME=GL_ACCOUNT_LEVEL, ALIAS=GLLEVEL, FORMAT=I3,
      TITLE=Lev, $
FIELDNAME=GL_ACCOUNT_CAPTION, ALIAS=GLCAP,  FORMAT=A30,
      TITLE=Caption,
      PROPERTY=CAPTION, REFERENCE=GL_ACCOUNT, $
FIELDNAME=SYS_ACCOUNT, ALIAS=ALINE,  FORMAT=A6,
      TITLE='System,Account,Line', MISSING=ON, $

```

## CENTGLL Structure Diagram

```
SECTION 01
      STRUCTURE OF FOCUS      FILE CENTGLL ON 05/15/03 AT 14.45.44

      ACCOUNTS
01      S1
*****
*GL_ACCOUNT  **I
*GL_ACCOUNT_> **
*GL_ACCOUNT_> **
*GL_ROLLUP_OP **
*              **
*****
*****
```





## Error Messages

---

To see the text or explanation for any error message, you can display it or find it in an errors file. All of the FOCUS error messages are stored in eight system ERRORS files.

❑ For UNIX, Windows, and Open/VMS, the extension is .err.

❑ For z/OS, the ddname is ERRORS.

**In this appendix:**

❑ [Displaying Messages](#)

---

### Displaying Messages

To display the text and explanation for any message, issue the following query command in a stored procedure

```
? n
```

where:

```
n
```

Is the message number.

The message number and text appear, along with a detailed explanation of the message (if one exists). For example, issuing the following command

```
? 210
```

displays the following:

```
(FOC210)      THE DATA VALUE HAS A FORMAT ERROR:
An alphabetic character has been found where all numerical digits are
required.
```



## Table Syntax Summary and Limits

---

This appendix summarizes WebFOCUS reporting commands and options.

**In this appendix:**

- ☐ [TABLE Syntax Summary](#)
  - ☐ [TABLEF Syntax Summary](#)
  - ☐ [MATCH Syntax Summary](#)
  - ☐ [FOR Syntax Summary](#)
  - ☐ [TABLE Limits](#)
-

## TABLE Syntax Summary

The syntax of a TABLE request is:

```

DEFINE FILE filename      CLEAR|ADD
tempfield [/format] [{DEFCENT|DFC} {cc|19} {YRTHRESH|YRT} {-|yy|0}]
  [MISSING {ON|OFF} [NEEDS] {SOME|ALL} [DATA]]
  [(GEOGRAPHIC_ROLE = georole) [WITH realfield]
  [TITLE 'line1[,line2 ...']]
  [DESCRIPTION 'description'] = expression;
tempfield [/format]      REDEFINES qualifier.fieldname = expression;
.
.
.
END
TABLE FILE filename
HEADING [CENTER]
"text"
{display_command} [SEG.] field [/R|/L|/C] [/format]
{display_command} [prefixop.] [field] [/R|/L|/C] [/format]
  [NOPRINT|AS 'title1,...,title5'] [AND|OVER] [obj2...obj1024]
  [WITHIN field] [IN [+]n]
COMPUTE field [/format] [(GEOGRAPHIC_ROLE = georole)] =
  expression; [AS 'title,...,title5'] [IN [+]n]
[AND] ROW-TOTAL [/R|/L|/C] [/format][AS 'name']
[AND] COLUMN-TOTAL [/R|/L|/C] [AS 'name']
ACROSS [HIGHEST] sortfieldn [IN-GROUPS-OF qty]
  [NOPRINT| AS 'title1,...,title5']
BY [HIGHEST] sortfieldn [IN-GROUPS-OF qty]
  [NOPRINT| AS 'title1,...,title5']
BY [HIGHEST|LOWEST{n}] TOTAL [prefix_operator] {field|code_value}
RANKED [AS 'name'] BY {TOP|HIGHEST|LOWEST} [n] field
  [PLUS OTHERS AS 'othertext']
  [IN-GROUPS-OF qty [TILES [TOP m]] [AS 'heading']]
  [NOPRINT|AS 'title1,...,title5']

{BY|ACROSS} sortfield IN-RANGES-OF value [TOP limit]
ON sfld option1 [AND] option2 [WHEN expression:... ]
ON sfld RECAP fld1 [/fmt] = FORECAST (fld2, intvl, npredict,
  '{MOVAVE|EXPAVE}', npnt);

ON sfld RECAP fld1[/fmt] = FORECAST(fld2, interval, npredict, 'DOUBLEXP',
  npoint1, npoint2);

ON sfld RECAP fld1[/fmt] = FORECAST(fld2, interval, npredict, 'SEASONAL',
  nperiod, npoint1, npoint2, npoint3);

ON sfld RECAP fld1 [/fmt] = FORECAST (fld2, intvl, npredict,
  'REGRESS');

```

```

ON {sortfield|TABLE} RECAP y[/fmt] = REGRESS(n, x1, [x2, [x3,]] z);
ON sfld RECAP fld1 [/fmt] = FORECAST (infield, interval, npredict,
'DOUBLEXP', npoint, npoint2);
ON sfld RECAP fld1 [/fmt] = FORECAST (infield, interval, npredict,
'SEASONAL', nperiod, npoint, npoint2, npoint3); {BY|ON} fieldname
SUBHEAD
[NEWPAGE]
"text"

{BY|ON} fieldname SUBFOOT [WITHIN] [MULTILINES][NEWPAGE]
"text" [<prefop.fieldname ... ]" [WHEN expression;]

WHERE [TOTAL] expression
WHERE {RECORDLIMIT|READLIMIT} EQ n
IF [TOTAL] field relation value [OR value...]
WHERE_GROUPED expression
ON TABLE SET parameter value
ON TABLE HOLD [VIA program][AS name] [FORMAT format] [DATASET dataset]
[MISSING {ON|OFF}] [PERSISTENCE {STAGE|PERMANENT}]
ON TABLE {PCHOLD|SAVE|SAVB} [AS name] [FORMAT format] [MISSING {ON|OFF}]
ON TABLE NOTOTAL
ON TABLE COLUMN-TOTAL [/R|/L|/C] [AS 'name'] fieldname
ON TABLE {ROW-TOTAL|ACROSS-TOTAL} [/R|/L|/C][format] [AS 'name'] fldname
{BY|ON} sfld [AS 'text1'] {SUBTOTAL|SUB-TOTAL|SUMMARIZE|RECOMPUTE}
[MULTILINES] [pref. ] [field1 [pref. ] field2 ...] [AS 'text2']
[WHEN expression;]
{ACROSS|ON} sfld [AS 'text1'] {SUBTOTAL|SUB-TOTAL|SUMMARIZE|RECOMPUTE}
[AS 'text2'] [COLUMNS c1 [AND c2 ...]]
ON TABLE {SUBTOTAL|SUB-TOTAL|SUMMARIZE|RECOMPUTE}
[pref. ] [field1 [pref. ] field2 ...] [AS 'text2']
FOOTING [CENTER] [BOTTOM]
"text"
MORE
FILE file2
[IF field relation value [OR value...]|WHERE expression]
{END|RUN|QUIT}

```

## Hierarchical Reporting Syntax Summary

```

SUM [FROLL.]measure_field ...
BY hierarchy_field [HIERARCHY [WHEN expression_using_hierarchy_fields;]
[SHOW [TOP|UP n] [TO {BOTTOM|DOWN m}] [byoption [WHEN condition] ...] ]
[WHERE expression_using_dimension_data]
[ON hierarchy_field HIERARCHY [WHEN expression_using_hierarchy_fields;]
[SHOW [TOP|UP n] [TO BOTTOM|DOWN m] [byoption [WHEN condition] ...]]

```

## TABLEF Syntax Summary

The syntax of a TABLEF request is:

```
TABLEF FILE filename
HEADING [CENTER]
"text"
```

```
{display_command} [SEG.]field [/R|/L|/C] [/format]
{display_command} [prefixop.]field [/R|/L|/C] [/format]
      [NOPRINT|AS 'title1,...,title5']      [AND|OVER]      [obj2...obj495]
      [IN n]
```

```
COMPUTE field [/format]=expression; [AS 'title1,...,title5']
[AND] ROW-TOTAL [AND] COLUMN-TOTAL
```

```
BY [HIGHEST] keyfieldn [NOPRINT]
```

```
ON keyfield option1 [AND] option2...
```

```
WHERE [TOTAL] expression
```

```
IF [TOTAL] field relation value [OR value...]
```

```
ON TABLE SET parameter value
```

```
ON TABLE HOLD [VIA program] [AS name] [FORMAT format] [MISSING {ON|OFF}]
ON TABLE PCHOLD                [AS name] [FORMAT format] [MISSING {ON|OFF}]
ON TABLE SAVE                   [AS name] [FORMAT format] [MISSING {ON|OFF}]
ON TABLE SAVB                   [AS name] [FORMAT format] [MISSING {ON|OFF}]
```

```
ON TABLE NOTOTAL
ON TABLE COLUMN-TOTAL fieldname
ON TABLE ROW-TOTAL fieldname
```

```
FOOTING [CENTER] [BOTTOM]
"text"
```

```
{END|RUN|QUIT}
```

### Note:

- ☐ Prefix operators for TABLEF can be: AVE., ASQ., MAX., MIN., FST., LST., CNT., or SUM.
- ☐ TABLEF requests cannot use:
  - ☐ Prefix operators DST., PCT., PCT.CNT., RPCT., and TOT.
  - ☐ Variables TABPAGENO, TABLASTPAGE, and BYLASTPAGE.

- ☐ SET SQUEEZE
- ☐ SET EMPTYREPORT.
- ☐ Border styling.
- ☐ ACROSS phrases.
- ☐ WITHIN phrases.
- ☐ RANKED BY phrases.
- ☐ NOSPLIT.
- ☐ Requests with multiple display commands (multi-verb requests).

## MATCH Syntax Summary

The syntax of a MATCH request is:

```
MATCH FILE filename      (the OLD file) report request
BY field1 [AS sortfield]
MORE
FILE file3
subrequest
RUN
.
.
.
FILE filename2           (the NEW file) report request
BY field1 [AS sortfield1]
.
.
.
[AFTER MATCH HOLD [AS filename] matchtype]
MORE
FILE file4
subrequest
END
```

where:

*matchtype*

Can be any of the following:

OLD

NEW

OLD-NOT-NEW

NEW-NOT-OLD

OLD-AND-NEW

OLD-OR-NEW

OLD-NOR-NEW

## FOR Syntax Summary

The formal syntax of the FOR statement is:

```
FOR fieldname [NOPRINT]row [OVER row]  
.  
.  
.  
.  
END
```

where:

*row*

Can be any of the following:

```
tag [OR tag...] [options]  
[fieldname]  
DATA n, [n,...] $  
DATA PICKUP [FROM filename] tag [LABEL label] [AS 'text']  
RECAP name [/format]=expression;  
BAR [AS 'character'] [OVER]  
"text"  
parentvalue {GET|WITH} CHILD[REN] [{n|ALL}] [ADD [m|ALL]]  
[AS {CAPTION|'text'}] [LABEL label]parentvalue ADD [{m|ALL}] [AS  
{CAPTION|'text'}] [LABEL label]  
PAGE-BREAK [OVER]
```

*tag*

Can be any of the following:

```
value [OR value...] value TO value
```

*options*

Can be any of the following:



```

AS 'text'
[INDENT m]
NOPRINT
[LABEL label]
WHEN EXISTS
[POST [TO filename]]

```

## TABLE Limits

The following limits apply to TABLE requests:

- ☐ There is no limit to the number of verb objects in a TABLE request. However, an error can occur if the report output format has a limit to the number of columns supported, the operating system has a maximum record length that cannot fit all of the columns, or the amount of memory needed to store the output is not available.
- ☐ Number of verb objects referenced in a MATCH request: 495
- ☐ Number of columns of report output: 1024
- ☐ Total length of all fields in the request or in a single Master File: 256K
- ☐ Total number of sort fields (combined BY and ACROSS): 128

An internal sort will be generated automatically under some circumstances, and these have to be counted in the total. HOLD FORMAT FOCUS/XFOCUS will add FOCLIST as a BY field in order to ensure uniqueness.

- ☐ Maximum size of the output record: 256K (FORMAT/USAGE)
- ☐ Maximum size of the output file: FOCUS partition 2GB, XFOCUS 32GB
- ☐ Maximum size of internal expression representation of a single DEFINE, COMPUTE, -SET, or WHERE phrase: 64K

The internal representation is generated in polish postfix notation, which is significantly smaller than the entered expression. In addition, the constants in expressions (as in DECODE or IF...THEN..ELSE conditions) are stored outside of the expression representations, reducing the space requirement for the expression representation itself.

- ☐ Maximum number of segments in a structure or file: 512.

This means that a total of 511 JOINS can be in effect at any one given time.

**Note:** FOCUS data sources are limited to 64 segments.

- ☐ Maximum size of Alphanumeric fields: 4K characters ( in UTF, this means 12K bytes)

- ❑ Maximum number of display commands in a TABLE request: 16

## Referring to Fields in a Report Request

---

When creating a report, you refer to fields in several parts of the request. For example, in display commands (PRINT, SUM), in sort phrases (BY, ACROSS), and in selection criteria (WHERE, WHERE TOTAL, IF).

Several methods are available for referring to a field. You can:

- ☐ Refer to individual fields by using the alias specified in the Master File, referring to the name defined in the Master File, or using the shortest unique truncation of the field name or alias. For details, see [Referring to an Individual Field](#) on page 1903.
- ☐ Refer to fields using qualified field names. For details, see [Referring to Fields Using Qualified Field Names](#) on page 1904.
- ☐ Refer to all fields in a segment using only one field name. For details, see [Referring to All of the Fields in a Segment](#) on page 1905.

You can also view a list of all the fields that are included in the currently active data source, or a specified Master File. For details, see the *Developing Reporting Applications* manual.

### In this appendix:

- ☐ [Referring to an Individual Field](#)
  - ☐ [Referring to Fields Using Qualified Field Names](#)
  - ☐ [Referring to All of the Fields in a Segment](#)
  - ☐ [Displaying a List of Field Names](#)
- 

## Referring to an Individual Field

You can refer to an individual field in any one of the following ways:

- ☐ Using the field name defined in the Master File.
- ☐ Using the alias (the field name synonym) defined in the Master File.
- ☐ Using the shortest unique truncation of the field name or the alias. When a truncation is used, it must be unique. If it is not unique, an error message is displayed.

- ❑ Adding the letter S to the end of a field name defined in the Master File.

### **Example:** Referring to an Individual Field

In the following requests, DEPARTMENT is the complete field name, DPT is the alias, and DEP is a unique truncation of DEPARTMENT. All these examples produce the same output.

```
1. TABLE FILE EMPLOYEE
   PRINT DEPARTMENT
   END
```

```
2. TABLE FILE EMPLOYEE
   PRINT DPT
   END
```

```
3. TABLE FILE EMPLOYEE
   PRINT DEP
   END
```

**Note:** If you use a truncation that is not unique, the following message will appear:

```
(FOC016) THE TRUNCATED FIELDNAME IS NOT UNIQUE : D
```

## Referring to Fields Using Qualified Field Names

Field names and aliases have a maximum length of 512 characters. They can also be qualified by prepending up to two qualifiers and qualification characters. However, text fields and indexed field names in Master Files for FOCUS data sources are limited to 12 characters, although the aliases for text and indexed fields can have the same length as general field names. Field names are always displayed as column titles in reports, unless a TITLE attribute or an AS phrase is used to provide an alternative name. For related information, see [Using Headings, Footings, Titles, and Labels](#) on page 1431.

You may use the file name, segment name, or both as a qualifier for a specified field.

### **Syntax:** How to Activate Long and Qualified Field Names

The SET FIELDNAME command enables you to activate long (up to 512 characters) and qualified field names.

```
SET FIELDNAME = fieldname
```

where:

*fieldname*

Specifies the activation status of long and qualified field names. Valid identifiers include:

[NEW](#) specifies that 512-character and qualified field names are supported. NEW is the default value.

[NOTRUNC](#) supports the 512-character maximum. It does not permit unique truncations of field names.

[OLD](#) specifies that 512-character and qualified field names are not supported. The maximum length is 12 characters. The limit may be different for some types of non-FOCUS data sources.

**Example:** Using a Qualified Field Name to Refer to a Field

`EMPLOYEE.EMPINFO.EMP_ID`

Is the fully-qualified name of the field EMP\_ID in the EMPINFO segment of the EMPLOYEE file.

**Reference:** Usage Notes for Long and Qualified Field Names

? SET displays the current value of FIELDNAME. In addition, a Dialogue Manager variable called &FOCFIELDNAME is available. &FOCFIELDNAME may have a value of NEW, OLD, or NOTRUNC.

When the value of FIELDNAME is changed within a session, JOIN, and DEFINE commands are affected as follows:

- ☐ When you change from a value of OLD to a value of NEW, all JOIN and DEFINE commands are cleared.
- ☐ When you change from a value of OLD to NOTRUNC, all JOIN and DEFINE commands are cleared.
- ☐ When you change from a value of NEW to OLD, all JOIN and DEFINE commands are cleared.
- ☐ When you change from a value of NOTRUNC to OLD, all JOIN and DEFINE commands are cleared.

All other changes to the FIELDNAME value have no effect on JOIN and DEFINE commands.

For additional information about using qualified field names in report requests, see the *Describing Data With WebFOCUS Language* manual.

## Referring to All of the Fields in a Segment

If you want to generate a report that displays all of a segment fields, you can refer to the complete segment without specifying every field. You only need to specify one field in the segment (any field will do) prefixed with the SEG. operator.

### **Example: Referring to All Fields in a Segment**

The segment PRODS01 in the GGPRODS Master File contains the PRODUCT\_ID, PRODUCT\_DESCRIPTION, VENDOR\_CODE, VENDOR\_NAME, PACKAGE\_TYPE, SIZE, and UNIT\_PRICE fields.

```
SEGMENT=PRODS01
FIELDNAME = PRODUCT_ID
FIELDNAME = PRODUCT_DESCRIPTION
FIELDNAME = VENDOR_CODE
FIELDNAME = VENDOR_NAME
FIELDNAME = PACKAGE_TYPE
FIELDNAME = SIZE
FIELDNAME = UNIT_PRICE
```

To write a report that includes data from every field in the segment, you can issue either of the following requests:

1. 

```
TABLE FILE GGPRODS
PRINT PRODUCT_ID AND PRODUCT_DESCRIPTION AND VENDOR_CODE AND
  VENDOR_NAME AND PACKAGE_TYPE AND SIZE AND UNIT_PRICE
END
```
2. 

```
TABLE FILE GGPRODS
PRINT SEG.PRODUCT_ID
END
```

## Displaying a List of Field Names

If you want to see a list of all the fields that are included in the currently active data source, you can issue the ?F field name query.

This is useful if you need to refer to a list of field names, or need to check the spelling of a field name, without exiting from the request process. It will also show you the entire 66-character field name.

You can issue the ?F query from the Editor in WebFOCUS.

More information on all of the query (?) commands appears in the *Developing Reporting Applications* manual.

## Listing Field Names, Aliases, and Format Information

The ?FF query displays field name, alias, and format information for a specified Master File, grouped by segment.

You can issue the ?FF query from the Editor in WebFOCUS.

If your software supports MODIFY or FSCAN, you can also issue ?FF from these facilities.

**Note:**

- ❑ If duplicate field names match a specified string, the display includes the field name qualified by the segment name with both ?F and ?FF.
- ❑ Field names longer than 31 characters are truncated in the display, and a caret (>) is appended in the 32nd position to indicate that the field name is longer than the display.





# Index

\_ masking character [248](#)  
- subtraction operator[/] [437](#)  
-n spot marker [1593](#)  
-SET command [431](#)  
? DEFINE command [288](#)  
? FILTER command [269](#), [270](#)  
? JOIN command [1072](#)  
? STAT command [191](#)  
?F command [1906](#)  
?FF command [1906](#)  
[F]DEFCENT attribute [453](#)  
[F]YRTHRESH attribute [453](#)  
\* multiplication operator [434](#), [437](#)  
\*\* exponentiation operator [437](#)  
/ division operator [434](#), [437](#)  
/n spot marker [1593](#)  
&FOCFIELDNAME variable [1905](#)  
% masking character [248](#), [250](#), [251](#)  
+ addition operator [437](#)  
\$ masking character [248](#), [250](#)  
\$\* masking character [248](#), [250](#), [252](#)  
  
OX spot marker [1434](#)  
3D graphs [1663](#)  
508 compliance [1387](#)

## A

absolute starting positions [1588](#)

ACCEPT attribute [495](#)  
Accordion By Column Reports [953](#)  
Accordion Reports [925](#), [927](#), [954](#)  
Accordion Reports and ACROSS phrase [925](#)  
Accordion Reports and BY phrase [925](#)  
Accordion Reports and distributing with ReportCaster [927](#)  
Accordion Reports and drill-downs [927](#)  
Accordion Reports and HTML report [925](#)  
ACROSS attribute [1183](#), [1184](#)  
ACROSS COLUMNS AND phrase [154](#), [156](#), [157](#)  
ACROSS phrase [45](#), [66](#), [91](#), [98–100](#), [110](#), [303](#), [925](#)  
ACROSS subtype [1183](#), [1192](#)  
ACROSS summary commands [415](#)  
ACROSS values [376](#), [377](#), [1191](#), [1193](#)  
ACROSS with ROW-TOTAL [373](#)  
ACROSS-TOTAL component [376](#), [377](#)  
ACROSSCOLUMN attribute [1170](#), [1172](#), [1418](#)  
ACROSSCOLUMN subtype [1419](#), [1701](#), [1704](#)  
ACROSSPRT parameter [110](#)  
ACROSSTITLE component [1192](#), [1193](#)  
ACROSSVALUE and drill-downs [1193](#)  
ACROSSVALUE component [1183](#), [1192](#), [1193](#)  
ACRSVRBTITL parameter [100](#)  
ADD command [54](#)  
ADD option [288](#)  
ADD parameter [1771](#)

adding attributes to WebFOCUS StyleSheets [917](#),  
[919](#)  
adding blank lines to headings and footings [1599](#)  
adding blank rows [1759](#)  
adding blank spaces around a report [1261](#)  
adding calculated values in financial reports [1761](#)  
adding color to graphs [1701](#), [1702](#)  
adding columns to financial reports [1761](#)  
adding comments to WebFOCUS StyleSheets  
[1124](#)  
adding conditional styling to graphs [1702](#)  
adding declarations to font map files [585](#)  
adding dynamic tables of contents to headings  
[919](#)  
adding dynamic tables of contents to reports [905](#),  
[907](#), [908](#)  
adding dynamic tables of contents to WebFOCUS  
StyleSheets [907](#), [908](#)  
adding dynamic TOCs (tables of contents) to  
headings [917](#)  
adding fields to graph footings [1710](#)  
adding fields to graph headings [1710](#)  
adding fields to headings and footings [1710](#)  
adding financial data to reports [1741](#)  
adding fonts for PostScript (PS) format [583](#)  
adding fonts in PS (PostScript) format [574](#)  
adding footings to graphs [1710](#)  
adding graphics to a report [874](#), [1376](#), [1380](#),  
[1382](#), [1388](#), [1390](#), [1395](#)  
adding headings to graphs [1710](#)

adding HTML reports to dynamic tables of  
contents [905](#)  
adding HTML tables of contents to headings [919](#)  
adding HTML tables of contents to reports [907](#),  
[908](#)  
adding HTML tables of contents to WebFOCUS  
StyleSheets [907](#), [908](#)  
adding HTML TOCs (tables of contents) to  
headings [917](#)  
adding images to a report [1376](#), [1382](#)  
adding labels to graphs [1710](#), [1712](#)  
adding line breaks to text fields [1653](#)  
adding PostScript fonts [583](#)  
adding PostScript Type 1 fonts [583](#)  
adding PostScript Type1 fonts in Unix [585](#)  
adding PostScript Type1 fonts in Windows [585](#)  
adding PostScript Type1 fonts in z/OS [587](#)  
adding rows to financial reports [1732](#), [1735](#)  
adding tables of contents (TOCs) to HTML reports  
[905](#)  
adding tag rows to financial reports [1732](#), [1733](#),  
[1767](#)  
adding text rows to financial reports [1759](#)  
adding TOCs (tables of contents) to HTML reports  
[905](#)  
adding underlines [1363](#), [1364](#)  
adding underlines to columns [1368](#), [1369](#)  
adding underlines to financial reports [1783](#)  
adding values [55](#)  
adding values for numeric fields [55](#)

- adding virtual fields [288](#)
- addition operator [437](#)
- addressing columns [1751](#)
- aggregate values [228](#)
- aggregation [197](#)
- aggregation and external sorting [198](#), [199](#)
- AHTML format [512](#)
- AHTMLTAB format [512](#)
- aliases [1825](#), [1903](#), [1904](#)
  - displaying [1906](#)
- aligning decimal points [1576](#), [1578](#), [1579](#)
- aligning footings [1579](#)
- aligning heading items [1590](#)
- aligning headings [1576](#), [1579](#)
- aligning headings and footings [1546](#), [1548](#), [1550](#), [1561](#), [1566](#)
- aligning sort headings [1579](#)
- aligning text fields [1564](#), [1565](#)
- aligning using HTML tables [1561](#)
- alignment methods [1546](#)
- alignment with OVER [1277](#)
- ALL parameter [228](#), [995](#), [1009](#)
- ALL parameter and JOIN command [1008](#), [1009](#)
- ALL parameter and missing values [995–997](#)
- ALL parameters [996](#)
- ALL prefix [995](#)
- ALL prefix and missing values [995](#)
- ALLOCATE command [472](#)
- ALPHA format [512](#)
- alphanumeric fields [247–255](#), [391](#)
- alphanumeric fields and text fields [362](#)
- ALT attribute [1379](#), [1380](#), [1387](#)
- alternate file views [1839](#), [1840](#)
- alternate indexes [274](#)
- alternating background color [1622](#)
- amper variables [1703](#), [1710](#)
- AND operator [237](#), [466](#)
- AnV fields [461](#)
- APDF display format [513](#)
- applying FORECAST command to conditional formatting [1155](#)
- applying FORECAST command to conditional styling [1143](#)
- applying graphs to columns [1413](#)
- applying graphs to report columns [1413](#)
- applying grids to a report [1348](#), [1349](#)
- applying macros [1125](#), [1126](#)
- applying selection criteria in graph requests [1694](#), [1710](#)
- area graphs [1663](#), [1678](#)
- arithmetic expressions [434](#)
- arithmetic operators [434](#), [437](#)
- around headings and footings of border [1321](#)
- AS CAPTION phrase [1767](#), [1768](#)
- AS phrase [473](#), [486](#)
- ascending sort order [152](#)
- ASNAMES command [484–488](#)
- ASQ calculations on field values [64](#)
- assigning cascading style sheet classes to report components [1224](#), [1225](#)

assigning classes in cascading style sheets to  
ACROSS values [1217](#)  
assigning classes in cascading style sheets to  
report components [1222](#), [1225](#)  
assigning external cascading style sheet classes  
to report components [1225](#)  
associating bar graphs with report columns [1413](#),  
[1418](#)  
associating graphs with columns [1418](#)  
associating graphs with report columns [1418](#)  
attribute inheritance [1127](#), [1128](#)  
attributes [764](#), [780](#), [814](#), [1122](#), [1422](#), [1709](#),  
[1783](#)  
augmenting attributes [1128](#)  
AUTOINDEX parameter [1842–1844](#)  
AUTOPATH parameter [1842](#)  
AUTOTABLEF parameter [69](#), [71](#)  
AVE field [63](#)  
AVE prefix operators [63](#)

## B

BACKCOLOR [1620](#), [1622](#)  
background color [1620](#)  
background color, alternating [1622](#)  
background images [1376](#), [1388](#), [1389](#)  
BACKIMAGE attribute [1376](#), [1379](#), [1388](#), [1389](#)  
bar graphs [1663](#), [1674](#)  
BAR  
    command [1368](#), [1369](#), [1783](#)  
    rows [1783](#)

base dates [441](#), [442](#)  
BASEURL SET parameter [813](#)  
basic date support in graphs [1692](#)  
BINARY format [513](#)  
BINARY HOLD format [513](#)  
BINARY output file format [513](#)  
BINS parameter [191](#)  
bipolar graphs [1663](#)  
blank lines [1354](#), [1356](#), [1598](#)  
blank lines, removing [1370](#)  
blank rows [1759](#)  
blank spaces [1258](#)  
blank spaces above data values [1262](#)  
blank spaces between columns [1264](#)  
blanks [986](#)  
BLEND-MODE parameter [1052](#)  
BLOB image in reports [1390](#), [1395](#)  
BODY element [1222](#)  
Boolean expressions [465](#), [467](#)  
Boolean operator [466](#)  
Boolean operators [465](#)  
BORDER attribute [1319](#), [1321](#), [1323](#), [1528](#),  
[1783](#)  
border colors [1321](#)  
border styles [1321](#)  
borders [1318](#), [1323](#), [1526](#), [1792](#)  
    adding for cells [1792](#), [1796](#)  
    adding for rows [1794](#), [1795](#)  
BOTTOM [209](#)  
bottom margins [1255](#)

BOTTOMGAP attribute [1258](#), [1261](#), [1600](#)  
 BOTTOMMARGIN attribute [1255](#), [1256](#)  
 breaks [1293](#)  
 browser fonts [1618](#), [1619](#)  
 browser fonts in HTML Reports [1618](#)  
 browser support for cascading style sheets [1241](#)  
 browser support for HTML tables of contents [925](#)  
 browser titles [1436](#)  
 bubble charts [1681](#)  
 bursting reports [965](#), [968](#)  
 BY [95](#)  
 BY HIERARCHY [209](#)  
 BY phrase [45](#), [56](#), [66](#), [91–94](#), [141](#), [1174](#), [1740](#)  
     with financial reports [1740](#), [1741](#)  
 BY ROWS OVER phrase [154–156](#)  
 BY subtype [1174](#), [1186](#), [1197](#)  
 BY TOTAL phrase [176–178](#)  
 BYLASTPAGE system variable [1305](#), [1308](#)  
 byte precision [58](#), [59](#)

## C

calculated values [278](#), [280](#), [303](#), [304](#), [369](#), [373](#),  
[385–387](#), [1501](#), [1761](#)  
     sorting reports [304](#)  
 calculating column and row totals [369](#), [370](#)  
 calculating column Percentages [66](#)  
 calculating column totals [369–373](#)  
 calculating dates [443](#)  
 calculating MAX field values [65](#)  
 calculating maximum field values [65](#)  
 calculating maximum values for field values [65](#)  
 calculating median of field values [65](#)  
 calculating MIN field values [65](#)  
 calculating minimum field values [65](#)  
 calculating minimum values for field values [65](#)  
 calculating mode of field values [65](#)  
 calculating row Percentages [66](#)  
 calculating row totals [369–372](#)  
 calculating trends [359](#)  
 calculating values for temporary fields [298](#)  
 calculation on field values [68](#)  
 calculations [1757](#)  
 calculations and functions [1757](#), [1758](#)  
 calculations counting field values [74](#)  
 calculations on counting field values [74](#)  
 calculations on field values [60](#), [63](#), [65](#), [141](#)  
 calculations on sum numeric field values [74](#)  
 calculations on SUM numeric field values [74](#)  
 calculations on TOT field values [74](#)  
 calculations on total field values [74](#)  
 calling functions [1757](#), [1758](#)  
 CAPTION parameter [1766](#), [1771](#), [1778](#), [1779](#)  
 captions [1436](#)  
     in a hierarchy [1770](#)  
     in Master Files [1766](#), [1778](#), [1779](#)  
 CAR data source [1855](#), [1857](#)  
 carriage-returns [1653](#)  
 Cartesian product [1104](#), [1105](#)  
 Cartesian product answer sets [1829](#)

Cartesian product answer sets and SQL Translator [1829](#)

cascading style sheet (CSS) [1113](#), [1140](#), [1141](#),  
[1211](#), [1220](#)

    browser support [1239](#), [1241](#), [1245](#)

    choosing [1221](#)

    class for ACROSS values [1217](#)

    classes [1212](#), [1222](#), [1241](#)

    conditional styling [1239](#)

    editing [1221](#)

    example [1218](#)

    external [1211](#)

    FAQS [1241](#)

    file location [1221](#)

    formatting [1224](#), [1226](#)

    images [1227](#), [1239](#)

    inheritance [1231](#), [1232](#), [1245](#)

    internal [1139](#)

    linking [1216](#), [1217](#), [1228–1231](#)

    location [1221](#)

    multiple [1221](#)

    multiple output formats [1234](#), [1236](#), [1238](#)

    naming classes [1223](#), [1225](#)

    personal [1245](#)

    refreshing [1221](#), [1245](#)

    report formatting [1214](#), [1216](#)

    requirements [1239](#)

    rules [1212](#), [1217](#), [1222](#), [1241](#)

    troubleshooting [1245](#)

CDN (Continental Decimal Notation) [1631](#), [1829](#)

CDN (Continental Decimal Notation) and SQL  
Translator [1829](#)

CDN parameter [1631](#)

cells [1755](#), [1756](#)

    formatting [1782](#), [1785](#), [1787](#), [1792](#)

    notation [1756](#)

CENTER command [1530](#)

CENTFIN data source [1877](#)

CENTHR data source [1877](#)

CENTINV data source [1877](#)

CENTORD data source [1877](#)

CENTQA data source [1877](#)

Century Corp data sources [1877](#)

changing column order [1272](#)

changing row titles [1781](#)

    PICKUP rows [1806](#)

    RECAP rows [1782](#)

    TAG rows [1781](#)

character expressions [432](#), [458](#)

character strings [247](#), [460](#)

Chart of Accounts hierarchy [1764](#), [1766](#)

charts [1663](#)

charts of accounts [1767](#)

CHECK FILE command [1071](#)

CHECK FILE command and join structures [1070](#),  
[1071](#)

CHECK PICTURE command [49](#)

CHECK STYLE command [1120](#)

choosing formatting reports style sheets [1113](#)

choosing report formatting style sheets [1113](#)

- CLASS attribute [1217](#), [1224](#), [1225](#), [1241](#)
- clearing conditional join structures [1073](#)
- clearing join structures [1072](#), [1073](#)
- clearing virtual fields [288](#), [289](#)
- CMS requirements [190](#)
- CNOTATION SET parameter [304](#), [305](#), [1752](#), [1753](#)
- CNT prefix operator [74](#)
- collapsing PRINT with ACROSS [110](#)
- collation sequence [144](#)
- COLLATION SET parameter [144](#)
- COLOR attribute [1614](#)
- color graph settings [1727](#)
- color settings [1254](#)
- color settings in graphs [1727](#)
- color values [1615](#)
- color, background [1620](#)
- color, background, alternating [1622](#)
- COLSPAN attribute [1566](#)
- column and row totals in calculated values [369](#)
- COLUMN attribute [1170](#), [1171](#), [1182](#), [1418](#)
- column formats [1627](#)
- column notation [304](#), [305](#)
- column reference numbers [304](#)
- COLUMN subtype [1538](#), [1649](#), [1786](#)
- column titles [87](#), [1501](#), [1503](#), [1504](#), [1627](#)
- column titles for calculated values [1506](#)
- column total labels [1513](#)
- column totals [369](#), [372](#), [373](#)
- COLUMN-TOTAL phrase [369–371](#)
- column
  - addresses [1751](#)
- columns [1170](#), [1626](#)
  - addresses [1750–1752](#)
  - formatting [1785](#)
  - in financial reports [1732](#)
  - notation [1752](#), [1753](#)
  - numbers [1748](#)
  - reference numbers [1752](#)
  - values [1753](#)
- COM format [514](#)
- COM HOLD format [514](#)
- COM output file format [514](#)
- COM PCHOLD format [514](#)
- COM SAVE format [514](#)
- combination of summary commands [410](#)
- combinations of subtotals [409](#)
- combining expressions [237](#)
- combining external cascading style sheets with other formatting methods [1226](#), [1227](#)
- combining fields in date expressions [445](#), [446](#)
- combining mixed format reports [871](#), [872](#), [874](#), [875](#)
- combining multiple values [1735](#)
- combining PDF reports [871](#), [872](#)
- combining records [1735](#), [1736](#)
- combining report formats [875](#)
- combining report formatting cascading style sheets and other formatting methods [1241](#)
- combining values [1735](#), [1736](#), [1738](#)

- COMMA format [514](#)
- COMMA HOLD format [514](#)
- COMMA output file format [514](#)
- COMMA SAVE format [514](#)
- comma-delimited files [1007](#)
- command support for Accordion Reports [953](#)
- commands [1124](#), [1125](#)
- comments [1122](#), [1124](#)
- comments in WebFOCUS StyleSheets [1122](#), [1124](#)
- common high-order sort fields [1086](#), [1087](#), [1089](#), [1100](#), [1101](#), [1103](#)
- comparing characters with masks [248–252](#)
- comparing decimal values [1578](#)
- comparing records [1078](#), [1084](#)
- compiling expressions [1846](#)
- compiling virtual fields [1846](#)
- complex expressions [431](#)
- COMPMISS parameter [990](#), [1629](#)
- compound display formats for reports [817](#), [870](#)
- compound expressions [237](#)
- COMPOUND parameter [871](#)
- compound report display formats [817](#), [870](#)
- Compound report syntax
  - COMPONENT [822](#)
  - displaying a grid [840](#)
  - draw objects [834](#)
  - example [823](#)
  - page overflow [831](#)
  - PAGELAYOUT [819](#)
  - SECTION [819](#)
- compound reports [526](#), [817](#), [870](#)
- compound reports in PDF format [872](#)
- compressing PDF output files [576](#)
- COMPUTE command [56](#), [298](#), [299](#), [301](#), [303](#), [385](#), [405](#), [423](#), [424](#), [426](#), [431](#)
- COMPUTE command expressions [431](#)
- COMPUTE component [1190](#)
- computing the average field values [63](#)
- COMT format [515](#)
- COMT HOLD format [515](#)
- COMT output file format [515](#)
- COMT PCHOLD format [515](#)
- COMT SAVE format [515](#)
- concatenated data sources and MATCH FILE command [1098](#)
- concatenating character strings [460](#)
- concatenating data sources [1093–1095](#)
- concatenating data sources and field names [1097](#)
- concatenation [460](#)
- concatenation data sources [1095](#), [1098](#)
- concatenation for data sources [1093](#)
- concatenation operators [460](#)
- concatenation usage formats [1096](#)
- conditional drill-down [806–808](#)
- conditional drill-down and multiple links [796](#)
- conditional expression types [467](#)
- conditional expressions [432](#), [467–469](#)
- conditional formatting [1465](#)
- conditional formatting and WHEN phrase [1156](#), [1158–1161](#), [1465](#)



- conditional grid formatting [1632](#), [1633](#)
- conditional join structures [1006](#), [1007](#), [1016](#), [1034](#), [1035](#), [1037](#), [1070](#)
- conditional operators [238](#), [243](#), [244](#)
- conditional sort headings [1460](#)
- conditional styling [796](#), [1139](#), [1142](#), [1156](#), [1158–1161](#), [1239](#)
- conditional styling and graphs [1701](#), [1702](#)
- conditional styling and style sheets [1145](#), [1152](#)
- conditional styling and WebFOCUS StyleSheets [1142](#), [1143](#), [1145](#), [1146](#), [1148](#), [1150–1152](#)
- conditional text [429](#), [430](#)
- conditionally displaying page breaks [1158](#)
- conditionally displaying skipped lines [1158](#)
- conditionally displaying summary lines [1158](#)
- conditionally displaying underlines [1158](#)
- configuring PostScript fonts [585](#), [587](#)
- configuring PostScript fonts in z/OS [587](#)
- configuring PostScript Type 1 fonts [585](#)
- connected point plot graphs [1663](#), [1673](#)
- consolidating financial data [1767](#), [1770](#), [1771](#), [1773](#), [1774](#)
  - in multiple rows [1774](#)
  - single row [1771](#), [1773](#)
- constant dates [440](#), [444](#)
- constants [1742](#)
  - in Financial Modeling Language (FML) [1741](#)
  - in financial reports [1741](#)
  - in FML (Financial Modeling Language) [1741](#)
  - in FML requests [1741](#), [1742](#)
- CONTAINS operator [247](#)
- contiguous columns [1749](#)
- contiguous columns in financial reports [1749](#)
- Continental Decimal Notation (CDN) [1631](#), [1829](#)
- Continental Decimal Notation (CDN) and SQL Translator [1829](#)
- controlling attributes [484](#), [490](#), [491](#), [495](#)
- controlling attributes and HOLD Master Files [492](#)
- controlling column order [1264](#), [1271](#), [1272](#)
- controlling column reference numbers [304](#), [1752](#)
- controlling column spacing [1264](#), [1270](#), [1271](#)
- controlling display of sort field values [925](#)
- controlling field names [485–488](#)
- controlling field names HOLD Master Files [485](#)
- controlling fields [490](#)
- converting data types for join structures [1034](#)
- converting TABLE requests [1658](#), [1660](#)
- converting TABLE requests to GRAPH requests [1658](#), [1660](#)
- COUNT \* command [57](#), [58](#)
- COUNT command [56](#), [57](#), [92](#)
- COUNT command for unique segments [57](#)
- count of occurrences [74](#)
- counting field values [56](#)
- COUNTWIDTH SET parameter [56](#), [58](#), [59](#)
- COURSE data source [1862](#), [1863](#)
- CREATE TABLE command [1826](#), [1827](#)
- CREATE VIEW command [1827](#), [1828](#)
- creating bar graphs [1668](#), [1669](#)
- creating calculated values [298](#), [301](#)

creating financial reports [1730](#), [1735](#)  
creating FOCUS data sources [479](#), [482–484](#)  
creating free-form reports [1809](#), [1810](#),  
[1812–1816](#)  
creating graphs [1657](#)  
creating HOLD files [473–476](#), [479](#), [480](#), [482](#)  
creating horizontal bar graphs [1668](#), [1669](#)  
creating links [763](#), [764](#)  
creating matrix report with OVER [1273](#)  
creating matrix reports with OVER [1275](#)  
creating multiple virtual fields [288](#)  
creating numeric expressions [434](#)  
creating output files [473](#)  
creating parameters [797](#), [799](#), [801](#), [802](#)  
creating parameters for drill-down reports [797](#)  
creating parameters for drill-downs [1710](#)  
creating PCHOLD files [509](#)  
creating pie graphs [1670](#), [1671](#)  
creating reports [36](#), [38](#)  
creating rows [1732–1734](#)  
    from multiple records [1735](#)  
    in financial reports [1732](#)  
creating SAVB files [508](#)  
creating SAVE files [506](#), [507](#)  
creating scatter graphs [1671](#), [1672](#)  
creating single-line footings [1444](#)  
creating single-line headings [1440](#), [1448](#)  
creating sort headings [1457](#)  
creating style sheets [1118](#)  
creating tag rows [1733](#), [1734](#)

creating temporary fields with COMPUTE phrases  
[298](#)  
creating temporary fields with DEFINE FUNCTION  
[366](#)  
creating vertical bar graphs [1668](#), [1669](#)  
creating virtual fields [280–282](#), [285](#), [287](#), [291](#)  
creating WebFOCUS StyleSheets [1117](#), [1118](#)  
cross-century dates [444](#)  
cross-referenced fields [1025](#)  
cross-referenced files [1008](#), [1016](#)  
CSS (cascading style sheets) [1113](#), [1140](#), [1141](#)  
CSS location [1221](#)  
CSSURL (StyleSheets) attribute [1217](#)  
CSSURL attribute [1217](#), [1221](#), [1228](#), [1229](#)  
CSSURL attribute example [1230](#)  
CSSURL attribute HTMTABLE format [1228](#)  
CSSURL parameter [1217](#), [1221](#), [1228](#), [1231](#)  
CSSURL parameter HTMTABLE format [1228](#)  
CT. prefix operator [1470](#)  
custom report titles [1436](#)  
custom sort order [154](#)  
custom worksheet names [1436](#)  
customizing column titles in a Master File [1507](#)  
customizing graphs [1700](#), [1718](#)  
customizing reports [41](#), [1813](#)  
customizing sort order [154](#)

## D

data [1180](#), [1181](#)  
    retrieval [1732](#), [1805](#), [1806](#)

- date constants [440](#)
- date expressions [440](#), [446](#)
- date fields [440](#)
- date formats [1830](#), [1831](#)
- date formats and SQL Translator [1830](#), [1831](#)
- date support in graphs [1692](#)
- date value formats [441](#), [442](#)
- date values [441](#)
- date-time data types [452](#)
- date-time expression types [440](#)
- date-time expressions [440](#)
- date-time field formats [452](#)
- date-time format [456](#)
- date-time format and display fields [452](#)
- date-time values [440](#), [1832–1834](#)
- date-time values and SQL Translator [1832–1834](#)
- DATEFORMAT parameter [451](#), [452](#)
- DATEFORMAT setting [452](#)
- dates [443](#)
- dates in graphs [1692](#), [1694](#)
- DATREC format [516](#)
- Db2 format [516](#)
- Db2 HOLD format [516](#)
- Db2 output file format [516](#)
- DBAFILE attribute [1008](#)
- DBAJJOIN [1061](#)
- DBASE format [517](#)
- DBASE HOLD format [517](#)
- DBASE output file format [517](#)
- decimal points [1576](#), [1578](#), [1579](#)
- decimal values [1576](#), [1579](#)
- declarations [1122](#), [1123](#)
- declaring filters [266](#)
- default browser type [1618](#)
- default column titles [1502](#)
- default font types [1617](#)
- default proportional fonts [1617](#)
- default StyleSheet values [1165](#)
- DEFAULT- FIXED attribute [1618](#)
- DEFAULT- PROPORTIONAL attribute [1618](#)
- DEFINE and dates [293](#), [294](#), [444](#)
- DEFINE attribute [431](#)
- DEFINE command [282](#), [285](#), [288](#), [431](#)
- DEFINE command and join structures [1027](#), [1028](#), [1030](#), [1065](#), [1066](#), [1068](#)
- DEFINE command and missing values [974–978](#)
- DEFINE command expressions [431](#)
- DEFINE compiler [1846](#)
- DEFINE FILE RETURN command [1068](#)
- DEFINE FILE SAVE command [292](#), [1068](#)
- DEFINE function [363](#)
  - command [363](#)
  - creating temporary fields [364](#)
  - deleting [363](#), [368](#)
  - displaying [363](#)
  - displaying [367](#)
  - limitations [363](#)
  - limitations [365](#)
  - querying [363](#)
- DEFINE functions [366](#)

- Define tool [288](#)
- defining custom groups [169](#), [170](#)
- defining field formats [1627](#)
- defining filters [265–267](#)
- defining macros [1126](#)
- defining virtual fields [285](#)
- DEFMACRO command [1124–1126](#)
- DEFMACRO commands [1124](#)
- DELETE command [1836](#)
- deleting underlines [1363](#), [1364](#)
- delimited file, creating [537](#)
- delimited output files [517](#)
- descending sort order [152](#), [153](#)
- designating missing values [1002](#)
- determining column width [1264–1268](#)
- DFIX [537](#)
- DFIX format [517](#)
- DFSORT utility [190](#)
- DHTML HOLD format [517](#)
- DHTML output format [517](#)
- Dialect Translation [1817](#)
- Dialogue Manager [431](#), [1729](#)
- Dialogue Manager variables [1469](#)
  - FOCFIELDNAME [1905](#)
- DIF format [518](#)
- DIF output file format [518](#)
- DIF PCHOLD format [518](#)
- DIF SAVE format [518](#)
- direct percent [68](#)
- direct percent of counts (PCT.CNT) [68](#)
- display ADD command [54](#)
- display commands [43](#), [92](#), [1090](#)
- display commands and graph format [1665](#)
- display commands and MATCH FILE command [1090](#), [1091](#)
- display COUNT command [56](#), [92](#)
- display DOC formats for reports [602](#)
- display field values [45](#), [46](#)
- display fields [60](#), [61](#), [1628](#)
- display fields and graph format [1665](#)
- display fields for prefix operators [60](#)
- display formats for compound reports [875](#)
- display formats for EXCEL reports [566](#)
- display formats for EXL2K PIVOT reports [566](#)
- display formats for EXL2K reports [566](#), [568](#)
- display formats for HTML reports [568](#)
- display formats for PDF reports [568](#), [574](#), [575](#)
- display formats for PostScript reports [568](#), [574](#), [580](#), [582](#)
- display formats for reports [565](#), [566](#), [568](#), [569](#)
- display LIST \* command [46](#)
- display LIST command [92](#)
- display LIST commands [43](#), [45](#)
- display PRINT \* command [46](#)
- display PRINT command [92](#)
- display PRINT commands [43](#), [45](#)
- display SUM command [54](#), [55](#), [92](#)
- display SUM commands [43](#)
- display values [43](#)
- display WRITE command [54](#)

- displaying ADD command [54](#)
- displaying all fields in a segment [48](#)
- displaying all fields in segments [48](#)
- displaying captions [1767](#)
- displaying children [1767](#), [1769](#), [1770](#)
- displaying compound reports [871](#)
- displaying empty reports [1163](#)
- displaying error messages [194](#), [195](#)
- displaying excluded values [95](#)
- displaying field descriptions [87](#)
- displaying field names [1906](#)
- displaying field values [43](#), [45–47](#)
- displaying grand totals [377](#), [378](#)
- displaying graph data [1683](#)
- displaying hierarchy values as captions [1770](#)
- displaying HOLD Master Files [473](#), [477](#)
- displaying join structures [1070–1072](#)
- displaying LIST \* command [46](#)
- displaying LIST command [45](#)
- displaying LIST commands [43](#)
- displaying missing values [1695](#)
- displaying missing values in graphs [1696](#)
- displaying parents and children [1767](#), [1769](#), [1770](#)
- displaying PRINT \* command [46](#)
- displaying PRINT command [45](#)
- displaying PRINT commands [43](#)
- displaying reports [219](#), [565](#), [566](#), [568](#), [571](#), [574](#), [575](#), [580](#), [602](#)
- displaying reports as PDF [870](#)
- displaying reports as PostScript [870](#)
- displaying reports in a browser [571](#)
- displaying reports in PDFs [817](#)
- displaying reports in PostScripts [817](#)
- displaying reports in WebFOCUS Viewer [957](#)
- displaying retrieval order [49](#)
- displaying retrieval order for multi-path data sources [51](#)
- displaying structure for multi-path data sources [49](#), [50](#)
- displaying sub-totals [383–385](#)
- displaying subtotals [377–379](#), [382–385](#)
- displaying SUM command [54](#)
- displaying summary lines [430](#)
- displaying values [43](#)
- displaying WRITE command [54](#)
- distinct prefix operators [69](#), [70](#)
- division operator [434](#), [437](#)
- DOC format [518](#), [566](#), [602](#)
- DOC output file format [518](#)
- DOC PCHOLD format [518](#)
- DOC report display formats [602](#)
- DOC SAVE format [518](#)
- double exponential smoothing [326](#), [349](#), [350](#)
  - FORECAST\_DOUBLEXP [326](#)
- drill through compound reports in PDF format [890–892](#), [895](#), [897](#), [903](#)
- drill through PDF compound reports [890–892](#), [895–897](#), [903](#)

drill through reports [890–892](#), [895–897](#), [900](#), [903](#)

drill-down reports [763](#), [764](#), [807](#), [808](#), [1709](#)

drill-down reports compared [890](#)

drill-downs [1709](#), [1710](#)

drill-downs and ACROSSVALUE [1193](#)

drill-downs and graphs [1703](#)

drilling down graphs [1703](#), [1705](#)

drilling down reports [764](#)

DRILLMENUITEM attribute [1709](#)

DRILLTHROUGH command [892](#)

DROP VIEW command [1827](#)

DROP VIEW command and SQL Translator [1828](#)

DROPBLNKLINE parameter [1370](#)

DST prefix operator [69](#), [70](#)

DST prefix operator restrictions [69](#)

DST prefix operators [69](#)

duplicate field names as column titles [1507](#)

DUPLICATECOL parameter [183](#)

dynamic reformatting [1650](#), [1651](#)

dynamic reporting [1764](#)

dynamic tables of contents [905](#)

dynamic tables of contents for multiple sort groups [912](#)

dynamically formatting virtual fields [294](#), [295](#)

## E

editing font map files in Windows [585](#)

editing font map files in z/OS [587](#)

editing metrics files in Windows [585](#)

editing metrics files in z/OS [587](#)

EDUCFILE data source [1852](#), [1853](#)

ELEMENT attribute [1425](#), [1426](#)

elements in footings [1469](#)

elements in headings [1469](#)

embedded fields [1202](#), [1203](#)

embedded fields in footings [1202](#)

embedded quotation marks [459](#), [460](#)

embedding compound reports graphs [874](#)

embedding compound reports images [874](#)

embedding graphics [874](#)

embedding images [874](#)

EMPDATA data source [1861](#)

EMPLOYEE data source [1847](#), [1849](#), [1850](#)

empty reports [1163](#)

EMPTYREPORT SET parameter [1163](#)

END command [38](#)

ending a report request [38](#)

EQ operator [246](#), [466](#)

equijoins [1006](#), [1007](#), [1018](#)

error files [1893](#)

error messages [1893](#)

escape characters [253–255](#)

ESSBASE hierarchies [1764](#)

establishing segment locations [291](#)

estimating number of records [192](#)

ESTLINES parameter [192](#)

ESTRECORDS parameter [192](#)

ex\_forecast\_dist [358](#)

ex\_forecast\_mov [357](#)

- ex\_forecast\_mult [356](#)
- ex\_regress\_mult [361](#)
- Excel 2000 alignment [1548](#)
- Excel 2007
  - TOCs (tables of contents) [661](#)
- EXCEL display format [566](#)
- EXCEL format [519](#)
- Excel formats [566](#)
- Excel formats supported with AnV fields [519](#)
- EXCEL output file format [519](#)
- EXCEL report display format [566](#)
- Excel reports [661](#)
- EXCEL SAVE format [519](#)
- EXCEPT operator [1830](#)
- EXCLUDES operator [256](#), [257](#)
- excluding missing values from tests [987](#)
- existing data [246](#), [247](#)
- EXL07 display format
  - TOCs (tables of contents) [661](#)
- EXL2K alignment [1548](#)
- EXL2K display format [520](#), [566](#), [568](#)
- EXL2K FORMULA [521](#)
- EXL2K FORMULA display format [521](#), [566](#)
- EXL2K output file format [520](#)
- EXL2K PCHOLD format [520](#)
- EXL2K PIVOT [521](#)
- EXL2K PIVOT display format [566](#)
- EXL2K PIVOT format [521](#)
- EXL2K PIVOT report display format [566](#)
- EXL2K report display format [566](#)
- EXL2K SAVE format [520](#)
- EXL97 display format [522](#)
- EXPANDABLE command [925](#), [954](#)
- expanding byte precision [59](#)
- expanding byte precision for COUNT command [58](#)
- expanding precision [59](#)
- explicit labels [1744–1746](#)
- EXPN and numeric functions [436](#)
- EXPN function [436](#)
- exponential moving average [315](#), [316](#), [322](#), [338](#), [339](#), [346](#), [347](#)
  - FORECAST\_EXPAVE [322](#)
- exponentiation operator [437](#)
- exporting from data sources [473](#), [476](#)
- expression dates [432](#), [444](#)
- expression types [432](#)
- expressions [277](#), [431](#), [1830](#)
- expressions and SQL Translator [1830](#)
- expressions IF phrase [431](#)
- expressions, relational [467](#)
- EXTAGGR parameter [197](#)
- extending heading and footing code [1434](#)
- extending underlines [1365](#)
- external cascading style sheet class for ACROSS values [1217](#)
- external cascading style sheet classes [1212](#)
- external cascading style sheet rules [1212](#), [1217](#)
- external cascading style sheet rules BODY element [1222](#)

external cascading style sheet rules TD element [1222](#)

external cascading style sheets [1113](#), [1213](#), [1220](#)

- benefits [1213](#)
- browser support [1239](#), [1241](#), [1245](#)
- choosing [1221](#)
- class names [1223](#)
- classes [1222](#), [1241](#)
- editing [1221](#)
- images [1239](#)
- linking to [1228](#)
- multiple [1221](#)
- multiple output formats [1234](#), [1236](#)
- report formatting [1214](#)
- requirements [1239](#)
- rules [1222](#), [1241](#)
- troubleshooting [1245](#)

external files [1740](#)

external report formatting cascading style sheets [1216](#)

external sorting [190](#), [191](#), [194](#), [195](#)

external sorting and aggregation [197](#)

external sorting and HOLD files [200](#), [201](#)

external sorting by aggregation [197–199](#)

external sorting requirements [190](#)

EXTHOLD parameter [201](#)

extract files [472](#), [500](#), [503](#), [505](#), [551](#)

extract files and missing values [987](#), [988](#)

EXTRACT function [1834](#)

extracting date components [445](#)

EXTSORT parameter [190](#), [191](#)

EXTUNDERLINE attribute [1365](#)

## F

FIELD attribute [1418](#)

field dates [440](#)

field format expressions [434](#)

field formats [434](#), [444](#)

field names [486–488](#), [1829](#), [1830](#), [1903](#)

- aliases [1903](#), [1904](#)
- displaying [1906](#)
- long [1904](#), [1905](#)
- qualified [1903–1905](#)
- truncated [1903](#), [1904](#)

field padding [546–548](#)

field references for COMPUTE command [302](#)

field reformatting and missing values [990](#), [1629](#)

field types not supported for EXCEL format [519](#)

field values [43](#), [141](#), [1459](#), [1471](#)

field values in headings and footings [1469–1471](#)

field-based reformatting [293–295](#), [1650–1652](#)

FIELDNAME command [1904](#)

fields [277](#), [434](#), [1627](#), [1903](#)

- in report requests [1903](#), [1904](#)

file location of external cascading style sheets [1221](#)

file names [764](#)

FILECOMPRESS parameter [576](#)

FILEDEF command [472](#), [498](#)



- FILTER parameter [265–268](#)
- FILTER query command [265](#), [269](#), [270](#)
- filters [265](#), [266](#), [268–270](#), [272](#)
- FINANCE data source [1859](#), [1860](#)
- financial data [1770](#)
  - retrieving [1804](#)
  - retrieving values for rows [1733](#), [1734](#)
- Financial Modeling Language (FML) [1173](#), [1729](#), [1900](#)
- Financial Modeling Language (FML) and Dialogue Manager [1729](#)
- Financial Report Painter [34](#)
- financial reports [34](#), [1107](#), [1729](#), [1764](#)
  - adding data [1732](#), [1741](#)
  - charts of accounts [1764](#), [1767](#), [1769](#), [1770](#)
  - external files [1740](#)
  - formatting [1782](#), [1783](#), [1785](#), [1787](#)
  - formatting options [1782](#), [1785](#)
  - hierarchies [1767](#), [1771](#), [1778–1780](#)
  - HOLD files [1807](#)
  - inserting text rows [1759](#)
  - inserting variables in text rows [1760](#), [1761](#)
  - inter-row calculations [1743](#), [1744](#)
  - records in multiple rows [1739](#)
  - recursive models [1763](#), [1764](#)
  - repeating rows [1747](#)
  - saving intermediate results [1804](#)
  - sorting with BY [1740](#)
  - sorting with FOR [1740](#)
  - supplying data as constants [1742](#)
- fixed scales [1712](#), [1718](#)
- FIXRETRIEVE parameter [496](#), [497](#)
- FLEX display format [522](#)
- FML (Financial Modeling Language) [1173](#), [1729](#), [1730](#), [1900](#)
  - and Dialogue Manager [1729](#)
- FML hierarchies [1764](#), [1766](#), [1767](#), [1769](#), [1770](#)
  - displaying [1764](#), [1767](#)
  - indenting captions [1800](#)
  - indenting row titles [1799](#)
  - indenting text or numbers [1783](#)
  - loading into memory [1778](#), [1779](#)
- FML Painter [34](#)
- FOCEXEC attribute [764](#)
- FOCEXEC attributes [764](#)
- FOCFIELDNAME variable [1905](#)
- FOCFIRSTPAGE SET parameter [1304](#), [1305](#), [1311](#)
- FOCHTMLURL parameter [927](#)
  - pop-up field descriptions and [89](#)
- FOCPOST files [1804](#)
- FOCUS data sources [220](#), [479](#)
- FOCUS file structure [479](#), [481](#)
- FOCUS format [522](#)
- FOCUS StyleSheets [1782](#)
- FOLD-LINE command [1265](#), [1274](#)
- FONT attribute [1617](#), [1618](#)
- font attributes [1522](#), [1597](#)
- font colors [1611](#), [1614](#), [1620](#), [1622](#)
- font file [585](#), [587](#)
- font files [583](#)

- font inherited styles [1614](#)
- font map files [583](#), [585](#)
- font sizes [1611](#), [1612](#)
- font styles [1613](#)
- fonts [583](#), [1520](#), [1611](#), [1617](#)
- fonts and labels [1522](#)
- fonts and system variables [1525](#)
- fonts and titles [1522](#)
- fonts in headings and footings [1522](#), [1524](#)
- fonts in HTML reports [1618](#)
- fonts in HTML Reports [1618](#)
- footing code [1434](#)
- FOOTING command [1449](#), [1530](#), [1813](#)
- FOOTING component [1195](#), [1197](#)
- footing limitations [1433](#)
- footings [1191](#), [1431](#), [1432](#), [1445](#), [1449](#)
- footings for bursted reports [966](#)
- FOR field
  - reusing values [1738](#)
- FOR phrase [69](#), [71](#), [155](#), [169](#), [170](#), [1729](#), [1730](#), [1733](#), [1740](#), [1741](#)
  - reusing values [1738](#)
  - syntax [1900](#)
- FORECAST [316](#), [341](#), [356–358](#)
  - calculating trends [338](#)
  - double exponential smoothing [349](#), [350](#)
  - exponential moving average [338](#), [339](#), [346](#), [347](#)
  - limit [317](#), [342](#)
  - linear regression analysis [338](#), [339](#)
  - FORECAST [316](#), [341](#), [356–358](#)
    - linear regression equation [353](#), [355](#)
    - predicting values [338](#)
    - processing [316](#), [339](#)
    - simple moving average [338](#), [339](#), [343](#), [345](#)
    - triple exponential smoothing [350](#), [352](#)
- format ALPHA [512](#)
- format dates [441](#), [442](#)
- FORMAT DFIX [537](#)
- format DHTML [517](#)
- formatting blank lines [1358](#), [1359](#)
- formatting carriage-returns for text fields [1653](#)
- formatting cells [1782](#), [1785](#), [1787](#)
- formatting cells and labels [1787](#)
- formatting columns [1626–1628](#), [1782](#), [1785](#), [1788](#)
- formatting dates in graphs [1694](#)
- formatting fields [434](#), [444](#)
- formatting financial reports [1368](#), [1369](#)
- formatting footings [1520](#)
- formatting graphs [1672](#), [1673](#), [1685](#)
- formatting heading and footing lines [1575](#)
- formatting headings [1520](#)
- formatting HOLD files [473](#)
- formatting labels [1520](#)
- formatting line-feeds for text fields [1653](#)
- formatting output files [511](#)
- formatting PCHOLD files [509](#)
- formatting reports [1107](#), [1109](#), [1211](#), [1214](#), [1216](#), [1241](#), [1415](#), [1416](#), [1821](#)

formatting reports and SQL Translator [1821](#)  
 formatting reports in FML (Financial Modeling Language) [1368](#), [1369](#)  
 formatting rows [1782](#), [1785–1787](#), [1789](#)  
 formatting rows and labels [1785–1787](#)  
 formatting SAVB files [508](#)  
 formatting SAVE files [506](#)  
 formatting skipped lines [1358](#), [1359](#)  
 formatting text [1791](#)  
 formatting text fields [1565](#), [1653](#)  
 formatting text rows [1790](#)  
 formatting titles [1520](#)  
 FORMULTIPLE parameter [1738](#), [1739](#)  
 free text [1191](#), [1759](#), [1760](#), [1790](#)  
     formatting [1785](#), [1789–1791](#)  
 free-form reports [34](#), [1107](#), [1241](#), [1576](#), [1579](#),  
[1809](#), [1816](#)  
 FREETEXT component [1194](#), [1195](#), [1790](#), [1791](#)  
 freezing HTML headings [1453](#)  
 FROM ... TO operator [243](#), [244](#)  
 FST prefix operator [72](#)  
 full outer join [1038](#)  
 functions [431](#), [1757](#)  
 FYRTHRESH attribute  
     date-time data type and [452](#)

## G

Gantt charts [1681](#)  
 GAPINTERNAL attribute [1277](#)  
 GE operator [245](#), [466](#)  
 generating TABLEF commands [1836](#)  
 GET CHILDREN parameter [1767](#), [1771](#)  
 GGDEMOG data source [1871](#)  
 GGORDER data source [1871](#)  
 GGPRODS data source [1871](#)  
 GGSales data source [1871](#)  
 GGSTORES data source [1871](#)  
 GIF files [522](#)  
 GIF format [522](#)  
 Gotham Grinds data sources [1871](#)  
 grand totals [377](#), [1515](#)  
 GRANDTOTAL component [1174](#), [1175](#), [1185](#),  
[1186](#), [1188](#)  
 GRAPH command [1658](#)  
 GRAPH command compared to TABLE command  
[1659](#)  
 graph footings [1710](#)  
 graph formats [1665](#), [1672](#), [1673](#)  
 graph formats and display commands [1665](#)  
 graph formats and display fields [1665](#)  
 graph formats and sort phrases [1665](#)  
 graph formatting [1216](#)  
 graph formatting BODY element [1222](#)  
 graph formatting in external cascading style  
 sheets [1224](#), [1241](#)  
 graph formatting TD element [1222](#)  
 graph formatting with external cascading style  
 sheets [1216](#)  
 graph headings [1710](#)  
 graph height [1712–1714](#)

- graph SET parameters [1713](#)
- graph styling [1710](#), [1712](#)
- graph titles [1436](#)
- graph types [1663](#)
- graph width [1712](#), [1714](#)
- GRAPHBASE attribute [1416](#)
- GRAPHCOLOR attribute [1150](#), [1415](#), [1416](#)
- graphic elements [809](#), [811](#)
- graphics [809](#), [1481](#)
- graphics in footings [1481](#)
- graphics in headings [1481](#)
- GRAPHLENGTH attribute [1415](#), [1416](#)
- GRAPHNEGColor attribute [1416](#)
- graphs [34](#), [1657](#), [1658](#)
  - displaying multiple graphs in columns [1690](#)
- GRAPHSCALE attribute [1416](#), [1420](#), [1422](#), [1423](#)
- GRAPHSERVURL SET parameter [1722–1724](#)
- GRAPHTYPE attribute [1150](#), [1413](#), [1415](#), [1418](#)
- GRAPHWIDTH attribute [1415](#), [1416](#)
- GRID attribute [1319](#), [1320](#), [1324](#), [1325](#)
- grids [1318](#), [1320](#), [1324](#), [1348](#), [1349](#), [1526](#)
- GRMERGE parameter [1686](#), [1694](#)
- group fields [1026](#)
- group fields and join structures [1026](#)
- group key values [257](#), [258](#)
- grouping numeric data [166–170](#)
- grouping numeric data into tiles [170](#), [171](#),  
[173–175](#)
- grouping sort fields [905](#), [912](#)

- groups of values [1738](#)
  - identifying [1738](#)
- GRWIDTH parameter [1690](#)
- GT operator [245](#), [246](#), [466](#)
- GTREND parameter [1684](#)
- GUTTER attribute [1425](#), [1426](#)

## H

- H data type [452](#)
- HAUTO parameter [1718](#)
- HAXIS parameter [1713](#)
- HEADALIGN attribute [1548](#), [1550](#), [1554](#), [1559](#)
- heading code [1434](#)
- HEADING command [1446](#), [1530](#), [1813](#)
- HEADING component [1195](#), [1197](#), [1198](#)
- heading limitations [1433](#)
- headings [1191](#), [1431](#), [1432](#), [1445](#)
- headings and footings for HTML index pages [966](#)
- headings for bursted reports [966](#)
- headings for graphs [1710](#)
- headings on panels [1484](#)
- HEADPANEL attribute [1484](#)
- helper applications [569](#)
- HGRID attribute [1319](#), [1348](#), [1349](#), [1527](#)
- HIDENULLACRS parameter [113](#)
- hiding columns [1288](#), [1291](#)
- hiding display fields [1287](#)
- hiding fields [1288](#), [1291](#)
- hiding fields in y-axis [1684](#)
- hiding rows [1802](#), [1803](#)

- hiding sort field values [179](#), [925](#)
- hiding y-axis fields [1684](#)
- hierarchical reporting
  - BOTTOM [209](#)
  - BY HIERARCHY [209](#)
  - hierarchical sort [209](#)
  - SHOW [209](#)
  - TOP [209](#)
  - using WHEN [209](#)
- hierarchies [1764](#), [1771](#), [1779](#), [1780](#)
  - displaying [1764](#)
- hierarchy of sort fields [907](#)
- high-order sort fields [1086](#), [1087](#), [1089](#), [1100](#), [1101](#), [1103](#)
- HMAX parameter [1718](#)
- HMIN parameter [1718](#)
- HOLD AT CLIENT command [473](#), [509](#)
- HOLD command [472](#), [473](#)
- HOLD file INTERNAL format [548](#), [549](#)
- HOLD file keys and indexes [505](#)
- HOLD file structured [551](#)
- HOLD file suppressing field padding [546–548](#)
- HOLD file text fields [535](#), [536](#)
- HOLD files [472](#), [473](#), [481](#), [496](#), [497](#), [551](#), [1078](#)
- HOLD files and external sorting [200](#), [201](#)
- HOLD files and merge phrases [1078](#), [1084](#)
- HOLD files and missing values [987](#), [988](#)
- HOLD files for financial reports [1807](#)
- HOLD files text fields [511](#)
- HOLD format ALPHA [512](#)
- HOLD format DATREC [516](#)
- HOLD format DFIX [517](#)
- HOLD format GIF [522](#)
- HOLD format INGRES [524](#)
- HOLD format INTERNAL [524](#), [546–548](#)
- HOLD format JPEG [524](#)
- HOLD format PowerPoint [527](#)
- HOLD format Red Brick [527](#)
- HOLD format SQL\_SCRIPT [528](#)
- HOLD format SQLDBC [528](#)
- HOLD format SQLINF [528](#)
- HOLD format SQLMAC [529](#)
- HOLD format SQLMSS [529](#)
- HOLD format SQLODBC [529](#)
- HOLD format SQLORA [529](#)
- HOLD format SQLPSTGR [530](#)
- HOLD format SQLSYB [530](#)
- HOLD format SYLK [530](#)
- HOLD format TAB [531](#)
- HOLD format TABT [531](#)
- HOLD FORMAT VISDIS [532](#)
- HOLD format WK1 [533](#)
- HOLD format WP [533](#)
- HOLD format XFOCUS [534](#)
- HOLD format
  - XML [535](#)
- HOLD formats [511](#)
- HOLD formats AHTML [512](#)
- HOLD formats AHTMLTAB [512](#)
- HOLD formats APDF [513](#)

HOLD formats EXL97 [522](#)  
HOLD formats FLEX [522](#)  
HOLD formats FOCUS [522](#)  
HOLD Master Files [473](#), [477](#), [484](#), [486–488](#),  
[490](#), [491](#), [495](#), [498](#)  
HOLDATTR command [484](#), [495](#)  
HOLDATTR parameter [495](#)  
HOLDLIST command [484](#), [490–492](#)  
horizontal bar graphs [1413](#)  
horizontal labels [1712](#)  
horizontal waterfall graphs [1681](#)  
host fields [1032](#)  
host files [1008](#), [1016](#)  
HTML alignment [1548](#)  
HTML display formats for reports [571](#)  
HTML format [523](#), [566](#), [568](#), [571](#)  
HTML heading freezing [1453](#)  
HTML index pages [965](#)  
HTML report display formats [571](#)  
HTML reports [87](#), [905](#), [958](#), [1618](#)  
    JavaScript files [1703](#)  
    pop-up field descriptions [89](#)  
    VBScript files [1703](#)  
HTML tables of contents [905](#), [907](#)  
HTML tables of contents for multiple sort groups  
[912](#)  
HTML5-only charts [1682](#)  
HTMLCSS SET parameter [1140](#)  
HTMLFORM command [1228](#)  
HTMTABLE format [523](#), [1228](#)

hyperlinks [958](#), [1239](#)  
hyperlinks in HTML reports [958](#)

## I

identifying across columns in a style sheet [1172](#)  
identifying ACROSS phrase sort data [1183](#)  
identifying across totals in a style sheet [1184](#)  
identifying ACROSS-TOTAL values [1184](#)  
identifying ACROSSVALUE component [1184](#)  
identifying cells [1755](#), [1756](#), [1785](#)  
identifying column report components [1170](#)  
identifying columns [1171](#), [1172](#), [1182](#), [1747](#),  
[1785](#)  
    by address [1750](#), [1751](#)  
    by number [1748](#)  
    by relative address [1751](#)  
    by value [1753](#)  
    contiguous columns [1749](#)  
    in financial reports [1747](#), [1748](#)  
identifying data [1179](#)  
identifying data report components [1180](#), [1181](#)  
identifying embedded fields in report components  
[1202](#), [1203](#)  
identifying footings in a style sheet [1195](#), [1197](#)  
identifying free text in FML reports [1195](#)  
identifying free text in report components [1195](#)  
identifying headings in a style sheet [1195](#), [1197](#),  
[1198](#)  
identifying page numbers in report components  
[1206–1208](#)

- identifying ranges of multiple values [1737](#)
- identifying report components [1167](#), [1168](#)
- identifying report components for totals and subtotals [1174–1176](#), [1185](#), [1186](#), [1188](#)
- identifying rows [1744–1746](#), [1785](#)
  - in financial reports [1745](#), [1746](#)
  - in WebFOCUS StyleSheets [1785–1787](#)
- identifying skipped lines in report components [1206–1208](#)
- identifying sort value report components [1193](#)
- identifying sort values [1191](#), [1193](#)
- identifying subtotal calculations in a style sheet [1190](#)
- identifying subtotals in a style sheet [1188](#), [1189](#)
- identifying text strings in report components [1200](#), [1202](#)
- identifying title report components [1191–1194](#)
- identifying totals in a style sheet [1188](#)
- identifying underlines in report components [1206](#), [1207](#)
- IF command with LIKE or UNLIKE [249](#)
- IF operator [238](#), [240](#)
- IF phrase [220](#), [249](#), [259](#), [260](#), [431](#), [1694](#)
- IF phrase expressions [431](#)
- IF-THEN-ELSE expressions
  - and missing tests [981](#)
- IF/THEN/ELSE statements [468](#)
- IMAGE attribute [809](#), [874](#), [1376](#), [1379–1381](#), [1389](#), [1390](#), [1395](#)
- IMAGEALIGN attribute [1379](#), [1380](#)
- IMAGEBREAK attribute [1379](#), [1380](#)
- images [809](#), [811](#), [1469](#), [1481](#)
- images and WebFOCUS StyleSheets [1380](#)
- images in footings [1469](#), [1481](#)
- images in headings [1469](#), [1481](#)
- improving performance [200](#), [1018](#), [1836](#), [1839](#)
- IN command [1284](#), [1285](#)
- IN-GROUPS-OF option [1691](#)
- IN-GROUPS-OF phrase [166](#), [167](#)
- IN-RANGES-OF phrase [166](#), [168](#)
- INCLUDES operator [256](#), [257](#)
- including JavaScript in an HTML report [572](#)
- indentation
  - specifying between levels [1799](#)
- indenting captions [1770](#)
- independent paths [223](#), [225](#)
- index optimized retrieval [1835](#)
- index pages [965](#)
- index pages for headings and footings [966](#)
- INGRES formats [524](#)
- inheritance in style sheets [1231](#), [1232](#), [1245](#)
- inheritance style sheets [1128](#)
- inheriting attributes [1127](#), [1128](#)
- inline StyleSheets [1118](#)
- inline WebFOCUS StyleSheets [1122](#)
- inner join [1006](#), [1020](#), [1023](#)
- inner join structures [1023](#)
- INSERT command [1836](#)
- INSERT INTO command [1826](#), [1827](#)
- inserting blank lines [1354](#)

inserting page breaks [1156](#)  
inserting skipped lines [1156](#)  
inserting summary lines [1156](#)  
inserting text in financial reports [1759](#), [1760](#)  
inserting underlines [1156](#), [1354](#)  
inserting variables in free text [1760](#), [1761](#)  
inter-row calculations [1743](#)  
internal cascading style sheets (CSS) [1140](#), [1141](#)  
internal cascading style sheets (CSS) and  
StyleSheets [1227](#)  
INTERNAL format [524](#), [546–548](#)  
internal matrixes [304](#), [1752](#)  
internal storage and field formats [442](#)  
interpolating X and Y axis values [1684](#)  
INTERSECT operator [1830](#)  
irrelevant report data [971](#), [972](#)  
IS NOT operator [248](#), [249](#)  
IS operator [248](#), [249](#)  
ISO standard date-time formats [456](#)  
ITEM attribute [1200](#)  
ITEM subtype [1200](#), [1202](#), [1567](#)  
ITEMS data source [1869](#), [1870](#)

## J

JavaScript files [1703](#)  
JavaScript functions [777](#), [778](#), [1708](#)  
JavaScript functions in HTML reports [572](#)  
JavaScript requirements [927](#)  
JavaScript requirements for Accordion Reports  
[927](#)

JavaScript  
    requirements for pop-up field descriptions [89](#)  
JOBFILE data source [1850](#), [1851](#)  
JOBHIST data source; sample data sources  
    JOBHIST [1863](#)  
JOBLIST data source; sample data sources  
    JOBLIST [1863](#)  
JOIN AS\_ROOT [1047](#)  
JOIN CLEAR command [1073](#)  
JOIN command [1007](#), [1012](#), [1013](#), [1016](#), [1018](#),  
[1028](#), [1035](#), [1823–1826](#)  
JOIN command and ALL parameter [1008](#), [1009](#)  
JOIN command and SQL Translator [1823–1826](#)  
join structures [1007](#), [1016](#), [1018](#), [1025](#), [1034](#),  
[1823–1825](#)  
join structures and CHECK FILE command [1070](#),  
[1071](#)  
join structures and DBA security [1008](#)  
join structures and DEFINE command [1027](#),  
[1028](#), [1030](#), [1065](#), [1066](#), [1068](#)  
join structures and group fields [1026](#)  
join structures and numeric data types [1034](#)  
join structures and qualified field names [1825](#)  
join structures and virtual fields [1027](#), [1028](#),  
[1030](#), [1066](#), [1068](#)  
join structures and WHERE phrase [1070](#)  
join types [1006](#)  
join  
    from multi-fact synonym [1056](#)  
    full outer [1038](#)



joining data sources [272](#), [1007](#), [1008](#), [1016](#),  
[1033](#)  
 joining fields [1033](#), [1034](#)  
 joins [1006](#)  
 JPEG format [524](#)  
 JSCHART format [525](#)  
 JSURL parameter [1703](#)  
 JSURL SET parameter [572](#)  
 justification regions [1534](#)  
 JUSTIFY attribute [1530](#), [1531](#), [1538](#), [1578](#), [1648](#)  
 justifying column titles [1537–1542](#)  
 justifying columns [1648](#)  
 justifying data [1633](#), [1649](#)  
 justifying field values [1570](#)  
 justifying footings [1529](#)  
 justifying grand totals [1544](#)  
 justifying headings [1529](#)  
 justifying headings and footings [1529–1533](#),  
[1535](#), [1536](#), [1568](#)  
 justifying labels [1529](#)  
 justifying report columns [1648](#)  
 justifying row totals [1542](#), [1543](#)  
 justifying subtotals [1544](#)  
 justifying titles [1529](#)

## K

KEEPDEFINES parameter [1065](#), [1066](#)  
 KEEPFILTERS SET parameter [271](#), [272](#)  
 key fields [479](#), [481](#)  
 keyed retrieval [496](#), [497](#)

## L

LABEL attribute [1173](#), [1194](#), [1785–1787](#)  
 LABEL subtype [1786](#), [1792](#)  
 LABELPROMPT attribute [1425](#), [1426](#)  
 labels [1424](#), [1426](#), [1427](#), [1431](#), [1513](#), [1744](#)  
     for rows [1745](#), [1746](#)  
     formatting rows [1790](#)  
 lagging values [1010](#)  
 landscape orientation [1253](#)  
 last page number [1306](#)  
 last page number in a sort group [1308](#)  
 LE operator [245](#), [466](#)  
 LEDGER data source [1859](#)  
 left margins [1255](#)  
 left outer join [1006](#), [1023](#)  
 left outer join structures [1023](#)  
 LEFTGAP attribute [1258](#), [1261](#), [1264](#)  
 LEFTMARGIN attribute [1255–1257](#)  
 legacy features [1114](#)  
 legacy report formatting [1114](#)  
 LIKE operator [248](#), [249](#)  
 limit FORECAST [317](#), [342](#)  
 limitations for headings and footings [1433](#)  
 limitations in display fields [59](#)  
 limitations of dynamic tables of contents [925](#)  
 limitations of HTML tables of contents [925](#)  
 limiting data for graphs [1694](#)  
 limiting display fields [59](#)  
 limits column titles [1503](#)  
 limits for display fields [59](#)

- LINE attribute [1199](#)
- line breaks [1653](#)
- line graphs [1663](#), [1666](#), [1667](#), [1673](#)
- LINE subtype [1199](#), [1200](#), [1202](#), [1567](#)
- line termination characters [601](#)
- line-by-line formatting [1575](#)
- line-feeds [1653](#)
- linear regression [315](#), [359](#), [361](#)
- linear regression analysis [316](#), [338](#), [339](#)
- linear regression analysis FORECAST [338](#), [339](#)
- linear regression equation [332](#), [353](#), [355](#)
  - FORECAST\_LINEAR [332](#)
- linear regression in graphs [1684](#)
- linear scales [1664](#), [1665](#)
- LINES SET parameter [1294](#)
- LINK element [1216](#), [1217](#), [1228](#)
- linking from graphics [809](#)
- linking graphic elements [809](#), [811](#)
- linking graphs [1705](#)
- linking images [809](#), [811](#)
- linking report components [763](#), [764](#), [766](#), [769](#),  
[770](#), [777](#), [778](#), [806](#), [809](#), [811](#), [813](#), [814](#), [1706](#),  
[1708](#)
- linking report pages [958](#)
- linking report pages and heading text [961](#)
- linking report pages and images [959](#)
- linking report pages and page numbers [961](#)
- linking report pages in HTML reports [958](#)
- linking reports [905](#), [958](#), [1142](#)
- linking reports with WebFOCUS StyleSheets [958](#)
- linking summary and detail data [890](#), [895](#)
- linking summary and detail data drill through reports [891](#), [892](#)
- linking to a Maintain procedure [779](#), [780](#)
- linking to external cascading style sheets [1217](#),  
[1228](#)
- linking to JavaScript functions [777](#), [778](#), [1708](#)
- linking to Maintain procedures [780](#)
- linking to Uniform Resource Locators (URLs) [769](#),  
[770](#), [1706](#)
- linking to URLs (Uniform Resource Locators) [769](#),  
[770](#), [1706](#)
- linking with conditions [806–808](#)
- links [763](#), [764](#)
- LIST \* command [46](#)
- LIST command [43](#), [45](#), [58](#), [92](#)
- list records [45](#)
- listing join structures [1072](#)
- listing records [45](#), [46](#)
- literals [1830](#)
- LOAD CHART command [1778](#), [1779](#)
- load procedures [1847](#)
- loading a hierarchy into memory [1778](#), [1779](#)
- LOCATOR data source;sample data sources
  - LOCATOR [1864](#)
- logarithmic scales [1664](#), [1665](#)
- logical expression types [465](#)
- logical expressions [237](#), [247](#), [432](#), [465](#)
- logical operator [247](#)
- logical operators [237](#), [238](#), [466](#)

long field names [1903–1905](#)  
 LOOKGRAPH parameter [1672](#), [1673](#)  
 LOTUS format [525](#)  
 LST prefix operator [72](#)  
 LT operator [245](#), [246](#), [466](#)

## M

MACRO attribute [1125](#)  
 macros [1114](#), [1124–1126](#)  
 macros style sheets [1124](#), [1125](#)  
 mailing labels [1424–1427](#)  
 Maintain procedures [779](#), [780](#)  
 maintaining across joins [272](#)  
 maintaining filters [272](#)  
 maintaining filters across joins [271](#)  
 margins [1255](#), [1256](#)  
 masked fields [248](#), [250](#)  
 masking characters [248](#), [250](#), [1738](#)  
     retrieving multiple values with [1738](#)  
     retrieving values with [1738](#)  
 masks [248](#), [249](#)  
 Master Files [33](#), [246](#), [265](#), [1766](#), [1847](#)  
     for financial reports [1766](#)  
     for FML hierarchies [1766](#), [1778](#), [1779](#)  
     for hierarchies [1766](#)  
     hierarchies in [1778](#), [1779](#)  
 MATCH command [1076](#), [1078–1080](#), [1507](#), [1899](#)  
 MATCH FILE command [1076](#), [1078](#), [1080](#), [1084](#)  
 MATCH FILE command and concatenated data  
 sources [1098](#)

MATCH FILE command and display commands  
[1090](#), [1091](#)  
 MATCH FILE command and merge phrases [1078](#),  
[1084](#)  
 MATCH FILE commands [1078](#)  
 MATCHCOLUMNORDER parameter [1078](#)  
 matrix reports [143](#), [144](#), [369](#), [372](#)  
 matrix type reports [143](#)  
 MATRIXORDER attribute [1425](#), [1426](#)  
 MAX prefix operator [65](#)  
 MAX prefix operators [65](#)  
 maximum prefix operators [65](#)  
 MDE prefix operator [65](#)  
 MDN prefix operator [65](#)  
 measurement units [1255](#)  
 measuring fonts [1578](#)  
 measuring for column width alignment [1578](#)  
 measuring for decimal alignment [1578](#)  
 merge phrases [1078](#), [1084](#)  
 merge phrases and HOLD files [1078](#), [1084](#)  
 merge phrases and MATCH FILE command [1078](#),  
[1084](#)  
 merging data sources [1076](#), [1078–1080](#), [1084](#),  
[1086](#), [1087](#), [1089–1091](#), [1098](#), [1100](#), [1101](#),  
[1103](#)  
 merging data sources and display commands  
[1090](#), [1091](#)  
 merging data sources and PRINT command [1090](#),  
[1091](#)  
 merging data sources and SUM command [1091](#)

- merging multiple graphs [1686](#)
- merging multiple OLAP graphs [1688](#)
- metrics file [583](#), [585](#), [587](#)
- MIME types [561](#), [563](#), [569](#)
- MIN prefix operator [65](#)
- MIN prefix operators [65](#)
- minimum prefix operators [65](#)
- MISSING attribute [246](#), [972](#), [985–987](#)
- MISSING attribute and extract files [988](#)
- MISSING attribute and Master Files [974](#), [975](#)
- MISSING attribute and virtual fields [975](#)
- MISSING attribute limits [975](#)
- missing descendants [995–997](#)
- missing instances [972](#)
- missing value data sources [971](#)
- missing values [246](#), [971–973](#), [995](#), [996](#), [1631](#), [1695](#)
- missing values and ALL parameter [995–997](#)
- missing values and ALL prefix [995](#)
- missing values and DEFINE command [974–978](#)
- missing values and extract files [988](#)
- missing values and segment instances [972](#), [992–994](#)
- missing values and temporary fields [976](#)
- missing values for reformatted fields [990](#), [1629](#)
- missing
  - in IF-THEN-ELSE expressions [981](#)
- MORE phrase [1093–1095](#), [1098](#)
- MORE phrase and universal concatenation [1094](#), [1095](#)
- MOVIES data source [1869](#)
- multi-fact synonym
  - and join [1047](#), [1056](#)
- multi-pane reports [1424](#), [1429](#)
- multi-path data sources [49–51](#), [54](#), [291](#)
- multi-segment data sources [256](#), [257](#)
- multi-segment files [256](#)
- multi-table HTML reports [1293](#), [1294](#), [1297](#)
- multi-verb requests [182](#)
- MULTILINES command [379](#), [385](#), [1515](#), [1518](#)
- multipath join structures [1023](#)
- MULTIPATH parameter [223–225](#), [228](#)
- multiple display commands [182](#)
- multiple display commands and ROW-TOTAL [373](#)
- multiple drill-down links and WHEN phrase [796](#)
- multiple drill-down links conditional styling [796](#)
- multiple drill-down reports [1709](#)
- multiple drill-downs [1709](#)
- multiple graphs [1685](#)
- multiple parameters [804](#)
- multiple records [1735](#)
- multiple sort fields [94](#), [95](#), [110](#), [182](#)
- multiple values [1735](#)
- multiple verbs [182](#)
- multiple virtual fields [287](#)
- multiple web pages [1293](#)
- multiple WHERE phrases [222](#)
- multiple Y-axis graphs [1681](#)
- multiple-line footings [1451](#)
- multiplication operator [434](#), [437](#)

multivariate REGRESS [359–361](#)

## N

naming extract files [472](#)

naming output files [472](#)

naming StyleSheet files [1121](#)

naming WebFOCUS StyleSheet files [1121](#)

National Language Support (NLS) [190](#)

native-mode arithmetic [438](#)

navigating between reports [924](#)

NE operator [246](#), [247](#), [466](#)

NLS (National Language Support) [190](#)

NOBREAK phrase [871](#)

NODATA character [971](#), [972](#), [1002](#), [1003](#), [1631](#),  
[1632](#)

non-numeric fields [54](#), [55](#)

non-recursive models [1763](#)

non-unique join structures [1006](#), [1008](#), [1009](#)

NOPRINT command [179](#), [1288](#), [1291](#), [1684](#),  
[1802](#)

NOSPLIT command [1299](#), [1300](#)

NOT FROM ... TO operator [243](#), [244](#)

NOT LIKE operator [248](#)

NOT operator [466](#)

NOTOTAL command [427](#), [428](#)

null values [1631](#)

numeric constants [1830](#)

numeric data [166](#), [170](#)

numeric data types [1033](#)

numeric data types and join structures [1034](#)

numeric expressions [432](#), [437–439](#)

evaluating [438](#)

numeric fields [54](#)

numeric functions [436](#)

numeric operator expressions [434](#), [437](#)

## O

OBJECT attribute [1200](#)

OBJECT subtype [1567](#), [1573](#)

OLAPGRMERGE parameter [1689](#)

OMITS operator [247](#)

ON phrase [1156](#)

ON TABLE SET command [1231](#)

on-demand paging [955](#), [957](#)

ONFIELD SET parameter [1157](#)

ONLINE-FMT parameter [568](#)

operand formats [439](#)

operators [434](#), [465](#)

operators prefix [61](#)

optimized join structures [1835](#)

optimizing join structures [1835](#)

optimizing sorting data [190](#)

OR operator [237](#), [466](#), [1736](#)

order of evaluation [437](#), [438](#)

ORIENTATION attribute [1250](#), [1253](#), [1254](#)

outer join [1006](#), [1052](#)

output file AHTML format [512](#)

output file AHTMLTAB format [512](#)

output file format [512](#)

output file format DATREC [516](#)

- output file format DFIX [517](#)
- output file format GIF [522](#)
- output file format HTML [523](#)
- output file format HTMTABLE [523](#)
- output file format INGRES [524](#)
- output file format INTERNAL [524](#)
- output file format JPEG [524](#)
- output file format JSCHART [525](#)
- output file format LOTUS [525](#)
- output file format PDF [525](#)
- output file format PDF OPEN/CLOSE [526](#)
- output file format PostScript (PS) [527](#)
- output file format PPT [527](#)
- output file format Red Brick [527](#)
- output file format SQL\_SCRIPT [528](#)
- output file format SQLDBC [528](#)
- output file format SQLINF [528](#)
- output file format SQLMAC [529](#)
- output file format SQLMSS [529](#)
- output file format SQLODBC [529](#)
- output file format SQLORA [529](#)
- output file format SQLPSTGR [530](#)
- output file format SQLSYB [530](#)
- output file format SYLK [530](#)
- output file format TAB [531](#)
- output file format TABT [531](#)
- output file format WK1 [533](#)
- output file format WP [533](#)
- output file format XFOCUS [534](#)

- output file formats [511](#), [521](#)
  - XML [535](#)
- output file text fields [535](#), [536](#)
- output files [472](#), [500](#)
- output files and missing values [988](#)
- output files text fields [511](#)
- output format VISDIS [532](#)
- output formats [511](#)
- OVER and column alignment [1277](#)
- OVER command [1265](#), [1275](#)
- overriding attribute inheritance [1130](#)
- overriding macros [1126](#)

## P

- padded fields [546–548](#)
- PAGE BREAK command [1784](#)
- page breaks [1293](#), [1295](#)
- page breaks in financial reports [1784](#)
- page colors [1249](#), [1254](#)
- page count [1306](#), [1308](#)
- page footings [1445](#), [1449](#), [1451](#)
- page headings [1445](#), [1446](#)
- page layout [1249](#)
- page margins [1255](#), [1256](#)
- page numbers [1206–1208](#), [1304](#), [1306](#), [1308](#),  
[1311](#), [1313](#)
- page numbers in footings [1469](#)
- page numbers in headings [1469](#)
- page numbers in headings and footings [1479](#)
- page orientation [1249](#), [1253](#), [1254](#)

- page size [1249](#), [1250](#)
- PAGE-BREAK command [1295](#), [1605](#)
- PAGE-NUM SET parameter [1304](#), [1305](#), [1313](#)
- PAGECOLOR attribute [1250](#), [1254](#)
- PAGEMATRIX attribute [1425](#), [1426](#)
- PAGENUM component [1206–1208](#)
- PAGESIZE attribute [1250](#)
- PAGESIZE option [581](#), [582](#)
- paginating a report [1293](#)
- panels, repeating headings [1484](#)
- paper size settings [581](#), [582](#)
- parameters [797](#), [799](#), [801](#), [802](#), [804](#)
- parent instances [995–997](#)
- parent segments in qualified field values [256](#)
- Pareto graphs [1681](#)
- PCHOLD AHTML format [512](#)
- PCHOLD command [472](#), [509](#), [566](#), [602](#), [908](#)
- PCHOLD command and PDF format [872](#)
- PCHOLD files [509](#)
- PCHOLD format [512](#)
- PCHOLD format DFIX [517](#)
- PCHOLD format HTML [523](#)
- PCHOLD format HTMTABLE [523](#)
- PCHOLD format JSCHART [525](#)
- PCHOLD format LOTUS [525](#)
- PCHOLD format PDF [525](#)
- PCHOLD format PDF OPEN/CLOSE [526](#)
- PCHOLD format PS (PostScript) [527](#)
- PCHOLD format TAB [531](#)
- PCHOLD format TABT [531](#)
- PCHOLD format WK1 [533](#)
- PCHOLD format WP [533](#)
- PCHOLD formats [511](#), [521](#), [897](#)
- PCHOLD formats APDF [513](#)
- PCHOLD formats FLEX [522](#)
- PCHOLD formats in PDF [895](#)
- PCHOLD formats PDF [900](#)
- PCHOLD formats PDF OPEN/CLOSE [900](#)
- PCSEND command [965](#)
- PCT percent [66](#)
- PCT prefix operators [66](#)
- PCT.CNT prefix operator [68](#)
- PDF (Portable Document Format) [525](#)
- PDF compound reports [872](#)
- PDF display format [566](#), [568](#), [575](#)
- PDF display format for compound reports [817](#), [870](#)
- PDF format [566](#), [574](#)
- PDF format on UNIX [600](#)
- PDF OPEN/CLOSE and PCHOLD formats [897](#)
- PDF OPEN/CLOSE format [526](#)
- PDF report display formats [566](#), [568](#), [575](#)
- PDFLINETERM parameter [600–602](#)
- percent (PCT) [66](#)
- percentiles [170](#)
- performance [190](#), [1836](#), [1839](#)
- performing calculations on dates [443](#)
- PERSINFO data source; sample data sources
  - PERSINFO [1865](#)
- PICTURE RETRIEVE command [49](#)

- pie graph [1670](#)
- pie graphs [1663](#), [1671](#), [1675](#)
- placing footings on a separate page [1605](#), [1607](#), [1608](#)
- placing headings on a separate page [1605](#), [1607](#), [1608](#)
- plotting dates [1691](#), [1692](#)
- plotting dates in graphs [1691](#), [1692](#)
- PLUS OTHERS phrase [95](#)
- polar charts [1680](#)
- pop-up field descriptions [87](#)
  - JavaScript requirements [89](#)
  - ReportCaster and [89](#)
- portrait orientation [1253](#)
- POSITION attribute [1258](#), [1259](#), [1261](#), [1379](#), [1390](#), [1393–1395](#), [1584](#), [1585](#)
- positional column referenced calculated values [302](#)
- positional field references for COMPUTE command [302](#)
- positional labels [1744–1746](#)
- positional referencing for columns [302](#)
- positioning columns [1258](#), [1259](#), [1264](#), [1283–1285](#)
- positioning columns for WebFOCUS StyleSheets [1648](#)
- positioning footings [1584](#), [1602–1604](#)
- positioning headings [1584](#), [1601](#)
- positioning headings and footings [1584–1587](#), [1604](#)
- positioning report components [1258](#)
- positioning reports for headers and footers [1648](#)
- positioning with spot markers [1593–1595](#)
- POST command [1805](#)
- posted data
  - retrieving [1805](#), [1806](#)
- posting data [1804](#), [1805](#)
  - in financial reports [1804](#)
- posting financial data [1804](#)
- PostScript (PS) format [527](#), [574](#)
- PostScript (PS) reports [581](#), [582](#)
- PostScript display format [566](#), [568](#), [580](#)
- PostScript display format for compound reports [817](#), [870](#)
- PostScript display formats for reports [581](#), [582](#)
- PostScript fonts [583](#), [585](#)
- PostScript fonts in UNIX [585](#)
- PostScript fonts in Windows [585](#)
- PostScript fonts in z/OS [587](#)
- PostScript format [566](#)
- PostScript formats [583](#)
- PostScript report display formats [568](#), [580–582](#)
- PostScript Type1 fonts [583](#)
- PowerPoint format [527](#)
- PPT format [527](#)
- precision [58](#)
- predicting values [359](#)
- prefix operators [60](#), [61](#), [391–393](#), [405](#), [407](#), [413](#), [415](#), [416](#), [1470](#)
  - MDE [65](#)



prefix operators [60](#), [61](#), [391–393](#), [405](#), [407](#), [413](#),  
[415](#), [416](#), [1470](#)

MDN [65](#)

preserving field names [484](#)

preserving missing values [987](#), [988](#), [992](#)

preserving virtual fields [292](#)

preventing breaks [1299](#)

preventing page breaks [1299](#)

PRINT \* command [46](#)

PRINT command [43](#), [45–47](#), [92](#), [1090](#), [1091](#)

PRINT command and merging data sources [1090](#),  
[1091](#)

PRINT command unique segments [54](#)

print display formats [574](#)

PRINT OFFLINE parameter [1727](#)

printing graphs [1727](#)

printing labels [1424](#), [1425](#), [1427](#)

printing multi-pane reports [1429](#)

PRINTONLY parameter [490](#), [492](#)

PRINTPLUS parameter [1591](#)

procedures sorting data [190](#)

producing a direct percent of a count [68](#)

product position graphs [1681](#)

protecting virtual fields [292](#)

PS (PostScript) format [574](#)

## Q

qualified field names [1825](#), [1829](#), [1830](#),  
[1903–1905](#)

qualified field names and SQL join structures  
[1825](#)

qualified field names and SQL Translator [1829](#),  
[1830](#)

qualified field values [141](#), [256](#)

QUALTITLES command [1507](#)

query ? STAT command [191](#)

query commands

? DEFINE [288](#)

?F [1906](#)

?FF [1906](#)

querying HOLD files [473](#), [477](#)

querying sort types [191](#)

QUIT command [38](#)

quotation marks [459](#)

quote-delimited string [459](#), [460](#)

## R

radar graphs [1663](#), [1680](#)

range of records

combining [1737](#)

range of values [1737](#)

combining [1737](#)

range tests [243–246](#)

ranges [166–170](#)

specifying in financial reports [1737](#)

RANKED BY phrase [160](#), [161](#)

RANKED BY TOTAL phrase [159](#), [178](#)

ranking columns [178](#)

ranking sort field values [76](#), [159–161](#), [175](#), [176](#)

reading selection values from a file [260–262](#)

reading values from a file [262](#), [263](#)

READLIMIT operator [258](#)

READLIMIT relational operator [258](#)

RECAP and sort footings [1467](#)

RECAP command [423–426](#), [431](#), [1729](#), [1730](#),  
[1743](#), [1744](#)

    and FML reports [1743](#)

RECAP component [1174](#), [1185](#), [1186](#), [1190](#)

RECAP expressions [1744](#)

    creating [1743](#)

RECAP rows [1743](#)

    formatting [1792](#)

RECOMPUTE command [385](#), [387](#), [391](#), [392](#), [405](#),  
[409–411](#), [413–416](#)

RECOMPUTE command and propagation to grand  
total [399](#)

RECOMPUTE prefix operators [415](#)

RECORDLIMIT operator [258](#), [259](#)

RECORDLIMIT relational operator [258](#), [259](#)

records [45](#), [220](#), [1735](#)

    in multiple rows [1738](#), [1739](#)

    reusing [1738](#), [1739](#)

recursive join structures [1012–1014](#), [1825](#)

recursive models [1763](#), [1764](#)

recursive structures [1012](#), [1013](#)

Red Brick format [527](#)

REDBRICK format [527](#)

redefining formats for fields [1627](#)

reducing report width [1273](#)

ref\_regress\_usage [360](#)

reformatting fields [293–295](#), [1650](#), [1651](#)

REGION data source [1860](#)

REGRESS method [359](#)

relational expressions [237](#), [465](#), [467](#)

relational operator [248](#)

relational operators [238](#), [240](#), [243](#), [245–248](#),  
[250](#), [465](#)

relative column addresses [1751](#), [1752](#)

relative point sizes and HTML fonts [1612](#)

relative starting positions [1589](#)

removing grids [1325](#)

renaming column titles [1501](#), [1503](#), [1504](#)

renaming column totals [369](#), [1513](#)

renaming HOLD files AS phrase [473](#)

renaming PCHOLD files [509](#)

renaming row totals [369](#), [1513](#)

REPAGE command [1295](#), [1304](#), [1305](#)

repeating fields [1012](#)

repeating fields in join structures [1012](#)

repeating rows [1747](#)

report columns [176](#)

REPORT component [1168](#), [1169](#)

report components [764](#), [766](#), [769](#), [770](#), [777](#),  
[778](#), [806](#), [809](#), [811](#), [813](#), [814](#), [1122](#), [1167](#),  
[1216](#), [1611](#), [1706](#), [1708](#)

report components, column [1167](#)

report components, entire report [1167](#)

report components, row [1167](#)

report display EXL2K formats [568](#)

- report display formats [565](#), [566](#), [568](#), [569](#)
- report display PDF formats [574](#)
- report display PostScript formats [574](#)
- report footings [1438](#), [1443](#)
- report formatting [1107](#), [1109](#), [1114](#)
- report formatting in external cascading style sheets [1224](#)
- report formatting methods [1111](#)
- report formatting with external cascading style sheets [1211](#), [1214](#), [1216](#)
- report formatting, inheritance in style sheets [1231](#)
- report headings [1438](#), [1439](#)
- report navigating [1115](#)
- report output formats [1234](#), [1236](#), [1238](#)
- report pagination [1293](#)
- report requests [1817](#)
- report requests and SQL statements [1817](#)
- report styling [1211](#)
- report styling in external cascading style sheets [1224](#)
- report styling with external cascading style sheets [1211](#), [1214](#), [1216](#)
- report SUM columns [176](#)
- report titles [1436](#), [1437](#)
- reporting against hierarchies [1764](#), [1767](#), [1769–1771](#), [1780](#)
- reporting commands [1895](#)
- reporting options [1895](#)
- reports [33](#), [36](#), [38](#), [764](#), [905](#), [965](#), [1611](#), [1617](#), [1903](#)
  - creating [33](#), [34](#), [39](#)
  - creating requests [36](#)
  - customizing [36](#), [41](#)
  - displaying [42](#)
  - displaying data [36](#)
  - financial [34](#)
  - free-form [34](#)
  - output [36](#)
  - printing [42](#)
  - requests [38](#)
  - running [38](#)
  - saving [42](#)
  - selecting data [36](#)
  - sorting data [36](#)
  - specifying fields [36](#)
  - types [34](#)
- requirements for external sorting [190](#)
- reserved words [1820](#)
- restricting sort field values [159](#), [161](#), [175](#), [176](#)
- restrictions for distinct prefix operators [69](#), [71](#)
- restrictions for DST prefix operators [71](#)
- restructuring data [1841](#)
- retrieval data [223](#), [225](#)
- retrieval limits [258](#), [259](#)
- retrieval logic [1839](#)
- retrieval order [198](#), [199](#)
- retrieving data [223](#), [225](#)
- retrieving HOLD Master Files [498](#)

retrieving records [72](#), [258](#), [259](#), [1018](#)  
returned fields [444](#)  
reusing output reports [471](#)  
reusing report output [471](#)  
right margins [1255](#)  
RIGHTGAP attribute [1258](#), [1261](#)  
RIGHTMARGIN attribute [1255](#), [1256](#)  
RNK. prefix operator [76](#)  
rotating data sources [1839](#)  
rounding numeric values [434](#)  
row formatting [1782](#), [1785](#)  
row labels [1744](#), [1746](#)  
row percent (RPCT) [66](#)  
row titles [1194](#), [1781](#), [1782](#)  
    PICKUP rows [1806](#)  
    RECAP rows [1782](#)  
    TAG rows [1781](#)  
row total labels [1513](#)  
row totals [369](#), [372](#), [373](#), [376](#), [377](#), [1513](#)  
ROW-TOTAL phrase [369–371](#)  
ROW-TOTAL with ACROSS and multiple display  
    commands [373](#)  
rows [1173](#), [1174](#)  
    retrieving values for [1733](#)  
ROWTOTAL attribute [1176](#)  
RPCT prefix operator [66](#)  
RPCT row percent [66](#)  
rules in external cascading style sheets [1217](#)  
RUN command [38](#)

## S

SALES data source [1853–1855](#)  
SALHIST data source; sample data sources  
    SALHIST [1866](#)  
SAME DB [500](#)  
SAME\_DB extract files [500](#), [501](#), [503](#), [505](#)  
SAME\_DB HOLD files [500](#), [501](#), [503](#), [505](#)  
SAME\_DB HOLD format [500](#), [501](#), [503](#)  
SAME\_DB HOLD format columns [505](#)  
SAME\_DB output files [500](#), [501](#), [503](#), [505](#)  
sample data sources [1847](#)  
    CAR [1855](#), [1857](#)  
    Century Corp [1877](#)  
    COURSE [1862](#), [1863](#)  
    EDUCFILE [1852](#), [1853](#)  
    EMPLOYEE [1847](#), [1849](#), [1850](#)  
    FINANCE [1859](#), [1860](#)  
    Gotham Grinds [1871](#)  
    ITEMS [1869](#), [1870](#)  
    JOBFILE [1850](#), [1851](#)  
    LEDGER [1859](#)  
    MOVIES [1869](#)  
    REGION [1860](#)  
    SALES [1853–1855](#)  
    TRAINING [1861](#), [1862](#)  
    VIDEOTR2 [1870](#), [1871](#)  
    VideoTrk [1866–1868](#)  
SAVB command [506](#)  
SAVB files [506](#)  
SAVE AHTML format [512](#)

- SAVE AHTMLTAB format [512](#)
- SAVE command [472](#), [506](#)
- SAVE files [506](#)
- SAVE format [512](#)
- SAVE format EXL2K [521](#)
- SAVE format HTML [523](#)
- SAVE format HTMTABLE [523](#)
- SAVE format LOTUS [525](#)
- SAVE format PDF [525](#)
- SAVE format SYLK [530](#)
- SAVE format TAB [531](#)
- SAVE format TABT [531](#)
- SAVE format WP [533](#)
- SAVE formats [511](#), [521](#)
- saving drill-down reports with HTMTABLE [1709](#)
- saving graphs as GIF files [1722](#), [1724](#)
- saving graphs as GIF files using SET  
GRAPHSEVURL [1722](#)
- saving HOLD Master Files [498](#), [499](#)
- saving intermediate report results [1804](#)
- saving output files [472](#)
- saving report output [471](#), [472](#)
- saving reports [471](#), [472](#)
- saving rows [1804](#), [1805](#)
- saving virtual fields [1064–1066](#), [1068](#)
- scalar functions [1830](#)
- scale graphs [1664](#), [1665](#)
- scales [1664](#), [1665](#)
- scaling [1420](#), [1422](#)
- scaling and vertical bar graphs [1423](#)
- scatter graphs [1663](#), [1671](#), [1672](#), [1676](#)
- screening conditions [265](#)
- screening segments [1070](#)
- screening values [304](#)
- scrollable area for HTML output [1453](#)
- search option of WebFOCUS Viewer [958](#)
- SEG. operator [1906](#)
- segment instances [971](#), [995–997](#)
- segment instances and missing values [992–994](#)
- segment locations [290](#)
- segment types [72](#), [73](#)
- segments [223](#), [228](#), [972](#), [1905](#), [1906](#)
- SEGTYPE parameter [496](#)
- selecting graph types [1663](#)
- selecting paper size [581](#), [582](#)
- selecting paper size for PS (PostScript) format  
[574](#)
- selecting paper size for style sheets [582](#)
- selecting paper size PostScript (PS) format [581](#)
- selecting PostScript (PS) format paper size [582](#)
- selecting records [219](#), [220](#), [223](#), [225](#), [228–230](#),  
[237](#), [238](#), [240](#), [243](#), [247](#), [248](#), [256–264](#), [274](#),  
[1816](#)
- selecting records with IF phrase [261](#), [262](#), [264](#)
- selecting records with VSAM [274](#)
- selecting sort procedures [191](#)
- selecting sort types [191](#)
- selecting style sheets [1113](#)
- selecting values using WHERE phrase [261](#)
- selecting values with IF phrase [264](#)

selection criteria [219–223](#), [225](#), [238](#), [240](#), [243](#),  
[260–264](#), [429](#)

selection values [263](#)

selection values with IF phrase [264](#)

sending graphs directly to a printer [1727](#)

SEQUENCE attribute [1272](#)

sequential conditional formatting [1143](#)

SET ACROSSPRT [110](#)

SET ALL parameter [995–997](#)

SET ASNAMES parameter [484](#)

SET AUTOINDEX parameter [1842](#)

SET AUTOPATH parameter [1842](#)

SET BLANKINDENT parameter [1799](#)

SET BYTOC parameter [907](#)

SET CDN parameter [1631](#)

SET CNOTATION parameter [304](#), [305](#), [1752](#),  
[1753](#)

SET commands [907](#), [908](#)

SET COMPMISS parameter [990](#), [1629](#)

SET COMPOUND parameter [871](#)

SET COMPUTE = NEW command [1846](#)

SET COUNTWIDTH parameter [58](#)

SET CSSURL command [1231](#)

SET DATEFORMAT parameter [452](#)

SET DEFINES command [1846](#)

SET DUPLICATECOL command [183](#)

SET EMPTYREPORT parameter [1163](#)

SET EXPANDABLE parameter [925](#), [954](#)

SET EXTSORT parameter [190](#)

SET FILECOMPRESS command [576](#)

SET FILTER parameter [265](#), [268](#)

SET FOCFIRSTPAGE parameter [1304](#)

SET FORMULTIPLE parameter [1738](#), [1739](#)

SET GRAPHSEVURL parameter [1722–1724](#)

SET GRMERGE parameter [1686](#), [1694](#)

SET GRWIDTH parameter [1690](#)

SET GTREND parameter [1684](#)

SET HAUTO parameter [1718](#)

SET HAXIS parameter [1713](#)

SET HMAX parameter [1718](#)

SET HMIN parameter [1718](#)

SET HOLDATTR parameter [484](#), [495](#)

SET HOLDLIST parameter [484](#), [490–492](#)

SET HOLDMISS parameter [987](#)

SET HTMLCSS parameter [1140](#)

SET JSURL parameter [572](#)

SET KEEPFILTER parameter [271](#), [272](#)

SET LINES parameter [1294](#)

SET LOOKGRAPH parameter [1672](#), [1673](#)

SET NODATA command [1631](#)

SET NODATA parameter [1631](#)

SET OLAPGRMERGE parameter [1689](#)

SET ONLINE-FMT parameter [568](#)

SET PAGE-NUM parameter [1304](#)

SET parameter ESTLINES [192](#)

SET parameter ESTRECORDS [192](#)

SET parameter EXTAGGR [197](#)

SET parameter EXTHOLD [201](#)

SET parameter NULL=ON [532](#)

SET parameters [569](#), [600](#), [1118](#), [1800](#), [1842](#)

ACRSVRBTITL [100](#)

DBAJJOIN [1061](#)

DROPBLNKLINE [1370](#)

DUPLICATECOL [183](#)

FIELDNAME [1904](#)

HIDENULLACRS [113](#)

JSURL [1703](#)

MATCHCOLUMNORDER [1078](#)

SET PDFLINETERM parameter [600–602](#)

SET PRINTPLUS parameter [1591](#)

SET PSPAGESETUP parameter [581](#), [582](#)

SET QUALITIES parameter [1507](#)

SET SHOWBLANKS parameter [569](#)

SET SPACES parameter [1265](#)

SET SQLTOPTTF parameter [1836](#)

SET STYLE \* parameter [1718](#), [1719](#)

SET STYLEMODE parameter [1294](#)

SET STYLESHEET parameter [1120](#), [1165](#)

SET SUMMARYLINES parameter [399](#)

SET SUMMARYLINES parameter [400](#), [410](#)

SET UNITS parameter [1141](#)

SET VAXIS parameter [1713](#)

SET VMAX parameter [1719](#)

SET VMIN parameter [1719](#)

SET WEBVIEWER parameter [1293](#)

SET WPMINWIDTH [534](#)

setting conditions for linking reports [1142](#)

setting fixed scales [1712](#), [1718](#), [1719](#)

setting graph height [1713](#)

setting graph height and width [1712](#)

setting page colors [1254](#)

setting paper size for PostScript (PS) reports [582](#)

setting retrieval order [199](#)

setting x-axis fixed scales [1712](#), [1718](#)

setting y-axis fixed scales [1712](#), [1719](#)

sheet names [1436](#)

SHOW for SAP BW [209](#)

SHOWBLANKS SET parameter [569](#)

simple moving average [316](#), [317](#), [338](#), [339](#), [343](#), [345](#)

FORECAST\_MOVAVE [317](#)

single-line footings [1444](#)

single-line headings [1440](#)

SIZE attribute [1379](#), [1390](#), [1393–1395](#)

SKIP-LINE option [1354](#), [1356](#), [1357](#)

SKIPLINE attribute [1358](#), [1359](#)

SKIPLINE component [1206–1208](#)

skipped lines [1206–1208](#), [1356](#), [1357](#)

sort field values [175](#)

sort fields [91](#), [905](#), [908](#), [1740](#)

sort fields for multi-path data sources [93](#)

sort fields using multi-path data sources [99](#)

sort footing limitations [1433](#)

sort footings [1432](#), [1456](#), [1461](#), [1464](#)

sort footings and RECAP [1467](#)

sort footings omitting a display command [1469](#)

sort heading limitations [1433](#)

sort headings [1432](#), [1456](#), [1457](#), [1459](#), [1576](#)

sort multiple fields [94](#), [110](#)

- sort order [94](#), [110](#), [166–170](#)
- sort phrases [45](#), [1665](#)
- sort phrases and graph format [1665](#)
- sort sequence [93](#), [99](#), [1740](#)
- sort temporary fields [93](#), [98](#)
- sort values [92](#), [93](#), [166](#)
- sorted by calculated values [178](#)
- sorting a hierarchy [209](#)
- sorting alphabetically [1293](#)
- sorting by calculated values [176–178](#)
- sorting by columns [99](#), [100](#), [110](#), [178](#)
- sorting by rows [93](#)
- sorting columns [177](#)
- sorting columns by [98](#)
- sorting data [95](#), [190](#), [191](#)
- sorting data by columns [99](#), [100](#)
- sorting data by multiple fields [94](#), [110](#)
- sorting data by rows [92–95](#)
- sorting report columns [176](#), [177](#)
- sorting reports [91](#), [92](#), [1816](#)
- sorting rows by [92–95](#)
- sorting with COMPUTE command [178](#)
- SORTWORK files [192](#)
- SPACES SET parameter [1265](#), [1271](#)
- spacing between columns [1264](#), [1270](#), [1271](#)
- specifying date-time values [452](#)
- specifying fields [36](#)
- specifying fonts for reports [1617](#)
- specifying sort order [152–157](#)
- specifying Uniform Resource Locators (URLs) [813](#)
- specifying URLs [1381](#)
- specifying URLs (Uniform Resource Locators) [813](#)
- spectral charts [1681](#)
- spot markers [1202](#), [1434](#)
- SQL join structures [1823–1825](#)
- SQL join structures and qualified field names [1825](#)
- SQL SELECT statement [1822](#)
- SQL statements [1817–1819](#)
- SQL statements and FOCUS TABLE requests [1817](#)
- SQL Translation Services [1817–1819](#)
- SQL Translator [1817](#)
- SQL Translator and aliases [1825](#)
- SQL Translator and Cartesian product answer sets [1829](#)
- SQL Translator and Continental Decimal Notation (CDN) [1829](#)
- SQL Translator and CREATE TABLE command [1826](#), [1827](#)
- SQL Translator and CREATE VIEW command [1827](#), [1828](#)
- SQL Translator and date formats [1830](#), [1831](#)
- SQL Translator and date-time values [1832–1834](#)
- SQL Translator and DELETE command [1836](#)
- SQL Translator and DROP VIEW command [1827](#), [1828](#)
- SQL Translator and expressions [1830](#)
- SQL Translator and field names [1829](#), [1830](#)
- SQL Translator and index optimized retrieval [1835](#)



- SQL Translator and INSERT command [1836](#)
- SQL Translator and INSERT INTO command [1826](#), [1827](#)
- SQL Translator and JOIN command [1823–1826](#)
- SQL Translator and join structures [1835](#)
- SQL Translator and reserved words [1820](#)
- SQL Translator and SQLTOPTTF parameter [1836](#)
- SQL Translator and time and timestamp fields [1830](#), [1831](#)
- SQL Translator and UPDATE command [1836](#)
- SQL Translator commands [1820](#), [1822](#)
- SQL Translator commands and formatting commands [1821](#)
- SQL\_SCRIPT format [528](#)
- SQLDBC format [528](#)
- SQLINF format [528](#)
- SQLMAC formats [529](#)
- SQLMSS formats [529](#)
- SQLODBC formats [529](#)
- SQLORA formats [529](#)
- SQLPSTGR formats [530](#)
- SQLSYB formats [530](#)
- SQLTOPTTF parameter [1836](#)
- SQUEEZE attribute [1266–1268](#)
- ST. prefix operator [1470](#)
- stacking columns [1264](#), [1273](#)
- stacking columns with FOLD-LINE [1274](#)
- stacking columns with OVER [1275](#)
- STAT query [191](#)
- stock charts [1678](#)
- storing StyleSheet files [1121](#)
- storing WebFOCUS StyleSheet files [1121](#)
- structure diagrams [1847](#)
- structured HOLD files [551](#)
- STYLE \* parameter [1718](#), [1719](#)
- STYLE attribute [1363](#), [1613](#)
- style sheet attributes [1216](#)
- style sheet CLASS attribute (StyleSheets) [1224](#), [1225](#)
- style sheet CSSURL attribute (StyleSheets) [1228](#), [1229](#)
- style sheet types [1140](#)
- style sheets [1110](#), [1117](#), [1141](#), [1611](#), [1782](#)
  - cascading style sheets (CSS) [1211](#)
  - multiple output formats [1238](#)
- STYLEMODE parameter [1239](#)
- STYLEMODE SET parameter [1294](#)
- StyleSheet attributes [1782](#)
- StyleSheet CLASS attribute [1224](#), [1225](#)
- STYLESHEET command [1120](#)
- StyleSheet CSSURL attribute [1228](#), [1229](#)
- StyleSheet declarations [1123](#), [1124](#)
- StyleSheet files [1120](#), [1121](#)
- STYLESHEET parameter [917](#)
- STYLESHEET SET parameter [1165](#)
- StyleSheets and external CSS [1227](#)
- styling ACROSS-TOTAL component [1184](#)
- styling free-form reports [1241](#)
- styling graphs [1712](#)
- styling reports [1211](#), [1415](#), [1416](#)

styling reports in external cascading style sheets  
[1224](#)

styling reports with external cascading style  
sheets [1211](#), [1214](#), [1216](#)

sub-total calculated values [385](#)

SUB-TOTAL command [379](#), [382–384](#), [391–393](#),  
[407](#), [409](#), [410](#), [412–415](#)

SUB-TOTAL command and propagation to grand  
total [399](#)

SUB-TOTAL prefix operators [415](#)

SUBFOOT command [1443](#)

SUBFOOT command report footings [1443](#)

SUBFOOT command sort footings [1461](#)

SUBFOOT component [1195](#), [1197](#)

SUBHEAD command [1439](#), [1457](#)  
report headings [1439](#)

SUBHEAD component [1195](#), [1197](#)

subquery file [528](#)

subroutines [1757](#)

subtotal calculated values [385](#)

SUBTOTAL command [379](#), [382–384](#), [391–393](#),  
[407](#), [409](#), [411–415](#)

SUBTOTAL command and propagation to grand  
total [399](#)

SUBTOTAL component [1174](#), [1185](#), [1186](#), [1189](#)

subtotal labels [1515](#)

SUBTOTAL prefix operators [415](#)

SUBTOTAL SUMMARYLINES command [410](#)

subtotals [385–387](#), [409](#), [410](#), [423–426](#),  
[1174–1176](#), [1185](#), [1186](#), [1513](#), [1515](#)

subtraction operator [437](#)

subtype attribute [764](#), [769](#), [777](#), [780](#), [1706](#),  
[1708](#), [1709](#)

subtypes [764](#), [1567](#), [1786](#)

SUM command [43](#), [54](#), [55](#), [92](#), [1090](#), [1091](#)

SUM command and merging data sources [1090](#),  
[1091](#)

SUM prefix operator [74](#)

SUMMARIZE command [385](#), [386](#), [391](#), [392](#), [405](#),  
[407](#), [409](#), [410](#), [415](#)

SUMMARIZE command and propagation to grand  
total [399](#)

SUMMARIZE prefix operators [415](#)

summary commands [409](#), [413](#)

summary lines [429](#), [430](#)

summary values [391](#), [392](#), [416](#)

SUMMARYLINES SET parameter [399](#), [400](#), [410](#)

summing columns [177](#)

summing field values [92](#)

summing report columns [177](#)

summing values [92](#)

SUMPREFIX parameter [199](#)

SUP-PRINT command [179](#), [1288](#), [1291](#)

supplying data directly in FML [1741](#), [1742](#)

supplying images descriptions for screen readers  
[1387](#)

supported data sources [1007](#)

suppressing column titles [1505](#)

suppressing display fields [1287](#)

- suppressing display in financial reports [1802](#), [1803](#)
- suppressing field padding [546–548](#)
- suppressing grand totals [427](#), [428](#)
- suppressing page numbers [1314](#)
- suppressing rows [1802](#), [1803](#)
- suppressing rows in financial reports [1802](#), [1803](#)
- suppressing sort field values [179](#)
- suppressing sort footings [1466](#)
- suppressing wrapping data [1636](#)
- SYLK format [530](#)
- syn\_regress\_mult [359](#)
- SyncSort utility [190](#)

## T

- TAB format [531](#)
- tab names [1438](#)
- tab-delimited output files [531](#)
- TABFOOTING component [1195](#), [1197](#)
- TABHEADING component [1195](#), [1197](#)
- TABLASTPAGE system variable [1304–1307](#)
- TABLE command [1896](#)
- TABLE command compared to GRAPH command [1659](#)
- TABLE FILE command [36](#), [38](#)
- TABLE requests [1817](#)
- TABLEF command [1836](#), [1845](#), [1898](#)
- TABLEF command and data retrieval [1845](#), [1846](#)
- TABLEF command and SQL Translator [1836](#)
- tables of contents (TOCs) [905](#)

- TABPAGENO system variable [1304–1306](#)
- TABT format [531](#)
- tag names [1013](#)
- TAG rows [1802](#), [1803](#)
  - suppressing display in financial reports [1802](#), [1803](#)
- tag values [1733](#)
  - reusing [1733](#), [1738](#), [1739](#), [1771](#)
- TARGET attribute [814](#)
- target frames [814–816](#)
- TARGET parameter [927](#)
- TARGETFRAME SET parameter [814–816](#)
- TD element [1222](#)
- temporary fields [277](#), [278](#), [280](#)
  - calculated values [298](#)
  - creating [36](#)
  - DEFINE FUNCTION [363](#)
  - evaluation [279](#)
  - types [278](#)
- temporary sort fields [99](#)
- temporary tables [500](#), [503](#), [505](#)
- temporary tables extract files [501](#)
- temporary tables HOLD files [503](#), [505](#)
- temporary tables output files [501](#), [503](#), [505](#)
- testing character strings [247–255](#)
- testing data fields [248](#), [249](#)
- testing for blanks or zeros [986](#)
- testing for existing data [246](#), [247](#), [986](#)
- testing for missing segment instances [1002](#)
- testing for missing values [984](#), [985](#), [1002](#)

- testing multi-segment files [256](#)
  - TEXT component [1200](#)
  - text field output files [535](#), [536](#)
  - text fields [93](#), [1565](#)
  - text fields and alphanumeric fields [362](#)
  - text fields in DEFINE and COMPUTE [362](#)
  - text fields in headings and footings [1477](#), [1478](#)
  - text fields output files [511](#)
  - text rows [1759](#), [1760](#)
    - formatting [1789](#)
  - text strings [1202](#)
  - three-dimensional graphs [1677](#)
  - TILE column [170](#), [171](#), [174](#)
  - tile fields [170](#), [171](#), [174](#), [175](#)
  - TILES phrase [170](#), [171](#), [174](#), [175](#)
  - time fields [1830](#)
  - time fields and SQL Translator [1831](#)
  - timestamp data type [452](#)
  - timestamp fields [1830](#)
  - timestamp fields and SQL Translator [1831](#)
  - TITLE attribute [495](#), [1503](#), [1507](#)
  - TITLE component [1192](#), [1193](#)
  - titles [1191](#), [1431](#), [1432](#), [1436](#)
  - titles of column [1193](#)
  - titles of columns [1191](#), [1192](#)
  - TITLETEXT attribute [1436](#), [1437](#)
  - TO phrase [1737](#)
  - TOCs (tables of contents) [905](#)
  - TOP [209](#)
  - top margins [1255](#)
  - TOPGAP attribute [1258](#), [1261](#), [1262](#), [1600](#)
  - TOPMARGIN attribute [1255](#), [1256](#)
  - TOT prefix operator [74](#)
  - total page count [1306](#)
  - totals [369](#), [376](#), [377](#), [1174–1176](#), [1185](#), [1186](#)
  - trailing blanks [1474](#)
  - TRAINING data source [1861](#), [1862](#)
  - treating as literal masking characters [253](#), [254](#)
  - treating as literal wildcard characters [253](#), [254](#)
  - treating literal masking characters [255](#)
  - treating literal wildcard characters [255](#)
  - triple exponential smoothing [328](#), [350](#), [352](#)
    - FORECAST\_SEASONAL [328](#)
  - truncated field names [1903](#)
  - truncating decimal values [439](#)
  - TYPE attribute [764](#), [769](#), [777](#), [780](#), [1122](#), [1363](#), [1422](#), [1522](#), [1706](#), [1708](#), [1709](#)
- ## U
- UNDER-LINE option [1354](#), [1360](#)
  - UNDERLINE attribute [1361](#), [1362](#)
  - UNDERLINE component [1206](#), [1207](#)
  - underlines [1206](#), [1207](#), [1354](#)
  - underlines in Financial Modeling Language (FML) reports [1368](#), [1369](#)
  - underlines in financial reports [1783](#)
  - underlining values [1360–1362](#)
  - Uniform Resource Locators (URLs) [769](#), [770](#), [813](#), [1381](#), [1706](#)
  - UNION operator [1830](#)

unique join structures [1006](#), [1008](#), [1009](#), [1020](#)

unique segments [46](#), [47](#), [57](#)

unique segments for PRINT command [49](#)

UNITS attribute [1141](#), [1255](#)

units of measurement [1141](#), [1255](#)

universal concatenation [1093](#)

universal concatenation and field names [1096](#), [1097](#)

universal concatenation and MORE phrase [1093–1095](#)

UNIX [600–602](#)

UNIX PDF files [600–602](#)

UNIX PDF format [602](#)

UNLIKE [249](#)

UPDATE command [1836](#)

URLs (Uniform Resource Locators) [769](#), [770](#), [813](#), [1381](#), [1706](#)

using ACROSS phrase with Accordion Reports [925](#)

using ACROSSVALUE component for a numeric column reference [1184](#)

using BY phrase with Accordion Reports [925](#)

using concatenation with AnV fields [461](#)

using CONTAINS and OMITS with AnV fields [462](#)

using drill-down reports conditions [807](#), [808](#)

using drill-downs with Accordion Reports [927](#)

using EDIT function with AnV fields [462](#)

using LIKE fields with AnV fields [462](#)

using multiple parameters [804](#)

using operators with AnV fields [463](#)

## V

value dates [441](#)

value format for dates [442](#)

values [1736](#)

- for columns [1753](#)
- in multiple rows [1738](#), [1739](#)
- reusing [1739](#)

varchar fields [519](#)

variable length character expressions [461](#)

variables [359](#)

- dependent and independent [359](#)

VAUTO parameter [1719](#)

VAXIS parameter [1713](#)

VBScript files [1703](#)

VBScript in an HTML report [572](#)

verbs [43](#), [92](#)

verbs, multiple [182](#)

VERBSET attribute [185](#)

verifying external sorting [191](#)

vertical bar graphs [1413](#)

vertical bar graphs and scaling [1420](#), [1422](#), [1423](#)

vertical labels [1712](#)

vertical scaling [1420](#), [1422](#), [1423](#)

vertical spacing [1598](#), [1600](#), [1602](#)

vertical waterfall graphs [1681](#)

VGRID attribute [1319](#), [1348](#)

VIDEOTR2 data source [1870](#), [1871](#)

VideoTrk data source [1866–1868](#)

viewing reports [905](#)

virtual fields [265](#), [266](#), [278](#), [280](#), [290](#), [291](#)

    calculated values [280](#)

VISDIS [532](#)

Visual Discovery format [532](#)

VMAX parameter [1719](#)

VMIN parameter [1719](#)

VMSORT utility [190](#)

VSAM data sources [274](#)

VSAM record selection efficiencies [274](#)

VZERO parameter [1696](#)

## W

waterfall graphs [1681](#)

web browser support for cascading style sheets  
[1241](#)

WebFOCUS font map files [585](#)

WebFOCUS Font Map files [583](#)

WebFOCUS inheritance StyleSheets [1127](#), [1128](#)

WebFOCUS macros StyleSheets [1124](#), [1125](#)

WebFOCUS style sheets [1122](#), [1124](#), [1127](#), [1128](#)

WebFOCUS StyleSheet CLASS attribute [1217](#)

WebFOCUS StyleSheet CSSURL attribute [1217](#)

WebFOCUS StyleSheet declarations [1122–1124](#)

WebFOCUS StyleSheet files [1120](#), [1121](#)

WebFOCUS StyleSheets [905](#), [917](#), [919](#), [958](#),  
[1113](#), [1117](#), [1118](#), [1142](#), [1415](#), [1418](#), [1611](#),  
[1648](#)

WebFOCUS StyleSheets and adding graphics  
[1376](#), [1388](#)

WebFOCUS StyleSheets and conditional styling  
[1142](#), [1143](#), [1145](#), [1146](#), [1148](#), [1150–1152](#)

WebFOCUS StyleSheets and data visualization  
[1413](#), [1415](#), [1416](#), [1418](#)

WebFOCUS StyleSheets and graphics [1380](#),  
[1390](#), [1395](#)

WebFOCUS StyleSheets and graphs [1700](#)

WebFOCUS StyleSheets and links [763](#), [764](#)

WebFOCUS StyleSheets and multi-pane reports  
[1424](#), [1429](#)

WebFOCUS StyleSheets and parameters [797](#),  
[801](#), [802](#)

WebFOCUS StyleSheets macros [1124](#)

WebFOCUS Viewer [905](#), [955](#)

WebFOCUS Viewer search option [958](#)

WEBVIEWER SET parameter [1293](#)

WHEN attribute [1142](#), [1143](#)

WHEN EXISTS phrase [1803](#)

WHEN for SAP BW [209](#)

WHEN phrase [380](#), [429](#), [431](#), [1158](#)

WHEN phrase and conditional formatting [1156](#),  
[1158–1161](#), [1465](#)

WHEN phrase expressions [431](#)

WHEN=FORECAST attribute [316](#), [342](#), [1145](#)

WHERE operator [238](#), [240](#), [245](#), [246](#)

WHERE phrase [220–222](#), [228](#), [230](#), [237](#), [248](#),  
[250](#), [260](#), [261](#), [263](#), [431](#), [1694](#)

WHERE phrase and existing data [986](#)

WHERE phrase and join structures [1070](#)

WHERE phrase and missing values [985](#), [987](#)

WHERE phrase expressions [431](#)  
WHERE tests [242](#)  
WHERE TOTAL phrase [228–230](#), [1694](#)  
WHERE\_GROUPED [230](#)  
WHERE-based join structures [1006](#)  
WIDTH attribute [1578](#)  
width of borders [1321](#)  
width of columns [1265–1268](#), [1270](#)  
wildcard characters [248](#), [250](#)  
window titles [1436](#)  
WITH CHILDREN parameter [1767](#)  
WITHIN phrase [141](#)  
WK1 format [533](#)  
worksheet names [1438](#)  
worksheet titles [1436](#)  
WP display formats for reports [602](#)  
WP format [533](#), [566](#), [602](#)  
WP report display formats [602](#)  
WPMINWIDTH parameter [534](#)  
WRAP attribute [1633](#), [1634](#)  
wrapgap StyleSheet attribute [1645](#)  
wrapping data [1633](#)  
wrapping data by Web browser functionality [1635](#)

wrapping data reports [1633](#)

WRITE command [54](#)

## X

x-axis [1683](#), [1685](#)

XFOCUS format [534](#)

XLSX display format [520](#)

XLSX output file format [520](#)

XLSX PCHOLD format [520](#)

XLSX SAVE format [520](#)

XML format [535](#)

## Y

y-axis [1683](#)

y-axis fields [1684](#)

Y2K attributes in Master Files [452](#)

Year 2000 attributes in Master Files [452](#)

YRTHRESH attribute [452](#)

## Z

z/OS requirements [190](#)

zeros [986](#)







# 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](mailto: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](mailto:Frances_Gambino@ibi.com).



# WebFOCUS

Creating Reports With WebFOCUS Language  
Release 8.2 Version 03



**Information  
Builders**

DN4501639.0418

Information Builders, Inc.  
Two Penn Plaza  
New York, NY 10121-2898