WebFOCUS

Creating Reports With WebFOCUS Language

Version 7 Release 6
Contents

Preface..............................................................................................................23
  Documentation Conventions...........................................................................26
  Related Publications.........................................................................................27
  Customer Support.............................................................................................27
  Information You Should Have...........................................................................27
  User Feedback....................................................................................................28
  Information Builders Consulting and Training...............................................28

1. Creating Reports Overview..............................................................................31
  Requirements for Creating a Report.................................................................32
  Report Types.....................................................................................................33
  Reporting Tools................................................................................................34
  Developing Your Report Request....................................................................35
    Starting a Report Request...............................................................................37
    Completing a Report Request.........................................................................37
    Creating a Report Example............................................................................37
  Customizing a Report.......................................................................................39
  Selecting a Report Output Destination............................................................41

2. Displaying Report Data....................................................................................43
  Using Display Commands in a Request............................................................44
  Displaying Individual Values............................................................................45
    Displaying All Fields.....................................................................................47
    Displaying All Fields in a Segment...............................................................47
    Displaying the Structure and Retrieval Order of a Multi-Path Data Source....49
  Adding Values..................................................................................................53
  Counting Values...............................................................................................55
    Counting Segment Instances.......................................................................56
  Expanding Byte Precision for COUNT and LIST..............................................57
  Maximum Number of Display Fields Supported in a Request.........................59
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manipulating Display Fields With Prefix Operators</td>
<td>60</td>
</tr>
<tr>
<td>Prefix Operator Basics</td>
<td>61</td>
</tr>
<tr>
<td>Averaging Values of a Field</td>
<td>63</td>
</tr>
<tr>
<td>Averaging the Sum of Squared Fields</td>
<td>64</td>
</tr>
<tr>
<td>Calculating Maximum and Minimum Field Values</td>
<td>64</td>
</tr>
<tr>
<td>Calculating Column and Row Percents</td>
<td>65</td>
</tr>
<tr>
<td>Producing a Direct Percent of a Count</td>
<td>67</td>
</tr>
<tr>
<td>Aggregating and Listing Unique Values</td>
<td>67</td>
</tr>
<tr>
<td>Retrieving First and Last Records</td>
<td>70</td>
</tr>
<tr>
<td>Summing and Counting Values</td>
<td>72</td>
</tr>
<tr>
<td>Ranking Sort Field Values With RNK</td>
<td>74</td>
</tr>
<tr>
<td>Displaying Pop-up Field Descriptions for Column Titles</td>
<td>77</td>
</tr>
<tr>
<td>3. Sorting Tabular Reports</td>
<td>81</td>
</tr>
<tr>
<td>Sorting Tabular Reports Overview</td>
<td>82</td>
</tr>
<tr>
<td>Sorting Rows</td>
<td>83</td>
</tr>
<tr>
<td>Displaying All Vertical (BY) Sort Field Values</td>
<td>85</td>
</tr>
<tr>
<td>Using Multiple Vertical (BY) Sort Fields</td>
<td>86</td>
</tr>
<tr>
<td>Displaying a Row for Data Excluded by a Sort Phrase</td>
<td>87</td>
</tr>
<tr>
<td>Sorting Columns</td>
<td>90</td>
</tr>
<tr>
<td>Controlling Underlines for ACROSS Objects</td>
<td>92</td>
</tr>
<tr>
<td>Using Multiple Horizontal (ACROSS) Sort Fields</td>
<td>94</td>
</tr>
<tr>
<td>Collapsing PRINT With ACROSS</td>
<td>95</td>
</tr>
<tr>
<td>Manipulating Display Field Values in a Sort Group</td>
<td>97</td>
</tr>
<tr>
<td>Creating a Matrix Report</td>
<td>99</td>
</tr>
<tr>
<td>Specifying the Sort Order</td>
<td>100</td>
</tr>
<tr>
<td>Specifying Your Own Sort Order</td>
<td>103</td>
</tr>
<tr>
<td>Ranking Sort Field Values</td>
<td>107</td>
</tr>
<tr>
<td>Grouping Numeric Data Into Ranges</td>
<td>109</td>
</tr>
<tr>
<td>Grouping Numeric Data Into Tiles</td>
<td>113</td>
</tr>
<tr>
<td>Restricting Sort Field Values by Highest/Lowest Rank</td>
<td>118</td>
</tr>
<tr>
<td>Sorting and Aggregating Report Columns</td>
<td>119</td>
</tr>
<tr>
<td>Hiding Sort Values</td>
<td>122</td>
</tr>
<tr>
<td>Sorting With Multiple Display Commands</td>
<td>124</td>
</tr>
</tbody>
</table>
Controlling Formatting of Reports With Multiple Display Commands..........................125
Improving Efficiency With External Sorts..................................................................................130
Providing an Estimate of Input Records or Report Size for Sorting..................................133
Sort Work Files and Return Codes..................................................................................133
Mainframe External Sort Utilities and Message Options..................................................135
Aggregation by External Sort (Mainframe Environments Only)...........................................138
Changing Retrieval Order With Aggregation (Mainframe Environments Only)...............140
Creating a HOLD File With an External Sort (Mainframe Environments Only)........141

4. Selecting Records for Your Report..................................................................................143
   Selecting Records Overview.............................................................................................144
   Choosing a Filtering Method.........................................................................................144
   Selections Based on Individual Values........................................................................145
      Controlling Record Selection in Multi-path Data Sources........................................148
   Selection Based on Aggregate Values........................................................................153
   Using Compound Expressions for Record Selection..................................................155
   Using Operators in Record Selection Tests..................................................................157
   Types of Record Selection Tests..................................................................................161
      Range Tests With FROM and TO...........................................................................162
      Range Tests With GE and LE or GT and LT..........................................................163
      Missing Data Tests..................................................................................................165
      Character String Screening With CONTAINS and OMITS..................................165
      Screening on Masked Fields With LIKE and IS....................................................167
      Using an Escape Character for LIKE...................................................................171
      Qualifying Parent Segments Using INCLUDES and EXCLUDES...........................174
   Selections Based on Group Key Values..........................................................................175
   Setting Limits on the Number of Records Read............................................................176
   Selecting Records Using IF Phrases...............................................................................177
   Reading Selection Values From a File............................................................................179
   Assigning Screening Conditions to a File......................................................................182
      Preserving Filters Across Joins................................................................................188
   VSAM Record Selection Efficiencies..............................................................................191
   Reporting From Files With Alternate Indexes.............................................................191
5. Creating Temporary Fields


What Is a Temporary Field? ................................................................. 194
Defining a Virtual Field ........................................................................ 197
Defining Multiple Virtual Fields ....................................................... 203
Displaying Virtual Fields .................................................................. 204
Clearing a Virtual Field ....................................................................... 204
Establishing a Segment Location for a Virtual Field ....................... 206
Defining Virtual Fields Using a Multi-Path Data Source ................... 207
Increasing the Speed of Calculations in Virtual Fields .................... 207
Preserving Virtual Fields Using DEFINE FILE SAVE and RETURN .... 208
Applying Dynamically Formatted Virtual Fields to Report Columns .. 209
Creating a Calculated Value ............................................................... 213
Using Positional Column Referencing With Calculated Values ......... 216
Using ACROSS With Calculated Values .............................................. 217
Sorting Calculated Values ................................................................ 218
Screening on Calculated Values ........................................................ 218
Assigning Column Reference Numbers ............................................. 218
Using Column Notation in a Report Request .................................... 219
Calculating Trends and Predicting Values With FORECAST .......... 229
FORECAST Processing ....................................................................... 230
Using a Simple Moving Average ......................................................... 234
Using Single Exponential Smoothing ............................................... 239
Using Double Exponential Smoothing .............................................. 241
Using Triple Exponential Smoothing ............................................... 242
Using a Linear Regression Equation ............................................... 245
FORECAST Reporting Techniques .................................................... 247
Calculating Trends and Predicting Values With Multivariate REGRESS . 250
Using Text Fields in DEFINE and COMPUTE ..................................... 254
Creating Temporary Fields Independent of a Master File ................. 255

6. Including Totals and Subtotals ......................................................... 261
Calculating Row and Column Totals ................................................ 262
Producing Row Totals for Horizontal (ACROSS) Sort Field Values ... 269
Including Section Totals and a Grand Total ....................................... 270
Including Subtotals........................................................................................................272
Recalculating Values for Subtotal Rows........................................................................276
Manipulating Summary Values With Prefix Operators........................................................280
  Controlling Summary Line Processing........................................................................288
  Using Prefix Operators With Calculated Values.........................................................295
  Using Multiple SUB-TOTAL or SUMMARIZE Commands With Prefix Operators........297
Combinations of Summary Commands............................................................................299
Producing Summary Columns for Horizontal Sort Fields..................................................306
Performing Calculations at Sort Field Breaks................................................................308
 Suppressing Grand Totals..............................................................................................312
 Conditionally Displaying Summary Lines and Text........................................................313
7. Using Expressions........................................................................................................315
Using Expressions in Commands and Phrases.................................................................316
Types of Expressions....................................................................................................317
  Expressions and Field Formats....................................................................................318
Creating a Numeric Expression..........................................................................................319
  Order of Evaluation....................................................................................................322
  Evaluating Numeric Expressions With Native-Mode Arithmetic...............................323
  Using Identical Operand Formats With Native-Mode Arithmetic...............................323
  Using Different Operand Formats With Native-Mode Arithmetic...............................324
Creating a Date Expression................................................................................................325
  Formats for Date Values.............................................................................................326
  Performing Calculations on Dates...............................................................................328
  Cross-Century Dates With DEFINE and COMPUTE..................................................329
  Returned Field Format Selection...............................................................................329
  Using a Date Constant in an Expression.....................................................................329
  Extracting a Date Component.....................................................................................330
  Combining Fields With Different Formats in an Expression.......................................330
Creating a Date-Time Expression......................................................................................331
  Specifying a Date-Time Value....................................................................................332
  Manipulating Date-Time Values..................................................................................336
Creating a Character Expression.......................................................................................338
  Embedding a Quotation Mark in a Quote-Delimited Literal String..............................339
Concatenating Character Strings ................................................................................... 340
Creating a Variable Length Character Expression ............................................................. 341
Using Concatenation With AnV Fields ............................................................................. 342
Using the EDIT Function With AnV Fields ..................................................................... 342
Using CONTAINS and OMITS With AnV Fields ............................................................... 342
Using LIKE With AnV Fields .......................................................................................... 343
Using the EQ, NE, LT, GT, LE, and GE Operators With AnV Fields ................................. 343
Using the DECODE Function With AnV Fields ............................................................... 344
Using the Assignment Operator With AnV Fields ............................................................ 344
Creating a Logical Expression ........................................................................................... 345
Creating a Conditional Expression .................................................................................... 347

8. Saving and Reusing Your Report Output .............................................................................. 351

Saving Your Report Output ............................................................................................... 352
Naming and Storing Report Output Files .......................................................................... 352
Creating a HOLD File .......................................................................................................... 353
Holding Report Output in FOCUS Format ......................................................................... 358
Controlling Attributes in HOLD Master Files .................................................................. 363
  Controlling Field Names in a HOLD Master File .......................................................... 364
  Controlling Fields in a HOLD Master File .................................................................... 368
  Controlling the TITLE and ACCEPT Attributes in the HOLD Master File ................. 370
Keyed Retrieval From HOLD Files ..................................................................................... 372
Saving and Retrieving HOLD Files .................................................................................... 374
Using DBMS Temporary Tables as HOLD Files ............................................................... 376
  Column Names in the HOLD File ................................................................................. 380
  Primary Keys and Indexes in the HOLD File ............................................................... 381
Creating SAVE and SAVB Files ......................................................................................... 381
Creating a PCHOLD File .................................................................................................... 385
Choosing Output File Formats .......................................................................................... 386
Using Text Fields in Output Files ....................................................................................... 407
Creating a Delimited Sequential File ............................................................................... 409
Saving Report Output in INTERNAL Format .................................................................... 414
Creating a Structured HOLD File ...................................................................................... 417
Specifying MIME Types for WebFOCUS Reports ................................................................ 426
9. Choosing a Display Format

Report Display Formats .................................................................430
Preserving Leading and Internal Blanks in Report Output ..................433
Using Web Display Format: HTML ................................................435
Using Print Display Formats: PDF, PS ..............................................438
  Using PDF Display Format .....................................................439
  Using PostScript (PS) Display Format ........................................440
  Adding PostScript Type 1 Fonts for PS and PDF Formats ................443
Creating PDF Files for Use With UNIX Systems .............................452
Using Word Processing Display Formats: DOC, WP .........................454
Using Excel Display Formats: Excel 2000, Excel 97, Excel .................455
  Assigning Excel 2000 Format to Your Report Output ......................457
  Drilling Down to a Procedure in Excel 2000 With a Table of Contents ....459
  Identifying Null Values in Excel 2000 .........................................461
  Displaying Formatted Dates and Numeric Values in Excel 2000 ..........464
  Controlling Column Width and Wrapping in Excel 2000 .....................467
  Locking Columns in Excel Report Output .................................471
  Generating Native Excel Formulas in Excel 2000 ..............................474
  Using PivotTables in Excel 2000 ..............................................480
  Enabling PivotTables ..................................................................484
  Designating CACHEFIELDS in PivotTables ....................................487
  Designating PAGEFIELDS in PivotTables ......................................488
  Viewing a Report in Excel 97 Format ............................................490
  Embedding a WebFOCUS Report in an Excel 97 Spreadsheet ..........492
  Embedding a WebFOCUS Report in an Excel 2000 Spreadsheet .......501
  Deploying a Spreadsheet With a WebFOCUS Web QUERY .............508
Creating Excel Templates .................................................................511
  Utilizing Excel Named Ranges ...................................................513
  Creating Excel Tables of Contents Reports ....................................516

10. Linking a Report to Other Resources ...........................................519

Linking Using StyleSheets .............................................................520
Linking to Another Report .............................................................520
Linking to a URL .........................................................................525
Linking to a JavaScript Function.................................................................528
Linking to a WebFOCUS Maintain Procedure........................................531
Creating Multiple Drill-Down Links..........................................................534
  Applying Conditional Styling.................................................................540
  Saving a Drill-Down Report to a Server With HTMTABLE.......................542
Creating Parameters..................................................................................544
Linking With Conditions..........................................................................554
Linking From a Graphic Image.................................................................556
Specifying a Base URL...............................................................................561
Specifying a Target Frame.........................................................................562
Creating a Compound Report..................................................................564
  Creating a Compound Layout Report With Document Composer Syntax....566
  Creating a Compound PDF or PS Report...............................................604
  Creating a Compound Excel Report......................................................612
Creating a PDF Compound Report With Drill Through Links...................623
  Sample Drill Through PDF Compound Reports......................................628

11. Navigating Within an HTML Report......................................................639
  Navigating Sort Groups From a Table of Contents...............................640
  Adding an HTML TOC to a Report.........................................................641
    Navigation Behavior in a Multi-Level TOC...........................................647
Creating an Active Report........................................................................658
  Controlling Active Report Formatting Options....................................662
  User Controlled Options in an Active Report.......................................671
  Modifying a Chart Legend With StyleSheet Syntax...............................672
  Controlling Expiration of an Active Report.........................................677
  Setting a Password for an Active Report..............................................678
  Controlling the Display of Sorted Data With Accordion Reports............678
    Requirements for Accordion Reports..................................................679
  Navigating a Multi-Page Report With the WebFOCUS Viewer.................682
    Using the WebFOCUS Viewer Search Option.......................................684
  Linking Report Pages.............................................................................685

12. Bursting Reports Into Multiple HTML Files.........................................691
  Bursting Reports Overview....................................................................692
13. Handling Records With Missing Field Values .............................................................. 697
   Irrelevant Report Data ................................................................................................. 698
   Missing Field Values .................................................................................................. 699
      MISSING Attribute in the Master File ...................................................................... 700
      MISSING Attribute in a DEFINE or COMPUTE Command ........................................ 702
   Testing for a Segment With a Missing Field Value .................................................... 705
   Preserving Missing Data Values in an Output File ..................................................... 709
   Propagating Missing Values to Reformatted Fields in a Request .............................. 711
   Handling a Missing Segment Instance ....................................................................... 713
      Including Missing Instances in Reports With the ALL Prefix ............................... 716
      Including Missing Instances in Reports With the SET ALL Parameter ................. 717
      Testing for Missing Instances in FOCUS Data Sources ......................................... 719
   Setting the NODATA Character String ...................................................................... 720

14. Joining Data Sources ................................................................................................ 723
   Types of Joins .............................................................................................................. 724
      Unique and Non-Unique Joined Structures ................................................................ 727
      Recursive Joined Structures .................................................................................... 731
   How the JOIN Command Works ............................................................................... 735
   Creating an Equijoin ................................................................................................ 737
      Joining From a Virtual Field to a Real Field Using an Equijoin ......................... 747
      Data Formats of Shared Fields ............................................................................... 751
      Joining Fields With Different Numeric Data Types .............................................. 752
   Using a Conditional Join .......................................................................................... 753
   Preserving Virtual Fields During Join Parsing ......................................................... 757
      Preserving Virtual Fields Using KEEPDEFINES .................................................. 757
      Preserving Virtual Fields Using DEFINE FILE SAVE and RETURN .................... 760
   Screening Segments With Conditional JOIN Expressions ..................................... 762
      Parsing WHERE Criteria in a Join ........................................................................ 762
   Displaying Joined Structures .................................................................................... 763
   Clearing Joined Structures ......................................................................................... 766
      Clearing a Conditional Join .................................................................................... 766

15. Merging Data Sources .............................................................................................. 769
   Merging Data .............................................................................................................. 770
MATCH Processing........................................................................................................773
MATCH Processing With Common High-Order Sort Fields..................................................776
Fine-Tuning MATCH Processing.......................................................................................780
Universal Concatenation................................................................................................783
  Field Name and Format Matching............................................................................786
Merging Concatenated Data Sources..............................................................................788
  Using Sort Fields in MATCH Requests.................................................................790
Cartesian Product.........................................................................................................794

16. Formatting Reports: An Overview........................................................................797
  What Kinds of Formatting Can I Do?..........................................................................798
  How to Specify Formatting in a Report........................................................................800
    How to Choose a Type of Style Sheet........................................................................803
  Standard and Legacy Formatting................................................................................804
  Techniques for Quick and Easy Formatting..............................................................805
  Navigating From a Report to Other Resources..........................................................806

17. Creating and Managing a WebFOCUS StyleSheet..............................................807
  Creating a WebFOCUS StyleSheet..........................................................................808
    Creating a WebFOCUS StyleSheet Within a Report Request................................808
    Creating and Applying a WebFOCUS StyleSheet File...........................................810
  General WebFOCUS StyleSheet Syntax.....................................................................812
    Improving WebFOCUS StyleSheet Readability....................................................814
    Adding a Comment to a WebFOCUS StyleSheet....................................................815
    Validating WebFOCUS StyleSheet Syntax.............................................................815
  Reusing WebFOCUS StyleSheet Declarations With Macros.....................................816
    Defining a WebFOCUS StyleSheet Macro...............................................................816
    Applying a WebFOCUS StyleSheet Macro..............................................................817
  WebFOCUS StyleSheet Attribute Inheritance..........................................................819

18. Controlling Report Formatting.............................................................................825
  Generating an Internal Cascading Style Sheet for HTML Reports.............................826
  Selecting a Unit of Measurement...............................................................................827
  Conditionally Formatting, Displaying, and Linking in a StyleSheet.............................828
    Applying Sequential Conditional Formatting......................................................829
19. Identifying a Report Component in a WebFOCUS StyleSheet

- Identifying an Entire Report, Column, or Row.........................................................850
- Identifying Data.........................................................................................................860
  - Identifying Totals and Subtotals..........................................................................866
- Identifying a Heading, Footing, Title, or FML Free Text.........................................873
  - Identifying a Column or Row Title.......................................................................874
  - Identifying a Heading or Footing.........................................................................879
- Identifying a Page Number, Underline, or Skipped Line........................................887

20. Using an External Cascading Style Sheet

- What Is a Cascading Style Sheet?..............................................................................892
  - What Are Cascading Style Sheet Rules and Classes?...........................................893
- Why Use an External Cascading Style Sheet?.........................................................894
- Formatting a Report With an External Cascading Style Sheet.................................895
- Working With an External Cascading Style Sheet....................................................901
  - Choosing an External Cascading Style Sheet.....................................................902
  - External Cascading Style Sheet Location.............................................................902
  - Using Several External Cascading Style Sheets....................................................902
  - Editing an External Cascading Style Sheet...........................................................903
  - Choosing a Cascading Style Sheet Rule..............................................................903
  - Naming a Cascading Style Sheet Class.................................................................904
- Applying External Cascading Style Sheet Formatting.................................................905
- Combining an External CSS With Other Formatting Methods....................................907
  - Combining an External CSS With WebFOCUS StyleSheets.................................908
- Linking to an External Cascading Style Sheet............................................................909
  - Using the CSSURL Attribute and Parameter.......................................................910
- Inheritance and External Cascading Style Sheets......................................................913
- Using External Cascading Style Sheets With Non-HTML Reports..............................916
- Requirements for Using an External Cascading Style Sheet......................................921
- FAQ About Using External Cascading Style Sheets................................................922
- Troubleshooting External Cascading Style Sheets....................................................925
21. Laying Out the Report Page ................................................................. 929
  Selecting Page Size, Orientation, and Color ........................................ 930
  Setting Page Margins ........................................................................... 935
  Positioning a Report Component .......................................................... 939
  Arranging Columns on a Page ............................................................... 946
    Determining Column Width ............................................................... 947
    Controlling Column Spacing ............................................................. 952
    Changing Column Order ................................................................... 953
    Stacking Columns ............................................................................ 955
    Positioning a Column ...................................................................... 957
  Suppressing Column Display ............................................................... 961
  Inserting a Page Break ........................................................................ 968
    Preventing an Undesirable Split ....................................................... 973
  Inserting Page Numbers ..................................................................... 979
    Inserting the Total Page Count ....................................................... 981
    Displaying the Total Page Count Within a Sort Group .................. 983
    Assigning Any Page Number to the First Page ............................... 986
    Controlling the Display of Page Numbers ...................................... 989
  Adding Grids and Borders .................................................................. 991
  Adding Underlines and Skipped Lines ................................................. 1000
  Adding an Image to a Report ............................................................... 1016
  Associating Bar Graphs With Report Data ......................................... 1040
    Controlling Bar Graph Scaling in Horizontal (ACROSS) Sort Fields .. 1043
    Formatting Data Visualization Bar Graphs ...................................... 1046
  Working With Mailing Labels and Multi-Pane Pages ......................... 1046

22. Using Headings, Footings, Titles, and Labels ..................................... 1053
  Creating Headings and Footings ......................................................... 1054
    Limits for Headings and Footings .................................................... 1055
    Extending Heading and Footing Code to Multiple Lines in a Report Request ........................................ 1056
  Creating a Custom Report or Worksheet Title .................................... 1059
  Creating a Report Heading or Footing ............................................... 1061
  Creating a Page Heading or Footing ................................................... 1069
  Freezing HTML Headings, Footings, and Column Titles .................. 1077
Creating a Sort Heading or Footing.................................................................1078
Including an Element in a Heading or Footing..................................................1093
  Including a Field Value in a Heading or Footing.............................................1093
  Including a Text Field in a Heading or Footing...............................................1101
  Including a Page Number in a Heading or Footing..........................................1103
  Including a Dialogue Manager Variable in a Heading or Footing..................1104
  Including an Image in a Heading or Footing..................................................1106
Customizing a Column Title..............................................................................1107
  Customizing a Column Title in a Master File..................................................1112
  Distinguishing Between Duplicate Field Names.............................................1112
Creating Labels to Identify Data........................................................................1113
  Creating a Label for a Row or Column Total..................................................1113
  Creating a Label for a Subtotal and a Grand Total..........................................1116
  Creating a Label for a Row in a Financial Report...........................................1120
Formatting a Heading, Footing, Title, or Label..................................................1120
  Applying Font Attributes to a Heading, Footing, Title, or Label.......................1122
Adding Borders and Grid Lines.........................................................................1126
Justifying a Heading, Footing, Title, or Label....................................................1130
  Justifying a Heading or Footing......................................................................1130
  Justifying a Column Title...............................................................................1138
  Justifying a Label for a Row or Column Total................................................1143
  Justifying a Label for a Subtotal or Grand Total..............................................1145
Choosing an Alignment Method for Heading and Footing Elements.....................1147
Aligning a Heading or Footing Element in an HTML Report................................1149
Aligning a Heading or Footing Element Across Columns in an HTML Report........1162
Aligning Content in a Multi-Line Heading or Footing.........................................1166
  Aligning Decimals in a Multi-Line Heading or Footing....................................1172
  Combining Column and Line Formatting in Headings and Footings...............1173
Positioning Headings, Footings, or Items Within Them....................................1180
  Using PRINTPLUS.........................................................................................1187
  Using Spot Markers to Refine Positioning......................................................1188
Controlling the Vertical Positioning of a Heading or Footing.............................1193
Placing a Report Heading or Footing on Its Own Page.......................................1201
23. Formatting Report Data .......................................................... 1205
   Specifying Font Format in a Report ........................................... 1206
   Specifying Fonts for Reports ................................................... 1212
   Specifying Background Color in a Report .............................. 1215
   Specifying Data Format in a Report ........................................... 1218
   Changing the Format of Values in a Report Column ................. 1219
   Controlling Missing Values for a Reformatted Field ................. 1221
   Using Commas vs. Decimals (Continental Decimal Notation) ....  1223
   Setting Characters to Represent Null and Missing Values .......... 1224
   Using Conditional Grid Formatting in a Field ......................... 1225
   Positioning Data in a Report .................................................... 1226
   Controlling Wrapping of Report Data ....................................... 1227
   Controlling Wrapping With Alternative Methods .................... 1233
   Justifying Report Columns ..................................................... 1235
   Field-Based Reformatting ....................................................... 1237
   Data Visualization ................................................................. 1240
   Displaying Multi-Line An and AnV Fields ............................... 1242

24. Creating a Graph ................................................................. 1245
   Content Analysis: Determining Graphing Objectives ................. 1246
   The GRAPH Command .......................................................... 1247
   Similarities Between GRAPH and TABLE ............................. 1247
   Differences Between GRAPH and TABLE ............................... 1248
   Selecting a Graph Type .......................................................... 1250
   Graph Types ......................................................................... 1250
   Selecting Scales ..................................................................... 1251
   Determining Graph Styles With Display Commands and Sort Phrases ........................................ 1252
   Determining Graph Styles Using LOOKGRAPH ..................... 1261
   Selecting Values for the X and Y Axes ..................................... 1270
   Hiding the Display of a Y-Axis Field ....................................... 1271
   Interpolating X and Y Axis Values Using Linear Regression ...... 1271
   Creating Multiple Graphs ...................................................... 1272
   Merging Multiple Graphs ...................................................... 1273
   Displaying Multiple Graphs in Columns ................................. 1274
<table>
<thead>
<tr>
<th>Data Source Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCATOR Structure Diagram</td>
<td>1462</td>
</tr>
<tr>
<td>PERSINFO Data Source</td>
<td>1462</td>
</tr>
<tr>
<td>PERSINFO Master File</td>
<td>1462</td>
</tr>
<tr>
<td>PERSINFO Structure Diagram</td>
<td>1463</td>
</tr>
<tr>
<td>SALHIST Data Source</td>
<td>1463</td>
</tr>
<tr>
<td>SALHIST Master File</td>
<td>1463</td>
</tr>
<tr>
<td>SALHIST Structure Diagram</td>
<td>1463</td>
</tr>
<tr>
<td>VIDEOTRK, MOVIES, and ITEMS Data Sources</td>
<td>1464</td>
</tr>
<tr>
<td>VIDEOTRK Master File</td>
<td>1464</td>
</tr>
<tr>
<td>VIDEOTRK Structure Diagram</td>
<td>1465</td>
</tr>
<tr>
<td>MOVIES Master File</td>
<td>1466</td>
</tr>
<tr>
<td>MOVIES Structure Diagram</td>
<td>1466</td>
</tr>
<tr>
<td>ITEMS Master File</td>
<td>1467</td>
</tr>
<tr>
<td>ITEMS Structure Diagram</td>
<td>1467</td>
</tr>
<tr>
<td>VIDEOTR2 Data Source</td>
<td>1468</td>
</tr>
<tr>
<td>VIDEOTR2 Master File</td>
<td>1468</td>
</tr>
<tr>
<td>VIDEOTR2 Structure Diagram</td>
<td>1469</td>
</tr>
<tr>
<td>Gotham Grinds Data Sources</td>
<td>1470</td>
</tr>
<tr>
<td>GGDEMOG Master File</td>
<td>1471</td>
</tr>
<tr>
<td>GGDEMOG Structure Diagram</td>
<td>1471</td>
</tr>
<tr>
<td>GGORDER Master File</td>
<td>1472</td>
</tr>
<tr>
<td>GGORDER Structure Diagram</td>
<td>1472</td>
</tr>
<tr>
<td>GGPRODS Master File</td>
<td>1473</td>
</tr>
<tr>
<td>GGPRODS Structure Diagram</td>
<td>1473</td>
</tr>
<tr>
<td>GGSALES Master File</td>
<td>1474</td>
</tr>
<tr>
<td>GGSALES Structure Diagram</td>
<td>1474</td>
</tr>
<tr>
<td>GGSTORES Master File</td>
<td>1475</td>
</tr>
<tr>
<td>GGSTORES Structure Diagram</td>
<td>1475</td>
</tr>
<tr>
<td>Century Corp Data Sources</td>
<td>1476</td>
</tr>
<tr>
<td>CENTCOMP Master File</td>
<td>1478</td>
</tr>
<tr>
<td>CENTCOMP Structure Diagram</td>
<td>1478</td>
</tr>
<tr>
<td>CENTFIN Master File</td>
<td>1479</td>
</tr>
<tr>
<td>CENTFIN Structure Diagram</td>
<td>1479</td>
</tr>
<tr>
<td>CENTHR Master File</td>
<td>1480</td>
</tr>
</tbody>
</table>
B. Error Messages .................................................................................. 1493
   Displaying Messages ........................................................................ 1494

C. Table Syntax Summary ...................................................................... 1495
   TABLE Syntax Summary ................................................................... 1496
      Hierarchical Reporting Syntax Summary ...................................... 1497
   TABLEF Syntax Summary ................................................................ 1498
   MATCH Syntax Summary ................................................................... 1499
   FOR Syntax Summary ...................................................................... 1500

D. Referring to Fields in a Report Request ........................................... 1501
   Referring to an Individual Field ...................................................... 1502
   Referring to Fields Using Long and Qualified Field Names ............ 1502
   Referring to All of the Fields in a Segment .................................... 1504
   Displaying a List of Field Names .................................................... 1505
      Listing Field Names, Aliases, and Format Information ................ 1505

Reader Comments ............................................................................... 1561
Preface

This documentation 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:

<table>
<thead>
<tr>
<th>Chapter/Appendix</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Creating Reports Overview Describes the tools available for creating and formatting reports.</td>
</tr>
<tr>
<td>2</td>
<td>Displaying Report Data Describes ways to retrieve field values from a data source and display them.</td>
</tr>
<tr>
<td>3</td>
<td>Sorting Tabular Reports Describes how to display report information grouped in a particular order by sorting.</td>
</tr>
<tr>
<td>4</td>
<td>Selecting Records for Your Report Describes how to use and specify selection criteria to display only the field values that meet your needs.</td>
</tr>
<tr>
<td>5</td>
<td>Creating Temporary Fields Describes how to use the DEFINE and COMPUTE commands to create temporary fields.</td>
</tr>
<tr>
<td>6</td>
<td>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.</td>
</tr>
<tr>
<td>7</td>
<td>Using Expressions Describes how to combine operators, field names, and constants in an expression to derive new values.</td>
</tr>
<tr>
<td>Chapter/Appendix</td>
<td>Contents</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
</tr>
<tr>
<td>8</td>
<td>Saving and Reusing Your Report Output</td>
</tr>
<tr>
<td></td>
<td>Describes how to save report data to files for reuse in different respects.</td>
</tr>
<tr>
<td>9</td>
<td>Choosing a Display Format</td>
</tr>
<tr>
<td></td>
<td>Describes the display formats available for viewing reports on the screen, including PDF and Excel 2000.</td>
</tr>
<tr>
<td>10</td>
<td>Linking a Report to Other Resources</td>
</tr>
<tr>
<td></td>
<td>Describes how to use StyleSheets to link reports to other reports, URLs, Maintain procedures, and JavaScript functions.</td>
</tr>
<tr>
<td>11</td>
<td>Navigating Within an HTML Report</td>
</tr>
<tr>
<td></td>
<td>Describes navigation in an HTML report including dynamic table of contents (TOC), the WebFOCUS Viewer, and linking report pages.</td>
</tr>
<tr>
<td>12</td>
<td>Bursting Reports Into Multiple HTML Files</td>
</tr>
<tr>
<td></td>
<td>Describes how to provide more efficient retrieval of reports by bursting, or separating, a single report into multiple HTML files.</td>
</tr>
<tr>
<td>13</td>
<td>Handling Records With Missing Field Values</td>
</tr>
<tr>
<td></td>
<td>Describes how missing data affects report results and how to treat and represent it.</td>
</tr>
<tr>
<td>14</td>
<td>Joining Data Sources</td>
</tr>
<tr>
<td></td>
<td>Describes how to join two or more related data sources to create a larger integrated data structure from which you can report.</td>
</tr>
<tr>
<td>15</td>
<td>Merging Data Sources</td>
</tr>
<tr>
<td></td>
<td>Describes how to merge and concatenate two or more data sources into a new permanent data source.</td>
</tr>
<tr>
<td>16</td>
<td>Formatting Reports: An Overview</td>
</tr>
<tr>
<td></td>
<td>Provides a brief overview of the formatting options available in WebFOCUS.</td>
</tr>
<tr>
<td>17</td>
<td>Creating and Managing a WebFOCUS StyleSheet</td>
</tr>
<tr>
<td></td>
<td>Describes how to create a StyleSheet and reviews general StyleSheet syntax.</td>
</tr>
<tr>
<td>18</td>
<td>Controlling Report Formatting</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Chapter/Appendix</td>
<td>Contents</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
</tr>
<tr>
<td><strong>19</strong></td>
<td>Identifying a Report Component in a WebFOCUS StyleSheet</td>
</tr>
<tr>
<td><strong>20</strong></td>
<td>Using an External Cascading Style Sheet</td>
</tr>
<tr>
<td><strong>21</strong></td>
<td>Laying Out the Report Page</td>
</tr>
<tr>
<td><strong>22</strong></td>
<td>Using Headings, Footings, Titles, and Labels</td>
</tr>
<tr>
<td><strong>23</strong></td>
<td>Formatting Report Data</td>
</tr>
<tr>
<td><strong>24</strong></td>
<td>Creating a Graph</td>
</tr>
<tr>
<td><strong>25</strong></td>
<td>Creating Financial Reports With Financial Modeling Language (FML)</td>
</tr>
<tr>
<td><strong>26</strong></td>
<td>Creating a Free-Form Report</td>
</tr>
<tr>
<td><strong>27</strong></td>
<td>Using SQL to Create Reports</td>
</tr>
<tr>
<td><strong>28</strong></td>
<td>Improving Report Processing</td>
</tr>
<tr>
<td><strong>A</strong></td>
<td>Master Files and Diagrams</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>Error Messages</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>Table Syntax Summary</td>
</tr>
</tbody>
</table>
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.

### Documentation Conventions

The following table lists and describes the conventions that apply in this manual.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>THIS TYPEFACE</strong></td>
<td>Denotes syntax that you must enter exactly as shown.</td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>this typeface</td>
<td></td>
</tr>
<tr>
<td><strong>this typeface</strong></td>
<td>Represents a placeholder (or variable), a cross-reference, or an important term.</td>
</tr>
<tr>
<td>underscore</td>
<td>Indicates a default setting.</td>
</tr>
<tr>
<td><strong>this typeface</strong></td>
<td>Highlights a file name or command. It may also indicate a button, menu item, or dialog box option you can click or select.</td>
</tr>
<tr>
<td>Key + Key</td>
<td>Indicates keys that you must press simultaneously.</td>
</tr>
<tr>
<td>{}</td>
<td>Indicates two or three choices; type one of them, not the braces.</td>
</tr>
<tr>
<td>[]</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (...).</td>
</tr>
<tr>
<td>.</td>
<td>Indicates that there are (or could be) intervening or additional commands.</td>
</tr>
</tbody>
</table>
Related Publications

To view a current listing of our publications and to place an order, visit our Technical Documentation Library, http://documentation.informationbuilders.com. You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have any questions about this product?

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

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

Call Information Builders Customer Support Service (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. 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 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?
- 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 Documentation Services staff welcomes your opinions regarding this manual. Please use the Reader Comments form at the end of this manual to communicate suggestions for improving this publication or to alert us to corrections. You can also use the Documentation Feedback form on our Web site, http://documentation.informationbuilders.com/feedback.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 World Wide Web site (http://www.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 graphs even if you do not know HTML or WebFOCUS reporting language commands and syntax.

Topics:
- Requirements for Creating a Report
- Report Types
- Reporting Tools
- 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. See the *Developing Reporting Applications* manual for information on data source locations.

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 comma-delimited format.
- Multi-dimensional data sources, such as Fusion 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. See the *Describing Data With WebFOCUS Language* manual for information on Master Files and Access Files.

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**, which display 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. For details, see Report Painter Basics in the Creating Reports With Report Painter manual. You can display these reports in formats such as HTML, Excel, and PDF.

- **Financial reports**, which are 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 FML (Financial Modeling Language) Painter, a spreadsheet-like tool that 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.

  For details about the FML language, see the Creating Financial Reports manual.

- **Free-form reports**, which present 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 1397.

- **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. You can create and customize graphs using Graph Assistant and the Graph Editor.

  For details, see Creating a Graph on page 1245 and Creating a Graph With Graph Assistant in the Creating Charts With Graph Tools manual.
SQL requests, which retrieve information using the SQL reporting language, and can directly incorporate FOCUS formatting commands. For details see Using SQL to Create Reports on page 1405.

OLAP reports, which allow users to drill down or roll up data hierarchies, pivot fields from columns to rows (or vice versa), and separate information by filtering or querying data sources based on specified criteria or thresholds.

Drill Through reports, which 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; no additional reports are run. Drill Through reports are static. You can save the PDF file to disk or distribute it via 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 623. For more information about Compound Reports, see Creating a Compound Report on page 564.

Excel Compound reports, which provide 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 on page 612.

Excel Table of Contents reports, which provide a way to generate a multiple worksheet report where a separate worksheet is generated for each value of the first BY field in the FOCUS report.

For more information, see Choosing a Display Format on page 429.

Reporting Tools

Developer Studio provides powerful reporting tools for analyzing and formatting information, and building report requests.


The graphical reporting tool is integrated with two language-based tools, which can be used by power reporters, analytical reporters, administrators, and developers:

- The Text editor for viewing the code generated by the graphical tools and for creating procedures and other types of application files. The Text editor can be accessed from the graphical tools and the graphical tools can be launched directly from the Text editor.

- The Command Console for issuing commands for monitoring messages, and for debugging running procedures. For details, see the Developing Reporting Applications manual.
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; that is, 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. (Except where otherwise noted, see Report Painter Basics in the Creating Reports With Report Painter manual for details about these and many other reporting features.)

- **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, horizontally, or 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. There are two aspects of a successful report: 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 193.

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

For details, see Joining Data Sources on page 723.

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 351.

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

**How to:**

Begin 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.

**Note:** In Developer Studio, the QUIT and RUN commands are only supported by entering these commands in the Command Console.

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.
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 using Developer Studio graphical tools, or 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

<table>
<thead>
<tr>
<th>DIV</th>
<th>DEPT</th>
<th>Education Cost</th>
<th>Current Salary</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>ADMIN SERVICES</td>
<td>1,250.00</td>
<td>$25,400.00</td>
<td>4.92</td>
</tr>
<tr>
<td></td>
<td>PROGRAMMING &amp; DVLPMT</td>
<td>1,480.00</td>
<td>$40,900.00</td>
<td>3.86</td>
</tr>
<tr>
<td>CORP</td>
<td>ACCOUNTING</td>
<td>2,050.00</td>
<td>$62,500.00</td>
<td>3.28</td>
</tr>
<tr>
<td>NE</td>
<td>CUSTOMER SUPPORT</td>
<td>1,800.00</td>
<td>$19,300.00</td>
<td>9.33</td>
</tr>
<tr>
<td></td>
<td>MARKETING</td>
<td>3,100.00</td>
<td>$32,300.00</td>
<td>9.60</td>
</tr>
<tr>
<td></td>
<td>SALES</td>
<td>1,800.00</td>
<td>$43,600.00</td>
<td>4.13</td>
</tr>
<tr>
<td>SE</td>
<td>CONSULTING</td>
<td>3,350.00</td>
<td>$35,900.00</td>
<td>9.33</td>
</tr>
<tr>
<td>WE</td>
<td>MARKETING</td>
<td>9,850.00</td>
<td>$102,200.00</td>
<td>9.64</td>
</tr>
<tr>
<td></td>
<td>PROGRAMMING &amp; DVLPMT</td>
<td>5,310.00</td>
<td>$42,900.00</td>
<td>12.38</td>
</tr>
<tr>
<td></td>
<td>SALES</td>
<td>4,500.00</td>
<td>$100,500.00</td>
<td>4.48</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>34,590.00</td>
<td>$505,500.00</td>
<td>6.84</td>
</tr>
</tbody>
</table>
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

There are two aspects of a successful report: 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 1205.
Customizing a Report

- Determine the width of a report column. For details, see Formatting Report Data on page 1205.

- Dynamically control the display of subtotals, headings, and footings based on conditions you define. For details, see Controlling Report Formatting on page 825.

- 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 1205.

- Format your report using external cascading style sheets. For details, see Using an External Cascading Style Sheet on page 891.

- 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 519.
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 Acrobat Reader or Microsoft Excel), within a Web browser. Reports run locally in Developer Studio appear in the browser.

  You can also view reports outside of the browser in a standalone helper application such as Adobe Acrobat 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 1245.

- **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 351.

- **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 Acrobat 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).

Topics:
- 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
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:**

```plaintext
display [THE] [SEG.] fieldname1 [AND] [THE] fieldname2 ...
```

or

```plaintext
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 is a re-defines of a real field.

## Displaying Individual Values

### In this section:
- Displaying All Fields
- Displaying All Fields in a Segment
- Displaying the Structure and Retrieval Order of a Multi-Path Data Source

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 47.

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 1431, 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 81.

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 [How to Use Display Commands in a Request](#) on page 44.
**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.

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

The following shows the report output.

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEVENS</td>
<td>ALFRED</td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
</tr>
<tr>
<td>SMITH</td>
<td>RICHARD</td>
</tr>
<tr>
<td>BANNING</td>
<td>JOHN</td>
</tr>
<tr>
<td>IRVING</td>
<td>JOAN</td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
</tr>
</tbody>
</table>

**Example: Listing Records**

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

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

The following shows the report output.

<table>
<thead>
<tr>
<th>LIST</th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>STEVENS</td>
<td>ALFRED</td>
</tr>
<tr>
<td>2</td>
<td>SMITH</td>
<td>MARY</td>
</tr>
<tr>
<td>3</td>
<td>JONES</td>
<td>DIANE</td>
</tr>
<tr>
<td>4</td>
<td>SMITH</td>
<td>RICHARD</td>
</tr>
<tr>
<td>5</td>
<td>BANNING</td>
<td>JOHN</td>
</tr>
<tr>
<td>6</td>
<td>IRVING</td>
<td>JOAN</td>
</tr>
<tr>
<td>7</td>
<td>ROMANS</td>
<td>ANTHONY</td>
</tr>
<tr>
<td>8</td>
<td>MCCOY</td>
<td>JOHN</td>
</tr>
<tr>
<td>9</td>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
</tr>
<tr>
<td>10</td>
<td>MCKNIGHT</td>
<td>ROGER</td>
</tr>
<tr>
<td>11</td>
<td>GREENSPAN</td>
<td>MARY</td>
</tr>
<tr>
<td>12</td>
<td>CROSS</td>
<td>BARBARA</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>LIST</th>
<th>COURSE_CODE</th>
<th>COURSE_NAME</th>
<th>DATE_ATTEND</th>
<th>EMP_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
<td>FILE DESCRPT &amp; MAINT</td>
<td>83/01/04</td>
<td>212289111</td>
</tr>
<tr>
<td>2</td>
<td>101</td>
<td>FILE DESCRPT &amp; MAINT</td>
<td>82/05/25</td>
<td>117593129</td>
</tr>
<tr>
<td>3</td>
<td>101</td>
<td>FILE DESCRPT &amp; MAINT</td>
<td>82/05/25</td>
<td>071382660</td>
</tr>
<tr>
<td>4</td>
<td>101</td>
<td>FILE DESCRPT &amp; MAINT</td>
<td>81/11/15</td>
<td>451123478</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>FILE DESCRPT &amp; MAINT</td>
<td>81/11/15</td>
<td>112847612</td>
</tr>
<tr>
<td>6</td>
<td>102</td>
<td>BASIC REPORT PREP NON-PROG</td>
<td>82/07/12</td>
<td>326179357</td>
</tr>
<tr>
<td>7</td>
<td>103</td>
<td>BASIC REPORT PREP NON-PROG</td>
<td>83/01/05</td>
<td>212289111</td>
</tr>
<tr>
<td>8</td>
<td>103</td>
<td>BASIC REPORT PREP NON-PROG</td>
<td>82/05/26</td>
<td>117593129</td>
</tr>
<tr>
<td>9</td>
<td>103</td>
<td>BASIC REPORT PREP NON-PROG</td>
<td>81/11/16</td>
<td>112847612</td>
</tr>
<tr>
<td>10</td>
<td>104</td>
<td>FILE DESC &amp; MAINT NON-PROG</td>
<td>82/07/14</td>
<td>326179357</td>
</tr>
<tr>
<td>11</td>
<td>106</td>
<td>TIMESHARING WORKSHOP</td>
<td>82/07/15</td>
<td>326179357</td>
</tr>
<tr>
<td>12</td>
<td>202</td>
<td>WHAT'S NEW IN FOCUS</td>
<td>82/10/28</td>
<td>326179357</td>
</tr>
<tr>
<td>13</td>
<td>301</td>
<td>DECISION SUPPORT WORKSHOP</td>
<td>82/09/03</td>
<td>326179357</td>
</tr>
<tr>
<td>14</td>
<td>107</td>
<td>BASIC REPORT PREP DP MGRS</td>
<td>82/08/02</td>
<td>818692173</td>
</tr>
<tr>
<td>15</td>
<td>302</td>
<td>HOST LANGUAGE INTERFACE</td>
<td>82/10/21</td>
<td>818692173</td>
</tr>
<tr>
<td>16</td>
<td>108</td>
<td>BASIC RPT NON-DP MGRS</td>
<td>82/10/10</td>
<td>315548712</td>
</tr>
<tr>
<td>17</td>
<td>108</td>
<td>BASIC RPT NON-DP MGRS</td>
<td>82/08/24</td>
<td>119265415</td>
</tr>
<tr>
<td>18</td>
<td>201</td>
<td>ADVANCED TECHNIQUES</td>
<td>82/07/26</td>
<td>117593129</td>
</tr>
<tr>
<td>19</td>
<td>203</td>
<td>FOCUS INTERNALS</td>
<td>82/10/28</td>
<td>117593129</td>
</tr>
</tbody>
</table>

Displaying All Fields in a Segment

**How to:** Display 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.

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

If you are using Developer Studio, you can issue these commands from the Developer Studio Console.

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

OINFO
01 S1
**************
*ORDER_NUM **I
*STORE_CODE **I
*PLANT **I
*ORDER_DATE **
*            *
**************
**************

+-----------------+
 I                 I
 I STOSEG         I PINFO
02 I KU 03 I S1

STORE_CODE :K *PRODUCT **I
STORENAME : *QUANTITY **
STATE : *LINEPRICE **
: : *            *
: : *            *
: : *            *
: : *            *
: : *            *
: : *            *
: : *            *
: : *            *
: : *            *
: : *            *
: : *            *
: : *            *
: : *            *
JOINED CENTCOMPFO**************

I
 I
 I INVSEG
04 I KU

PRODUCT :K
PRODNAME :
QTY_IN_STOCK:
PRICE :
 : :
: : :
: : :
JOINED CENTINV FOCUS A1
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:

```sql
check file employee picture retrieve
NUMBER OF ERRORS= 0
NUMBER OF SEGMENTS= 11 ( REAL= 6 VIRTUAL= 5 )
NUMBER OF INDEXES= 0 FILES= 3
TOTAL LENGTH OF ALL FIELDS= 385
SECTION 01
    RETRIEVAL VIEW OF FOCUS   FILE EMPLOYEE ON 12-29-93 AT 14:42:10

EMPINFO
01 S1
    =EMP_ID ==
    =LAST_NAME ==
    =FIRST_NAME ==
    =HIRE_DATE ==

    ************
    1
    1
    1 FUNCTION
02 I U
    ************
    =BANK_NAME ==
    =BANK_CODE ==
    =BANK_ACCT ==
    =EFFECT_DATE ==

    ************
7
7

    ************
7
7

I PAYINFO I ADDRESS I SALINFO I ATTONES
03 I SH1 07 I S1 00 I SH1 10 I RM

    ************
    1
    1
    1

I JOBSEG I DEDUCT I COURSEG
04 I KLU

    ************
    1
    1
    1

I SECSEG
05 I KLU

    ************
    1
    1
    1

I JOBFIL
06 I KL

    ************
    1
    1
    1

I SKILLSD

    ************
    1
    1
    1
```
**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
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, it displays the last value retrieved from the data source.

For SUM, WRITE, and ADD syntax, see *How to Use Display Commands in a Request* on page 44.

**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:** If you are using the external sorting product DFSORT, you can set the SUMPREFIX parameter to FST or LST to control the sort order. For details, see *Sorting Tabular Reports* on page 81.
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 81.

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

For COUNT syntax, see How to Use Display Commands in a Request on page 44.

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 193. 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:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>6</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>6</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>DEPARTMENT</th>
<th>JOBCODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>COUNT</td>
<td>COUNT</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>19</td>
</tr>
</tbody>
</table>

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 697.
**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 JOBFILE and EDUCFILE 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**

**How to:**

Set the 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

```plaintext
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 | COUNT | ****
value
```

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

```
TABLE FILE filename
COUNT Fldxx
BY Fldyy
END

Fldxx | COUNT | 999999999
value
```

**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.
The maximum number of display fields you can include in a report request is approximately 1024 (495 for MATCH requests). However, when adding fields to a request, it is important to be aware that the allowable number of fields includes all named fields, whether printed or not, including data source fields, temporary fields (virtual fields and calculated values), certain internal fields (for example, TAMPA Geno), and fields used in headings and footings. The total does not include sort fields.

This field limit is also affected by the combined length of fields in the request. The field limit represents the maximum number of fields allowed when each field has the smallest length possible (A4 ACTUAL). Longer field lengths reduce the total number of printable fields.

When you create a report, the fields specified in the request are stored in a 32K (3956 bytes for MATCH requests) data area. The capacity of the data area is affected by a number of factors:

- Every field is rounded up to a full word boundary (a multiple of 4).
- Every field is associated with a four-byte counter field, which affects the total number of bytes in this data area.
- Field prefixes and formatting options affect the available data area.

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.
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

**How to:**
Use Prefix Operators

**Reference:**
Usage Notes for Prefix Operators
Functions You Can Perform With Prefix Operators

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 81.

- 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.

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

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

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASQ.</td>
<td>Computes the average sum of squares for standard deviation in statistical analysis.</td>
</tr>
<tr>
<td>AVE.</td>
<td>Computes the average value of the field.</td>
</tr>
<tr>
<td>CNT.</td>
<td>Counts the number of occurrences of the field. The data type of the result is always Integer.</td>
</tr>
<tr>
<td>CNT.DST.</td>
<td>Counts the number of distinct values within a field.</td>
</tr>
<tr>
<td>CT.</td>
<td>Produces a cumulative total of the specified field. This operator only applies when used in subfootings. For details, see Using Headings, Footings, Titles, and Labels on page 1053.</td>
</tr>
<tr>
<td>DST.</td>
<td>Determines the total number of distinct values in a single pass of a data source.</td>
</tr>
<tr>
<td>FST.</td>
<td>Generates the first physical instance of the field. Can be used with numeric or text fields.</td>
</tr>
<tr>
<td>LST.</td>
<td>Generates the last physical instance of the field. Can be used with numeric or text fields.</td>
</tr>
<tr>
<td>MAX.</td>
<td>Generates the maximum value of the field.</td>
</tr>
<tr>
<td>MIN.</td>
<td>Generates the minimum value of the field.</td>
</tr>
<tr>
<td>PCT.</td>
<td>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.</td>
</tr>
<tr>
<td>PCT.CNT.</td>
<td>Computes a field percentage based on the number of instances found. The format of the result is always F6.2 and cannot be reformatted.</td>
</tr>
</tbody>
</table>
### 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.

```
  AVE      ED_HRS
 DEPARTMENT ------
    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).

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

If the field format is integer and you get a large set of numbers, the ASQ. result may be negative as a result of field overflow.

**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.

<table>
<thead>
<tr>
<th>CITY</th>
<th>DELIVER_AMT</th>
<th>DELIVER_AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW YORK</td>
<td>300</td>
<td>980</td>
</tr>
<tr>
<td>NEWARK</td>
<td>60</td>
<td>900</td>
</tr>
<tr>
<td>STAMFORD</td>
<td>430</td>
<td>3637</td>
</tr>
<tr>
<td>UNIONDALE</td>
<td>80</td>
<td>1600</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>MAX SALARY</th>
<th>MIN SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>$29,700.00</td>
<td>$8,650.00</td>
</tr>
</tbody>
</table>
Calculating Column and Row Percents

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 Percents

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:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>ED_HRS</th>
<th>PCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANNING</td>
<td>.00</td>
<td>.00</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>75.00</td>
<td>21.37</td>
</tr>
<tr>
<td>CROSS</td>
<td>45.00</td>
<td>12.82</td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>25.00</td>
<td>7.12</td>
</tr>
<tr>
<td>IRVING</td>
<td>30.00</td>
<td>8.55</td>
</tr>
<tr>
<td>JONES</td>
<td>50.00</td>
<td>14.25</td>
</tr>
<tr>
<td>MCCOY</td>
<td>.00</td>
<td>.00</td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>50.00</td>
<td>14.25</td>
</tr>
<tr>
<td>ROMANS</td>
<td>5.00</td>
<td>1.42</td>
</tr>
<tr>
<td>SMITH</td>
<td>46.00</td>
<td>13.11</td>
</tr>
<tr>
<td>STEVENS</td>
<td>25.00</td>
<td>7.12</td>
</tr>
<tr>
<td>TOTAL</td>
<td>351.00</td>
<td>100.00</td>
</tr>
</tbody>
</table>

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 Percents

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:

<table>
<thead>
<tr>
<th>CITY</th>
<th>NEW YORK</th>
<th>STAMFORD</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PROD_CODE</td>
<td>UNIT_SOLD</td>
<td>RPCT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RPCT</td>
</tr>
<tr>
<td>B10</td>
<td>30</td>
<td>33</td>
<td>60</td>
</tr>
<tr>
<td>B12</td>
<td>.</td>
<td>.</td>
<td>40</td>
</tr>
<tr>
<td>B17</td>
<td>20</td>
<td>40</td>
<td>29</td>
</tr>
<tr>
<td>B20</td>
<td>15</td>
<td>100</td>
<td>.</td>
</tr>
<tr>
<td>C13</td>
<td>.</td>
<td>.</td>
<td>25</td>
</tr>
<tr>
<td>C17</td>
<td>12</td>
<td>100</td>
<td>.</td>
</tr>
<tr>
<td>C7</td>
<td>.</td>
<td>.</td>
<td>45</td>
</tr>
<tr>
<td>D12</td>
<td>20</td>
<td>42</td>
<td>27</td>
</tr>
<tr>
<td>E1</td>
<td>30</td>
<td>100</td>
<td>.</td>
</tr>
<tr>
<td>E2</td>
<td>.</td>
<td>.</td>
<td>80</td>
</tr>
<tr>
<td>E3</td>
<td>35</td>
<td>33</td>
<td>70</td>
</tr>
</tbody>
</table>

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:

\[ \text{PCT.CNT.} \text{fieldname} \]

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:

<table>
<thead>
<tr>
<th>PCT.CNT</th>
<th>EMP_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPARTMENT</td>
<td>------</td>
</tr>
<tr>
<td>MIS</td>
<td>50.00</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>50.00</td>
</tr>
</tbody>
</table>

Aggregating and Listing Unique Values

**How to:**

Use the Distinct Operator

**Reference:**

Distinct Operator Limitations

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.
**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 excludes the second records for values 50.00, 25.00, and .0, resulting in nine unique ED_HRS values.

When used with PRINT, DST. acts in the same manner as a BY phrase. It can be attached to several PRINT display fields, but not more than 31 (the current limit for BY sort fields with PRINT).
DST. display fields must precede all non-distinct display fields named by the PRINT command. If this rule is not observed, the following error is displayed:

(FOC1855) DISTINCT FIELDS MUST PRECEDE THE NONDISTINCT ONES

**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 or TABLEF command:
  
  (FOC1854) THE DST OPERATOR IS ONLY SUPPORTED IN TABLE REQUESTS

- The following error occurs if you code more than one DST. operator for the SUM command:
  
  (FOC1856) ONLY ONE DISTINCT FIELD IS ALLOWED IN AGGREGATION

- 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 in an ACROSS or FOR phrase:
  
  (FOC1864) THE DST OPERATOR IS NOT SUPPORTED FOR ACROSS OR FOR

- 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.

- TABLE requests that contain the DST. operator are not candidates for AUTOTABLEF.
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 1427.

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     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:

```
TABLE FILE EMPLOYEE
SUM FST.SALARY
BY EMP_ID
END
```
The output is:

<table>
<thead>
<tr>
<th>EMP_ID</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>071382660</td>
<td>$11,000.00</td>
</tr>
<tr>
<td>112847612</td>
<td>$13,200.00</td>
</tr>
<tr>
<td>117593129</td>
<td>$18,480.00</td>
</tr>
<tr>
<td>119265415</td>
<td>$9,500.00</td>
</tr>
<tr>
<td>119329144</td>
<td>$29,700.00</td>
</tr>
<tr>
<td>123764317</td>
<td>$26,862.00</td>
</tr>
<tr>
<td>126724188</td>
<td>$21,120.00</td>
</tr>
<tr>
<td>219984371</td>
<td>$18,480.00</td>
</tr>
<tr>
<td>326179357</td>
<td>$21,780.00</td>
</tr>
<tr>
<td>451123478</td>
<td>$16,100.00</td>
</tr>
<tr>
<td>543729165</td>
<td>$9,000.00</td>
</tr>
<tr>
<td>818692173</td>
<td>$27,062.00</td>
</tr>
</tbody>
</table>

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.

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

The output is:

<table>
<thead>
<tr>
<th>Product Code</th>
<th>Unit Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>Price</td>
</tr>
<tr>
<td>10</td>
<td>660.00</td>
</tr>
</tbody>
</table>
**Example:**  **Summing Values With SUM**

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

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

The output is:

<table>
<thead>
<tr>
<th>Product Code</th>
<th>Unit Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>660.00</td>
</tr>
</tbody>
</table>

**Example:**  **Summing Values With TOT**

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

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

The output is:

<table>
<thead>
<tr>
<th>Department</th>
<th>Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>SMITH</td>
</tr>
<tr>
<td></td>
<td>JONES</td>
</tr>
<tr>
<td></td>
<td>MCCOY</td>
</tr>
<tr>
<td></td>
<td>BLACKWOOD</td>
</tr>
<tr>
<td></td>
<td>GREENSPAN</td>
</tr>
<tr>
<td></td>
<td>CROSS</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>STEVENS</td>
</tr>
<tr>
<td></td>
<td>SMITH</td>
</tr>
<tr>
<td></td>
<td>BANNING</td>
</tr>
<tr>
<td></td>
<td>IRVING</td>
</tr>
<tr>
<td></td>
<td>ROMANS</td>
</tr>
<tr>
<td></td>
<td>MCKNIGHT</td>
</tr>
</tbody>
</table>

Total salaries equal: $222,284.00
Ranking Sort Field Values With RNK.

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

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:

<table>
<thead>
<tr>
<th>DEPT</th>
<th>YRS_SERVICE</th>
<th>SALARY</th>
<th>RANKING BY SERVICE</th>
<th>SALARY RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARKETING</td>
<td>17</td>
<td>$55,500.00</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>$55,500.00</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>$62,500.00</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>$62,500.00</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>$62,500.00</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>$58,800.00</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>$52,000.00</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>$35,200.00</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>$32,300.00</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>SALES</td>
<td>17</td>
<td>$50,500.00</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>$50,500.00</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>$43,400.00</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>$43,000.00</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>$30,500.00</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
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:

```sql
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:

<table>
<thead>
<tr>
<th>YRS_SERVICE</th>
<th>SALARY</th>
<th>LASTNAME</th>
<th>FIRSTNAME</th>
<th>RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>$115,000.00</td>
<td>LASTRA</td>
<td>KAREN</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>$80,500.00</td>
<td>NOZAWA</td>
<td>JIM</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>$83,000.00</td>
<td>SANCHEZ</td>
<td>EVELYN</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>$70,000.00</td>
<td>CASSANOVA</td>
<td>LOIS</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>$62,500.00</td>
<td>HIRSCHMAN</td>
<td>ROSE</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>$50,500.00</td>
<td>LEWIS</td>
<td>CASSANDRA</td>
<td>2</td>
</tr>
</tbody>
</table>

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:

```sql
DEFINE FILE EMPDATA
   TODAY/YYMD = &YYMD;
   YRS_SERVICE/I9 = DATEDIF(HIREDATE,TODAY,'Y');
END
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
```
Table: Salary and Years of Service by Department

<table>
<thead>
<tr>
<th>DEPT</th>
<th>SALARY</th>
<th>YRS_SERVICE</th>
<th>RANK</th>
<th>RANK</th>
<th>SALARY</th>
<th>YRS_SERVICE</th>
<th>FLAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARKETING</td>
<td>$32,300.00</td>
<td>16</td>
<td>1</td>
<td></td>
<td>$35,200.00</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$43,400.00</td>
<td>15</td>
<td>2</td>
<td></td>
<td>$50,500.00</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>$52,000.00</td>
<td>16</td>
<td>4</td>
<td></td>
<td>$55,500.00</td>
<td>17</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>$58,800.00</td>
<td>16</td>
<td>6</td>
<td></td>
<td>$55,500.00</td>
<td>17</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>$62,500.00</td>
<td>16</td>
<td>7</td>
<td></td>
<td>$55,500.00</td>
<td>17</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>$75,000.00</td>
<td>17</td>
<td>8</td>
<td></td>
<td>$70,000.00</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>$115,000.00</td>
<td>17</td>
<td>9</td>
<td></td>
<td>$54,100.00</td>
<td>17</td>
<td>9</td>
</tr>
</tbody>
</table>

Displaying Pop-up Field Descriptions for Column Titles

How to:

Use the POPUPDESC Command

Reference:

Distributing Reports With Pop-up Field Descriptions Using ReportCaster

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 *Describing an Individual Field* 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.
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_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_html` is the site-customized Web server alias pointing to the WebFOCUS76/ibi_html directory (where ibi_html is the default value).

For more information about coding reports for use with ReportCaster, see the Tips and Techniques for Coding a ReportCaster Report appendix in your ReportCaster manual.
Displaying Pop-up Field Descriptions for Column Titles
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.

**Topics:**

- Sorting Tabular Reports Overview
- Sorting Rows
- Sorting Columns
- Manipulating Display Field Values in a Sort Group
- Creating a Matrix Report
- Specifying the Sort Order
- Ranking Sort Field Values
- Grouping Numeric Data Into Ranges
- Restricting Sort Field Values by Highest/Lowest Rank
- Sorting and Aggregating Report Columns
- Hiding Sort Values
- Sorting With Multiple Display Commands
- Improving Efficiency With External Sorts
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.

Additional sorting options include:

- Sorting from low to high values or from high to low values, and defining your own sorting 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, showing 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 124.

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 32 BY phrases per report request (31 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 on page 85.

**Syntax:**

```
BY sortfield
```

where:

`sortfield`

Is the name of the sort field.

**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 and VM 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 100.

- 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 351).

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 on page 85.

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:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>EMP_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>112847612</td>
</tr>
<tr>
<td></td>
<td>117593129</td>
</tr>
<tr>
<td></td>
<td>219984371</td>
</tr>
<tr>
<td></td>
<td>326179357</td>
</tr>
<tr>
<td></td>
<td>543729165</td>
</tr>
<tr>
<td></td>
<td>818692173</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>071382660</td>
</tr>
<tr>
<td></td>
<td>119265415</td>
</tr>
<tr>
<td></td>
<td>119329144</td>
</tr>
<tr>
<td></td>
<td>123764317</td>
</tr>
<tr>
<td></td>
<td>126724188</td>
</tr>
<tr>
<td></td>
<td>451123478</td>
</tr>
</tbody>
</table>
Displaying All Vertical (BY) Sort Field Values

**How to:**
Display All Vertical (BY) Sort Field Values

Within a vertical sort group, the sort field value displays only on the first line of the rows for its sort group, and on the first line of a page. However, using the SET BYDISPLAY command, you can display the appropriate BY field on every row of a report produced in a styled output format.

Although SET BYDISPLAY is supported for all styled output formats, it is especially important for making report output more usable by Excel, which cannot sort columns properly when they have blank values in some rows.

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 Display All Vertical (BY) Sort Field Values

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

or

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

where:

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

- **ON**
  - Displays the associated BY field value on every line of report output produced in a styled format.
**Example:** Displaying All Vertical (BY) Sort Field Values

The following illustrates how you can display every instance of a vertical (BY) sort field value in a styled report using SET BYDISPLAY.

```
SET BYDISPLAY = ON
TABLE FILE CENTHR
PRINT LNAME
BY FNAME
WHERE FNAME EQ 'CAROLYN' OR 'DAVID' ON
TABLE PCHOLD FORMAT EXL2K
END
```

The output is:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>First</td>
<td>Last</td>
<td>Plant</td>
<td>Pay</td>
</tr>
<tr>
<td></td>
<td>Name</td>
<td>Name</td>
<td>Location</td>
<td>Level</td>
</tr>
<tr>
<td>2</td>
<td>CAROLYN</td>
<td>MILLER</td>
<td>LA</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>CAROLYN</td>
<td>STRICKER</td>
<td>SEA</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>DAVID</td>
<td>JAMES</td>
<td>ORL</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>DAVID</td>
<td>KERNER</td>
<td>SEA</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>DAVID</td>
<td>MONTGOMERY</td>
<td>ORL</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>DAVID</td>
<td>RODRIGUEZ</td>
<td>SEA</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>DAVID</td>
<td>TEPPER</td>
<td>BOS</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>DAVID</td>
<td>HAZARD</td>
<td>BOS</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>DAVID</td>
<td>SEGAL</td>
<td>BOS</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>DAVID</td>
<td>KNUDSEN</td>
<td>LA</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>DAVID</td>
<td>PARADISE</td>
<td>LA</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>DAVID</td>
<td>HENSLEE</td>
<td>DAL</td>
<td>5</td>
</tr>
</tbody>
</table>

**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.

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

The output is:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>LAST_NAME</th>
<th>CURR_SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>BLACKWOOD</td>
<td>$21,780.00</td>
</tr>
<tr>
<td></td>
<td>CROSS</td>
<td>$27,062.00</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>BANNING</td>
<td>$29,700.00</td>
</tr>
<tr>
<td></td>
<td>IRVING</td>
<td>$26,862.00</td>
</tr>
</tbody>
</table>

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

**How to:**

Display Data Excluded by a Sort Phrase

**Reference:**

Usage Notes for PLUS OTHERS

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} nsrtfield [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:

<table>
<thead>
<tr>
<th>ED_HRS</th>
<th>CURR_SAL</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>75.00</td>
<td>$21,780.00</td>
<td>BLACKWOOD</td>
</tr>
<tr>
<td>50.00</td>
<td>$18,480.00</td>
<td>JONES</td>
</tr>
<tr>
<td></td>
<td>$16,100.00</td>
<td>MCKNIGHT</td>
</tr>
<tr>
<td>Others</td>
<td>$165,924.00</td>
<td></td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>ED_HRS</th>
<th>CURR_SAL</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.00</td>
<td>$18,480.00</td>
<td>JONES</td>
</tr>
<tr>
<td>25.00</td>
<td>$11,000.00</td>
<td>STEVENS</td>
</tr>
<tr>
<td></td>
<td>$13,200.00</td>
<td>SMITH</td>
</tr>
<tr>
<td></td>
<td>$26,862.00</td>
<td>IRVING</td>
</tr>
<tr>
<td></td>
<td>$9,000.00</td>
<td>GREENSPAN</td>
</tr>
<tr>
<td></td>
<td>$27,062.00</td>
<td>CROSS</td>
</tr>
<tr>
<td>Others</td>
<td>$78,800.00</td>
<td></td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>ED_HRS</th>
<th>CURR_SAL</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>.00</td>
<td>$9,500.00</td>
<td>SMITH</td>
</tr>
<tr>
<td>$29,700.00</td>
<td>BANNING</td>
<td></td>
</tr>
<tr>
<td>$21,120.00</td>
<td>ROMANS</td>
<td></td>
</tr>
<tr>
<td>$18,480.00</td>
<td>MCCOY</td>
<td></td>
</tr>
<tr>
<td>25.00</td>
<td>$11,000.00</td>
<td>STEVENS</td>
</tr>
<tr>
<td>$13,200.00</td>
<td>SMITH</td>
<td></td>
</tr>
<tr>
<td>$26,862.00</td>
<td>IRVING</td>
<td></td>
</tr>
<tr>
<td>$9,000.00</td>
<td>GREENSPAN</td>
<td></td>
</tr>
<tr>
<td>$27,062.00</td>
<td>CROSS</td>
<td></td>
</tr>
<tr>
<td>Others</td>
<td>$56,360.00</td>
<td></td>
</tr>
</tbody>
</table>

**Sorting Columns**

**In this section:**
- Controlling Underlines for ACROSS Objects
- Using Multiple Horizontal (ACROSS) Sort Fields
- Collapsing PRINT With ACROSS

**How to:**
- Sort Columns

**Reference:**
- Usage Notes for Sorting Columns

You can sort report information horizontally using the ACROSS phrase. This creates columns in your report. You can have up to five ACROSS phrases per report request. Each ACROSS phrase can generate up to 255 columns of data. 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.
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 261.

**Syntax:**

How to Sort Columns

```plaintext
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 351.

- 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 929.

- Sort phrases cannot contain format information for fields.

- If you specify several sort fields when reporting from a multi-path data source, all the sort fields must be in the same path.

- 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

  ```plaintext
  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 Underlines for ACROSS Objects

How to: Control Underlining for ACROSS Objects

The SET ACROSSLINE command allows users to turn off/on optional underlining in reports to highlight ACROSS objects. The feature is only available for report output formats WP, HTML, and PDF.

Syntax: How to Control Underlining for ACROSS Objects

Issue the following command in any supported profile, or in a FOCEXEC:

```
SET ACROSSLINE= (ON|OFF|SKIP)
```

where:

ON
- Underlines ACROSS objects in report headings with a dashed line. ON is the default value.

OFF
- Replaces the underline with a blank line.

SKIP
- Specifies no underline and no blank line.
**Example:** Underlining ACROSS Objects With a Dashed Line (SET ACROSSLINE=ON)

```
SET ACROSSLINE=ON
TABLE FILE GGSALES
SUM UNITS BY PRODUCT
ACROSS REGION
END
```

The output is:

<table>
<thead>
<tr>
<th>Product</th>
<th>Midwest</th>
<th>Northeast</th>
<th>Southeast</th>
<th>West</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscotti</td>
<td>86105</td>
<td>145242</td>
<td>119594</td>
<td>70436</td>
</tr>
<tr>
<td>Cappuccino</td>
<td></td>
<td>44785</td>
<td>73264</td>
<td>71168</td>
</tr>
<tr>
<td>Coffee Grinder</td>
<td>50393</td>
<td>40977</td>
<td>47083</td>
<td>48081</td>
</tr>
<tr>
<td>Coffee Pot</td>
<td>47156</td>
<td>46185</td>
<td>49922</td>
<td>47432</td>
</tr>
<tr>
<td>Croissant</td>
<td>139182</td>
<td>137394</td>
<td>156456</td>
<td>197022</td>
</tr>
<tr>
<td>Espresso</td>
<td>101154</td>
<td>68127</td>
<td>68030</td>
<td>71675</td>
</tr>
<tr>
<td>Latte</td>
<td>231623</td>
<td>222866</td>
<td>209654</td>
<td>213920</td>
</tr>
<tr>
<td>Mug</td>
<td>86718</td>
<td>91497</td>
<td>88474</td>
<td>93881</td>
</tr>
<tr>
<td>Scone</td>
<td>116127</td>
<td>70732</td>
<td>73779</td>
<td>72776</td>
</tr>
<tr>
<td>Thermos</td>
<td>46587</td>
<td>48870</td>
<td>48976</td>
<td>45648</td>
</tr>
</tbody>
</table>

**Example:** Removing Underlines for ACROSS Objects (SET ACROSSLINE=SKIP)

```
SET ACROSSLINE=SKIP
TABLE FILE GGSALES
SUM UNITS BY PRODUCT
ACROSS REGION
END
```

The output is:

<table>
<thead>
<tr>
<th>Product</th>
<th>Midwest</th>
<th>Northeast</th>
<th>Southeast</th>
<th>West</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscotti</td>
<td>86105</td>
<td>145242</td>
<td>119594</td>
<td>70436</td>
</tr>
<tr>
<td>Cappuccino</td>
<td></td>
<td>44785</td>
<td>73264</td>
<td>71168</td>
</tr>
<tr>
<td>Coffee Grinder</td>
<td>50393</td>
<td>40977</td>
<td>47083</td>
<td>48081</td>
</tr>
<tr>
<td>Coffee Pot</td>
<td>47156</td>
<td>46185</td>
<td>49922</td>
<td>47432</td>
</tr>
<tr>
<td>Croissant</td>
<td>139182</td>
<td>137394</td>
<td>156456</td>
<td>197022</td>
</tr>
<tr>
<td>Espresso</td>
<td>101154</td>
<td>68127</td>
<td>68030</td>
<td>71675</td>
</tr>
<tr>
<td>Latte</td>
<td>231623</td>
<td>222866</td>
<td>209654</td>
<td>213920</td>
</tr>
<tr>
<td>Mug</td>
<td>86718</td>
<td>91497</td>
<td>88474</td>
<td>93881</td>
</tr>
<tr>
<td>Scone</td>
<td>116127</td>
<td>70732</td>
<td>73779</td>
<td>72776</td>
</tr>
<tr>
<td>Thermos</td>
<td>46587</td>
<td>48870</td>
<td>48976</td>
<td>45648</td>
</tr>
</tbody>
</table>
Example: Replacing the Underline With a Blank Line (SET ACROSSLINE=OFF)

```
SET ACROSSLINE=OFF
TABLE FILE GGSALES
SUM UNITS BY PRODUCT
ACROSS REGION
END
```

Turning ACROSSLINE=OFF replaces the (default) dashed line with an extra blank line between the report heading and the detail lines:

<table>
<thead>
<tr>
<th>Product</th>
<th>Midwest</th>
<th>Northeast</th>
<th>Southeast</th>
<th>West</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscotti</td>
<td>86105</td>
<td>145242</td>
<td>119594</td>
<td>70436</td>
</tr>
<tr>
<td>Cappuccino</td>
<td>.</td>
<td>44785</td>
<td>73264</td>
<td>71168</td>
</tr>
<tr>
<td>Coffee Grinder</td>
<td>50393</td>
<td>40977</td>
<td>47083</td>
<td>48081</td>
</tr>
<tr>
<td>Coffee Pot</td>
<td>47156</td>
<td>46185</td>
<td>49922</td>
<td>47432</td>
</tr>
<tr>
<td>Croissant</td>
<td>139182</td>
<td>137394</td>
<td>156456</td>
<td>197022</td>
</tr>
<tr>
<td>Espresso</td>
<td>101154</td>
<td>68127</td>
<td>68030</td>
<td>71675</td>
</tr>
<tr>
<td>Latte</td>
<td>231623</td>
<td>222866</td>
<td>209654</td>
<td>213920</td>
</tr>
<tr>
<td>Mug</td>
<td>86718</td>
<td>91497</td>
<td>88474</td>
<td>93881</td>
</tr>
<tr>
<td>Scone</td>
<td>116127</td>
<td>70732</td>
<td>73779</td>
<td>72776</td>
</tr>
<tr>
<td>Thermos</td>
<td>46587</td>
<td>48870</td>
<td>48976</td>
<td>45648</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>CURR_JOBCODE</th>
<th>PRODUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>A17</td>
<td>$27,062.00</td>
</tr>
<tr>
<td></td>
<td>B04</td>
<td>$21,780.00</td>
</tr>
<tr>
<td></td>
<td>A15</td>
<td>$26,862.00</td>
</tr>
<tr>
<td></td>
<td>A17</td>
<td>$29,700.00</td>
</tr>
</tbody>
</table>
Collapsing PRINT With ACROSS

**How to:**

Compress Report Lines

**Reference:**

Usage Notes for SET ACROSSPRT

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**

```plaintext
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:

<table>
<thead>
<tr>
<th>Region</th>
<th>Midwest</th>
<th>Northeast</th>
<th>Southeast</th>
<th>West</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Unit Sales</td>
<td>Unit Sales</td>
<td>Unit Sales</td>
<td>Unit Sales</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Capuccino</td>
<td>.</td>
<td>936</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>116</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>136</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>1616</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>1118</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>774</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>1696</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>1519</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>836</td>
<td>.</td>
</tr>
<tr>
<td>Espresso</td>
<td>1333</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>280</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>139</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>1363</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>634</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>406</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>1028</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>1014</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>885</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>1782</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>1399</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>551</td>
</tr>
</tbody>
</table>
Setting ACROSSPRRT 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 ACROSSPRRT COMPRESS ON TABLE SET PAGE NOPAGE
END
```

The output is:

```
<table>
<thead>
<tr>
<th>Region</th>
<th>Midwest Unit Sales</th>
<th>Northeast Unit Sales</th>
<th>Southeast Unit Sales</th>
<th>West Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capuccino</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>936</td>
<td>1616</td>
<td>1696</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>116</td>
<td>1118</td>
<td>1519</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>136</td>
<td>774</td>
<td>836</td>
<td></td>
</tr>
<tr>
<td>Espresso</td>
<td>1333</td>
<td>1363</td>
<td>1028</td>
<td>1782</td>
</tr>
<tr>
<td></td>
<td>280</td>
<td>634</td>
<td>1014</td>
<td>1399</td>
</tr>
<tr>
<td></td>
<td>139</td>
<td>406</td>
<td>885</td>
<td>551</td>
</tr>
</tbody>
</table>
```

### Manipulating Display Field Values in a Sort Group

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

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.

You can use up to 64 fields in a display command when using the WITHIN phrase. 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**

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

where:

- \text{display_field} is the object of a SUM or COUNT display command.
- \text{by_sort_field} is the object of a BY phrase.
- \text{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
```
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:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>PRODUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>A01</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>$9,500.00</td>
</tr>
<tr>
<td></td>
<td>A07</td>
</tr>
<tr>
<td></td>
<td>$9,000.00</td>
</tr>
<tr>
<td></td>
<td>$11,000.00</td>
</tr>
<tr>
<td></td>
<td>A15</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>$26,862.00</td>
</tr>
<tr>
<td></td>
<td>A17</td>
</tr>
<tr>
<td></td>
<td>$27,062.00</td>
</tr>
<tr>
<td></td>
<td>$29,700.00</td>
</tr>
<tr>
<td></td>
<td>B02</td>
</tr>
<tr>
<td></td>
<td>$18,480.00</td>
</tr>
<tr>
<td></td>
<td>$16,100.00</td>
</tr>
<tr>
<td></td>
<td>B03</td>
</tr>
<tr>
<td></td>
<td>$18,480.00</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>B04</td>
</tr>
<tr>
<td></td>
<td>$21,780.00</td>
</tr>
<tr>
<td></td>
<td>$21,120.00</td>
</tr>
<tr>
<td></td>
<td>B14</td>
</tr>
<tr>
<td></td>
<td>$13,200.00</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>LAST_NAME</th>
<th>ED_HRS</th>
<th>CROSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>BLACKWOOD</td>
<td>45.00</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>45.00</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>45.00</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>45.00</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>75.00</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>75.00</td>
<td>.</td>
</tr>
</tbody>
</table>

**Specifying the Sort Order**

In this section:
- Specifying Your Own Sort Order

How to:
- Specify 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 and VM 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
```
**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:

```
CURR_SAL   LAST_NAME
$29,700.00  BANNING
$27,062.00  CROSS
$26,862.00  IRVING
$21,780.00  BLACKWOOD
$21,120.00  ROMANS
$18,480.00  JONES
$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

How to:
Define Your Own Sort Order
Define Column Sort Sequence

Reference:
Usage Notes for Defining Your Sort Order
Usage Notes for Defining Column Sort Sequence

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...

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 ROWS value1 OVER value2 [... OVER 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.
Specifying the Sort Order

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

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 1307.

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.
- 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 1307.

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:

![Table with columns and values]

**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.

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

The output is:

<table>
<thead>
<tr>
<th>BANK_NAME</th>
<th>BEST BANK</th>
<th>STATE</th>
<th>ASSOCIATED</th>
<th>BANK ASSOCIATION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$29,700.00</td>
<td>$18,480.00</td>
<td>$64,742.00</td>
<td>$27,062.00</td>
</tr>
</tbody>
</table>
Ranking Sort Field Values

How to: Rank 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 1053.

You can rank aggregated values using the syntax RANKED BY TOTAL. For details, see Sorting and Aggregating Report Columns on page 119.

Syntax: How to Rank Sort Field Values

RANKED [AS 'name'] BY sortfield

where:

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

name
  Is the new name for the RANK column title.

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
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:

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

```

<table>
<thead>
<tr>
<th>RANK</th>
<th>CURR_SAL</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$29,700.00</td>
<td>BANNING</td>
</tr>
<tr>
<td>2</td>
<td>$27,062.00</td>
<td>CROSS</td>
</tr>
<tr>
<td>3</td>
<td>$26,862.00</td>
<td>IRVING</td>
</tr>
<tr>
<td>4</td>
<td>$21,780.00</td>
<td>BLACKWOOD</td>
</tr>
<tr>
<td>5</td>
<td>$21,120.00</td>
<td>ROMANS</td>
</tr>
</tbody>
</table>
Grouping Numeric Data Into Ranges

In this section:
Grouping Numeric Data Into Tiles

How to:
Define Groups of Equal Range
Define Equal Ranges
Define Custom Groups of Data Values

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. 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 113.

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 1307.

Note: If there is not any data for a group, a row for the group still appears in the report.
How to Define Groups of Equal Range

**Syntax:**

```
{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.

```language
TABLE FILE EMPLOYEE
PRINT LAST_NAME
BY CURR_SAL IN-GROUPS-OF 5000
END
```

The output is:

<table>
<thead>
<tr>
<th>CURR_SAL</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>$5,000.00</td>
<td>SMITH</td>
</tr>
<tr>
<td>$10,000.00</td>
<td>GREENSPAN</td>
</tr>
<tr>
<td>$15,000.00</td>
<td>STEVENS</td>
</tr>
<tr>
<td>$20,000.00</td>
<td>JONES</td>
</tr>
<tr>
<td>$25,000.00</td>
<td>MCCOY</td>
</tr>
<tr>
<td>$30,000.00</td>
<td>MCKNIGHT</td>
</tr>
<tr>
<td>$35,000.00</td>
<td>ROMANS</td>
</tr>
<tr>
<td>$40,000.00</td>
<td>BLACKWOOD</td>
</tr>
<tr>
<td>$45,000.00</td>
<td>BANNING</td>
</tr>
<tr>
<td>$50,000.00</td>
<td>IRVING</td>
</tr>
<tr>
<td>$55,000.00</td>
<td>CROSS</td>
</tr>
</tbody>
</table>

How to Define Equal Ranges

**Syntax:**

```
{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.

Example: Defining Equal Ranges

TABLE FILE EMPLOYEE
PRINT LAST_NAME
BY CURR_SAL IN-RANGES-OF 5000
END

The output is:

<table>
<thead>
<tr>
<th>CURR_SAL</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>$5,000.00 - $9,999.99</td>
<td>SMITH</td>
</tr>
<tr>
<td></td>
<td>GREENSPAN</td>
</tr>
<tr>
<td>$10,000.00 - $14,999.99</td>
<td>STEVENS</td>
</tr>
<tr>
<td></td>
<td>SMITH</td>
</tr>
<tr>
<td>$15,000.00 - $19,999.99</td>
<td>JONES</td>
</tr>
<tr>
<td></td>
<td>MCCOY</td>
</tr>
<tr>
<td></td>
<td>MCKNIGHT</td>
</tr>
<tr>
<td>$20,000.00 - $24,999.99</td>
<td>ROMANS</td>
</tr>
<tr>
<td></td>
<td>BLACKWOOD</td>
</tr>
<tr>
<td>$25,000.00 - $29,999.99</td>
<td>BANNING</td>
</tr>
<tr>
<td></td>
<td>IRVING</td>
</tr>
<tr>
<td></td>
<td>CROSS</td>
</tr>
</tbody>
</table>

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.
**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:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEVENS</td>
</tr>
<tr>
<td>SMITH</td>
</tr>
<tr>
<td>SMITH</td>
</tr>
<tr>
<td>GREENSPAN</td>
</tr>
<tr>
<td>JONES</td>
</tr>
<tr>
<td>MCCOY</td>
</tr>
<tr>
<td>MCKNIGHT</td>
</tr>
<tr>
<td>BANNING</td>
</tr>
<tr>
<td>IRVING</td>
</tr>
<tr>
<td>ROMANS</td>
</tr>
<tr>
<td>BLACKWOOD</td>
</tr>
<tr>
<td>CROSS</td>
</tr>
</tbody>
</table>
Grouping Numeric Data Into Tiles

How to:
Group Numeric Data Into Tiles

Reference:
Usage Notes for Tiles

You can group numeric data into any number of tiles (percentiles, deciles, quartiles, etc.) in tabular reports. For example, you can group students’ test scores into deciles to determine which students are in the top ten percent of the class, or determine which salesmen are in the top half of all salesmen 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 1053.

- 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:

<table>
<thead>
<tr>
<th>Group</th>
<th>Data Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,5</td>
</tr>
<tr>
<td>2</td>
<td>5,5</td>
</tr>
<tr>
<td>3</td>
<td>8,9</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Tile Number</th>
<th>Data Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,5,5,5</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8,9</td>
</tr>
</tbody>
</table>

**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 and VM, 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.

```plaintext
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME
BY DEPARTMENT
BY CURR_SAL IN-GROUPS-OF 5 TILES
END
```
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:

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:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>CURR_SAL</th>
<th>DECILE</th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>$9,000.00</td>
<td>1</td>
<td>GREENSPAN</td>
<td>MARY</td>
</tr>
<tr>
<td></td>
<td>$13,200.00</td>
<td>1</td>
<td>SMITH</td>
<td>MARY</td>
</tr>
<tr>
<td></td>
<td>$18,480.00</td>
<td>2</td>
<td>JONES</td>
<td>DIANE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MCCOY</td>
<td>JOHN</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>$9,500.00</td>
<td>1</td>
<td>SMITH</td>
<td>RICHARD</td>
</tr>
<tr>
<td></td>
<td>$11,000.00</td>
<td>1</td>
<td>STEVENS</td>
<td>ALFRED</td>
</tr>
<tr>
<td></td>
<td>$16,100.00</td>
<td>2</td>
<td>MCKNIGHT</td>
<td>ROGER</td>
</tr>
<tr>
<td></td>
<td>$21,120.00</td>
<td>3</td>
<td>ROMANS</td>
<td>ANTHONY</td>
</tr>
</tbody>
</table>

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**

**How to:**
Restrict 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

```plaintext
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:

<table>
<thead>
<tr>
<th>CURR_SAL</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>$29,700.00</td>
<td>BANNING</td>
</tr>
<tr>
<td>$27,062.00</td>
<td>CROSS</td>
</tr>
<tr>
<td>$26,862.00</td>
<td>IRVING</td>
</tr>
<tr>
<td>$21,780.00</td>
<td>BLACKWOOD</td>
</tr>
<tr>
<td>$21,120.00</td>
<td>ROMANS</td>
</tr>
</tbody>
</table>

**Sorting and Aggregating Report Columns**

**How to:**

Sort and Aggregate a Report Column

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 and VM, 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

\[
\begin{align*}
\text{[RANKED]} & \quad \text{BY} \quad \text{[HIGHEST|LOWEST} \quad [n] \quad ] \\
\text{TOTAL} & \quad \{\text{display\_field|}\text{COMPUTE name/format=expression;}\}
\end{align*}
\]

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:

```
<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>SALARIES</th>
<th>CURR_JOBCODE</th>
<th>SALARY</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>$26,418.50</td>
<td>A17</td>
<td>$52,837.00</td>
<td>2</td>
</tr>
<tr>
<td>MIS</td>
<td>$21,780.00</td>
<td>B04</td>
<td>$21,780.00</td>
<td>1</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>$29,700.00</td>
<td>A17</td>
<td>$29,700.00</td>
<td>1</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>$25,641.00</td>
<td>A15</td>
<td>$51,282.00</td>
<td>2</td>
</tr>
</tbody>
</table>
```
**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:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>RANK</th>
<th>SALARIES</th>
<th>CURR_JOBCODE</th>
<th>SALARY</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>1</td>
<td>$26,418.50</td>
<td>A17</td>
<td>$52,837.00</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>$21,780.00</td>
<td>B04</td>
<td>$21,780.00</td>
<td>1</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>1</td>
<td>$29,700.00</td>
<td>A17</td>
<td>$29,700.00</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>$25,641.00</td>
<td>A15</td>
<td>$51,282.00</td>
<td>2</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>SALARIES</th>
<th>CURR_JOBCODE</th>
<th>SALARY</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>$4,403.08</td>
<td>A17</td>
<td>$52,837.00</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$3,019.17</td>
<td>B03</td>
<td>$36,230.00</td>
<td>2</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>$4,273.50</td>
<td>A15</td>
<td>$51,282.00</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$2,591.67</td>
<td>B02</td>
<td>$31,100.00</td>
<td>2</td>
</tr>
</tbody>
</table>
Hiding Sort Values

**How to:**
Hide 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:**

```plaintext
{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.

```plaintext
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME HIRE_DATE
END
```
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:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>HIRE_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEVENS</td>
<td>ALFRED</td>
<td>80/06/02</td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td>81/07/01</td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td>82/05/01</td>
</tr>
<tr>
<td>SMITH</td>
<td>RICHARD</td>
<td>82/01/04</td>
</tr>
<tr>
<td>BANNING</td>
<td>JOHN</td>
<td>82/08/01</td>
</tr>
<tr>
<td>IRVING</td>
<td>JOAN</td>
<td>82/01/04</td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>82/07/01</td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td>81/07/01</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>82/04/01</td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>82/02/02</td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
<td>82/04/01</td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>81/11/02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sorting With Multiple Display Commands

A request can consist of up to sixteen 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 sixteen 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 143.

- 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:

```plaintext
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.

```plaintext
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:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>CURR_SAL</th>
<th>DEPARTMENT</th>
<th>CURR_SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>$222,284.00</td>
<td>BLACKWOOD</td>
<td>$21,780.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CROSS</td>
<td>$27,062.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GREENSPAN</td>
<td>$9,000.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JONES</td>
<td>$18,480.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MCCOY</td>
<td>$18,480.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SMITH</td>
<td>$13,200.00</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>$114,282.00</td>
<td>BANNING</td>
<td>$29,700.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IRVING</td>
<td>$26,862.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MCKNIGHT</td>
<td>$16,100.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ROMANS</td>
<td>$21,120.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SMITH</td>
<td>$9,500.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STEVENS</td>
<td>$11,000.00</td>
</tr>
</tbody>
</table>

**Controlling Formatting of Reports With Multiple Display Commands**

**How to:**

Control the Format of Reports With Multiple Display Commands

Style a Report With SET DUPLICATECOL=ON

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
----------         --------  -----
MIS             $222,284.00  351.00
PRODUCTION      $108,002.00  231.00
$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:

<table>
<thead>
<tr>
<th>DEPT</th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>CURR_SAL</th>
<th>ED_HRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td></td>
<td>$21,780.00</td>
<td>75.00</td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td></td>
<td>$27,062.00</td>
<td>45.00</td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
<td></td>
<td>$0,000.00</td>
<td>25.00</td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td></td>
<td>$18,480.00</td>
<td>50.00</td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td></td>
<td>$18,480.00</td>
<td>.00</td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td></td>
<td>$13,200.00</td>
<td>36.00</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td></td>
<td></td>
<td>$114,282.00</td>
<td>120.00</td>
</tr>
<tr>
<td>BANNING</td>
<td>JOAN</td>
<td></td>
<td>$26,862.00</td>
<td>30.00</td>
</tr>
<tr>
<td>IRVING</td>
<td>JOAN</td>
<td></td>
<td>$26,862.00</td>
<td>30.00</td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td></td>
<td>$16,100.00</td>
<td>50.00</td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td></td>
<td>$21,120.00</td>
<td>5.00</td>
</tr>
<tr>
<td>SMITH</td>
<td>RICHARD</td>
<td></td>
<td>$9,500.00</td>
<td>10.00</td>
</tr>
<tr>
<td>STEVENS</td>
<td>ALFRED</td>
<td></td>
<td>$11,000.00</td>
<td>25.00</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>DEPT</th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>CURR_SAL</th>
<th>ED_HRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td></td>
<td>$21,780.00</td>
<td>75.00</td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td></td>
<td>$27,062.00</td>
<td>45.00</td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
<td></td>
<td>$0,000.00</td>
<td>25.00</td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td></td>
<td>$18,480.00</td>
<td>50.00</td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td></td>
<td>$18,480.00</td>
<td>.00</td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td></td>
<td>$13,200.00</td>
<td>36.00</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td></td>
<td></td>
<td>$114,282.00</td>
<td>120.00</td>
</tr>
<tr>
<td>BANNING</td>
<td>JOAN</td>
<td></td>
<td>$26,862.00</td>
<td>30.00</td>
</tr>
<tr>
<td>IRVING</td>
<td>JOAN</td>
<td></td>
<td>$26,862.00</td>
<td>30.00</td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td></td>
<td>$16,100.00</td>
<td>50.00</td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td></td>
<td>$21,120.00</td>
<td>5.00</td>
</tr>
<tr>
<td>SMITH</td>
<td>RICHARD</td>
<td></td>
<td>$9,500.00</td>
<td>10.00</td>
</tr>
<tr>
<td>STEVENS</td>
<td>ALFRED</td>
<td></td>
<td>$11,000.00</td>
<td>25.00</td>
</tr>
</tbody>
</table>

**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).
### 3. Sorting Tabular Reports

The output is:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>CURR_SAL</th>
<th>ED_HRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>$108,002.00</td>
<td>231.00</td>
</tr>
<tr>
<td></td>
<td>CROSS</td>
<td>BARBARA</td>
<td>$21,780.00</td>
<td>75.00</td>
</tr>
<tr>
<td></td>
<td>GREENSPAN</td>
<td>MARY</td>
<td>$27,062.00</td>
<td>45.00</td>
</tr>
<tr>
<td></td>
<td>JONES</td>
<td>DIANE</td>
<td>$9,000.00</td>
<td>25.00</td>
</tr>
<tr>
<td></td>
<td>MCCOY</td>
<td>JOHN</td>
<td>$18,480.00</td>
<td>50.00</td>
</tr>
<tr>
<td></td>
<td>SMITH</td>
<td>MARY</td>
<td>$13,200.00</td>
<td>36.00</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>BANNING</td>
<td>JOHN</td>
<td>$18,480.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>IRVING</td>
<td>JOAN</td>
<td>$18,480.00</td>
<td>30.00</td>
</tr>
<tr>
<td></td>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>$13,200.00</td>
<td>50.00</td>
</tr>
<tr>
<td></td>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>$13,200.00</td>
<td>5.00</td>
</tr>
<tr>
<td></td>
<td>SMITH</td>
<td>RICHARD</td>
<td>$13,200.00</td>
<td>5.00</td>
</tr>
<tr>
<td></td>
<td>STEVENS</td>
<td>ALFRED</td>
<td>$11,000.00</td>
<td>25.00</td>
</tr>
</tbody>
</table>
Improving Efficiency With External Sorts

In this section:
- Providing an Estimate of Input Records or Report Size for Sorting
- Sort Work Files and Return Codes
- Mainframe External Sort Utilities and Message Options
- Aggregation by External Sort (Mainframe Environments Only)
- Changing Retrieval Order With Aggregation (Mainframe Environments Only)
- Creating a HOLD File With an External Sort (Mainframe Environments Only)

How to:
- Determine the Type of Sort Used
- Control External Sorting
- Query the Sort Type

Reference:
- Requirements for External Sorting
- Usage Notes for External Sorting in Non-Mainframe Environments

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 132) 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 133). 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

**How to:**
Provide 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 \textit{n}
ON TABLE SET ESTLINES \textit{n}

where:

\textit{n}

Is the estimated number of records or lines to be sorted.

**Sort Work Files and Return Codes**

**Reference:**
Sort Work Files on UNIX, Windows, and OpenVMS
Sort Work Files on i5/OS

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, FOCSORT (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 FOCSORT.)
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 IBITMPDIRnn variables. The UNIX command for unsetting the TMPDIR variable is:

  ```bash
  unset TMPDIR
  ```

  The Windows command for unsetting the TMP variable is:

  ```bash
  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 i5/OS

On i5/OS (formerly AS/400), the number of work files is fixed at 9. They are virtual files.
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

In this section:
Diagnosing External Sort Errors

How to:
Select a Sort Utility 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.
MVSMSGSS for SyncSort with standard messages.
MVSMSGSD 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

**How to:**

Trace Sort Processing

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 136.

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 135 and How to Trace Sort Processing on page 136) 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 MVSMMSGDJ 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 yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
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 135) 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 133.

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 $ORTPARM without direct instructions from technical support. The instructions in *How to Select a Sort Utility and Message Options* on page 135, *How to Trace Sort Processing* on page 136, and *Providing an Estimate of Input Records or Report Size for Sorting* on page 133 should provide equivalent capabilities.

**Aggregation by External Sort (Mainframe Environments Only)**

**How to:**

Use Aggregation in Your External Sort

**Reference:**

Usage Notes for Aggregating With an External Sort

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 1434.
- 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.
- CMS DFSORT does not support aggregation of numeric data types.
- 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. Turning on aggregation in the external sort displays the first record instead. However, you can control the order of display using the SUMPREFIX parameter. With SUMPREFIX = LST (the default), the last instance displays even with EXTAGGR = ON.
The following command turns aggregation ON and leaves SUMPREFIX set to LST (the default) and, therefore, displays the last record:

```sql
SET EXTAGGR = ON
SET SUMPREFIX = LST
TABLE FILE CAR
SUM CAR BY COUNTRY
END
```

The output is:

<table>
<thead>
<tr>
<th>Country</th>
<th>Car</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGLAND</td>
<td>TRIUMPH</td>
</tr>
<tr>
<td>FRANCE</td>
<td>PEUGEOT</td>
</tr>
<tr>
<td>ITALY</td>
<td>MASERATI</td>
</tr>
<tr>
<td>JAPAN</td>
<td>TOYOTA</td>
</tr>
<tr>
<td>W GERMANY</td>
<td>BMW</td>
</tr>
</tbody>
</table>

**Note:** SUMPREFIX is described in *Changing Retrieval Order With Aggregation (Mainframe Environments Only)* on page 140.

With SUMPREFIX = FST, the output is:

<table>
<thead>
<tr>
<th>Country</th>
<th>Car</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGLAND</td>
<td>JAGUAR</td>
</tr>
<tr>
<td>FRANCE</td>
<td>PEUGEOT</td>
</tr>
<tr>
<td>ITALY</td>
<td>ALFA ROMEO</td>
</tr>
<tr>
<td>JAPAN</td>
<td>DATSUN</td>
</tr>
<tr>
<td>W GERMANY</td>
<td>AUDI</td>
</tr>
</tbody>
</table>

### Changing Retrieval Order With Aggregation (Mainframe Environments Only)

**How to:**

Set Retrieval Order

When an external sort product performs aggregation of alphanumeric or smart date formats, the order of the answer set returned differs from the order of the internally sorted answer sets.

External sort products return the first alphanumeric or smart date record that was aggregated. Conversely, internal sorting returns the last record.

The SUMPREFIX command allows users to choose the answer set display order.
**Syntax:** How to Set Retrieval Order

```
SET SUMPREFIX = {LST | FST}
```

where:

- **LST**
  - Displays the last value when alphanumeric or smart date data types are aggregated. This value is the default.

- **FST**
  - Displays the first value when alphanumeric or smart date data types are aggregated.

---

**Creating a HOLD File With an External Sort (Mainframe Environments Only)**

**How to:**

Create HOLD Files With an External Sort

**Reference:**

Usage Notes for Creating a HOLD File With an External Sort

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.
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.

**Topics:**
- Selecting Records Overview
- Choosing a Filtering Method
- Selections Based on Individual Values
- Selection Based on Aggregate Values
- 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 145.
- The aggregate value of a field (for example, the sum or average of field values). See Selection Based on Aggregate Values on page 153.
- 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 161.
- 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 176.
- 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 175.

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 177.
Selections Based on Individual Values

**In this section:**
- Controlling Record Selection in Multi-path Data Sources

**How to:**
Select Records With WHERE

**Reference:**
- Usage Notes for WHERE Phrases

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 147. For additional information, see *Using Compound Expressions for Record Selection* on page 155.

**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 148.

**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 315. 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 157.

```
;
```

- 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 155.

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:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>CURR_SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANNING</td>
<td>JOHN</td>
<td>$29,700.00</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>$21,780.00</td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>$27,062.00</td>
</tr>
<tr>
<td>IRVING</td>
<td>JOAN</td>
<td>$26,862.00</td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>$21,120.00</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>EMP_ID</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>119329144</td>
<td>BANNING</td>
</tr>
<tr>
<td>818692173</td>
<td>CROSS</td>
</tr>
</tbody>
</table>

For related information, see *Using Compound Expressions for Record Selection* on page 155.
Controlling Record Selection in Multi-path Data Sources

How to:
Control Record Selection in Multi-path Data Sources

Reference:
Requirements and Usage Notes for MULTIPATH = COMPOUND
MULTIPATH and SET ALL Combinations
Rules for Determining If a Segment Is Required

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

```sql
SET MULTIPATH = {SIMPLE|COMPOUND}
```
To set MULTIPATH in a report request, use

```sql
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:

  ```text
  (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 152 and *Rules for Determining If a Segment Is Required* on page 153.

**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.

- SET MULTIPATH = COMPOUND creates a pool boundary when reports are pooled.

- 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:

```
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.
If you run the same request with \texttt{MULTIPATH = COMPOUND}, the employees without instances for \texttt{COURSE\_NAME} are omitted from the report output, and the warning message is not generated.

The output is:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>GROSS</th>
<th>DATE_ATTEND</th>
<th>COURSE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANNING</td>
<td>JOHN</td>
<td>$2,475.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>$1,815.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>$2,255.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
<td>$750.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IRVING</td>
<td>JOAN</td>
<td>$2,238.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td>$1,540.00</td>
<td>82/05/26</td>
<td>BASIC REPORT PREP FOR PROG</td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td>$1,540.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>$1,342.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>$1,760.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td>$1,100.00</td>
<td>81/11/16</td>
<td>BASIC REPORT PREP FOR PROG</td>
</tr>
<tr>
<td>STEVENS</td>
<td>RICHARD</td>
<td>$791.67</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ALFRED</td>
<td>$916.67</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**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.

<table>
<thead>
<tr>
<th>Request</th>
<th>MULTIPATH=SIMPLE</th>
<th>MULTIPATH=COMPOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET ALL = OFF PRINT EMP_ID PAY_DATE DATE_ATTEND</td>
<td>Shows employees who have <em>either</em> SALINFO data or ATTNDSEG data.</td>
<td>Shows employees who have <em>both</em> SALINFO and ATTNDSEG data.</td>
</tr>
<tr>
<td>SET ALL = ON PRINT EMP_ID PAY_DATE DATE_ATTEND</td>
<td>Shows employees who have SALINFO data or ATTNDSEG data or no child data at all.</td>
<td>Same as SIMPLE.</td>
</tr>
<tr>
<td>SET ALL = OFF PRINT EMP_ID PAY_DATE DATE_ATTEND WHERE PAY_DATE EQ 980115</td>
<td>Shows employees who have <em>either</em> SALINFO data for 980115 or any ATTNDSEG data. Produces (FOC144) message.</td>
<td>Shows employees who have <em>both</em> SALINFO data for 980115 and ATTNDSEG data.</td>
</tr>
<tr>
<td>SET ALL = ON PRINT EMP_ID PAY_DATE DATE_ATTEND WHERE PAY_DATE EQ 980115</td>
<td>Shows employees who have <em>either</em> SALINFO data for 980115 or any ATTNDSEG data. Produces (FOC144) message.</td>
<td>Shows employees who have SALINFO data for 980115. Any DATE_ATTEND data is also shown.</td>
</tr>
<tr>
<td>SET ALL = OFF PRINT ALL.EMP_ID DATE_ATTEND WHERE PAY_DATE EQ 980115</td>
<td>Shows employees who have <em>either</em> SALINFO data for 980115 or any ATTNDSEG data. Produces (FOC144) message.</td>
<td>Shows employees who have SALINFO data for 980115. Any DATE_ATTEND data is also shown.</td>
</tr>
<tr>
<td>SET ALL = ON or OFF PRINT EMP_ID PAY_DATE DATE_ATTEND WHERE PAY_DATE EQ 980115 AND COURSE_CODE EQ '103'</td>
<td>Shows employees who have <em>either</em> SALINFO data for 980115 or COURSE 103. <strong>Note:</strong> SIMPLE treats AND in the WHERE clause as OR. Produces (FOC144) message.</td>
<td>Shows employees who have <em>both</em> SALINFO data for 980115 and COURSE 103.</td>
</tr>
</tbody>
</table>

**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 697.

**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 697, and the *Describing Data With WebFOCUS Language* manual.

**Selection Based on Aggregate Values**

**How to:**

Select Records With WHERE TOTAL

**Reference:**

Usage Notes for WHERE TOTAL

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 154.
### 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 315. 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 157.

; 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
```
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
----------------       --------
MIS               $108,002.00
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
```

**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
```
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.

<table>
<thead>
<tr>
<th>WHERE Operator</th>
<th>IF Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ</td>
<td>EQ</td>
<td>Tests for and selects values equal to the test expression.</td>
</tr>
<tr>
<td>IS</td>
<td>IS</td>
<td></td>
</tr>
<tr>
<td>NE</td>
<td>NE</td>
<td>Tests for and selects values not equal to the test expression.</td>
</tr>
<tr>
<td>IS-NOT</td>
<td>IS-NOT</td>
<td></td>
</tr>
<tr>
<td>GE</td>
<td>GE</td>
<td>Tests for and selects values greater than or equal to the test value.</td>
</tr>
<tr>
<td>FROM</td>
<td>IS-FROM</td>
<td>(based on the characters 0 to 9 for numeric values, A to Z and a to z for alphanumeric values).</td>
</tr>
<tr>
<td>GT</td>
<td>GT</td>
<td>Tests for and selects values greater than the test value.</td>
</tr>
<tr>
<td>EXCEEDS</td>
<td>EXCEEDS</td>
<td></td>
</tr>
<tr>
<td>IS-MORE-THAN</td>
<td>IS-MORE-THAN</td>
<td></td>
</tr>
<tr>
<td>LT</td>
<td>LT</td>
<td>Tests for and selects values less than the test value.</td>
</tr>
<tr>
<td>IS-LESS-THAN</td>
<td>IS-LESS-THAN</td>
<td></td>
</tr>
<tr>
<td>LE</td>
<td>LE</td>
<td>Tests for and selects values less than or equal to the test value.</td>
</tr>
<tr>
<td>TO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GE lower AND ... LE upper</td>
<td></td>
<td>Tests for and selects values within a range of values.</td>
</tr>
<tr>
<td>LT lower OR ... GT upper</td>
<td></td>
<td>Tests for and selects values outside of a range of values.</td>
</tr>
</tbody>
</table>
### Using Operators in Record Selection Tests

<table>
<thead>
<tr>
<th>WHERE Operator</th>
<th>IF Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FROM lower TO upper</td>
<td></td>
<td>Tests for and selects values within a range of values.</td>
</tr>
<tr>
<td>IS-FROM lower TO upper</td>
<td>IS-FROM lower TO upper</td>
<td>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.</td>
</tr>
<tr>
<td>NOT-FROM lower TO upper</td>
<td>NOT-FROM lower TO upper</td>
<td>Tests for and selects values that are outside a range of values.</td>
</tr>
<tr>
<td>IS MISSING</td>
<td>IS NOT MISSING</td>
<td>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 Handling Records With Missing Field Values on page 697.</td>
</tr>
<tr>
<td>IS NOT MISSING</td>
<td>NE MISSING</td>
<td></td>
</tr>
<tr>
<td>CONTAINS LIKE</td>
<td>CONTAINS LIKE</td>
<td>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.</td>
</tr>
<tr>
<td>OMITS UNLIKE</td>
<td>OMITS NOT LIKE</td>
<td>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.</td>
</tr>
<tr>
<td>INCLUDES</td>
<td>INCLUDES</td>
<td>Tests whether a chain of values of a given field in a child segment includes all of a list of literals.</td>
</tr>
</tbody>
</table>
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:

![Output Image]

Please select a Region: CE
Submit
Reset
Done
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

<table>
<thead>
<tr>
<th>DIV DEPT</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE CUSTOMER SUPPORT</td>
<td>$81,800.00</td>
</tr>
<tr>
<td>MARKETING</td>
<td>$139,800.00</td>
</tr>
<tr>
<td>SALES</td>
<td>$82,600.00</td>
</tr>
</tbody>
</table>

Types of Record Selection Tests

In this section:
- Range Tests With FROM and TO
- Range Tests With GE and LE or GT and LT
- Missing Data Tests
- Character String Screening With CONTAINS and OMITS
- Screening on Masked Fields With LIKE and IS
- Using an Escape Character for LIKE
- Qualifying Parent Segments Using INCLUDES and EXCLUDES

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 157. You can test for:

- Values that lie within or outside of a range. See Range Tests With FROM and TO on page 162 and Range Tests With GE and LE or GT and LT on page 163.
- Missing or existing data. See Missing Data Tests on page 165.
- The existence or absence of a character string. See Character String Screening With CONTAINS and OMITS on page 165.
- Partially defined character strings in a data field. See Screening on Masked Fields With LIKE and IS on page 167.
Range Tests With FROM and TO

**How to:**
Specify a Range Test (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.

**Syntax:**

```
How to Specify a Range Test (FROM and TO)

WHERE [TOTAL] fieldname {FROM|IS-FROM} lower TO upper
WHERE [TOTAL] fieldname NOT-FROM lower TO 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:**

**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)
```

**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:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>CURR_SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANNING</td>
<td>$29,700.00</td>
</tr>
<tr>
<td>CROSS</td>
<td>$27,062.00</td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>$9,000.00</td>
</tr>
<tr>
<td>IRVING</td>
<td>$26,862.00</td>
</tr>
<tr>
<td>SMITH</td>
<td>$9,500.00</td>
</tr>
<tr>
<td>STEVENS</td>
<td>$11,000.00</td>
</tr>
</tbody>
</table>

**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**

**How to:**

Specify Range Tests (GE and LE)

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.

```sql
WHERE UNITS GE 10000 AND UNITS LE 14000
```
This example is equivalent to:

```sql
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.

```sql
TABLE FILE EMPLOYEE
PRINT CURR_SAL
BY LAST_NAME
WHERE CURR_SAL LT 12000 OR CURR_SAL GT 22000
END
```
The output is:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>CURR_SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANNING</td>
<td>$29,700.00</td>
</tr>
<tr>
<td>CROSS</td>
<td>$27,062.00</td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>$9,000.00</td>
</tr>
<tr>
<td>IRVING</td>
<td>$26,862.00</td>
</tr>
<tr>
<td>SMITH</td>
<td>$9,500.00</td>
</tr>
<tr>
<td>STEVENS</td>
<td>$11,000.00</td>
</tr>
</tbody>
</table>
Missing Data Tests

**How to:**
- Test for Missing Data
- Test for Existing Data

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 697, and the *Describing Data With WebFOCUS Language* manual.

**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:

  ```sql
  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:

  ```sql
  WHERE LAST_NAME OMITS 'JOHN'
  ```

- In this example, all names that contain the letters ING are retrieved.

  ```sql
  TABLE FILE EMPLOYEE
  LIST LAST_NAME AND FIRST_NAME
  WHERE LAST_NAME CONTAINS 'ING'
  END
  ```

  The output is:

<table>
<thead>
<tr>
<th></th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BANNING</td>
<td>JOHN</td>
</tr>
<tr>
<td>2</td>
<td>IRVING</td>
<td>JOAN</td>
</tr>
</tbody>
</table>
Screening on Masked Fields With LIKE and IS

How to:

Screen Fields Based on a Mask (Using LIKE and NOT LIKE)
Screen Using LIKE and UNLIKE in an IF Phrase
Screen Fields Based on a Mask (Using IS and IS-NOT)

Reference:
Restrictions on Masking Characters

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.

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 169.

**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.

\textbf{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 \textit{Restrictions on Masking Characters} on page 169.

\textbf{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.

\textbf{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:

\begin{verbatim}
TABLE FILE EMPLOYEE
PRINT COURSE_NAME COURSE_CODE
BY LAST_NAME BY FIRST_NAME
WHERE COURSE_NAME LIKE 'BASIC%'
END
\end{verbatim}
### 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):

```plaintext
TABLE FILE EMPLOYEE
PRINT COURSE_NAME COURSE_CODE
BY LAST_NAME BY FIRST_NAME
WHERE COURSE_NAME LIKE '%FOCUS%'
END
```

The output is:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>COURSE_NAME</th>
<th>COURSE_CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>WHAT'S NEW IN FOCUS</td>
<td>202</td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td>FOCUS INTERNALS</td>
<td>203</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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:

```plaintext
TABLE FILE EMPLOYEE
PRINT COURSE_NAME COURSE_CODE
BY LAST_NAME BY FIRST_NAME
WHERE COURSE_CODE LIKE '20_'
END
```

The output is:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>COURSE_NAME</th>
<th>COURSE_CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>WHAT'S NEW IN FOCUS</td>
<td>202</td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td>FOCUS INTERNALS</td>
<td>203</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADVANCED TECHNIQUES</td>
<td>201</td>
</tr>
</tbody>
</table>
**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 7 characters in length (the three initial characters BAN and the 4 placeholders, in this case the dollar sign). The remaining characters in the field (positions 8 through 15) must be blank.

```sql
TABLE FILE EMPLOYEE
PRINT LAST_NAME
WHERE LAST_NAME IS 'BAN$**$
END
```

The output is:

```
LAST_NAME
---------
BANNING
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.

**Using an Escape Character for LIKE**

**How to:**

- Use an Escape Character in a WHERE Phrase
- Specify an Escape Character for a Mask in an IF Phrase

**Reference:**

Usage Notes for Escape Characters

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 With LIKE and IS* on page 167.
**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 e-mail address field. To search for the e-mail 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:

<table>
<thead>
<tr>
<th>CUSTID</th>
<th>LASTNAME</th>
<th>FIRSTNAME</th>
<th>EMAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0944</td>
<td>HANDLER</td>
<td>EVAN</td>
<td><a href="mailto:handy_man@usa.com">handy_man@usa.com</a></td>
</tr>
<tr>
<td>0944</td>
<td>HANDLER</td>
<td>EVAN</td>
<td><a href="mailto:handyman@usa.com">handyman@usa.com</a></td>
</tr>
</tbody>
</table>

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 e-mail field:

```
TABLE FILE VIDEOTR2
PRINT CUSTID LASTNAME FIRSTNAME EMAIL
WHERE EMAIL LIKE 'handy\_%' ESCAPE '\'
END
```

The output is:

<table>
<thead>
<tr>
<th>CUSTID</th>
<th>LASTNAME</th>
<th>FIRSTNAME</th>
<th>EMAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0944</td>
<td>HANDLER</td>
<td>EVAN</td>
<td><a href="mailto:handy_man@usa.com">handy_man@usa.com</a></td>
</tr>
</tbody>
</table>
Example: Using an Escape Character in an IF Phrase

The VIDEOTR2 data source contains an e-mail address field. To search for e-mail addresses with the characters 'handy_' you can issue the following request:

```
TABLE FILE VIDEOTR2
PRINT CUSTID LASTNAME FIRSTNAME EMAIL
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:

<table>
<thead>
<tr>
<th>CUSTID</th>
<th>LASTNAME</th>
<th>FIRSTNAME</th>
<th>EMAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0944</td>
<td>HANDLER</td>
<td>EVAN</td>
<td><a href="mailto:handy_man@usa.com">handy_man@usa.com</a></td>
</tr>
<tr>
<td>0944</td>
<td>HANDLER</td>
<td>EVAN</td>
<td><a href="mailto:handyman@usa.com">handyman@usa.com</a></td>
</tr>
</tbody>
</table>

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 e-mail field:

```
TABLE FILE VIDEOTR2
PRINT CUSTID LASTNAME FIRSTNAME EMAIL
IF EMAIL LIKE 'handy\_%' ESCAPE '\'
END
```

The output is:

<table>
<thead>
<tr>
<th>CUSTID</th>
<th>LASTNAME</th>
<th>FIRSTNAME</th>
<th>EMAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0944</td>
<td>HANDLER</td>
<td>EVAN</td>
<td><a href="mailto:handy_man@usa.com">handy_man@usa.com</a></td>
</tr>
</tbody>
</table>

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 155.

**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, 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**

### How to:

**Limit 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.

**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**

**How to:**

Select Records Using the IF Phrase

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 *Using Operators in Record Selection Tests* on page 157.

**Tip:** Unless you specifically require IF syntax (for example, to support legacy applications), we recommend using WHERE.
How to Select Records Using the IF Phrase

Syntax:

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 157.

literal
Can be the MISSING keyword (as described in Missing Data Tests on page 165) 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 315. You can accomplish this more easily with WHERE. See Using Compound Expressions for Record Selection on page 155.

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

**How to:**

Read Selection Values From a File (WHERE)
Read Selection Values From a File (IF)

**Reference:**

Usage Notes for Reading 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.

  If you use a HOLD file, it must either be in BINARY format (the default) or in ALPHA (simple character) format. If you use a SAVE file, it must be in ALPHA format (the default). You can also use a SAVB file if the selection values are alphanumeric. For information on HOLD and SAVE files, see *Saving and Reusing Your Report Output* on page 351.

  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)

```
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.
For z/OS, this is the ddname assigned by a DYNAM or TSO ALLOCATE command. On CMS, the ddname is assigned by a FILEDEF command.

For related information, see *Usage Notes for Reading Values From a File* on page 180.

**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 157).

*file*

Is the name of the file.

For z/OS, this is the ddname assigned by a DYNAM or TSO ALLOCATE command.

For CMS, this is the ddname assigned by a FILEDEF command.

**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).

- For IF, the total of all files can be up to 32,767 literals, including new line and other formatting characters. Lower limits apply to fixed sequential and other non-relational data sources.

- For WHERE, the file can be approximately 16,000 bytes. If the file is too large, an error message displays.

- 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. Actual literals may also be mixed with the file names. For example:

IF filename operator (filename) OR literal...etc...

**Example:** **Reading Selection Values From a File (WHERE)**

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:

```asp
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:

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>French Roast</td>
<td>81.00</td>
</tr>
<tr>
<td>Hazelnut</td>
<td>58.00</td>
</tr>
</tbody>
</table>

**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:

```asp
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:

<table>
<thead>
<tr>
<th>Product</th>
<th>Unit Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>French Roast</td>
<td>81.00</td>
</tr>
<tr>
<td>Hazelnut</td>
<td>58.00</td>
</tr>
</tbody>
</table>

**Assigning Screening Conditions to a File**

**In this section:**
- Preserving Filters Across Joins

**How to:**
- Declare a Filter
- Activate or Deactivate Filters
- Query the Status of Filters

**Reference:**
- Usage Notes for Virtual Fields Used in Filters

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 183.

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.
- Restrict 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 185.

### 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=filtername2 [,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 184.
- `filtername1...filtername2` 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.
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.

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.

* **ON**
  
  Activates all (*) or specifically named filters for the data source. 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. 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:

<table>
<thead>
<tr>
<th>Set File</th>
<th>Filter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>CAR</td>
<td>PETER</td>
</tr>
<tr>
<td>*</td>
<td>EMPLOYEE</td>
<td>DAVE</td>
</tr>
</tbody>
</table>

The asterisk in the first column indicates that a filter is activated.

**Preserving Filters Across Joins**

**How to: Preserve Filter Definitions With KEEPFILTERS**

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.

**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

In this section:

Reporting From Files With Alternate Indexes

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.
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.

Topics:
- What Is a Temporary Field?
- Defining a Virtual Field
- Creating a Calculated Value
- Assigning Column Reference Numbers
- 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:

\[
\text{deduction} \, / \, \text{salary}
\]

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:

1. Locates the Master Files and data sources.
2. Retrieves records that meet selection criteria.
3. Determines values of virtual fields (DEFINE).
4. Applies filtering criteria to virtual field values.
5. Sorts the data.
6. Applies a display command.
7. Determines calculated values (COMPUTE).
8. Applies filtering criteria to total values or calculated values.
10. Formats the report.
11. Sends the report to a screen, printer or file.

Record-by-record processing against data source values (create the internal matrix)

Processing against the internal matrix
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:

```
DELIVER_AMT  OPENING_AMT  DRATIO  CRATIO
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

In this section:
Defining Multiple Virtual Fields
Displaying Virtual Fields
Clearing a Virtual Field
Establishing a Segment Location for a Virtual Field
Defining Virtual Fields Using a Multi-Path Data Source
Increasing the Speed of Calculations in Virtual Fields
Preserving Virtual Fields Using DEFINE FILE SAVE and RETURN
Applying Dynamically Formatted Virtual Fields to Report Columns

How to:
Create a Virtual Field

Reference:
Usage Notes for Creating Virtual Fields

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 723.
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 723.

- 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 206.

- 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]
filename[/format]=expression;
filename[/format][WITH realfield]=expression;
filename[/format] REDEFINES qualifier.fieldname=expression;
```

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 or field names that exceed the 12-character limit. For information on alternate views, see *Rotating a Data Structure for Enhanced Retrieval* on page 1428.

  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.
**Defining a Virtual Field**

`fieldname`

Is a name of up to 66 characters. Indexed field names 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 Cn, En, or 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. All formats except text fields (TX) are allowed. The default value is D12.2. For information on field formats, see the *Describing Data With WebFOCUS Language* manual.

**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 198.

**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 315). 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 198.

**END**

Is required to end the DEFINE FILE command. END must be on its own line in the procedure.
**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:

<table>
<thead>
<tr>
<th>DELIVER_AMT</th>
<th>OPENING_AMT</th>
<th>RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>65</td>
<td>1.23</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>1.00</td>
</tr>
<tr>
<td>80</td>
<td>90</td>
<td>0.89</td>
</tr>
</tbody>
</table>

**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:

```
DEFINE FILE EMPDATA
SALARY REDEFINES EMPDATA.SALARY =
  IF TITLE CONTAINS 'EXECUTIVE' THEN 999999999999 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:

<table>
<thead>
<tr>
<th>TITLE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC MANAGER</td>
<td>$54,100.00</td>
</tr>
<tr>
<td>EXECUTIVE MANAGER</td>
<td>**</td>
</tr>
<tr>
<td>MANAGER</td>
<td>$270,500.00</td>
</tr>
<tr>
<td>MARKETING DIRECTOR</td>
<td>$176,800.00</td>
</tr>
<tr>
<td>MARKETING EXECUTIVE</td>
<td>**</td>
</tr>
<tr>
<td>MARKETING SUPERVISOR</td>
<td>$50,500.00</td>
</tr>
<tr>
<td>SALES EXECUTIVE</td>
<td>**</td>
</tr>
<tr>
<td>SALES MANAGER</td>
<td>$70,000.00</td>
</tr>
<tr>
<td>SALES SPECIALIST</td>
<td>$82,000.00</td>
</tr>
<tr>
<td>SENIOR SALES EXEC.</td>
<td>$43,400.00</td>
</tr>
</tbody>
</table>

**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 replace by asterisks:

```sql
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:

<table>
<thead>
<tr>
<th>LASTNAME</th>
<th>ORIGINAL</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIRSCMAN</td>
<td>$62,500.00</td>
<td>$75,000.00</td>
</tr>
<tr>
<td>KASHMAN</td>
<td>$33,300.00</td>
<td>$39,960.00</td>
</tr>
<tr>
<td>LASTRA</td>
<td>************</td>
<td>$138,000.00</td>
</tr>
<tr>
<td>LEWIS</td>
<td>************</td>
<td>$62,400.00</td>
</tr>
<tr>
<td>LIEBER</td>
<td>************</td>
<td>$31,680.00</td>
</tr>
<tr>
<td>LOPEZ</td>
<td>************</td>
<td>$31,680.00</td>
</tr>
<tr>
<td>MARTIN</td>
<td>$49,000.00</td>
<td>************</td>
</tr>
<tr>
<td>MEDINA</td>
<td>$39,000.00</td>
<td>************</td>
</tr>
<tr>
<td>MORAN</td>
<td>$30,800.00</td>
<td>************</td>
</tr>
<tr>
<td>NOZAWA</td>
<td>$80,500.00</td>
<td>$96,600.00</td>
</tr>
<tr>
<td>OLSON</td>
<td>$30,500.00</td>
<td>$36,600.00</td>
</tr>
</tbody>
</table>

**Defining Multiple Virtual Fields**

**How to:**

Add a Virtual Field to Existing 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.
**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

**How to:**

Display 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.

### 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 *Developing Reporting Applications* 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 1427.

**Increasing the Speed of Calculations in Virtual Fields**

**How to:**

Set Compilation of Virtual Fields

You can prevent virtual fields from being compiled, and have them interpreted at execution time, by changing the setting of the DEFINES SET parameter. Calculations are then interpreted each time the calculation is performed, which increases the execution time of the procedure.
In order to compile calculations, there must be a C compiler available on Windows and UNIX. On z/OS, calculations are compiled directly into machine code. If, for any reason compilation fails, WebFOCUS automatically suppresses the SET DEFINES=COMPILED option and reverts to the default. For details on this parameter, see the Developing Reporting Applications manual.

Syntax: **How to Set Compilation of Virtual Fields**

```
SET DEFINES = \{COMPILED \| OLD\}
```
where:

- **COMPILED**
  - Compiles virtual fields.
- **OLD**
  - Does not compile virtual fields. OLD is the default value.

Preserving Virtual Fields Using DEFINE FILE SAVE and RETURN

**How to:**

Protect Virtual Fields From Being Overwritten

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 723.

**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**

**How to:**

Define and Apply a Format Field

**Reference:**

Usage Notes for Field-Based Reformatting

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.
Defining a Virtual Field

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
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 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.
**Example:** Creating Dynamically Formatted Fields

The following request formats the SALES field according to the value of the COUNTRY field:

```plaintext
DEFINE FILE CAR
MYFORMAT/A8=DECODE COUNTRY ('ENGLAND' 'P15.3C' 'JAPAN' 'P15.0' ELSE 'P15.2M');
END

TABLE FILE CAR
SUM SALES/MYFORMAT BY COUNTRY
END
```

The output is:

```
COUNTRY          SALES
-------          -----   
ENGLAND     12,000.000  
FRANCE            $.00  
ITALY       $30,200.00  
JAPAN           78030.  
W GERMANY   $88,190.00  
```

**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 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.
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 261.

- 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 315.

- 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]= expression;[AS 'title'] [NOPRINT] [IN [+n]]
```

where:

- **fld**
  
  Is the name of the calculated value.
  
  The name can be up to 66 characters long and can include any combination of letters, digits, and underscores (_). It should begin with a letter. Other characters are not recommended, and may cause problems in some operating environments or when resolving expressions.
  
  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. All formats except text fields (TX) are allowed. The default is D12.2. For information on formats, see the *Describing Data With WebFOCUS Language* manual.

- **expression**
  
  Can be an arithmetic and/or logical expression or function (see *Using Expressions* on page 315). 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 929.

- **AS 'title'**
  
  Changes the name of the calculated value. For more information, see *Using Headings, Footings, Titles, and Labels* on page 1053.

- **IN [+n]**
  
  Specifies the location of the column. For more information, see *Using Headings, Footings, Titles, and Labels* on page 1053. 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 up to 66 characters long and can include any combination of letters, digits, and underscores (_). It should begin with a letter. Other characters are not recommended, and may cause problems in some operating environments or when resolving expressions.

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. All formats except text fields (TX) are allowed. 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
```
Creating a Calculated Value

The output is:

NEW YORK PROFIT REPORT

<table>
<thead>
<tr>
<th>PROD CODE</th>
<th>UNITS SOLD</th>
<th>RETAIL PRICE</th>
<th>REVENUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>B10</td>
<td>30</td>
<td>$.85</td>
<td>$25.50</td>
</tr>
<tr>
<td>B17</td>
<td>20</td>
<td>$1.89</td>
<td>$37.80</td>
</tr>
<tr>
<td>B20</td>
<td>15</td>
<td>$1.99</td>
<td>$29.85</td>
</tr>
<tr>
<td>C13</td>
<td>15</td>
<td>$1.99</td>
<td>$29.85</td>
</tr>
<tr>
<td>C14</td>
<td>18</td>
<td>$2.05</td>
<td>$36.90</td>
</tr>
<tr>
<td>C17</td>
<td>12</td>
<td>$2.09</td>
<td>$25.08</td>
</tr>
<tr>
<td>D12</td>
<td>20</td>
<td>$2.09</td>
<td>$41.80</td>
</tr>
<tr>
<td>E1</td>
<td>30</td>
<td>$.89</td>
<td>$26.70</td>
</tr>
<tr>
<td>E2</td>
<td>33</td>
<td>$.99</td>
<td>$32.67</td>
</tr>
<tr>
<td>E3</td>
<td>35</td>
<td>$1.09</td>
<td>$38.15</td>
</tr>
</tbody>
</table>

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 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 218.

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:

<table>
<thead>
<tr>
<th>AVE DEALER_COST</th>
<th>COUNTRY</th>
<th>AVE DEALER_COST</th>
<th>RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>7,989</td>
<td>ENGLAND</td>
<td>9,463</td>
<td>.84</td>
</tr>
<tr>
<td>FRANCE</td>
<td>4,631</td>
<td>1.73</td>
<td></td>
</tr>
<tr>
<td>ITALY</td>
<td>10,309</td>
<td>.77</td>
<td></td>
</tr>
<tr>
<td>JAPAN</td>
<td>2,756</td>
<td>2.90</td>
<td></td>
</tr>
<tr>
<td>W GERMANY</td>
<td>7,795</td>
<td>1.02</td>
<td></td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>CITY</th>
<th>UNIT_SOLD</th>
<th>NEWVAL</th>
<th>UNIT_SOLD</th>
<th>NEWVAL</th>
<th>UNIT_SOLD</th>
<th>NEWVAL</th>
<th>UNIT_SOLD</th>
<th>NEWVAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW YORK</td>
<td>162</td>
<td>1,764.18</td>
<td>42</td>
<td>104.16</td>
<td>376</td>
<td>4,805.28</td>
<td>65</td>
<td>297.70</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>DATE</th>
<th>UNIT_SOLD</th>
<th>RETURNS</th>
<th>UNIT_SOLD</th>
<th>RETURNS</th>
<th>UNIT_SOLD</th>
<th>RETURNS</th>
<th>TOT_UNITS</th>
<th>TOT_RETURNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/17</td>
<td>162</td>
<td>15</td>
<td>78</td>
<td>2</td>
<td>29</td>
<td>1</td>
<td>269</td>
<td>18.00</td>
</tr>
</tbody>
</table>

## 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 119.

## 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 143.

## Assigning Column Reference Numbers

**In this section:**
- Using Column Notation in a Report Request

**How to:**
- Control 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 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

**Reference:** Usage Notes for Column Numbers

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:

<table>
<thead>
<tr>
<th>TRANSDATE</th>
<th>TRANSTOT</th>
<th>QUANTITY</th>
<th>PRODUCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>91/06/17</td>
<td>57.03</td>
<td>12.00</td>
<td>3,252.42</td>
</tr>
<tr>
<td>91/06/18</td>
<td>21.25</td>
<td>2.00</td>
<td>451.56</td>
</tr>
<tr>
<td>91/06/19</td>
<td>38.17</td>
<td>5.00</td>
<td>1,456.95</td>
</tr>
<tr>
<td>91/06/20</td>
<td>14.23</td>
<td>3.00</td>
<td>202.49</td>
</tr>
<tr>
<td>91/06/21</td>
<td>44.72</td>
<td>7.00</td>
<td>1,999.88</td>
</tr>
<tr>
<td>91/06/24</td>
<td>126.28</td>
<td>12.00</td>
<td>15,946.63</td>
</tr>
<tr>
<td>91/06/25</td>
<td>47.74</td>
<td>8.00</td>
<td>2,279.11</td>
</tr>
<tr>
<td>91/06/26</td>
<td>40.97</td>
<td>2.00</td>
<td>817.85</td>
</tr>
<tr>
<td>91/06/27</td>
<td>60.24</td>
<td>9.00</td>
<td>3,628.85</td>
</tr>
<tr>
<td>91/06/28</td>
<td>31.00</td>
<td>3.00</td>
<td>961.00</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>TRANSDATE</th>
<th>TRANSTOT</th>
<th>QUANTITY</th>
<th>PRODUCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>91/06/17</td>
<td>57.03</td>
<td>12.00</td>
<td>684.36</td>
</tr>
<tr>
<td>91/06/18</td>
<td>21.25</td>
<td>2.00</td>
<td>42.50</td>
</tr>
<tr>
<td>91/06/19</td>
<td>38.17</td>
<td>5.00</td>
<td>190.85</td>
</tr>
<tr>
<td>91/06/20</td>
<td>14.23</td>
<td>3.00</td>
<td>42.69</td>
</tr>
<tr>
<td>91/06/21</td>
<td>44.72</td>
<td>7.00</td>
<td>313.04</td>
</tr>
<tr>
<td>91/06/24</td>
<td>126.28</td>
<td>12.00</td>
<td>1,515.36</td>
</tr>
<tr>
<td>91/06/25</td>
<td>47.74</td>
<td>8.00</td>
<td>381.92</td>
</tr>
<tr>
<td>91/06/26</td>
<td>40.97</td>
<td>2.00</td>
<td>81.94</td>
</tr>
<tr>
<td>91/06/27</td>
<td>60.24</td>
<td>9.00</td>
<td>542.16</td>
</tr>
<tr>
<td>91/06/28</td>
<td>31.00</td>
<td>3.00</td>
<td>93.00</td>
</tr>
</tbody>
</table>
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.

```ml
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(%)' OVER
END
```

The output is:

<table>
<thead>
<tr>
<th></th>
<th>CURRENT YEAR</th>
<th>LAST YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH ON HAND</td>
<td>8784.00</td>
<td>7216.00</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
<td>4494.00</td>
<td>3483.00</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
<td>7961.00</td>
<td>6499.00</td>
</tr>
<tr>
<td>TOTAL CASH</td>
<td>21239.00</td>
<td>17198.00</td>
</tr>
<tr>
<td>CASH GROWTH(%)</td>
<td>23.50</td>
<td></td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th></th>
<th>NEXT_YR</th>
<th>CUR_YR</th>
<th>LAST_YR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH</td>
<td>25992.00</td>
<td>21239.00</td>
<td>17198.00</td>
</tr>
<tr>
<td>ACCOUNTS RECEIVABLE</td>
<td>21941.00</td>
<td>18829.00</td>
<td>15954.00</td>
</tr>
<tr>
<td>INVENTORY</td>
<td>31522.00</td>
<td>27307.00</td>
<td>23329.00</td>
</tr>
<tr>
<td>ASSETS ACTUAL</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>67,375</td>
<td>56,478</td>
<td></td>
</tr>
</tbody>
</table>
**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.

```sql
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)/I5SC = TOTCASH(*) - TOTCASH(*+1); AS 'CHANGE IN CASH'
END
```

The output is:

<table>
<thead>
<tr>
<th></th>
<th>NEXT_YR</th>
<th>CUR_YR</th>
<th>LAST_YR</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL CASH</td>
<td>25992.00</td>
<td>21239.00</td>
<td>17198.00</td>
</tr>
<tr>
<td>CHANGE IN CASH</td>
<td>4,752</td>
<td>4,044</td>
<td></td>
</tr>
</tbody>
</table>

**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).

```sql
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
```
Assigning Column Reference Numbers

The output is:

<table>
<thead>
<tr>
<th></th>
<th>E ACTUAL</th>
<th>E BUDGET</th>
<th>W ACTUAL</th>
<th>W BUDGET</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALES</td>
<td>6000.00</td>
<td>4934.00</td>
<td>7222.00</td>
<td>7056.00</td>
</tr>
<tr>
<td>COST</td>
<td>4650.00</td>
<td>3760.00</td>
<td>5697.00</td>
<td>5410.00</td>
</tr>
<tr>
<td>PROFIT</td>
<td>1,350</td>
<td>1,174</td>
<td>1,525</td>
<td>1,646</td>
</tr>
</tbody>
</table>

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                   OVER
RECAP ATOT/I8C = R1 + R2 + R3; AS 'TOTAL'      OVER
RECAP DIFF(2,10,2)/D8  = ATOT(*) - ATOT(*-1);
END
```
The output is:

<table>
<thead>
<tr>
<th></th>
<th>CUR_YR</th>
<th>AMT2</th>
<th>LAST_YR</th>
<th>AMOUNT</th>
<th>NEXT_YR</th>
<th>AMT3</th>
<th>AMT4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH</td>
<td>21,239</td>
<td>42,478</td>
<td>17,198</td>
<td>21,239</td>
<td>25,992</td>
<td>63,717</td>
<td>84,956</td>
</tr>
<tr>
<td>ACCTS. REC.</td>
<td>18,829</td>
<td>37,658</td>
<td>15,954</td>
<td>18,829</td>
<td>21,941</td>
<td>56,487</td>
<td>75,316</td>
</tr>
<tr>
<td>INVENTORY</td>
<td>27,307</td>
<td>54,614</td>
<td>23,329</td>
<td>27,307</td>
<td>31,522</td>
<td>81,921</td>
<td>109,228</td>
</tr>
<tr>
<td>TOTAL</td>
<td>67,375</td>
<td>134,750</td>
<td>56,481</td>
<td>67,375</td>
<td>79,455</td>
<td>202,125</td>
<td>269,500</td>
</tr>
<tr>
<td>DIFF</td>
<td>67,375</td>
<td>134,750</td>
<td>56,481</td>
<td>67,375</td>
<td>79,455</td>
<td>202,125</td>
<td>269,500</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>TRANSDATE</th>
<th>QUANTITY</th>
<th>TRANSCODE</th>
<th>TTOT2</th>
<th>UNIT_COST1</th>
<th>UNIT_COST2</th>
</tr>
</thead>
<tbody>
<tr>
<td>91/06/17</td>
<td>12.00</td>
<td>10</td>
<td>57.03</td>
<td>1.00</td>
<td>4.75</td>
</tr>
<tr>
<td>91/06/18</td>
<td>2.00</td>
<td>2</td>
<td>21.25</td>
<td>1.00</td>
<td>10.63</td>
</tr>
<tr>
<td>91/06/19</td>
<td>5.00</td>
<td>4</td>
<td>38.17</td>
<td>1.00</td>
<td>7.63</td>
</tr>
<tr>
<td>91/06/20</td>
<td>3.00</td>
<td>3</td>
<td>14.23</td>
<td>1.00</td>
<td>4.74</td>
</tr>
<tr>
<td>91/06/21</td>
<td>7.00</td>
<td>6</td>
<td>44.72</td>
<td>1.00</td>
<td>6.39</td>
</tr>
<tr>
<td>91/06/24</td>
<td>12.00</td>
<td>9</td>
<td>126.28</td>
<td>1.00</td>
<td>10.52</td>
</tr>
<tr>
<td>91/06/25</td>
<td>8.00</td>
<td>7</td>
<td>47.74</td>
<td>1.00</td>
<td>5.97</td>
</tr>
<tr>
<td>91/06/26</td>
<td>2.00</td>
<td>2</td>
<td>40.97</td>
<td>1.00</td>
<td>20.48</td>
</tr>
<tr>
<td>91/06/27</td>
<td>9.00</td>
<td>7</td>
<td>60.24</td>
<td>1.00</td>
<td>6.69</td>
</tr>
<tr>
<td>91/06/28</td>
<td>3.00</td>
<td>3</td>
<td>31.00</td>
<td>1.00</td>
<td>10.33</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>TRANSDATE</th>
<th>QUANTITY</th>
<th>TRANSCODE</th>
<th>TTOT2</th>
<th>UNIT_COST1</th>
<th>UNIT_COST2</th>
</tr>
</thead>
<tbody>
<tr>
<td>91/06/17</td>
<td>12.00</td>
<td>10</td>
<td>12.00</td>
<td>1.20</td>
<td>1.00</td>
</tr>
<tr>
<td>91/06/18</td>
<td>2.00</td>
<td>2</td>
<td>2.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>91/06/19</td>
<td>5.00</td>
<td>4</td>
<td>5.00</td>
<td>1.25</td>
<td>1.00</td>
</tr>
<tr>
<td>91/06/20</td>
<td>3.00</td>
<td>3</td>
<td>3.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>91/06/21</td>
<td>7.00</td>
<td>6</td>
<td>7.00</td>
<td>1.17</td>
<td>1.00</td>
</tr>
<tr>
<td>91/06/24</td>
<td>12.00</td>
<td>9</td>
<td>12.00</td>
<td>1.33</td>
<td>1.00</td>
</tr>
<tr>
<td>91/06/25</td>
<td>8.00</td>
<td>7</td>
<td>8.00</td>
<td>1.14</td>
<td>1.00</td>
</tr>
<tr>
<td>91/06/26</td>
<td>2.00</td>
<td>2</td>
<td>2.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>91/06/27</td>
<td>9.00</td>
<td>7</td>
<td>9.00</td>
<td>1.29</td>
<td>1.00</td>
</tr>
<tr>
<td>91/06/28</td>
<td>3.00</td>
<td>3</td>
<td>3.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>TRANSDATE</th>
<th>QUANTITY</th>
<th>TRANSCODE</th>
<th>TTOT2</th>
<th>UNIT_COST1</th>
<th>UNIT_COST2</th>
</tr>
</thead>
<tbody>
<tr>
<td>91/06/17</td>
<td>12.00</td>
<td>10</td>
<td>57.03</td>
<td>4.75</td>
<td>4.75</td>
</tr>
<tr>
<td>91/06/18</td>
<td>2.00</td>
<td>2</td>
<td>21.25</td>
<td>10.63</td>
<td>10.63</td>
</tr>
<tr>
<td>91/06/19</td>
<td>5.00</td>
<td>4</td>
<td>38.17</td>
<td>7.63</td>
<td>7.63</td>
</tr>
<tr>
<td>91/06/20</td>
<td>3.00</td>
<td>3</td>
<td>14.23</td>
<td>4.74</td>
<td>4.74</td>
</tr>
<tr>
<td>91/06/21</td>
<td>7.00</td>
<td>6</td>
<td>44.72</td>
<td>6.39</td>
<td>6.39</td>
</tr>
<tr>
<td>91/06/24</td>
<td>12.00</td>
<td>9</td>
<td>126.28</td>
<td>10.52</td>
<td>10.52</td>
</tr>
<tr>
<td>91/06/25</td>
<td>8.00</td>
<td>7</td>
<td>47.74</td>
<td>5.97</td>
<td>5.97</td>
</tr>
<tr>
<td>91/06/26</td>
<td>2.00</td>
<td>2</td>
<td>40.97</td>
<td>20.48</td>
<td>20.48</td>
</tr>
<tr>
<td>91/06/27</td>
<td>9.00</td>
<td>7</td>
<td>60.24</td>
<td>6.69</td>
<td>6.69</td>
</tr>
<tr>
<td>91/06/28</td>
<td>3.00</td>
<td>3</td>
<td>31.00</td>
<td>10.33</td>
<td>10.33</td>
</tr>
</tbody>
</table>

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)/I5SC=(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:

<table>
<thead>
<tr>
<th>LAST_YR</th>
<th>NEXT_YR</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL CASH</td>
<td>17,195</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>LAST_YR</th>
<th>NEXT_YR</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL CASH</td>
<td>17,195</td>
</tr>
<tr>
<td>CHANGE FROM CURRENT</td>
<td>-4,044</td>
</tr>
</tbody>
</table>

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
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 234.

- **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 239.
  - **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 241.
  - **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 242.

- **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 198.
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.

You can conditionally format forecasted values. For details, see *Controlling Report Formatting* on page 825.

**FORECAST Processing**

**How to:**
Calculate Trends and Predict Values

**Reference:**
Usage Notes for FORECAST
FORECAST Limits

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**

```plaintext
ON sortfield RECAP result_field[/fmt] = FORECAST(infield, interval, npredict, 'MOVAVE', npoint1); sendstyle
```

**EXPAVE calculation**

```plaintext
ON sortfield RECAP result_field[/fmt] = FORECAST(infield, interval, npredict, 'EXPAVE', npoint1);
```

**DOUBLEXP calculation**

```plaintext
ON sortfield RECAP fld1[/fmt] = FORECAST(infield, interval, npredict, 'DOUBLEXP', npoint1, npoint2);
```

**SEASONAL calculation**

```plaintext
ON sortfield RECAP fld1[/fmt] = FORECAST(infield, interval, npredict, 'SEASONAL', nperiod, npoint1, npoint2, npoint3);
```

**REGRESS calculation**

```plaintext
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=`. 

Creating Reports With WebFOCUS Language 231
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:

\[ nperiod \times npredict \]

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:

\[ k = \frac{2}{1 + npoint1} \]

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:

\[ g = \frac{2}{1 + npoint2} \]
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:

\[
p = \frac{2}{1 + n_{\text{point3}}}
\]

**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:

```plaintext
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 \( n \)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 \( n \)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.

```plaintext
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
```
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.

```groovy
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
```
### Calculating Trends and Predicting Values With FORECAST

The output is:

<table>
<thead>
<tr>
<th>Category</th>
<th>PERIOD</th>
<th>Unit Sales</th>
<th>DOLLARS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1</td>
<td>61666</td>
<td>801,123.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>54870</td>
<td>682,340.0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>61608</td>
<td>765,078.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>57050</td>
<td>691,274.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>59229</td>
<td>720,444.0</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>58466</td>
<td>742,457.0</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>60771</td>
<td>747,253.0</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>54633</td>
<td>655,896.0</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>57829</td>
<td>730,317.0</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>57012</td>
<td>724,412.0</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>51110</td>
<td>620,264.0</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>58981</td>
<td>762,328.0</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>0</td>
<td>694,975.0</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>0</td>
<td>719,879.0</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0</td>
<td>705,729.0</td>
</tr>
<tr>
<td>Food</td>
<td>1</td>
<td>54394</td>
<td>672,727.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>54894</td>
<td>699,073.0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>52713</td>
<td>642,802.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>58026</td>
<td>718,514.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>53289</td>
<td>660,740.0</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>58742</td>
<td>734,705.0</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>60127</td>
<td>760,586.0</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>55622</td>
<td>695,235.0</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>55787</td>
<td>683,140.0</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>57340</td>
<td>713,768.0</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>57459</td>
<td>710,138.0</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>57290</td>
<td>705,315.0</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>0</td>
<td>708,397.0</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>0</td>
<td>707,817.0</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0</td>
<td>708,651.0</td>
</tr>
</tbody>
</table>
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.

\[ k = \frac{2}{1+n} \]

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:

\[ EMA = (EMA \times (1-k)) + (datavalue \times k) \]

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
```

Creating Reports With WebFOCUS Language 239

5. Creating Temporary Fields
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:

\[ n=3 \text{ (number used to calculate weights)} \]
\[ k = \frac{2}{(1+n)} = \frac{2}{4} = 0.5 \]
\[ \text{EXPAVE} = (\text{EXPAVE}*(1-k))+(\text{new-DOLLARS}*k) = (801123*0.5) + (682340*0.50) \]
\[ = 400561.5 + 341170 = 741731.5 \]

The third EXPAVE value (753,404.8) is calculated as follows:

\[ \text{EXPAVE} = (\text{EXPAVE}*(1-k))+(\text{new-DOLLARS}*k) = (741731.5*0.5)+(765078*0.50) \]
\[ = 370865.75 + 382539 = 753404.75 \]

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:

\[ (706,741.6 * 0.5) + (706,741.6 * 0.5) = 706,741.6 \]

\[ \text{EXPAVE} = (\text{EXPAVE} * (1-k)) + (\text{new-DOLLARS} * k) = (706741.6*0.5) + (706741.6*0.50) \]
\[ = 353370.8 + 353370.8 = 706741.6 \]

**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 \( b(t) \) that represents the trend for the previous period. The weight constant is \( k \):

  \[ \text{DOUBLEXP}(t) = k * \text{datavalue}(t) + (1-k) * ((\text{DOUBLEXP}(t-1) + b(t-1)) \]

- 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 \( g \):

  \[ b(t) = g * (\text{DOUBLEXP}(t)-\text{DOUBLEXP}(t-1)) + (1-g) * (b(t-1)) \]

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

\[ \text{forecast}(t+m) = \text{DOUBLEXP}(t) + m \times b(t) \]

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:

```sql
SET HISTOGRAM = OFF
TABLE FILE CENTSTMT
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:

<table>
<thead>
<tr>
<th>PERIOD</th>
<th>Actual</th>
<th>EXP</th>
<th>DOUBLEXP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002/01</td>
<td>12,957,681</td>
<td>12,957,681</td>
<td>12,957,681</td>
</tr>
<tr>
<td>2002/02</td>
<td>25,441,971</td>
<td>19,199,826</td>
<td>22,439,246</td>
</tr>
<tr>
<td>2002/03</td>
<td>39,164,321</td>
<td>29,182,073</td>
<td>34,791,885</td>
</tr>
<tr>
<td>2002/04</td>
<td>52,733,326</td>
<td>40,957,699</td>
<td>48,845,816</td>
</tr>
<tr>
<td>2002/05</td>
<td>66,765,920</td>
<td>53,861,809</td>
<td>63,860,955</td>
</tr>
<tr>
<td>2002/06</td>
<td>80,952,492</td>
<td>67,407,150</td>
<td>79,188,052</td>
</tr>
</tbody>
</table>

**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’s 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:

  \[ SEASONAL(t) = k \times \frac{\text{datavalue}(t)}{I(t-L)} + (1-k) \times (SEASONAL(t-1) + b(t-1)) \]

- 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:

  \[ b(t) = g \times (SEASONAL(t) - SEASONAL(t-1)) + (1-g) \times (b(t-1)) \]

- 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:

  \[ I(t) = p \times \frac{\text{datavalue}(t)}{SEASONAL(t)} + (1-p) \times I(t-L) \]

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:

   \[ b(0) = \frac{1}{L} \left( \frac{(y(L+1)-y(1))}{L} + \frac{(y(L+2)-y(2))}{L} + \ldots + \frac{(y(2L)-y(L))}{L} \right) \]

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):

   \[ A(j) = \frac{y((j-1)L+1) + y((j-1)L+2) + \ldots + y(jL))}{L} \]

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):

   \[ I(n) = \frac{y(n)/A(1) + y(L+n)/A(2) + \ldots + y((N-1)L+n)/A(N)}{N} \]

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:

\[ \text{forecast}(t+m) = \frac{\text{SEASONAL}(t) + m \times b(t)}{I(t-L+\text{MOD}(m/L))} \]

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:

```sql
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, \( nperiod \times npredict \)) are generated.

<table>
<thead>
<tr>
<th>TRANSDATE</th>
<th>TRANSTOT</th>
<th>SEASONAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>91/06/18</td>
<td>21.25</td>
<td>21.3</td>
</tr>
<tr>
<td>91/06/19</td>
<td>38.17</td>
<td>31.0</td>
</tr>
<tr>
<td>91/06/20</td>
<td>14.23</td>
<td>34.6</td>
</tr>
<tr>
<td>91/06/21</td>
<td>44.72</td>
<td>53.2</td>
</tr>
<tr>
<td>91/06/24</td>
<td>126.28</td>
<td>75.3</td>
</tr>
<tr>
<td>91/06/25</td>
<td>47.74</td>
<td>82.7</td>
</tr>
<tr>
<td>91/06/26</td>
<td>40.97</td>
<td>73.7</td>
</tr>
<tr>
<td>91/06/27</td>
<td>60.24</td>
<td>62.9</td>
</tr>
<tr>
<td>91/06/28</td>
<td>31.00</td>
<td>66.3</td>
</tr>
<tr>
<td>91/06/29</td>
<td></td>
<td>45.7</td>
</tr>
<tr>
<td>91/06/30</td>
<td></td>
<td>94.1</td>
</tr>
<tr>
<td>91/07/01</td>
<td></td>
<td>53.4</td>
</tr>
<tr>
<td>91/07/02</td>
<td></td>
<td>72.3</td>
</tr>
<tr>
<td>91/07/03</td>
<td></td>
<td>140.0</td>
</tr>
<tr>
<td>91/07/04</td>
<td></td>
<td>75.8</td>
</tr>
<tr>
<td>91/07/05</td>
<td></td>
<td>98.9</td>
</tr>
<tr>
<td>91/07/06</td>
<td></td>
<td>185.8</td>
</tr>
<tr>
<td>91/07/07</td>
<td></td>
<td>98.2</td>
</tr>
</tbody>
</table>
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.

\[ m = \frac{\left( \sum xy - \left( \sum x \cdot \sum y \right)/n \right)}{\left( \sum x^2 - \left( \sum x \right)^2/n \right)} \]

\[ b = \frac{\left( \sum y \right)/n - \left( m \cdot \left( \sum x \right)/n \right)} \]

where:

\[ n \]
Is the number of data points.

\[ y \]
Are the data values (dependent variables).

\[ x \]
Are 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

```plaintext
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:

<table>
<thead>
<tr>
<th>DEALER_COST</th>
<th>MPG</th>
<th>FORMPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,886</td>
<td>27</td>
<td>25.51</td>
</tr>
<tr>
<td>4,292</td>
<td>25</td>
<td>23.65</td>
</tr>
<tr>
<td>4,631</td>
<td>21</td>
<td>23.20</td>
</tr>
<tr>
<td>4,915</td>
<td>21</td>
<td>22.82</td>
</tr>
<tr>
<td>5,063</td>
<td>23</td>
<td>22.63</td>
</tr>
<tr>
<td>5,660</td>
<td>21</td>
<td>21.83</td>
</tr>
<tr>
<td>5,800</td>
<td>24</td>
<td>21.65</td>
</tr>
<tr>
<td>6,000</td>
<td>24</td>
<td>21.38</td>
</tr>
<tr>
<td>7,427</td>
<td>16</td>
<td>19.49</td>
</tr>
<tr>
<td>8,300</td>
<td>18</td>
<td>18.33</td>
</tr>
<tr>
<td>8,400</td>
<td>18</td>
<td>18.20</td>
</tr>
<tr>
<td>10,000</td>
<td>18</td>
<td>16.08</td>
</tr>
<tr>
<td>11,000</td>
<td>18</td>
<td>14.75</td>
</tr>
<tr>
<td>11,194</td>
<td>9</td>
<td>14.50</td>
</tr>
<tr>
<td>14,940</td>
<td>11</td>
<td>9.53</td>
</tr>
<tr>
<td>15,940</td>
<td>0</td>
<td>8.21</td>
</tr>
<tr>
<td>16,940</td>
<td>0</td>
<td>6.88</td>
</tr>
<tr>
<td>17,940</td>
<td>0</td>
<td>5.55</td>
</tr>
</tbody>
</table>

### 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 MPFGPFORECAST trend and predicted values.

In this case, the equation is approximately as follows:

\[
FORMPG = (-0.001323 \times DEALER\_COST) + 29.32
\]
The predicted values are (the values are not exactly as calculated by FORECAST because of rounding, but they show the calculation process):

<table>
<thead>
<tr>
<th>DEALER_COST</th>
<th>Calculation</th>
<th>FORMPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>15,940</td>
<td>$(-0.001323 \times 15,940) + 29.32$</td>
<td>8.23</td>
</tr>
<tr>
<td>16,940</td>
<td>$(-0.001323 \times 16,940) + 29.32$</td>
<td>6.91</td>
</tr>
<tr>
<td>17,940</td>
<td>$(-0.001323 \times 17,940) + 29.32$</td>
<td>5.59</td>
</tr>
</tbody>
</table>

**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 DOLEXPAVE/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:

<table>
<thead>
<tr>
<th>PER</th>
<th>DOLLARS</th>
<th>BUDGET</th>
<th>DOLMOVAVE</th>
<th>DOLEXPAVE</th>
<th>BUDMOVAVE</th>
<th>BUDEXPAVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>801123</td>
<td>801375</td>
<td>801,123.0</td>
<td>801,123.0</td>
<td>801,375.0</td>
<td>801,375.0</td>
</tr>
<tr>
<td>2</td>
<td>682340</td>
<td>725117</td>
<td>741,731.5</td>
<td>753,609.8</td>
<td>763,246.0</td>
<td>770,871.8</td>
</tr>
<tr>
<td>3</td>
<td>765078</td>
<td>810367</td>
<td>749,513.7</td>
<td>758,197.1</td>
<td>778,953.0</td>
<td>786,669.9</td>
</tr>
<tr>
<td>4</td>
<td>691274</td>
<td>717688</td>
<td>712,897.3</td>
<td>731,427.8</td>
<td>751,057.3</td>
<td>759,077.1</td>
</tr>
<tr>
<td>5</td>
<td>720444</td>
<td>739999</td>
<td>725,598.7</td>
<td>727,034.3</td>
<td>756,018.0</td>
<td>751,445.9</td>
</tr>
<tr>
<td>6</td>
<td>742457</td>
<td>742586</td>
<td>718,058.3</td>
<td>733,203.4</td>
<td>733,424.3</td>
<td>747,901.9</td>
</tr>
<tr>
<td>7</td>
<td>747253</td>
<td>773136</td>
<td>736,718.0</td>
<td>738,823.2</td>
<td>751,907.0</td>
<td>757,995.6</td>
</tr>
<tr>
<td>8</td>
<td>655896</td>
<td>685170</td>
<td>715,202.0</td>
<td>705,652.3</td>
<td>733,630.7</td>
<td>728,865.3</td>
</tr>
<tr>
<td>9</td>
<td>730317</td>
<td>753760</td>
<td>711,155.3</td>
<td>715,518.2</td>
<td>737,355.3</td>
<td>738,823.2</td>
</tr>
<tr>
<td>10</td>
<td>724412</td>
<td>709397</td>
<td>703,541.7</td>
<td>719,075.7</td>
<td>716,109.0</td>
<td>727,052.7</td>
</tr>
<tr>
<td>11</td>
<td>620264</td>
<td>630452</td>
<td>691,664.3</td>
<td>679,551.0</td>
<td>697,869.7</td>
<td>688,412.4</td>
</tr>
<tr>
<td>12</td>
<td>762328</td>
<td>718837</td>
<td>702,334.7</td>
<td>712,661.8</td>
<td>686,228.7</td>
<td>700,582.3</td>
</tr>
</tbody>
</table>

Calculating Trends and Predicting Values With FORECAST
**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.

```plaintext
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:

<table>
<thead>
<tr>
<th>Category</th>
<th>PERIOD</th>
<th>Unit Sales</th>
<th>MOVAVE</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1</td>
<td>61666</td>
<td>801,123</td>
<td>801123</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>54870</td>
<td>741,731</td>
<td>682340</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>61608</td>
<td>749,513</td>
<td>765078</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>57050</td>
<td>712,897</td>
<td>691274</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>59229</td>
<td>725,598</td>
<td>720444</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>58466</td>
<td>718,058</td>
<td>742457</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>60771</td>
<td>736,718</td>
<td>747253</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>54633</td>
<td>715,202</td>
<td>655896</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>57829</td>
<td>711,155</td>
<td>730317</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>57012</td>
<td>703,541</td>
<td>724412</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>51110</td>
<td>691,664</td>
<td>620264</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>58981</td>
<td>702,334</td>
<td>762328</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>0</td>
<td>694,975</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>0</td>
<td>719,879</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0</td>
<td>705,729</td>
<td>0</td>
</tr>
</tbody>
</table>

Creating Reports With WebFOCUS Language 249
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.

```plaintext
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:

<table>
<thead>
<tr>
<th>DEALER_COST</th>
<th>DATA_ROW</th>
<th>PREDICT</th>
<th>MPG</th>
<th>FORMPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,886</td>
<td>1</td>
<td>NO</td>
<td>27.00</td>
<td>25.65</td>
</tr>
<tr>
<td>4,292</td>
<td>1</td>
<td>NO</td>
<td>25.00</td>
<td>23.91</td>
</tr>
<tr>
<td>4,631</td>
<td>1</td>
<td>NO</td>
<td>21.00</td>
<td>23.49</td>
</tr>
<tr>
<td>4,915</td>
<td>1</td>
<td>NO</td>
<td>21.00</td>
<td>23.14</td>
</tr>
<tr>
<td>5,063</td>
<td>1</td>
<td>NO</td>
<td>23.00</td>
<td>22.95</td>
</tr>
<tr>
<td>5,660</td>
<td>1</td>
<td>NO</td>
<td>21.00</td>
<td>22.21</td>
</tr>
<tr>
<td>5,800</td>
<td>1</td>
<td>NO</td>
<td>21.00</td>
<td>22.21</td>
</tr>
<tr>
<td>6,000</td>
<td>1</td>
<td>NO</td>
<td>24.20</td>
<td>22.04</td>
</tr>
<tr>
<td>7,000</td>
<td>0</td>
<td>YES</td>
<td>20.56</td>
<td>20.56</td>
</tr>
<tr>
<td>8,000</td>
<td>0</td>
<td>YES</td>
<td>19.32</td>
<td>19.32</td>
</tr>
<tr>
<td>9,000</td>
<td>0</td>
<td>YES</td>
<td>18.08</td>
<td>18.08</td>
</tr>
</tbody>
</table>
```

**Calculating Trends and Predicting Values With Multivariate REGRESS**

**How to:**

Create a Multivariate Linear Regression Column

**Reference:**

Usage Notes for 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 (\(x_1\), \(x_2\), \(x_3\)) are related by the following linear equation:

\[
y = a_1 x_1 + a_2 x_2 + a_3 x_3 + b
\]

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 (\(a_1\), \(a_2\), \(a_3\), and \(b\)) that minimize the sum of the squared differences between the data and the resulting line, plane, or hyperplane.

**Syntax:**

```
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:

```sql
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
```
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:

<table>
<thead>
<tr>
<th>ADESC</th>
<th>NEWDESC</th>
</tr>
</thead>
<tbody>
<tr>
<td>This course provides the DP</td>
<td>This course provides the DP</td>
</tr>
<tr>
<td>Anyone responsible for designing FOC</td>
<td>Anyone responsible for designing FOC</td>
</tr>
<tr>
<td>This is a course in FOCUS</td>
<td>This is a course in FOCUS</td>
</tr>
<tr>
<td>efficiency</td>
<td>efficiency</td>
</tr>
</tbody>
</table>

Creating Temporary Fields Independent of a Master File

How to:
- Display DEFINE Functions
- Define a Function
- Clear DEFINE Functions

Reference:
- DEFINE Function Limits and Restrictions

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/formata = expressiona;]
    .
    .
    [tempvariablex/formatx = expressionx;]
name/format = [result_expression];
END
```

where:

- **name**
  - Is the name of the function. 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.

- **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.

- **tempformata...tempformatx**
  - Are the formats of the temporary fields.

- **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.
- Function names are limited to eight characters.
- Argument names are limited to twelve characters. There is no limit to the number of arguments.
- 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 WHOLESALEPR AND
  COMPUTE PROFIT/D8.2 = SUBTRACT(LISTPR, WHOLESALEPR);
  BY CATEGORY
    WHERE CATEGORY EQ 'MYSTERY' OR 'ACTION'
END
```
Creating Temporary Fields Independent of a Master File

The output is:

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>TITLE</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
<th>PROFIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION</td>
<td>JAWS</td>
<td>19.95</td>
<td>10.99</td>
<td>8.96</td>
</tr>
<tr>
<td></td>
<td>ROBOCOP</td>
<td>19.98</td>
<td>11.50</td>
<td>8.48</td>
</tr>
<tr>
<td></td>
<td>TOTAL RECALL</td>
<td>19.99</td>
<td>11.99</td>
<td>8.00</td>
</tr>
<tr>
<td></td>
<td>TOP GUN</td>
<td>14.95</td>
<td>9.99</td>
<td>4.96</td>
</tr>
<tr>
<td></td>
<td>RAMBO III</td>
<td>19.95</td>
<td>10.99</td>
<td>8.96</td>
</tr>
<tr>
<td>MYSTERY</td>
<td>REAR WINDOW</td>
<td>19.98</td>
<td>9.00</td>
<td>10.98</td>
</tr>
<tr>
<td></td>
<td>VERTIGO</td>
<td>19.98</td>
<td>9.00</td>
<td>10.98</td>
</tr>
<tr>
<td></td>
<td>FATAL ATTRACTION</td>
<td>29.98</td>
<td>15.99</td>
<td>13.99</td>
</tr>
<tr>
<td></td>
<td>NORTH BY NORTHWEST</td>
<td>19.98</td>
<td>9.00</td>
<td>10.98</td>
</tr>
<tr>
<td></td>
<td>DEAD RINGERS</td>
<td>25.99</td>
<td>15.99</td>
<td>10.00</td>
</tr>
<tr>
<td></td>
<td>PSYCHO</td>
<td>19.98</td>
<td>9.00</td>
<td>10.98</td>
</tr>
<tr>
<td></td>
<td>BIRDS, THE</td>
<td>19.98</td>
<td>9.00</td>
<td>10.98</td>
</tr>
<tr>
<td></td>
<td>SEA OF LOVE</td>
<td>59.99</td>
<td>30.00</td>
<td>29.99</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>FUNCTIONS CURRENTLY ACTIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>SUBTRACT</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

If you issue the ? FUNCTION command when no functions are defined, the following appears:

```
NO FUNCTIONS CURRENTLY IN EFFECT
```
How to Clear DEFINE Functions

Syntax:

```
DEFINE FUNCTION {name|*} CLEAR
```

where:

- `name`
  - Is the name of the function name to clear.
- `*`
  - Clears all active DEFINE functions.
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.

**Topics:**
- Calculating Row and Column Totals
- Including Section Totals and a Grand Total
- Including Subtotals
- Recalculating Values for Subtotal Rows
- 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

In this section:
Producing Row Totals for Horizontal (ACROSS) Sort Field Values

How to:
Calculate Row and Column Totals

Reference:
Using ROW-TOTAL With ACROSS and Multiple Display Commands

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 1053.

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.

ame

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']
Calculating Row and Column Totals

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:

<table>
<thead>
<tr>
<th>PROD_CODE</th>
<th>RETURNS</th>
<th>DAMAGED</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>B10</td>
<td>13</td>
<td>10</td>
<td>23</td>
</tr>
<tr>
<td>B12</td>
<td>4</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>B17</td>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>B20</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>C13</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>C17</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C7</td>
<td>5</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>D12</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>E1</td>
<td>4</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>E2</td>
<td>9</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>E3</td>
<td>12</td>
<td>11</td>
<td>23</td>
</tr>
<tr>
<td>TOTAL</td>
<td>58</td>
<td>45</td>
<td>103</td>
</tr>
</tbody>
</table>

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
```
### 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:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>MIS</th>
<th>PRODUCTION</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANK_NAME</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>ASSOCIATED</td>
<td>$21,780.00</td>
<td>$42,962.00</td>
<td>$64,742.00</td>
</tr>
<tr>
<td>BANK ASSOCIATION</td>
<td>$27,062.00</td>
<td>.</td>
<td>$27,062.00</td>
</tr>
<tr>
<td>BEST BANK</td>
<td>.</td>
<td>$29,700.00</td>
<td>$29,700.00</td>
</tr>
<tr>
<td>STATE</td>
<td>$18,480.00</td>
<td>.</td>
<td>$18,480.00</td>
</tr>
<tr>
<td>TOTAL</td>
<td>$108,002.00</td>
<td>$114,282.00</td>
<td>$222,284.00</td>
</tr>
</tbody>
</table>
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.

```sql
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:

<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>DEALER_COST</th>
<th>RETAIL_COST</th>
<th>PROFIT</th>
<th>TOTAL_COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGLAND</td>
<td>37,853</td>
<td>45,319</td>
<td>7,466</td>
<td>90,638</td>
</tr>
<tr>
<td>FRANCE</td>
<td>4,631</td>
<td>5,610</td>
<td>979</td>
<td>11,220</td>
</tr>
<tr>
<td>ITALY</td>
<td>41,235</td>
<td>51,065</td>
<td>9,830</td>
<td>102,130</td>
</tr>
<tr>
<td>JAPAN</td>
<td>5,512</td>
<td>6,478</td>
<td>966</td>
<td>12,956</td>
</tr>
<tr>
<td>W GERMANY</td>
<td>54,563</td>
<td>64,732</td>
<td>10,169</td>
<td>129,464</td>
</tr>
<tr>
<td>FINAL_TOTAL</td>
<td>143,794</td>
<td>173,204</td>
<td>29,410</td>
<td>346,408</td>
</tr>
</tbody>
</table>

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:

```sql
TABLE FILE GGSALES
SUM UNITS AS 'U' DOLLARS AS 'D' BY CITY
ACROSS ST
IF ST EQ 'CA'
IF BUDUNITS NE MISSING
END
```

The output is:

<table>
<thead>
<tr>
<th>State</th>
<th>U</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>City</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Los Angeles</td>
<td>298070</td>
<td>3772014</td>
</tr>
<tr>
<td>San Francisco</td>
<td>312500</td>
<td>3870258</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>State</th>
<th>CA</th>
<th>U</th>
<th>D</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>U</td>
<td>D</td>
<td>U</td>
<td>D</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>---------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>298070</td>
<td>3772014</td>
<td>298070</td>
<td>3772014</td>
</tr>
<tr>
<td>San Francisco</td>
<td>312500</td>
<td>3870258</td>
<td>312500</td>
<td>3870258</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>City</th>
<th>U</th>
<th>D</th>
<th>BU</th>
<th>BD</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Los Angeles</td>
<td>298070</td>
<td>3772014</td>
<td>295637</td>
<td>3669484</td>
<td>593707</td>
</tr>
<tr>
<td>San Francisco</td>
<td>312500</td>
<td>3870258</td>
<td>314725</td>
<td>3916863</td>
<td>627225</td>
</tr>
</tbody>
</table>
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:

```sql
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:

<table>
<thead>
<tr>
<th>City</th>
<th>U</th>
<th>D</th>
<th>BU</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Los Angeles</td>
<td>298070</td>
<td>3772014</td>
<td>295637</td>
<td>593707</td>
</tr>
<tr>
<td>San Francisco</td>
<td>312500</td>
<td>3870258</td>
<td>314725</td>
<td>627225</td>
</tr>
</tbody>
</table>

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:

```sql
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:

<table>
<thead>
<tr>
<th>City</th>
<th>U</th>
<th>D</th>
<th>BU</th>
<th>ROW-TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Los Angeles</td>
<td>298070</td>
<td>3772014</td>
<td>295637</td>
<td>4365721</td>
</tr>
<tr>
<td>San Francisco</td>
<td>312500</td>
<td>3870258</td>
<td>314725</td>
<td>4497483</td>
</tr>
</tbody>
</table>
Producing Row Totals for Horizontal (ACROSS) Sort Field Values

**How to:**
Produce Row Totals for Horizontal (ACROSS) Sort Field Values

**Reference:**
Usage Notes for ACROSS-TOTAL

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
```
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 at the end of the report.

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**

**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 865.
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 313). You can also suppress grand totals using the NOTOTAL command. For details, see *Suppressing Grand Totals* on page 312.

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 280. In addition, you can combine different summary operations in a single request. For information, see *Combinations of Summary Commands* on page 299.

**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
```
Including Subtotals

The first and last portions of the output are:

<table>
<thead>
<tr>
<th>DED_CODE</th>
<th>DEPARTMENT</th>
<th>BANK_ACCT</th>
<th>DED_AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CITY</td>
<td>MIS</td>
<td>40950036</td>
<td>$14.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>122850108</td>
<td>$31.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>163800144</td>
<td>$82.70</td>
</tr>
<tr>
<td>*TOTAL DEPARTMENT MIS</td>
<td></td>
<td></td>
<td>$128.45</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td></td>
<td>160633</td>
<td>$7.42</td>
</tr>
<tr>
<td></td>
<td></td>
<td>136500120</td>
<td>$18.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>819000702</td>
<td>$60.20</td>
</tr>
<tr>
<td>*TOTAL DEPARTMENT PRODUCTION</td>
<td></td>
<td></td>
<td>$85.87</td>
</tr>
<tr>
<td>FED</td>
<td>MIS</td>
<td>40950036</td>
<td>$1,190.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>122850108</td>
<td>$2,699.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>163800144</td>
<td>$7,028.30</td>
</tr>
<tr>
<td>*TOTAL DEPARTMENT MIS</td>
<td></td>
<td></td>
<td>$10,918.87</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td></td>
<td>160633</td>
<td>$631.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>136500120</td>
<td>$1,552.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>819000702</td>
<td>$5,120.04</td>
</tr>
<tr>
<td>*TOTAL DEPARTMENT PRODUCTION</td>
<td></td>
<td></td>
<td>$7,303.26</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td>$18,436.45</td>
</tr>
</tbody>
</table>

Including Subtotals

**How to:**
Create Subtotals

**Reference:**
Usage Notes for 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 for all numeric values 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 312.
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:**

```
{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.

- **AS 'text'**
  
  Enables you to specify a different label. For related information, see *Using Headings, Footings, Titles, and Labels* on page 1053.

- **WHEN expression**
  
  Specifies the conditional display of subtotals as determined by a Boolean expression. You must end the expression with a semicolon.
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 subtotaling 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 1187.

- Subtotals display on the next line if the subtotal text does not fit on the line prior to the displayed field columns.

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:

<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>BODYTYPE</th>
<th>AVE</th>
<th>SALES</th>
<th>AVE RETAIL_COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGLAND</td>
<td>CONVERTIBLE</td>
<td>16</td>
<td>0</td>
<td>8,878</td>
</tr>
<tr>
<td></td>
<td>HARDTOP</td>
<td>25</td>
<td>0</td>
<td>5,100</td>
</tr>
<tr>
<td></td>
<td>SEDAN</td>
<td>10</td>
<td>12000</td>
<td>15,671</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12000</td>
</tr>
<tr>
<td>FRANCE</td>
<td>SEDAN</td>
<td>21</td>
<td>0</td>
<td>5,610</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>ITALY</td>
<td>COUPE</td>
<td>11</td>
<td>12400</td>
<td>19,160</td>
</tr>
<tr>
<td></td>
<td>ROADSTER</td>
<td>21</td>
<td>13000</td>
<td>6,820</td>
</tr>
<tr>
<td></td>
<td>SEDAN</td>
<td>21</td>
<td>4800</td>
<td>5,925</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30200</td>
</tr>
<tr>
<td>JAPAN</td>
<td>SEDAN</td>
<td>14</td>
<td>78030</td>
<td>3,239</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>78030</td>
</tr>
<tr>
<td>W GERMANY</td>
<td>SEDAN</td>
<td>20</td>
<td>88190</td>
<td>9,247</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>88190</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td></td>
<td>208420</td>
</tr>
</tbody>
</table>

### 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.
```
### Recalculating Values for Subtotal Rows

The first portion of the output is:

<table>
<thead>
<tr>
<th>DED_CODE</th>
<th>DEPARTMENT</th>
<th>BANK_ACCT</th>
<th>DED_AMT</th>
<th>GROSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CITY</td>
<td>MIS</td>
<td>40950036</td>
<td>$14.00</td>
<td>$6,099.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>122850108</td>
<td>$31.75</td>
<td>$9,075.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>163800144</td>
<td>$82.70</td>
<td>$22,013.75</td>
</tr>
</tbody>
</table>

*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 DEPARTMENT CITY $214.32 $65,887.25

The last portion of the output is:

<table>
<thead>
<tr>
<th>DED_CODE</th>
<th>DEPARTMENT</th>
<th>BANK_ACCT</th>
<th>DED_AMT</th>
<th>GROSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT</td>
<td>MIS</td>
<td>40950036</td>
<td>$196.13</td>
<td>$6,099.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>122850108</td>
<td>$444.65</td>
<td>$9,075.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>163800144</td>
<td>$1,157.60</td>
<td>$22,013.75</td>
</tr>
</tbody>
</table>

*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

---

**How to:**

**Subtotal Calculated Values**

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 SUBTOTAL 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 312.
**Syntax:**

```
{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 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.

- **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 312.

- **AS 'text'**
  
  Enables you to specify a different label. For related information, see *Using Headings, Footings, Titles, and Labels* on page 1053.

- **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.

- **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:

```sql
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:

<table>
<thead>
<tr>
<th>PAY_DATE</th>
<th>DEPARTMENT</th>
<th>BANK_ACCT</th>
<th>GROSS</th>
<th>DED_AMT</th>
<th>DG_RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>82/08/31</td>
<td>MIS</td>
<td>40950036</td>
<td>$1,540.00</td>
<td>$725.34</td>
<td>.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>122850108</td>
<td>$1,815.00</td>
<td>$1,261.40</td>
<td>.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>163800144</td>
<td>$2,255.00</td>
<td>$1,668.69</td>
<td>.74</td>
</tr>
</tbody>
</table>

*TOTAL DEPARTMENT MIS $5,610.00 $3,655.43 .65

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>PRODUCTION</th>
<th>160633</th>
<th>$2,475.00</th>
<th>$1,427.24</th>
<th>.58</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>136500120</td>
<td>$1,342.00</td>
<td>$522.28</td>
<td>.39</td>
<td></td>
</tr>
<tr>
<td></td>
<td>819000702</td>
<td>$2,238.50</td>
<td>$1,746.03</td>
<td>.78</td>
<td></td>
</tr>
</tbody>
</table>

*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:

<table>
<thead>
<tr>
<th>PAY_DATE</th>
<th>DEPARTMENT</th>
<th>BANK_ACCT</th>
<th>GROSS</th>
<th>DED_AMT</th>
<th>DG_RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>82/01/29</td>
<td>PRODUCTION</td>
<td>819000702</td>
<td>$2,035.00</td>
<td>$1,241.33</td>
<td>.61</td>
</tr>
</tbody>
</table>

*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
```

The first portion of the output is:

<table>
<thead>
<tr>
<th>PAY_DATE</th>
<th>DEPARTMENT</th>
<th>BANK_ACCT</th>
<th>GROSS</th>
<th>DED_AMT</th>
<th>DG_RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>82/08/31</td>
<td>MIS</td>
<td>40950036</td>
<td>$1,540.00</td>
<td>$725.34</td>
<td>.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>122850108</td>
<td>$1,815.00</td>
<td>$1,261.40</td>
<td>.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>163800144</td>
<td>$2,255.00</td>
<td>$1,668.69</td>
<td>.74</td>
</tr>
<tr>
<td></td>
<td>*TOTAL DEPARTMENT MIS</td>
<td></td>
<td>$5,610.00</td>
<td>$3,655.43</td>
<td>.65</td>
</tr>
<tr>
<td></td>
<td>PRODUCTION</td>
<td>160633</td>
<td>$2,475.00</td>
<td>$1,427.24</td>
<td>.58</td>
</tr>
<tr>
<td></td>
<td></td>
<td>136500120</td>
<td>$1,342.00</td>
<td>$522.28</td>
<td>.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>819000702</td>
<td>$2,238.50</td>
<td>$1,746.03</td>
<td>.78</td>
</tr>
<tr>
<td></td>
<td>*TOTAL DEPARTMENT PRODUCTION</td>
<td></td>
<td>$6,055.50</td>
<td>$3,695.55</td>
<td>.61</td>
</tr>
<tr>
<td>82/07/30</td>
<td>MIS</td>
<td>40950036</td>
<td>$1,540.00</td>
<td>$725.34</td>
<td>.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>122850108</td>
<td>$1,815.00</td>
<td>$1,261.40</td>
<td>.69</td>
</tr>
</tbody>
</table>

The last portion of the output is:

<table>
<thead>
<tr>
<th>PAY_DATE</th>
<th>DEPARTMENT</th>
<th>BANK_ACCT</th>
<th>GROSS</th>
<th>DED_AMT</th>
<th>DG_RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>82/01/29</td>
<td>MIS</td>
<td>163800144</td>
<td>$2,147.75</td>
<td>$1,406.79</td>
<td>.66</td>
</tr>
<tr>
<td></td>
<td>*TOTAL DEPARTMENT MIS</td>
<td></td>
<td>$2,147.75</td>
<td>$1,406.79</td>
<td>.66</td>
</tr>
<tr>
<td></td>
<td>PRODUCTION</td>
<td>819000702</td>
<td>$2,035.00</td>
<td>$1,241.33</td>
<td>.61</td>
</tr>
<tr>
<td></td>
<td>*TOTAL DEPARTMENT PRODUCTION</td>
<td></td>
<td>$2,035.00</td>
<td>$1,241.33</td>
<td>.61</td>
</tr>
<tr>
<td>81/12/31</td>
<td>MIS</td>
<td>163800144</td>
<td>$2,147.75</td>
<td>$1,406.79</td>
<td>.66</td>
</tr>
<tr>
<td></td>
<td>*TOTAL DEPARTMENT MIS</td>
<td></td>
<td>$2,147.75</td>
<td>$1,406.79</td>
<td>.66</td>
</tr>
<tr>
<td>81/11/30</td>
<td>MIS</td>
<td>163800144</td>
<td>$2,147.75</td>
<td>$1,406.79</td>
<td>.66</td>
</tr>
<tr>
<td></td>
<td>*TOTAL DEPARTMENT MIS</td>
<td></td>
<td>$2,147.75</td>
<td>$1,406.79</td>
<td>.66</td>
</tr>
<tr>
<td></td>
<td>TOTAL</td>
<td></td>
<td>$65,887.25</td>
<td>$41,521.18</td>
<td>.63</td>
</tr>
</tbody>
</table>
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.

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.
The following prefix operators are supported for alphanumeric fields:

- CNT.
- FST.
- LST.
- MAX.
- MIN.
- SUM.

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:

\[ \text{BY STORE_CODE SUMMARIZE AVE. UNIT_SOLD RETURNS CNT. DAMAGED} \]

'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 WHOLESALEPR 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
```
### Manipulating Summary Values With Prefix Operators

The output is:

<table>
<thead>
<tr>
<th>RATING</th>
<th>CATEGORY</th>
<th>COPIES</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>G</td>
</tr>
<tr>
<td>G</td>
<td>CHILDREN</td>
<td>2</td>
<td>44.95</td>
<td>29.99</td>
<td>SHAGGY DOG, THE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>29.95</td>
<td>12.50</td>
<td>ALICE IN WONDERLAND</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>26.99</td>
<td>12.00</td>
<td>BAMBI</td>
</tr>
<tr>
<td>*Sum:</td>
<td>CHILDREN</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CLASSIC</td>
</tr>
<tr>
<td>CLASSIC</td>
<td></td>
<td>3</td>
<td>89.95</td>
<td>40.99</td>
<td>GONE WITH THE WIND</td>
</tr>
<tr>
<td>*Sum:</td>
<td>CLASSIC</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*Ave:</td>
<td>G</td>
<td>47.96</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 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 *Using Prefix Operators With SUBTOTAL* on page 283 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 WHOLESALEPR 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:

<table>
<thead>
<tr>
<th>RATING</th>
<th>CATEGORY</th>
<th>COPIES</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>CHILDREN</td>
<td>2</td>
<td>44.95</td>
<td>29.99</td>
<td>SHAGGY DOG, THE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>29.95</td>
<td>12.50</td>
<td>ALICE IN WONDERLAND</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>26.99</td>
<td>12.00</td>
<td>Bambi</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*Sum: CHILDREN 7</td>
</tr>
<tr>
<td></td>
<td>CLASSIC</td>
<td>3</td>
<td>89.95</td>
<td>40.99</td>
<td>GONE WITH THE WIND</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*Sum: CLASSIC 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*Ave: G 47.96</td>
</tr>
<tr>
<td>NR</td>
<td>CHILDREN</td>
<td>1</td>
<td>19.95</td>
<td>10.00</td>
<td>SMURFS, THE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>19.95</td>
<td>9.75</td>
<td>SCOOBY-DOO-A DOG IN THE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>14.95</td>
<td>7.65</td>
<td>SESAME STREET-BEDTIME S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>14.98</td>
<td>7.99</td>
<td>ROMPER ROOM-ASK MISS MO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>29.95</td>
<td>15.99</td>
<td>SLEEPING BEAUTY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*Sum: CHILDREN 5</td>
</tr>
<tr>
<td></td>
<td>CLASSIC</td>
<td>1</td>
<td>24.98</td>
<td>14.99</td>
<td>EAST OF EDEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>39.99</td>
<td>20.00</td>
<td>CITIZEN KANE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>29.95</td>
<td>15.99</td>
<td>CYRANO DE BERGERAC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>19.99</td>
<td>10.95</td>
<td>MARTY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>19.99</td>
<td>10.95</td>
<td>MALTESE FALCON, THE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>19.95</td>
<td>9.99</td>
<td>ON THE WATERFRONT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>89.99</td>
<td>40.99</td>
<td>MUTINY ON THE BOUNTY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>19.99</td>
<td>10.95</td>
<td>PHILADELPHIA STORY, THE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>19.98</td>
<td>10.99</td>
<td>CAT ON A HOT TIN ROOF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>29.95</td>
<td>15.00</td>
<td>CASABLANCA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*Sum: CLASSIC 18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*Ave: NR 27.64</td>
</tr>
</tbody>
</table>
```

| TOTAL  | 1    | 89.99 |
**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 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. LISTPR MIN. DIRECTOR AS '*A/N:'
END
```

The output is:

<table>
<thead>
<tr>
<th>RATING</th>
<th>CATEGORY</th>
<th>COPIES</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
<th>DIRECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>CHILDREN</td>
<td>2</td>
<td>44.95</td>
<td>29.99</td>
<td>BARTON C.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>29.95</td>
<td>12.50</td>
<td>GEROMINI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>26.99</td>
<td>12.00</td>
<td>DISNEY W.</td>
</tr>
<tr>
<td></td>
<td>CLASSIC</td>
<td>3</td>
<td>89.95</td>
<td>40.99</td>
<td>FLEMING V</td>
</tr>
</tbody>
</table>

*A/N: G  191.84  BARTON C.

<table>
<thead>
<tr>
<th>RATING</th>
<th>CATEGORY</th>
<th>COPIES</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
<th>DIRECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>NR</td>
<td>CHILDREN</td>
<td>1</td>
<td>29.95</td>
<td>15.99</td>
<td>DISNEY W.</td>
</tr>
<tr>
<td></td>
<td>CLASSIC</td>
<td>1</td>
<td>24.98</td>
<td>14.99</td>
<td>KAZAN E.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>39.99</td>
<td>20.00</td>
<td>WELLES O.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>29.95</td>
<td>15.99</td>
<td>GORDON M.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>19.99</td>
<td>10.95</td>
<td>MANN D.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>19.99</td>
<td>10.95</td>
<td>HUSTON J.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>19.95</td>
<td>9.99</td>
<td>KAZAN E.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>89.99</td>
<td>40.99</td>
<td>MILESTONE L.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>19.99</td>
<td>10.95</td>
<td>CUKOR G.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>19.98</td>
<td>10.99</td>
<td>BROOKS R.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>29.95</td>
<td>15.00</td>
<td>CURTIZ M.</td>
</tr>
</tbody>
</table>

*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:

<table>
<thead>
<tr>
<th>RATING</th>
<th>CATEGORY</th>
<th>COPIES</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
<th>DIRECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>CHILDREN</td>
<td>2</td>
<td>44.95</td>
<td>29.99</td>
<td>BARTON C.</td>
</tr>
<tr>
<td>G</td>
<td>CHILDREN</td>
<td>2</td>
<td>29.95</td>
<td>12.50</td>
<td>GEROMINI</td>
</tr>
<tr>
<td>G</td>
<td>CHILDREN</td>
<td>3</td>
<td>26.99</td>
<td>12.00</td>
<td>DISNEY W.</td>
</tr>
<tr>
<td>CLASSIC</td>
<td></td>
<td>3</td>
<td>89.95</td>
<td>40.99</td>
<td>FLEMING V</td>
</tr>
<tr>
<td>*All:</td>
<td></td>
<td>10</td>
<td>191.84</td>
<td>95.48</td>
<td>FLEMING V</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RATING</th>
<th>CATEGORY</th>
<th>COPIES</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
<th>DIRECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>NR</td>
<td>CHILDREN</td>
<td>1</td>
<td>29.95</td>
<td>15.99</td>
<td>DISNEY W.</td>
</tr>
<tr>
<td>NR</td>
<td>CLASSIC</td>
<td>3</td>
<td>39.99</td>
<td>20.00</td>
<td>WELLES O.</td>
</tr>
<tr>
<td>NR</td>
<td>CHILDREN</td>
<td>1</td>
<td>24.98</td>
<td>14.99</td>
<td>KAZAN E.</td>
</tr>
<tr>
<td>NR</td>
<td>CLASSIC</td>
<td>1</td>
<td>19.99</td>
<td>10.95</td>
<td>MANN D.</td>
</tr>
<tr>
<td>NR</td>
<td>CLASSIC</td>
<td>2</td>
<td>19.99</td>
<td>10.95</td>
<td>HUSTON J.</td>
</tr>
<tr>
<td>NR</td>
<td>CLASSIC</td>
<td>2</td>
<td>19.95</td>
<td>9.99</td>
<td>KAZAN E.</td>
</tr>
<tr>
<td>NR</td>
<td>CLASSIC</td>
<td>2</td>
<td>89.99</td>
<td>40.99</td>
<td>MILESTONE L.</td>
</tr>
<tr>
<td>NR</td>
<td>CLASSIC</td>
<td>2</td>
<td>19.99</td>
<td>10.95</td>
<td>CUKOR G.</td>
</tr>
<tr>
<td>NR</td>
<td>CLASSIC</td>
<td>2</td>
<td>19.98</td>
<td>10.99</td>
<td>BROOKS R.</td>
</tr>
<tr>
<td>NR</td>
<td>CLASSIC</td>
<td>2</td>
<td>29.95</td>
<td>15.00</td>
<td>CURTIZ M.</td>
</tr>
<tr>
<td>*All:</td>
<td>NR</td>
<td>19</td>
<td>344.71</td>
<td>176.79</td>
<td>CURTIZ M.</td>
</tr>
</tbody>
</table>

TOTAL  29  536.55  272.27  CURTIZ M.
Controlling Summary Line Processing

How to:
Control Summary Line Processing

Reference:
Usage Notes for SET SUMMARYLINES

When processing summary lines, you can control whether prefix operator processing is used and whether SUBTOTAL and RECOMPUTE commands are propagated to the grand total row of a report.

Summary line processing with prefix operators differs from processing without prefix operators in both the types of operations supported and the fields affected.

By default, you cannot mix these two styles of processing in a request, and the syntax used in the request (prefix operators or no prefix operators on summary lines) determines which type of processing is used.

One function of the SUMMARYLINES setting is to allow you to combine fields with and without prefix operators on summary lines in one request. In this case, prefix operator processing is used for all summary lines. Those fields without prefix operators are processed as though they were specified with the operator SUM. The new processing is required to display alphanumeric fields on summary lines.

Processing of reports that use prefix operators on summary lines differs from processing without prefix operators. In some cases, a different style of report output results from each type of request.

If one summary command specifies one field name and another summary command specifies a second field name:

- In a report without summary prefix operators, both columns are populated on both summary lines.
- In a report with summary prefix operators, only the specified column is populated on each summary line.

If a prefix operator is used in any summary command, prefix operator processing is required for the request. In most requests it is clear which type of processing to use, even if prefix operators are specified in some summary commands but not in others.
However, if the first time a summary prefix operator is encountered occurs after a field name has been specified in a summary command without an accompanying prefix operator, neither type of processing can be implemented. In this case, by default, processing stops and the following error message is generated:

(FOC36376) CANNOT COMBINE SUBTOTAL/RECOMPUTE STYLES WHEN SUMMARYLINES=OLD

For example:

ON RATING SUBTOTAL COPIES AVE. LISTPR

or

ON RATING SUBTOTAL LISTPR
ON CATEGORY SUBTOTAL AVE. COPIES

You can eliminate this problem by issuing the SET SUMMARYLINES=NEW command to specify that prefix operator processing should be used. The SUM. operator is then applied to any field that does not have a prefix operator.

The new processing is required to display alphanumeric fields on summary lines.

The second 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 = {OLD|NEW|EXPLICIT}

where:

OLD

Does not allow summary fields with and without prefix operators and propagates all summary operations to the grand total line. Fields specified on summary lines must all have prefix operators applied or must all exclude them. This syntax determines which type of processing is applied. OLD is the default value.

Note that a prefix operator preceding a list of field names is applied to each of those report columns (and, therefore, is not considered mixing). You can specify the SUM operator for fields for which you want a standard subtotal. This produces the same value that would have been generated without prefix operators. Alphanumeric fields are not included on summary lines.

NEW

Propagates all summary operations to the grand total line. Uses prefix operator processing for all summary commands (all summary fields without prefix operators are processed as though they had a SUM operator). 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. Supports display of alphanumeric fields on summary lines.

EXPLICIT

Does not propagate SUBTOTAL and RECOMPUTE to the grand total line. Uses prefix operator processing for all summary commands (all summary fields without prefix operators are processed as though they had a SUM operator). 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. Supports display of alphanumeric fields on summary lines.

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.
Even if prefix operators are not used on summary lines, report output generated by the two settings for the same request may be slightly different. With SUMMARYLINES NEW, a summary command with a list of field names populates only those columns on the associated summary line, while SUMMARYLINES OLD populates every column specified in any summary command.

For example:

```
TABLE FILE MOVIES
PRINT COPIES LISTPR WHOLESALEPR
  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 when SUMMARYLINES=OLD has subtotals for both COPIES and LISTPR on both sort breaks. WHOLESALEPR is not referenced in either SUBTOTAL command and, therefore, is not on any summary line:

<table>
<thead>
<tr>
<th>RATING</th>
<th>CATEGORY</th>
<th>COPIES</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>CHILDREN</td>
<td>2</td>
<td>44.95</td>
<td>29.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>29.95</td>
<td>12.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>26.99</td>
<td>12.00</td>
</tr>
<tr>
<td>*COPY</td>
<td>CHILDREN</td>
<td>7</td>
<td></td>
<td>101.89</td>
</tr>
<tr>
<td>*LIST</td>
<td>G</td>
<td>7</td>
<td></td>
<td>101.89</td>
</tr>
</tbody>
</table>

TOTAL 7 101.89

The output when SUMMARYLINES=NEW 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. WHOLESALEPR is not referenced in either SUBTOTAL command and, therefore, is not on any summary line:

<table>
<thead>
<tr>
<th>RATING</th>
<th>CATEGORY</th>
<th>COPIES</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>CHILDREN</td>
<td>2</td>
<td>44.95</td>
<td>29.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>29.95</td>
<td>12.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>26.99</td>
<td>12.00</td>
</tr>
<tr>
<td>*COPY</td>
<td>CHILDREN</td>
<td>7</td>
<td></td>
<td>101.89</td>
</tr>
<tr>
<td>*LIST</td>
<td>G</td>
<td></td>
<td></td>
<td>101.89</td>
</tr>
</tbody>
</table>

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:

```
SET SUMMARYLINES=OLD
TABLE FILE MOVIES
SUM COPIES LISTPR WHOLESALEPR
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=OLD subtotals both COPIES and LISTPR at every sort break and propagates them 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=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:

<table>
<thead>
<tr>
<th>RATING</th>
<th>CATEGORY</th>
<th>COPIES</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>CHILDREN</td>
<td>7</td>
<td>101.89</td>
<td>54.49</td>
</tr>
</tbody>
</table>

*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 WHOLESALEPR
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:

<table>
<thead>
<tr>
<th>RATING</th>
<th>CATEGORY</th>
<th>COPIES</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>CHILDREN</td>
<td>7</td>
<td>101.89</td>
<td>54.49</td>
</tr>
</tbody>
</table>

*TOTAL CHILDREN 101.89
*TOTAL G 7

TOTAL                        101.89 54.49
The following request has an ON TABLE SUBTOTAL WHOLESALEPR command. It also has an ON TABLE COLUMN-TOTAL phrase:

```
SET SUMMARYLINES=EXPLICIT
TABLE FILE MOVIES
SUM COPIES LISTPR WHOLESALEPR
BY RATING
BY CATEGORY
WHERE CATEGORY EQ 'CHILDREN'
WHERE RATING   EQ 'G'
ON RATING SUBTOTAL COPIES
ON CATEGORY SUBTOTAL LISTPR
ON TABLE SUBTOTAL WHOLESALEPR
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
```
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.

With prefix operator processing, 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_Inter`, 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_Inter = 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.

<table>
<thead>
<tr>
<th>Region</th>
<th>Category</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
<th>Budget Dollars</th>
<th>DIFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midwest</td>
<td>Coffee</td>
<td>332777</td>
<td>4178513</td>
<td>4086032</td>
<td>92481</td>
</tr>
<tr>
<td></td>
<td>Food</td>
<td>341414</td>
<td>4338271</td>
<td>4220721</td>
<td>117550</td>
</tr>
<tr>
<td>*TOTAL Midwest</td>
<td></td>
<td>674191</td>
<td>8516784</td>
<td>8306753</td>
<td>210031</td>
</tr>
<tr>
<td>West</td>
<td>Coffee</td>
<td>356763</td>
<td>4473517</td>
<td>4523963</td>
<td>-50446</td>
</tr>
<tr>
<td></td>
<td>Food</td>
<td>340234</td>
<td>4202337</td>
<td>4183244</td>
<td>19093</td>
</tr>
<tr>
<td>*TOTAL West</td>
<td></td>
<td>696997</td>
<td>8675854</td>
<td>8707207</td>
<td>-31353</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Region</th>
<th>Category</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
<th>Budget Dollars</th>
<th>DIFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midwest</td>
<td>Coffee</td>
<td>332777</td>
<td>4178513</td>
<td>4086032</td>
<td>92481</td>
</tr>
<tr>
<td></td>
<td>Food</td>
<td>341414</td>
<td>4338271</td>
<td>4220721</td>
<td>117550</td>
</tr>
<tr>
<td>*TOTAL Midwest</td>
<td></td>
<td>674191</td>
<td>8516784</td>
<td>8306753</td>
<td>210031</td>
</tr>
<tr>
<td>West</td>
<td>Coffee</td>
<td>356763</td>
<td>4473517</td>
<td>4523963</td>
<td>-50446</td>
</tr>
<tr>
<td></td>
<td>Food</td>
<td>340234</td>
<td>4202337</td>
<td>4183244</td>
<td>19093</td>
</tr>
<tr>
<td>*TOTAL West</td>
<td></td>
<td>696997</td>
<td>8675854</td>
<td>8707207</td>
<td>-31353</td>
</tr>
</tbody>
</table>

TOTAL                      1371188     17192638       17013960     178678
**Example:** Using RECOMPUTE at the Sort Break and Grand Total Levels

The following example adds the ON TABLE RECOMPUTE command to the request in *Using Prefix Operators With RECOMPUTE* on page 295 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:

<table>
<thead>
<tr>
<th>Region</th>
<th>Category</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
<th>Budget Dollars</th>
<th>DIFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midwest</td>
<td>Coffee</td>
<td>332777</td>
<td>4178513</td>
<td>4086032</td>
<td>92481</td>
</tr>
<tr>
<td></td>
<td>Food</td>
<td>341414</td>
<td>4338271</td>
<td>4220721</td>
<td>117550</td>
</tr>
<tr>
<td>*TOTAL Midwest</td>
<td></td>
<td></td>
<td>4338271</td>
<td>4086032</td>
<td>252239</td>
</tr>
<tr>
<td>West</td>
<td>Coffee</td>
<td>356763</td>
<td>4473527</td>
<td>4523963</td>
<td>-50436</td>
</tr>
<tr>
<td></td>
<td>Food</td>
<td>340234</td>
<td>4202338</td>
<td>4183244</td>
<td>19094</td>
</tr>
<tr>
<td>*TOTAL West</td>
<td></td>
<td></td>
<td>4473527</td>
<td>4183244</td>
<td>290283</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>342797</td>
<td>4298162</td>
<td>4253490</td>
<td>44672</td>
</tr>
</tbody>
</table>

**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.
Combinations of Summary Commands

Reference:
Usage Notes for 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 or if you issue the SET SUMMARYLINES=OLD command, prefix operator processing is not used for the request. In this case, 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 or if you issue the SET SUMMARYLINES=NEW command, prefix operator processing is used for the request. In this case, 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, 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:

<table>
<thead>
<tr>
<th>COPIES</th>
<th>RATING</th>
<th>DIRECTOR</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
<th>RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NR</td>
<td>DISNEY W.</td>
<td>29.95</td>
<td>15.99</td>
<td>1.87</td>
</tr>
<tr>
<td></td>
<td>*SUB:</td>
<td>NR</td>
<td>29.95</td>
<td>15.99</td>
<td>1.87</td>
</tr>
<tr>
<td></td>
<td>*REC:</td>
<td>1</td>
<td>29.95</td>
<td>15.99</td>
<td>1.87</td>
</tr>
<tr>
<td>2</td>
<td>NR</td>
<td>HITCHCOCK A.</td>
<td>19.98</td>
<td>9.00</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td>*SUB:</td>
<td>NR</td>
<td>19.98</td>
<td>9.00</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PG</td>
<td>19.98</td>
<td>9.00</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HITCHCOCK A.</td>
<td>19.98</td>
<td>9.00</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td>*SUB:</td>
<td>PG</td>
<td>39.96</td>
<td>18.00</td>
<td>4.44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PG13</td>
<td>19.98</td>
<td>9.00</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td>*SUB:</td>
<td>PG13</td>
<td>19.98</td>
<td>9.00</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td>*REC:</td>
<td>R</td>
<td>19.98</td>
<td>9.00</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R</td>
<td>19.98</td>
<td>9.00</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>*REC:</td>
<td>99.90</td>
<td>45.00</td>
<td>2.22</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td>129.85</td>
<td>60.99</td>
<td>2.13</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>TOTAL</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>129.85</td>
<td>60.99</td>
<td>12.97</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>DIRECTOR</th>
<th>RATING</th>
<th>COPIES</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
<th>RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>HITCHCOCK A.</td>
<td>NR</td>
<td>2</td>
<td>19.98</td>
<td>9.00</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td>*SUB: 2</td>
<td></td>
<td>19.98</td>
<td>9.00</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td>*REC: NR</td>
<td></td>
<td>19.98</td>
<td>9.00</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td>PG</td>
<td>2</td>
<td>19.98</td>
<td>9.00</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td>*SUB: 2</td>
<td></td>
<td>39.96</td>
<td>18.00</td>
<td>4.44</td>
</tr>
<tr>
<td></td>
<td>*REC: PG</td>
<td></td>
<td>39.96</td>
<td>18.00</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td>PG13</td>
<td>2</td>
<td>19.98</td>
<td>9.00</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td>*SUB: 2</td>
<td></td>
<td>19.98</td>
<td>9.00</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>*REC: PG13</td>
<td></td>
<td>19.98</td>
<td>9.00</td>
<td>2.2</td>
</tr>
<tr>
<td>HITCHCOCK A.</td>
<td>R</td>
<td>2</td>
<td>19.98</td>
<td>9.00</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>*SUB: 2</td>
<td></td>
<td>19.98</td>
<td>9.00</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>*REC: R</td>
<td></td>
<td>19.98</td>
<td>9.00</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>*TOTAL DIRECTOR HITCHCOCK A.</td>
<td>99.90</td>
<td>45.00</td>
<td>11.1</td>
<td></td>
</tr>
</tbody>
</table>

| 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:

<table>
<thead>
<tr>
<th>DIRECTOR</th>
<th>RATING</th>
<th>COPIES</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
<th>RATIO</th>
<th>RATIO1</th>
</tr>
</thead>
<tbody>
<tr>
<td>KAZAN E.</td>
<td>NR</td>
<td>1</td>
<td>24.98</td>
<td>14.99</td>
<td>1.67</td>
<td>1.67</td>
</tr>
<tr>
<td>*SUB: 1</td>
<td></td>
<td></td>
<td>24.98</td>
<td>14.99</td>
<td>1.67</td>
<td>1.67</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>19.95</td>
<td>9.99</td>
<td>2.00</td>
<td>2.00</td>
</tr>
<tr>
<td>*SUB: 2</td>
<td></td>
<td></td>
<td>19.95</td>
<td>9.99</td>
<td>2.00</td>
<td>2.00</td>
</tr>
<tr>
<td>*REC: NR</td>
<td></td>
<td></td>
<td>22.46</td>
<td>24.98</td>
<td>.90</td>
<td>3.66</td>
</tr>
<tr>
<td>*TOTAL DIRECTOR KAZAN E.</td>
<td></td>
<td></td>
<td>44.93</td>
<td>24.98</td>
<td>3.66</td>
<td>3.66</td>
</tr>
</tbody>
</table>

**Example:** Propagation of Summary Commands With Field Lists

In the following request, the RECOMPUTE command has a field list.

```
SET SUMMARYLINES = OLD
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
```
With SUMMARYLINES=OLD, only those fields have values on the report output:

<table>
<thead>
<tr>
<th>DIRECTOR</th>
<th>RATING</th>
<th>COPIES</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
<th>RATIO</th>
<th>RATIO1</th>
</tr>
</thead>
<tbody>
<tr>
<td>KAZAN E.</td>
<td></td>
<td>1</td>
<td>24.98</td>
<td>14.99</td>
<td>1.67</td>
<td>1.67</td>
</tr>
</tbody>
</table>

*SUB: 1

*SUB: 2

*REC: NR

*TOTAL DIRECTOR KAZAN E.

TOTAL

With SUMMARYLINES=NEW, 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:

<table>
<thead>
<tr>
<th>DIRECTOR</th>
<th>RATING</th>
<th>COPIES</th>
<th>LISTPR</th>
<th>WHOLESALEPR</th>
<th>RATIO</th>
<th>RATIO1</th>
</tr>
</thead>
<tbody>
<tr>
<td>KAZAN E.</td>
<td></td>
<td>1</td>
<td>24.98</td>
<td>14.99</td>
<td>1.67</td>
<td>1.67</td>
</tr>
</tbody>
</table>

*SUB: 1

*SUB: 2

*REC: NR

*TOTAL DIRECTOR KAZAN E.

TOTAL

Reference: Usage Notes for Combinations of Summary Commands

- Summary processing differs for summary commands that have prefix operators and summary lines that do not use prefix operators. If any summary command uses prefix operators, the entire request uses prefix operator processing. If processing without prefix operators is initiated, but a subsequent field requires prefix operator processing, the following message is generated and processing halts:

  (FOC36376) CANNOT COMBINE SUBTOTAL/RECOMPUTE STYLES WHEN SUMMARYLINES=OLD

  You can prevent this message by setting SUMMARYLINES=NEW to invoke prefix operator processing.

- 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

How to:
Produce a Summary Operation on a Horizontal Sort Field

Reference:
Usage Notes for Summaries on ACROSS Fields

The summary commands SUBTOTAL, SUB-TOTAL, SUMMARIZE, and RECOMPUTE can be used with horizontal sort breaks.

Syntax: How to Produce a Summary Operation on a Horizontal Sort Field

\{ACROSS|ON\} breakfield [AS 'text1'] sumoption [AS 'text2']
[COLUMNS c1 [AND c2 ...]]

where:

**breakfield**

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.

**COLUMNS c1, 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.

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.
- 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.

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
```
Performing Calculations at Sort Field Breaks

How to:
Use Subtotals in Calculations

Reference:
Usage Notes for RECAP and COMPUTE

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 315. 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 314). 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.

- 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 1053.)

- 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 1053.)

- In an ON phrase, a COMPUTE command is the same as a RECAP command.
The word RECAP may not be specified more than seven times. However, more than seven RECAP calculations are permitted. Use the following syntax:

```
ON fieldname 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:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>PAY_DATE</th>
<th>DED_AMT</th>
<th>GROSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>81/11/30</td>
<td>$1,406.79</td>
<td>$2,147.75</td>
</tr>
<tr>
<td></td>
<td>81/12/31</td>
<td>$1,406.79</td>
<td>$2,147.75</td>
</tr>
<tr>
<td></td>
<td>82/01/29</td>
<td>$1,740.89</td>
<td>$3,247.75</td>
</tr>
<tr>
<td></td>
<td>82/02/26</td>
<td>$1,740.89</td>
<td>$3,247.75</td>
</tr>
<tr>
<td></td>
<td>82/03/31</td>
<td>$1,740.89</td>
<td>$3,247.75</td>
</tr>
<tr>
<td></td>
<td>82/04/30</td>
<td>$3,386.73</td>
<td>$5,890.84</td>
</tr>
<tr>
<td></td>
<td>82/05/28</td>
<td>$3,954.35</td>
<td>$6,649.50</td>
</tr>
<tr>
<td></td>
<td>82/06/30</td>
<td>$4,117.03</td>
<td>$7,460.00</td>
</tr>
<tr>
<td></td>
<td>82/07/30</td>
<td>$4,117.03</td>
<td>$7,460.00</td>
</tr>
<tr>
<td></td>
<td>82/08/31</td>
<td>$4,575.72</td>
<td>$9,000.00</td>
</tr>
</tbody>
</table>

** DEPT_NET $22,311.98

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>PAY_DATE</th>
<th>DED_AMT</th>
<th>GROSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTION</td>
<td>81/11/30</td>
<td>$141.66</td>
<td>$833.33</td>
</tr>
<tr>
<td></td>
<td>81/12/31</td>
<td>$141.66</td>
<td>$833.33</td>
</tr>
<tr>
<td></td>
<td>82/01/29</td>
<td>$1,560.09</td>
<td>$3,705.84</td>
</tr>
<tr>
<td></td>
<td>82/02/26</td>
<td>$2,061.69</td>
<td>$4,959.84</td>
</tr>
<tr>
<td></td>
<td>82/03/31</td>
<td>$2,061.69</td>
<td>$4,959.84</td>
</tr>
<tr>
<td></td>
<td>82/04/30</td>
<td>$2,061.69</td>
<td>$4,959.84</td>
</tr>
<tr>
<td></td>
<td>82/05/28</td>
<td>$3,483.88</td>
<td>$7,048.84</td>
</tr>
<tr>
<td></td>
<td>82/06/30</td>
<td>$3,483.88</td>
<td>$7,048.84</td>
</tr>
<tr>
<td></td>
<td>82/07/30</td>
<td>$3,483.88</td>
<td>$7,048.84</td>
</tr>
<tr>
<td></td>
<td>82/08/31</td>
<td>$4,911.12</td>
<td>$9,523.84</td>
</tr>
</tbody>
</table>

** 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 273.

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
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. For complete details on using the WHEN phrase, see *Conditionally Including Summary Lines, Underlines, Skipped Lines, and Page Breaks* on page 841.
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:

<table>
<thead>
<tr>
<th>Region</th>
<th>Category</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midwest</td>
<td>Coffee</td>
<td>332777</td>
<td>4178513</td>
</tr>
<tr>
<td></td>
<td>Food</td>
<td>341414</td>
<td>4338271</td>
</tr>
<tr>
<td></td>
<td>Gifts</td>
<td>230854</td>
<td>2883881</td>
</tr>
</tbody>
</table>

The total for the Midwest region is less than 11500000.

<table>
<thead>
<tr>
<th>Region</th>
<th>Category</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Coffee</td>
<td>335778</td>
<td>4164017</td>
</tr>
<tr>
<td></td>
<td>Food</td>
<td>353368</td>
<td>4379994</td>
</tr>
<tr>
<td></td>
<td>Gifts</td>
<td>227529</td>
<td>2848289</td>
</tr>
</tbody>
</table>

The total for the Northeast region is less than 11500000.

<table>
<thead>
<tr>
<th>Region</th>
<th>Category</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Southeast</td>
<td>Coffee</td>
<td>350948</td>
<td>4415408</td>
</tr>
<tr>
<td></td>
<td>Food</td>
<td>349829</td>
<td>4308731</td>
</tr>
<tr>
<td></td>
<td>Gifts</td>
<td>234455</td>
<td>2986240</td>
</tr>
</tbody>
</table>

*TOTAL Southeast 935232 11710379

<table>
<thead>
<tr>
<th>Region</th>
<th>Category</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>West</td>
<td>Coffee</td>
<td>356763</td>
<td>4473517</td>
</tr>
<tr>
<td></td>
<td>Food</td>
<td>340234</td>
<td>4202337</td>
</tr>
<tr>
<td></td>
<td>Gifts</td>
<td>235042</td>
<td>2977092</td>
</tr>
</tbody>
</table>

*TOTAL West 932039 11652946

TOTAL 3688991 46156290
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.

**Topics:**

- 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 amper 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

In this section:
Expressions and Field Formats

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 319.

- **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 325.

  - 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$9$9$9$9$9') ;`

  A character expression returns an alphanumeric value. For details, see *Creating a Character Expression* on page 338.

**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.
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
```
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:
  ```plaintext
  COMPUTE COUNT/I2 = 1;
  ```

- A numeric constant in scientific notation. For example:
  ```plaintext
  COMPUTE COST/D12.2 = EXPN(8E+3);
  
  For syntax usage, see How to Express a Number in Scientific Notation on page 320.
  ```

- A numeric field. For example:
  ```plaintext
  COMPUTE RECOUNT/I2 = COUNT;
  ```

- Two numeric constants or fields joined by an arithmetic operator. For example:
  ```plaintext
  COMPUTE BONUS/D12.2 = CURR_SAL * 0.05;
  
  For a list of arithmetic operators, see Arithmetic Operators on page 321.
  ```

- A numeric function. For example:
  ```plaintext
  COMPUTE LONGEST_SIDE/D12.2 = MAX(WIDTH, HEIGHT);
  ```
Creating a Numeric Expression

- 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 322.

  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 323.

  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:

\[
\text{COMPUTE PROFIT/D12.2} = \text{RETAIL_PRICE} - \text{UNIT_COST} \times \text{UNIT_SOLD} ;
\]
\[
\text{COMPUTE PROFIT/D12.2} = (\text{RETAIL_PRICE} - \text{UNIT_COST}) \times \text{UNIT_SOLD} ;
\]

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:

\[
a \times -1
\]

To make it valid, you must add parentheses:

\[
a \times (-1)
\]

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

\[
\text{COMPUTE RETAIL_LOSS/D12.2} = \text{RETAIL_PRICE} \times \text{RETURNS} + \text{DAMAGED};
\]

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:

<table>
<thead>
<tr>
<th>PROD_CODE</th>
<th>RETAIL_PRICE</th>
<th>RETURNS</th>
<th>DAMAGED</th>
<th>RETAIL_LOSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>$.89</td>
<td>4</td>
<td>7</td>
<td>9.79</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operational Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>+ Native</td>
</tr>
<tr>
<td>Subtraction</td>
<td>- Native</td>
</tr>
<tr>
<td>Multiplication</td>
<td>* Native</td>
</tr>
<tr>
<td>Full Division</td>
<td>/ Accepts single or double-precision floating point, converts all others to double-precision floating point</td>
</tr>
<tr>
<td>Exponentiation</td>
<td>** Double-precision floating point</td>
</tr>
</tbody>
</table>
**Example:** Using Identical Operand Formats (Native-mode Arithmetic)

The following variables are defined as integers in WebFOCUS Maintain:

```plaintext
COMPUTE OPERANDONE/I4  
OPERANDTWO/I4  
RESULT/I4  
```

The required multiplication is done in native-mode arithmetic (integer arithmetic):

```plaintext
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.

<table>
<thead>
<tr>
<th>Order</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16-byte packed decimal</td>
</tr>
<tr>
<td>2</td>
<td>Double-precision floating point</td>
</tr>
<tr>
<td>3</td>
<td>8-byte packed-decimal</td>
</tr>
<tr>
<td>4</td>
<td>Single-precision floating point</td>
</tr>
<tr>
<td>5</td>
<td>Integer</td>
</tr>
<tr>
<td>6</td>
<td>Character (alphanumeric and text)</td>
</tr>
</tbody>
</table>

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

In this section:
- Formats for Date Values
- Performing Calculations on Dates
- Cross-Century Dates With DEFINE and COMPUTE
- Returned Field Format Selection
- Using a Date Constant in an Expression
- Extracting a Date Component
- Combining Fields With Different Formats in an 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:

\[
\text{COMPUTE DIFF/I4 = YMD (HIRE\_DATE,FST\_DAT\_INC);}\]

### Formats for Date Values

**Reference:**

- Base Dates for Date Formats
- Impact of Date Formats on Storage and Display

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 VM 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:

\[
\text{COMPUTE ALPHADATE/A6MDY = '120599' ;}
\]

\[
\text{REALDATE/MDY = ALPHADATE;}\]
**Reference: Base Dates for Date Formats**

The following table shows the base date for each supported date format:

<table>
<thead>
<tr>
<th>Format</th>
<th>Base Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>YMD, YYMD, MDYY, DMYY, MDY, and DMY</td>
<td>1900/12/31</td>
</tr>
</tbody>
</table>
| YM, YYM, MYY, and MY | 1901/01 on z/OS and VM  
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Stored</th>
<th>Displayed</th>
<th>Stored</th>
<th>Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 28, 1999</td>
<td>35853</td>
<td>02/28/1999</td>
<td>02281999</td>
<td>02/28/1999</td>
</tr>
<tr>
<td>March 1, 1999</td>
<td>35854</td>
<td>03/01/1999</td>
<td>03011999</td>
<td>03/01/1999</td>
</tr>
</tbody>
</table>
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:

\[
\text{COMPUTE TURNAROUND/I4 = SHIP\_DATE - ORDER\_DATE;}\]

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:

<table>
<thead>
<tr>
<th>Value in Date Format</th>
<th>Value in Integer Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHIP_DATE = March 1, 1999</td>
<td>35854</td>
</tr>
<tr>
<td>ORDER_DATE = February 28, 1999</td>
<td>35853</td>
</tr>
<tr>
<td>TURNAROUND</td>
<td>1</td>
</tr>
</tbody>
</table>

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:

\[
\text{COMPUTE TURNAROUND/I4 = MDY(ORDER\_DATE, SHIP\_DATE);}\]
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.

For details on the sliding window technique, and the use of expressions in this technique, see *Working with Cross-Century Dates* in the Supplementary Documentation section of the WebFOCUS Documentation CD.

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.

```plaintext
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.
**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:

```plaintext
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:

```plaintext
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.

**Syntax:** How to Specify the Order of Date Components in a Date-Time Field

```plaintext
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.

For an example, see *Specifying the Order of Date Components for a Date-Time Field* on page 334.
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:

\[
\text{time_string} \ [\text{date_string}]
\]
\[
\text{date_string} \ [\text{time_string}]
\]

A date-time constant in a COMPUTE, DEFINE, or WHERE expression must have one of the following formats:

\[
\text{DT(} \text{time_string} \ [\text{date_string}]\text{)}
\]
\[
\text{DT(} \text{date_string} \ [\text{time_string}]\text{)}
\]

A date-time constant in an IF expression has one of the following formats:

\[
'\text{time_string} \ [\text{date_string}]'
\]
\[
'\text{date_string} \ [\text{time_string}]'
\]

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:

- \text{time_string}

  Cannot contain blanks. Time components are separated by colons, and may be followed by AM, PM, am, or pm. For example:

\[
14:30:20:99 \quad \text{(99 milliseconds)}
\]
\[
14:30 \quad \text{02:30:20:500pm}
\]

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 331 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 331. 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 in a COMPUTE, DEFINE, or WHERE expression to assign a date-time literal to a date-time field. For example:

\[
\begin{align*}
\text{DT2/HYYMDS} &= \text{DT(20051226 05:45);} \\
\text{DT3/HYYMDS} &= \text{DT(2005 DEC 26 05:45);} \\
\text{DT4/HYYMDS} &= \text{DT(December 26 2005 05:45);} \\
\end{align*}
\]

**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
DTFLDYYMD/HYYMDI = DT(APR 04 05);
END
```

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL DTFLDYYMD
END
```

The output shows that the natural date literal 'APR 04 05' is interpreted as April 5, 1904:

<table>
<thead>
<tr>
<th>CURR_SAL</th>
<th>DTFLDYYMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$11,000.00</td>
<td>1904/04/05 00:00</td>
</tr>
<tr>
<td>$13,200.00</td>
<td>1904/04/05 00:00</td>
</tr>
<tr>
<td>$18,480.00</td>
<td>1904/04/05 00:00</td>
</tr>
<tr>
<td>$9,500.00</td>
<td>1904/04/05 00:00</td>
</tr>
<tr>
<td>$29,700.00</td>
<td>1904/04/05 00:00</td>
</tr>
<tr>
<td>$26,862.00</td>
<td>1904/04/05 00:00</td>
</tr>
<tr>
<td>$21,120.00</td>
<td>1904/04/05 00:00</td>
</tr>
<tr>
<td>$18,480.00</td>
<td>1904/04/05 00:00</td>
</tr>
<tr>
<td>$21,780.00</td>
<td>1904/04/05 00:00</td>
</tr>
<tr>
<td>$16,100.00</td>
<td>1904/04/05 00:00</td>
</tr>
<tr>
<td>$9,000.00</td>
<td>1904/04/05 00:00</td>
</tr>
<tr>
<td>$27,062.00</td>
<td>1904/04/05 00:00</td>
</tr>
</tbody>
</table>

**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=HYMDS ,$

**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:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>NEWSAL</th>
<th>RAISETIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td>$14,520.00</td>
<td>2000/01/01 9:00AM</td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td>$20,328.00</td>
<td>2000/01/01 9:00AM</td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>$23,232.00</td>
<td>2000/01/01 9:00AM</td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td>$20,328.00</td>
<td>2000/01/01 9:00AM</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>$23,958.00</td>
<td>2000/01/01 9:00AM</td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>$17,710.00</td>
<td>2000/01/01 9:00AM</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>CUSTID</th>
<th>TRANSDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1118</td>
<td>2000/06/26 05:45</td>
</tr>
<tr>
<td>1237</td>
<td>2000/02/05 03:30</td>
</tr>
</tbody>
</table>

7. Using Expressions

Creating Reports With WebFOCUS Language
Creating a Date-Time Expression

**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:

```sql
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:

<table>
<thead>
<tr>
<th>Input Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:30[:20,99]</td>
<td>Comma separates time components instead of period</td>
</tr>
<tr>
<td>14:30[:20.99]Z</td>
<td>Universal time</td>
</tr>
<tr>
<td>15:30[:20,99]+01</td>
<td>Each of these is the same as above in Central European Time</td>
</tr>
<tr>
<td>15:30[:20,99]+0100</td>
<td></td>
</tr>
<tr>
<td>15:30[:20,99]+01:00</td>
<td></td>
</tr>
<tr>
<td>09:30[:20.99]−05</td>
<td>Same as above in Eastern Standard Time</td>
</tr>
</tbody>
</table>

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**

The only direct operations that can be performed on date-time variables and constants are comparison using a logical expression, and simple assignment of the form A = B.

Computations only allow direct assignment within data types: alpha to alpha, numeric to numeric, date to date, and date-time to date-time. All other operations are accomplished through a set of date-time functions.
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 arithmetic, 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:

```
<table>
<thead>
<tr>
<th>ID</th>
<th>DT1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2000/01/01 02:57:25</td>
</tr>
<tr>
<td>2</td>
<td>1999/12/31 00:00:00</td>
</tr>
<tr>
<td>3</td>
<td>.</td>
</tr>
</tbody>
</table>
```

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:

<table>
<thead>
<tr>
<th>ID</th>
<th>DT1</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>2000/01/01 02:57:25</td>
</tr>
</tbody>
</table>

**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/HYYMDH = TRANSDATE;
   T1/HHIS = TRANSDATE;
WHERE DATE EQ 2000
END
```

The output is:

<table>
<thead>
<tr>
<th>CUSTID</th>
<th>TRANSDATE</th>
<th>DT2</th>
<th>T1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1118</td>
<td>2000/06/26 05:45</td>
<td>2000/06/26 05</td>
<td>05:45:00</td>
</tr>
<tr>
<td>1237</td>
<td>2000/02/05 03:30</td>
<td>2000/02/05 03</td>
<td>03:30:00</td>
</tr>
</tbody>
</table>

**Creating a Character Expression**

**In this section:**

Embedding a Quotation Mark in a Quote-Delimited Literal String
Concatenating Character Strings

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.

**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:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Represents</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`</td>
<td></td>
<td>`</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
<td>Weak concatenation</td>
</tr>
</tbody>
</table>

**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:
   \[ \text{Banning, J.} \]

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

In this section:
- Using Concatenation With AnV Fields
- Using the EDIT Function With AnV Fields
- Using CONTAINS and OMITS With AnV Fields
- Using LIKE With AnV Fields
- Using the EQ, NE, LT, GT, LE, and GE Operators With AnV Fields
- Using the DECODE Function With AnV Fields
- Using the Assignment Operator With AnV Fields

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.

Using the EDIT Function With AnV Fields

The following expression results in an AnV format only when x has AnV format.

\[ \text{EDIT}(x, \text{mask}) \]

The actual length of the result is the number of characters in \( \text{mask} \) other than '‐'.

Note that an actual length of zero may result.

\[ \text{EDIT}(x) \] 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:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z CONTAINS Y</td>
<td>FALSE</td>
</tr>
<tr>
<td>X CONTAINS Z</td>
<td>TRUE</td>
</tr>
<tr>
<td>Z CONTAINS Z</td>
<td>TRUE</td>
</tr>
<tr>
<td>Z OMITS Y</td>
<td>TRUE</td>
</tr>
<tr>
<td>X OMITS Z</td>
<td>FALSE</td>
</tr>
<tr>
<td>Z OMITS Z</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
Using LIKE With AnV Fields

The only difference in evaluation of the following expression occurs when \( x \) has an actual length of zero:

\[
x \text{ LIKE mask } ...
\]

In the following example, the field instance \( Z \) has an actual length of zero:

\[
Z \text{ LIKE mask } ...
\]

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:

\[
Z \text{ LIKE ' '}
\]

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:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Z \text{ EQ } Z )</td>
<td>TRUE</td>
</tr>
<tr>
<td>( Z \text{ GE } Z )</td>
<td></td>
</tr>
<tr>
<td>( Z \text{ LE } Z )</td>
<td></td>
</tr>
<tr>
<td>( Z \text{ NE } Z )</td>
<td>FALSE</td>
</tr>
<tr>
<td>( Z \text{ LT } Z )</td>
<td></td>
</tr>
<tr>
<td>( Z \text{ GT } Z )</td>
<td></td>
</tr>
<tr>
<td>( Z \text{ EQ } X )</td>
<td>FALSE</td>
</tr>
<tr>
<td>( Z \text{ NE } X )</td>
<td>TRUE</td>
</tr>
<tr>
<td>( Z \text{ LT } X )</td>
<td>TRUE</td>
</tr>
<tr>
<td>( Z \text{ GT } X )</td>
<td>FALSE</td>
</tr>
<tr>
<td>( Z \text{ LE } X )</td>
<td>TRUE</td>
</tr>
</tbody>
</table>
### Creating a Variable Length Character Expression

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z GE X</td>
<td>FALSE</td>
</tr>
<tr>
<td>X EQ Z</td>
<td>FALSE</td>
</tr>
<tr>
<td>X NE Z</td>
<td>TRUE</td>
</tr>
<tr>
<td>X LT Z</td>
<td>FALSE</td>
</tr>
<tr>
<td>X GT Z</td>
<td>TRUE</td>
</tr>
<tr>
<td>X LE Z</td>
<td>FALSE</td>
</tr>
<tr>
<td>X GE Z</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

#### 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.

\[ \text{fld/An} = \text{AnV\_type\_expression}; \]

- 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.

\[ \text{fld/AnV} = \text{AnV\_type\_expression}; \]

- 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 \((n)\).
Creating a Logical Expression

How to:
Write a Relational Expression
Write a Boolean Expression

Reference:
Logical Operators

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 143.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ</td>
<td>Returns the value TRUE if the value on the left is equal to the value on the right.</td>
</tr>
<tr>
<td>NE</td>
<td>Returns the value TRUE if the value on the left is not equal to the value on the right.</td>
</tr>
<tr>
<td>GE</td>
<td>Returns the value TRUE if the value on the left is greater than or equal to the value on the right.</td>
</tr>
<tr>
<td>GT</td>
<td>Returns the value TRUE if the value on the left is greater than the value on the right.</td>
</tr>
<tr>
<td>LE</td>
<td>Returns the value TRUE if the value on the left is less than or equal to the value on the right.</td>
</tr>
<tr>
<td>LT</td>
<td>Returns the value TRUE if the value on the left is less than the value on the right.</td>
</tr>
<tr>
<td>AND</td>
<td>Returns the value TRUE if both operands are true.</td>
</tr>
<tr>
<td>OR</td>
<td>Returns the value TRUE if either operand is true.</td>
</tr>
<tr>
<td>NOT</td>
<td>Returns the value TRUE if the operand is false.</td>
</tr>
<tr>
<td>CONTAINS</td>
<td>Contains the specified character strings.</td>
</tr>
<tr>
<td>OMITS</td>
<td>Omits the specified character strings.</td>
</tr>
<tr>
<td>IS MISSING</td>
<td>Returns the value TRUE if the field is missing.</td>
</tr>
<tr>
<td>IS-NOT MISSING</td>
<td>Returns the value TRUE if the field is not missing.</td>
</tr>
</tbody>
</table>
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} value character_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**

**How to:**

Write 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'.
Creating a Conditional Expression

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.

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

348 WebFOCUS
The output is:

<table>
<thead>
<tr>
<th>EMP_ID</th>
<th>BANK_ACCT</th>
<th>CURR_SAL</th>
<th>BANK_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>071382660</td>
<td></td>
<td>$11,000.00</td>
<td>NONE</td>
</tr>
<tr>
<td>112847612</td>
<td></td>
<td>$13,200.00</td>
<td>NONE</td>
</tr>
<tr>
<td>117593129</td>
<td>40950036</td>
<td>$18,480.00</td>
<td>STATE</td>
</tr>
<tr>
<td>119265415</td>
<td></td>
<td>$9,500.00</td>
<td>NONE</td>
</tr>
<tr>
<td>119329144</td>
<td>160633</td>
<td>$29,700.00</td>
<td>BEST BANK</td>
</tr>
<tr>
<td>123764317</td>
<td>819000702</td>
<td>$26,862.00</td>
<td>ASSOCIATED</td>
</tr>
<tr>
<td>126724188</td>
<td></td>
<td>$21,120.00</td>
<td>NONE</td>
</tr>
<tr>
<td>219984371</td>
<td></td>
<td>$18,480.00</td>
<td>NONE</td>
</tr>
<tr>
<td>326179357</td>
<td>122850108</td>
<td>$21,780.00</td>
<td>ASSOCIATED</td>
</tr>
<tr>
<td>451123478</td>
<td>136500120</td>
<td>$16,100.00</td>
<td>ASSOCIATED</td>
</tr>
<tr>
<td>543729165</td>
<td></td>
<td>$9,000.00</td>
<td>NONE</td>
</tr>
<tr>
<td>818692173</td>
<td>163800144</td>
<td>$27,062.00</td>
<td>BANK ASSOCIATION</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>EMP_ID</th>
<th>CURR_SAL</th>
<th>DEPARTMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>071382660</td>
<td>$11,000.00</td>
<td>PRODUCTION</td>
</tr>
<tr>
<td>112847612</td>
<td>$13,200.00</td>
<td>MIS</td>
</tr>
<tr>
<td>117593129</td>
<td>$18,480.00</td>
<td>MIS</td>
</tr>
<tr>
<td>119329144</td>
<td>$29,700.00</td>
<td>PRODUCTION</td>
</tr>
<tr>
<td>123764317</td>
<td>$26,862.00</td>
<td>PRODUCTION</td>
</tr>
<tr>
<td>126724188</td>
<td>$21,120.00</td>
<td>PRODUCTION</td>
</tr>
<tr>
<td>219984371</td>
<td>$18,480.00</td>
<td>MIS</td>
</tr>
<tr>
<td>326179357</td>
<td>$21,780.00</td>
<td>MIS</td>
</tr>
<tr>
<td>451123478</td>
<td>$16,100.00</td>
<td>PRODUCTION</td>
</tr>
<tr>
<td>543729165</td>
<td>$9,000.00</td>
<td>MIS</td>
</tr>
<tr>
<td>818692173</td>
<td>$27,062.00</td>
<td>MIS</td>
</tr>
</tbody>
</table>

**Note:** Testing for a TRUE or FALSE condition is valid only with the IF command. It is not valid with WHERE.
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 Web page, 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.

**Topics:**

- Saving Your Report Output
- Creating a HOLD File
- Holding Report Output in FOCUS Format
- Controlling Attributes in HOLD Master Files
- Keyed Retrieval From HOLD Files
- Saving and Retrieving HOLD Files
- Using DBMS Temporary Tables as HOLD Files
- Creating SAVE and SAVB Files
- Creating a PCHOLD File
- Choosing Output File Formats
- Using Text Fields in Output Files
- Creating a Delimited Sequential File
- Saving Report Output in INTERNAL Format
- Creating a Structured HOLD File
- Specifying MIME Types for WebFOCUS Reports
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 353.

- **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 381.

- **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 385.

  **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 or ALLOCATE 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 a FILEDEF or ALLOCATE command. For details, see Saving and Retrieving HOLD Files on page 374. In the VM/CMS environment you can use a FILEDEF command to save an output file to a specific location and assign it a file name, file type, and file mode. In z/OS, you can dynamically allocate an output file using the DYNAM ALLOCATE or TSO ALLOCATE command.

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. 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

**How to:**

Create a HOLD File
Set the Default HOLD Format
Query a HOLD Master 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 364.
Creating a HOLD File

- 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 386.

- 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] [MISSING {ON|OFF}])
```

or

```
hold_field HOLD [AS filename] [FORMAT fmt] [MISSING {ON|OFF}])
```

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.

- **FORMAT fmt**
  
  Specifies the format of the HOLD output file. BINARY is the default format for reporting servers.

- To display as a Web page, choose: HTML, HTMTable, DHTML

- To display as a printed document, choose: PDF, PS
To use in a text document, choose: **ALPHA, DOC, WP**

In addition, Developer Studio supports: **CLIP**

To use in a spreadsheet application, choose: **DIF, EXCEL, EXL97, EXL2K [PIVOT], LOTUS, SYLK, TAB, TABT**

To use in a database application, choose: **COMMA, COM, COMT, FOCUS, DB2, FUSION, INGRES, REDBRICK, SQL, SQLDBC, SQLORA, SQLINF, SQLMSS, SQLSYB, SQLODBC, TABT**

In addition, Developer Studio supports: **SQLMAC**

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 386.

**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 697. 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.

**Syntax:**

**How to Set the Default HOLD Format**

```plaintext
SET HOLDFORMAT = {BINARY | ALPHA}
```

or

```plaintext
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.

```plaintext
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 EMPLOYEE SALARY OF
------ ---------- --------        ---------  ---------        ----------
   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
   120.00 PRODUCTION  50.00       $16,100.00  MCKNIGHT    ROGER
   30.00 $26,862.00    IRVING    JOAN
   25.00 $11,000.00    STEVENS    ALFRED
   10.00 $9,500.00      SMITH    RICHARD
    5.00 $21,120.00    ROMANS    ANTHONY
     .00 $29,700.00    BANNING    JOHN

Holding Report Output in FOCUS Format

How to:
Create HOLD Files in FOCUS Format

Reference:
Operating System Notes for HOLD Files in FOCUS Format
Controlling the FOCUS File Structure

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 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 be up to 64 characters long.

- **Note:** If you use a name longer than eight characters on OS/390, 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 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

In CMS, a USE command is issued and a new data source and Master File are created on the disk that has WRITE permission and the most available space.
In z/OS, the HOLD file is dynamically allocated if it is not currently allocated. 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 363.
**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, $
```
Creating a Two-Segment FOCUS Data Source

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:

```
SEGO1 S1
  COUNTRY
  SALES

SEGO2 S2
  CAR
  MODEL
  YEAR
```

The Master File for this newly-created FOCUS data source is:

```
FILE=X3, SUFFIX=FOC
SEGMENT=SEGO1, SEGTYPE=S01
  FIELDNAME=COUNTRY , ALIAS=E01 , USAGE=A10 , $
  FIELDNAME=SALES  , ALIAS=E02 , USAGE=I6    , $
SEGMENT=SEGO2, SEGTYPE=S02, PARENT=SEGO1
  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.

```plaintext
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:

```plaintext
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=SEG01
  FIELDNAME=BODYTYPE  ,ALIAS=E06     ,USAGE=A12   ,$
  FIELDNAME=FOCLIST   ,ALIAS=E07     ,USAGE=I5    ,$
  FIELDNAME=SALES     ,ALIAS=E08     ,USAGE=I6    ,$
```

Controlling Attributes in HOLD Master Files

In this section:

- Controlling Field Names in a HOLD Master File
- Controlling Fields in a HOLD Master File
- Controlling the TITLE and ACCEPT Attributes in the HOLD Master File

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 364.
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. See Controlling Fields in a HOLD Master File on page 368.

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 370.

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

How to:
Control Field Names in a HOLD Master File

Reference:
Usage Notes for Controlling Field Names in HOLD Files

When SET ASNAMES is set to ON 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|FOCUS]
```

where:

ON
Uses the literal specified in an AS phrase for the field name and controls the way ACROSS fields are named in HOLD files of any format.

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.

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.
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 HOLDATTR 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.

- All characters are converted to uppercase.

- 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.

- Aliases are not carried over into the HOLD Master File. A new set of aliases is supplied automatically. These aliases are named E01 for the first field, E02 for the second, and so forth.

- 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,

  \texttt{PRINT COUNTRY AS 'PLACE,OF,ORIGIN'}

  produces the field name \texttt{PLACE} in the HOLD Master File.

- 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.

```sql
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:

```plaintext
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.

```sql
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:

```plaintext
FILE=HOLD2, SUFFIX=FIX
SEGMENT=HOLD2, SEGTYPE=S01,$
  FIELDNAME=CAR           ,ALIAS=E01 ,USAGE=A16 ,ACTUAL=A16 ,$
  FIELDNAME=SALES         ,ALIAS=E02 ,USAGE=I16 ,ACTUAL=I04 ,$
  FIELDNAME=AVERAGESALES  ,ALIAS=E03 ,USAGE=I16 ,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 ,$
```
Controlling Fields in a HOLD Master File

How to:
Control Fields in a HOLD 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}
where:

PRINTONLY
Specifications 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) are not included in the HOLD file.

ALL
Specifications 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.

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
```
Controlling Attributes in HOLD Master Files

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

Controlling the TITLE and ACCEPT Attributes in the HOLD Master File

How to:
Control TITLE and ACCEPT Attributes

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:

- `FILENAME=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

How to: Control Keyed Retrieval for a HOLD File

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 $S{n}$ or $S{H{n}}$ 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

```plaintext
SET FIXRETRIEVE = {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,$
SEGNAME=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:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1200</td>
</tr>
<tr>
<td>2</td>
<td>2340</td>
</tr>
<tr>
<td>3</td>
<td>4875</td>
</tr>
</tbody>
</table>

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

**How to:**
Specify a Storage Location for a HOLD Master File (Windows, UNIX, OpenVMS)

**Reference:**
Allocating HOLD Files on z/OS

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.

For an illustration, see *Specifying a Storage Location for HOLD Data and Master Files* on page 375.

**Syntax:**

**How to Specify a Storage Location for a HOLD Master File (Windows, UNIX, OpenVMS)**

```plaintext
APP MAP appname path_to_directory
APP HODDATA appname
APP HOLDMETA appname
```

where:

- **APP MAP**
  Associates a directory location with an application name.

- **APP HODDATA**
  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:

```plaintext
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:

```plaintext
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:

```plaintext
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:

```plaintext
APP APPENDPATH holdapp
```

Then you can issue a request against the HOLD file:

```plaintext
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):

```plaintext
TABLE FILE holdapp/sales
PRINT *
END
```
Allocating HOLD Files on z/OS

On z/OS, the HOLD file is dynamically allocated if it is not currently allocated. 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

**In this section:**
- Column Names in the HOLD File
- Primary Keys and Indexes in the HOLD File

**How to:**
Save Report Output as a Native Temporary Table Using Commands

**Reference:**
Temporary Table Properties for SAME_DB Persistence Values

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 379, 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 379, 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 379, 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.
**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.

<table>
<thead>
<tr>
<th>DBMS</th>
<th>VOLATILE</th>
<th>GLOBAL_TEMPORARY</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB2</strong></td>
<td>DB2 on UNIX, 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.</td>
<td>DB2 Release 7.1 and up for z/OS only: 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.</td>
</tr>
<tr>
<td><strong>Informix</strong></td>
<td>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.</td>
<td>This type of table is not supported by Informix.</td>
</tr>
<tr>
<td><strong>Microsoft SQL Server</strong></td>
<td>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.</td>
<td>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.</td>
</tr>
</tbody>
</table>
### 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.

<table>
<thead>
<tr>
<th>DBMS</th>
<th>VOLATILE</th>
<th>GLOBAL TEMPORARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>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.</td>
<td>This type of table is not supported by MySQL.</td>
</tr>
<tr>
<td>Oracle</td>
<td>This type of table is not supported by Oracle.</td>
<td>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.</td>
</tr>
<tr>
<td>Teradata</td>
<td>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.</td>
<td>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.</td>
</tr>
</tbody>
</table>
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

How to:
Create a SAVE File
Create a SAVB File

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 Web page, 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 386.

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.
How to Create a SAVE File

Syntax:

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 Web page:
  - 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 386.

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 697.
**Example:** Creating a SAVE File

The following request extracts data from the EMPLOYEE data source and creates a SAVE file.

```sql
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:

```
NUMBER OF RECORDS IN TABLE= 12 LINES= 12

ALPHANUMERIC RECORD NAMED SAVE
FIELDNAME            ALIAS  FORMAT  LENGTH
DEPARTMENT            DPT    A10     10
LAST_NAME             LN     A15     15
FIRST_NAME            FN     A10     10
TOTAL                  35
```

**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 697.

**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:

```
NUMBER OF RECORDS IN TABLE= 10  LINES= 10

INTERNAL RECORD NAMED SAVB
FIELDNAME     ALIAS     FORMAT  LENGTH
DATE          DTE       A4MD     4
PROD_CODE     PCODE     A3       4
AREA          LOC       A1       4
TOTAL         "         "       12
```
Creating a PCHOLD File

How to:
Create 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), as is the case for Developer Studio used in client/server mode, 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 353.

If you specify an ON TABLE PCHOLD command without a FORMAT, XML/HTML code is returned to the browser. In WebFOCUS Version 4.2.1a, the Master File and report output are returned to the browser. To avoid this, use ON TABLE HOLD FORMAT ALPHA.
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 Web page, 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, EXL2K [PIVOT], LOTUS
- To use for additional reporting, choose:
  ALPHA, BINARY

For details about all available formats, see Choosing Output File Formats on page 386.

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 Web page, 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.

<table>
<thead>
<tr>
<th>AHTML</th>
<th>DFIX</th>
<th>GIF</th>
<th>POSTSCRIPT (PS)</th>
<th>SQLORA</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHTMLTAB</td>
<td>DHTML</td>
<td>HTML</td>
<td>PPT</td>
<td>SQLSYB</td>
</tr>
<tr>
<td>ALPHA</td>
<td>DIF</td>
<td>HTMTABLE</td>
<td>REDBRICK</td>
<td>SYLK</td>
</tr>
<tr>
<td>APDF</td>
<td>DOC</td>
<td>INGRES</td>
<td>SQL</td>
<td>TAB</td>
</tr>
<tr>
<td>BINARY</td>
<td>EXCEL</td>
<td>INTERNAL</td>
<td>SQLDB</td>
<td>TABT</td>
</tr>
<tr>
<td>CLIP</td>
<td>EXL2K</td>
<td>JPEG</td>
<td>VISDIS</td>
<td>VISDIS</td>
</tr>
<tr>
<td>COMMA</td>
<td>FORMULA</td>
<td>LOTUS</td>
<td>WK1</td>
<td>WP</td>
</tr>
<tr>
<td>COM</td>
<td>EXL2K PIVOT</td>
<td>PDF</td>
<td>EXL2K</td>
<td>XFOCUS</td>
</tr>
<tr>
<td>DB2</td>
<td>EXL97</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBASE</td>
<td>FLEX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FOCUS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 426.

**Reference:**  **FORMAT AHTML**

**Description:** Saves report output as an 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 e-mail and transparent to security systems.

**Use:** For offline analysis of data. For details, see *Creating an Active Report* on page 658.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

**Available in:** WebFOCUS, Developer 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 Report 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, Developer 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 407.

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 Developer Studio. As a transaction file for modifying a data source.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

**Available in:** WebFOCUS, Developer 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 runtime 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, Developer 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 414.)

The output file contains data only.

**Use:** For further reporting in FOCUS, WebFOCUS, or Developer Studio. As a transaction file for modifying a data source.

**Supported with the commands:** HOLD, PCHOLD

**Available in:** WebFOCUS, Developer Studio, FOCUS.

**Reference:** FORMAT CLIP

**Description:** Copies the report text in tab delimited format to the Windows Clipboard, from which you can paste the output into any Windows application that supports pasting from the Clipboard.

**Use:** For display in a text document.

**Supported with the command:** HOLD.
Available in: Developer Studio.

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 or file type 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, Developer 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 or file type 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, Developer 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 or file type 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, Developer Studio, WebFOCUS.

**Reference:** FORMAT DB2

**Description:** Creates a DB2 table, if you have the DB2 Data Adapter and permission to create tables.

**Use:** For further processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, Developer 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:** Developer 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 409.

**Use:** For importing data to Windows-based applications such as MS Access and Excel.

**Supported with the command:** HOLD, PCHOLD.

**Available in:** WebFOCUS, Developer Studio, FOCUS.
8. Saving and Reusing Your Report Output

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 Web page.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

**Available in:** WebFOCUS, Developer 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, Developer 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, Developer 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.

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, PCHOLD, SAVE.

Available in: WebFOCUS, Developer Studio, FOCUS.

In FOCUS, the recommended transfer mechanism is FTP in binary mode. In CMS, the file type of the resulting file is XLS. On a PC, the extension 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:

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 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 429.

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 dropped from the report.

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 429.

Available in: WebFOCUS, Developer Studio.
Choosing Output File Formats

**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 429.

**Use:** For display or processing in a spreadsheet application.

**Supported with the commands:** HOLD, PCHOLD, SAVE.

**Available in:** WebFOCUS, FOCUS, Developer 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 429.

**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, Developer 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 429.

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, Developer Studio.

**Reference:** FORMAT FLEX

**Description:** Saves report output as an Active Report in an Adobe Flash (.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 Report Flash/Flex (FLEX) format, Adobe Flash Player 9.0.28 or higher is required.

**Use:** For offline analysis of data.
Supported with the command: HOLD, PCHOLD, SAVE.
Available in: FOCUS, WebFOCUS, Developer 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 358.

Text fields are supported for FOCUS output files. See Using Text Fields in Output Files on page 407.

Use: For further processing in a database application.
Supported with the command: HOLD.
Available in: WebFOCUS, Developer 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/footing 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, Developer Studio.

For details see Creating a Graph on page 1245.

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, PAGEMATRIX, GUTTER, BACKIMAGE, BACKCOLOR, IMAGE, GRIDS.

Use: For display as a Web page.
Supported with the commands: HOLD, PCHOLD, SAVE.
Available in: WebFOCUS, Developer 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, Developer Studio, FOCUS.

Reference: FORMAT INGRES

Description: Creates an Ingres table, if you have the Ingres Data Adapter and permission to create tables.

Use: For further processing in a database application.

Supported with the command: HOLD.

Available in: WebFOCUS, Developer 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 414.

Use: For accurate processing by 3GL programs.

Supported with the command: HOLD, SAVB.

Available in: WebFOCUS, Developer 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/footing 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 files behave the same as GIF files.

**Supported with the commands:** HOLD.

**Available in:** WebFOCUS, Developer Studio.

For details, see *Linking From a Graphic Image* on page 556 and *Adding an Image to a Report* on page 1016.

**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, Developer Studio, FOCUS.

In VM/CMS, the LOTUS file has a file type of PRN and allocates a scratch data set to the file HOLD.

**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:

<table>
<thead>
<tr>
<th>Object</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pages</td>
<td>10,000</td>
</tr>
<tr>
<td>Images</td>
<td>900</td>
</tr>
<tr>
<td>Hyperlinks per page</td>
<td>500</td>
</tr>
<tr>
<td>Total pages with hyperlinks</td>
<td>100</td>
</tr>
<tr>
<td>Total hyperlinks</td>
<td>44,500</td>
</tr>
</tbody>
</table>

PDF format retains all formatting options, such as 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, Developer 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 929.

**Supported with the command:** PCHOLD.

**Available in:** WebFOCUS, Developer 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. In CMS, the file type is PS.

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, Developer 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, Developer Studio, FOCUS.

**Reference: FORMAT REDBRICK**

**Description:** Creates a Red Brick table, if you have the Redbrick Data Adapter and permission to create tables.

**Use:** For further processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, Developer Studio, FOCUS when used as a client to the WebFOCUS Reporting Server.

**Reference: FORMAT SQL**

**Description:** Creates an SQL/DS table, if you have the adapter for SQL/DS and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, Developer Studio, FOCUS.
Choosing Output File Formats

Reference: **FORMAT SQLDBC**

**Description:** Creates a Teradata table, if you have the Teradata Data Adapter and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, Developer Studio, FOCUS.

Reference: **FORMAT SQLINF**

**Description:** Creates an Informix table, if you have the Informix Data Adapter and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, Developer 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 Microsoft Access Data Adapter and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, Developer Studio.

Reference: **FORMAT SQLMSS**

**Description:** Creates a Microsoft SQL Server table, if you have the Microsoft SQL Data Adapter and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, Developer 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 ODBC Data Adapter and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, Developer Studio, FOCUS when used as a client to the WebFOCUS Reporting Server.

**Reference: FORMAT SQLORA**

**Description:** Creates an Oracle table, if you have the Oracle Data Adapter and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, Developer Studio, FOCUS.

**Reference: FORMAT SQLSYB**

**Description:** Creates a Sybase table, if you have the Sybase Data Adapter and permission to create tables.

**Use:** For processing in a database application.

**Supported with the command:** HOLD.

**Available in:** WebFOCUS, Developer 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, Developer 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, Developer 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, Developer 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:

<table>
<thead>
<tr>
<th>FOCUS Format</th>
<th>Visual Discovery Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>I</td>
</tr>
<tr>
<td>Decimal/Packed</td>
<td>R</td>
</tr>
<tr>
<td>Alphanumeric</td>
<td>S</td>
</tr>
<tr>
<td>Date format (smart date)</td>
<td>D%format%format%format (for example, D%Y%M%D). If the year is not a four-digit year, the format returned is S.</td>
</tr>
<tr>
<td>Other</td>
<td>S</td>
</tr>
</tbody>
</table>

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:** Developer 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 407.

To control whether a carriage control character is included in column 1 of each page of the report output, use:

```plaintext
[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 TAPBAGENO 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 TAPBAGENO is included in the heading and omits the carriage control character if TAPBAGENO 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.
In z/OS, the WP file is created with a record format of VB 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. On z/OS, The LRECL of the output file will be four bytes more than the report width 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, Developer 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, Developer Studio, FOCUS.

**Using Text Fields in Output Files**

**Reference:** Rules for 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 697.

In environments that support FIXFORM, due to limitations in the use of text fields with FIXFORM, the following restrictions apply:

- 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:

<table>
<thead>
<tr>
<th>COURSECODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMA130</td>
<td>FOR PROJECT LEADERS AND SYS ANALYSTS.</td>
</tr>
<tr>
<td>AMA680</td>
<td>FOR INDUSTRIAL MARKETERS.</td>
</tr>
<tr>
<td>AMA800</td>
<td>FOR MANAGERS AND HUMAN RESOURCE STAFF.</td>
</tr>
<tr>
<td>BIT420</td>
<td>FOR SENIOR MANAGERS. ANALYZE AND IMPROVE</td>
</tr>
<tr>
<td>BIT640</td>
<td>FOR ADMINISTRATOR OF LABOR CONTRACTS.</td>
</tr>
<tr>
<td>BIT650</td>
<td>FOR PROGRAMMERS, PROJECT LEADERS AND</td>
</tr>
<tr>
<td>EDP090</td>
<td>FOR PRODUCTION MANAGERS. TO DELEGATE AND</td>
</tr>
<tr>
<td>EDP130</td>
<td>FOR PROGRAMMERS, PROJECT LEADERS AND</td>
</tr>
<tr>
<td>EDP390</td>
<td>FOR MANAGERS. HELP APPLY MANAGERIAL</td>
</tr>
<tr>
<td>EDP690</td>
<td>FOR EXEC MANAGERS AND MKTG RESEARCH</td>
</tr>
<tr>
<td>EDP750</td>
<td>FOR MARKETING MANAGERS. ENABLE TO</td>
</tr>
<tr>
<td>MC230</td>
<td>FOR PROGRAMMERS, PROJECT LEADERS AND</td>
</tr>
<tr>
<td>MC90</td>
<td>FOR VICE PRES OF SALES AND MARKETING.</td>
</tr>
<tr>
<td>NAMA40</td>
<td>FOR MARKETING VICE PRES AND MANAGERS.</td>
</tr>
<tr>
<td>NAMA730</td>
<td>FOR MANAGERS OF SALESPEOPLE. TO ENHANCE</td>
</tr>
<tr>
<td>NAMA930</td>
<td>FOR FINANCIAL MANAGERS. EXPLORE CONCEPTS</td>
</tr>
<tr>
<td>PDR330</td>
<td>FOR EXECUTIVE SECRETARIES. TO KNOW THE</td>
</tr>
<tr>
<td>PDR740</td>
<td>FOR EXPERIENCED MARKETING MANAGERS. HELP</td>
</tr>
<tr>
<td>PDR870</td>
<td>FOR DESIGNERS AND TRAINING COORDINATORS.</td>
</tr>
<tr>
<td>PU168</td>
<td>FOR MARKETING, PRODUCT, ADVERTISING</td>
</tr>
<tr>
<td>PU440</td>
<td>FOR GENERAL MANAGERS. TO EXPLORE</td>
</tr>
<tr>
<td>SFC280</td>
<td>FOR MANAGERS AND SECRETARIES. HELP</td>
</tr>
</tbody>
</table>

### Creating a Delimited Sequential File

**How to:**

Create a Delimited Sequential File

**Reference:**

Usage Notes for HOLD FORMAT DFIX

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 and whether to include a header record that lists the names of the fields.

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), and whether there is a header record. The Master and Access Files are useful if you will later read the sequential file using WebFOCUS.
Creating a Delimited Sequential File

Syntax: How to Create a Delimited Sequential File

ON TABLE \{HOLD|PCHOLD\} [AS filename] FORMAT DFIX
   DELIMITER delimiter [ENCLOSURE enclosure] [HEADER {YES|NO}]

where:

filename
Is the name of the file to be created. If you do not specify a name, the default name is HOLD.

delimiter
Consists of up to 30 printable or non-printable non-null characters. 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. 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).

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.

closure
Consists of up to four printable characters used to enclose each alphanumeric value in the file. 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.

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.

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.
**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:

```plaintext
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:

```plaintext
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:

```plaintext
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 ("|"):

```plaintext
ON TABLE HOLD AS PIPE1 FORMAT DFIX DELIMITER | ENCLOSURE "
```

The Master File remains the same, but the Access File now specifies the enclosure character:

```plaintext
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: 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;
MQUANTITY/I9 MISSING ON = IF REGION EQ 'WEST' THEN MISSING ELSE 200;
END

TABLE FILE CENTORD
SUM QUANTITY MQUANTITY 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

How to:
Suppress Field Padding in HOLD Files

Reference:
Usage Notes for Suppressing Padded Fields in HOLD Files

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 Fields in a HOLD Master File on page 368.

`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 415.
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.

For an illustration, see *Creating a HOLD File With HOLD FORMAT INTERNAL* on page 416.

**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

  $$\text{bytes} = \text{INT} \left( \frac{n}{2} \right) + 1$$

  where:

  $$\text{INT} \left( \frac{n}{2} \right)$$

  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.

```plaintext
TABLE FILE CAR
PRINT CAR COUNTRY RETAIL_COST DEALER_COST SEATS
ON TABLE HOLD AS DJG
END
```

The request creates the following Master File:

```plaintext
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.

```plaintext
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:

```plaintext
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.
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.
Creating a Structured HOLD File

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 419.

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 418. 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 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, $
    FIELDDNAME=RECTYPE, ALIAS=R, USAGE=A3, ACTUAL=A3, $
    FIELDDNAME=LAST_NAME, ALIAS='LN', USAGE=A15, ACTUAL=A15, $
    FIELDDNAME=FIRST_NAME, ALIAS='FN', USAGE=A10, ACTUAL=A10, $
    FIELDDNAME=DEPARTMENT, ALIAS='DPT', USAGE=A10, ACTUAL=A10, $
    FIELDDNAME=ED_HRS, ALIAS='OJT', USAGE=F6.2, ACTUAL=A06, $
  SEGMENT=PAYINFO, SEGTYPE=S0, PARENT=EMPINFO, $
    FIELDDNAME=RECTYPE, ALIAS=1, USAGE=A3, ACTUAL=A3, $
    FIELDDNAME=SALARY, ALIAS='SAL', USAGE=D12.2M, ACTUAL=A12, $
    FIELDDNAME=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 *Creating a Structured HOLD File in ALPHA Format* on page 419:

```
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:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>SALARY</th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>JOBCODE</th>
<th>ED_HRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>$27,062.00</td>
<td>CROSS</td>
<td>BARBARA</td>
<td>A17</td>
<td>45.00</td>
</tr>
<tr>
<td></td>
<td>$25,775.00</td>
<td>CROSS</td>
<td>BARBARA</td>
<td>A16</td>
<td>45.00</td>
</tr>
<tr>
<td></td>
<td>$21,780.00</td>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>B04</td>
<td>75.00</td>
</tr>
<tr>
<td></td>
<td>$18,480.00</td>
<td>JONES</td>
<td>DIANE</td>
<td>B03</td>
<td>50.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MCCOY</td>
<td>JOHN</td>
<td>B02</td>
<td>.00</td>
</tr>
<tr>
<td></td>
<td>$17,750.00</td>
<td>JONES</td>
<td>DIANE</td>
<td>B02</td>
<td>50.00</td>
</tr>
<tr>
<td></td>
<td>$13,200.00</td>
<td>SMITH</td>
<td>MARY</td>
<td>B14</td>
<td>36.00</td>
</tr>
<tr>
<td></td>
<td>$9,000.00</td>
<td>GREENSPAN</td>
<td>MARY</td>
<td>A07</td>
<td>25.00</td>
</tr>
<tr>
<td></td>
<td>$8,650.00</td>
<td>GREENSPAN</td>
<td>MARY</td>
<td>B01</td>
<td>25.00</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>$29,700.00</td>
<td>BANNING</td>
<td>JOHN</td>
<td>A17</td>
<td>.00</td>
</tr>
<tr>
<td></td>
<td>$26,862.00</td>
<td>IRVING</td>
<td>JOAN</td>
<td>A15</td>
<td>30.00</td>
</tr>
<tr>
<td></td>
<td>$24,420.00</td>
<td>IRVING</td>
<td>JOAN</td>
<td>A14</td>
<td>30.00</td>
</tr>
<tr>
<td></td>
<td>$21,120.00</td>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>B04</td>
<td>5.00</td>
</tr>
<tr>
<td></td>
<td>$16,100.00</td>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>B02</td>
<td>50.00</td>
</tr>
<tr>
<td></td>
<td>$15,000.00</td>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>B02</td>
<td>50.00</td>
</tr>
<tr>
<td></td>
<td>$11,000.00</td>
<td>STEVENS</td>
<td>ALFRED</td>
<td>A07</td>
<td>25.00</td>
</tr>
<tr>
<td></td>
<td>$10,000.00</td>
<td>STEVENS</td>
<td>ALFRED</td>
<td>A07</td>
<td>25.00</td>
</tr>
<tr>
<td></td>
<td>$9,500.00</td>
<td>SMITH</td>
<td>RICHARD</td>
<td>A01</td>
<td>10.00</td>
</tr>
<tr>
<td></td>
<td>$9,050.00</td>
<td>SMITH</td>
<td>RICHARD</td>
<td>B01</td>
<td>10.00</td>
</tr>
</tbody>
</table>

**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 and 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=S0, as neither KEY nor INDEX information is retained. An INDEX can be reinserted using REBUILD INDEX.

Specifying MIME Types for WebFOCUS Reports

**How to:**
Specify MIME Types in Windows

**Reference:**
Supported MIME Types and Applications

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 e-mail. 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 e-mail attachment in PDF or DOC format and launch the report from your browser in Adobe Acrobat Reader or in Microsoft Word, respectively.
Reference: Supported MIME Types and Applications

WebFOCUS supports the following MIME types and desktop applications:

<table>
<thead>
<tr>
<th>MIME Type</th>
<th>Application</th>
<th>File Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>application/PostScript</td>
<td>GhostView. GhostView is a third-party product and is available at the following Web site: <a href="http://www.cs.wisc.edu/~ghost/gsview/index.htm">www.cs.wisc.edu/~ghost/gsview/index.htm</a>.</td>
<td>.ps</td>
</tr>
<tr>
<td>application/x-prn</td>
<td>Microsoft Excel or equivalent spreadsheet application.</td>
<td>.prn</td>
</tr>
<tr>
<td>application/x-dif</td>
<td>Microsoft Excel or equivalent spreadsheet application.</td>
<td>.dif</td>
</tr>
<tr>
<td>application/ms word</td>
<td>Microsoft Word.</td>
<td>.doc</td>
</tr>
<tr>
<td>application/vnd.ms-excel</td>
<td>Microsoft Excel.</td>
<td>.xls</td>
</tr>
<tr>
<td>application/x-msexcel</td>
<td>Microsoft Excel 2000.</td>
<td>.htx</td>
</tr>
<tr>
<td>application/pdf</td>
<td>Adobe Acrobat Reader.</td>
<td>.pdf</td>
</tr>
<tr>
<td>text/plain</td>
<td>Microsoft Notepad or equivalent text processor.</td>
<td>.wp or .hts</td>
</tr>
<tr>
<td>text/html</td>
<td>Microsoft Notepad or equivalent HTML editor.</td>
<td>.html</td>
</tr>
<tr>
<td>text/htm</td>
<td>Microsoft Notepad or equivalent HTML editor.</td>
<td>.htm</td>
</tr>
<tr>
<td>image/gif</td>
<td>Microsoft Paint or equivalent graphic editor.</td>
<td>.gif</td>
</tr>
<tr>
<td>image/jpg</td>
<td>Microsoft Paint or equivalent graphic editor.</td>
<td>.jpg</td>
</tr>
</tbody>
</table>
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 chose 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.
- An Excel 2000 spreadsheet, where you can work with the data in Excel.

You can learn which display formats are available in Report Display Formats on page 430.

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 351.

Topics:

- 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
- Using Excel Display Formats: Excel 2000, Excel 97, Excel
Report Display Formats

How to:
Choose a Display Format Using PCHOLD
Choose a Display Format Using SET ONLINE-FMT

Reference:
Specifying MIME Types for WebFOCUS Reports

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 435.

- **Print formats**: PDF (Adobe Acrobat Portable Document Format) and PostScript (PS). For more information, see *Using PDF Display Format* on page 439, and *Using PostScript (PS) Display Format* on page 440.

- **Word-processing formats**: WP and DOC. For more information, see *Using Word Processing Display Formats: DOC, WP* on page 454.

- **Spreadsheet formats**: Excel 2000 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 *Using Excel Display Formats: Excel 2000, Excel 97, Excel* on page 455.

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 351.
**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:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOC</td>
<td>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 <em>Using Word Processing Display Formats: DOC, WP</em> on page 454.</td>
</tr>
<tr>
<td>EXCEL</td>
<td>Specifies that the report will be displayed as an Excel spreadsheet. See <em>Using Excel Display Formats: Excel 2000, Excel 97, Excel</em> on page 455.</td>
</tr>
<tr>
<td>Excel 2000 (EXL2K FORMULA)</td>
<td>Specifies that the report will be displayed as an Excel 2000 spreadsheet, 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.) See <em>Generating Native Excel Formulas in Excel 2000</em> on page 474.</td>
</tr>
<tr>
<td>Excel 2000 (EXL2K PIVOT)</td>
<td>Specifies that the report will be displayed as an Excel 2000 PivotTable. See <em>Using PivotTables in Excel 2000</em> on page 480.</td>
</tr>
<tr>
<td>Excel 97 (EXL97)</td>
<td>Specifies that the report will be displayed as an Excel 97 worksheet. See <em>Viewing a Report in Excel 97 Format</em> on page 490.</td>
</tr>
<tr>
<td>HTML</td>
<td>Specifies that the report will be displayed as an HTML page. See <em>Using Web Display Format: HTML</em> on page 435.</td>
</tr>
</tbody>
</table>
Specifies that the report will be displayed as a PDF document (Adobe Acrobat’s Portable Document Format). See Using PDF Display Format on page 439.

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 Using PostScript (PS) Display Format on page 440.

WP

Specifies that the report will be displayed as a plain-text word processing document in the Web browser. See Using Word Processing Display Formats: DOC, WP on page 454.

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:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML (default)</td>
<td>Specifies that the report will be displayed as an HTML page. See Using Web Display Format: HTML on page 435.</td>
</tr>
<tr>
<td>PDF</td>
<td>Specifies that the report will be displayed as a PDF document (Adobe Acrobat Portable Document Format). See Using PDF Display Format on page 439.</td>
</tr>
<tr>
<td>Excel 97 (EXL97)</td>
<td>Specifies that the report will be displayed as an Excel 97 worksheet. See Viewing a Report in Excel 97 Format on page 490.</td>
</tr>
</tbody>
</table>
### 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 *Using PostScript (PS) Display Format* on page 440.

| 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 431). 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

**How to:**

Preserve Leading and Internal Blanks in HTML and EXL2K Reports

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 FOCEXC or in a profile, use the following syntax:

```plaintext
SET SHOWBLANKS = {OFF | ON}
```

In a request, use the following syntax

```plaintext
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:

```plaintext
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:

<table>
<thead>
<tr>
<th>NEWCAT</th>
<th>CATEGORY</th>
<th>LISTPR COPIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAINING EXERCISE</td>
<td>TRAIN/EX</td>
<td>119.87 10</td>
</tr>
<tr>
<td>ACTION</td>
<td>ACTION</td>
<td>94.82 14</td>
</tr>
<tr>
<td>COMEDY</td>
<td>COMEDY</td>
<td>154.80 19</td>
</tr>
<tr>
<td>GENERAL</td>
<td>MYSTERY</td>
<td>1,216.82 67</td>
</tr>
<tr>
<td>SCIENCE FICTION</td>
<td>SCI/FI</td>
<td>114.84 7</td>
</tr>
</tbody>
</table>

With SHOWBLANKS ON, the additional leading and internal blanks are preserved. Note that trailing blanks are not preserved:

<table>
<thead>
<tr>
<th>NEWCAT</th>
<th>CATEGORY</th>
<th>LISTPR COPIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAINING EXERCISE</td>
<td>TRAIN/EX</td>
<td>119.87 10</td>
</tr>
<tr>
<td>ACTION</td>
<td>ACTION</td>
<td>94.82 14</td>
</tr>
<tr>
<td>COMEDY</td>
<td>COMEDY</td>
<td>154.80 19</td>
</tr>
<tr>
<td>GENERAL</td>
<td>MYSTERY</td>
<td>1,216.82 67</td>
</tr>
<tr>
<td>SCIENCE FICTION</td>
<td>SCI/FI</td>
<td>114.84 7</td>
</tr>
</tbody>
</table>

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. For more information, see Controlling Report Formatting on page 825. 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 433.

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 431.

- **ONLINE-FMT parameter** of the SET command. For more information, see How to Choose a Display Format Using SET ONLINE-FMT on page 432.

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. For examples, see Customizing the Display of an HTML Report on page 436 and Disabling Default WebFOCUS JavaScript Functions on page 437.

**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:

   ```javascript
   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_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.

```sql
SET JSURL=/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_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.

   ```javascript
   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.

```sql
-OLAP ON
SET AUTODRILL = ON
SET JSURL=/ibi_html/disable.js
TABLE FILE CENTORD
SUM QUANTITY
BY PLANTLNG
END
```
Using Print Display Formats: PDF, PS

In this section:
- Using PDF Display Format
- Using PostScript (PS) Display Format
- Adding PostScript Type 1 Fonts for PS and PDF Formats
- Creating PDF Files for Use With UNIX Systems

How to:
- Specify Line Termination Characters When Creating a PDF File

Reference:
- Required PDFLINETERM Settings Based on Environment

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 439.

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 440.
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 431.

- **ONLINE-FMT parameter** of the SET command. For more information, see *How to Choose a Display Format Using SET ONLINE-FMT* on page 432.

You can combine multiple styled reports into a single PDF or PS file. For details, see *Laying Out the Report Page* on page 929.

PDF and PS reports, including compound reports, can be distributed using Report Caster. See the ReportCaster documentation for details.

### Using PDF Display Format

**How to:**

Compress a PDF Output File

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](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 351.

**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 440.

**Syntax:**

**How to Compress a PDF Output File**

```sql
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.

Using PostScript (PS) Display Format

**How to:**

Select Paper Size in a PostScript (PS) Report

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 440. 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 439.

**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

```plaintext
SET PAGESIZE= option
```
or

```plaintext
ON TABLE SET PAGESIZE option
```
or

```plaintext
PAGESIZE=option, $
```
where:

```plaintext
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.

```plaintext
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
```
Example: 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
```
Adding PostScript Type 1 Fonts for PS and PDF Formats

In this section:
- How WebFOCUS Uses Type 1 Fonts
- WebFOCUS StyleSheet Declaration

How to:
- Configure Type 1 Postscript Fonts on the Windows and UNIX Platforms
- Configure Type 1 Postscript Fonts on z/OS Under PDS Deployment

Reference:
- Support for the Symbol Font

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 Acrobat Reader. Thousands of PostScript fonts are available to make your reports more stylish and useful, including some that support symbols and bar codes.

How WebFOCUS Uses Type 1 Fonts

WebFOCUS generates a PDF, DHTML, PPT, 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.

- **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 don’t require font files, since the fonts are built in to Acrobat.)
**Note:** A Printer Metrics File (PFM) is also available. This file is used by applications such as Acrobat 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 and font metrics files (AFM and PFB or PFA). There are two versions of this file: PDF.FMP and PSCRIPT.FMP. For DHTML format, there is DHTML.FMP. You can update either one or both, depending on the output format (PDF or PS) with which you plan to use the Type 1 fonts. For illustrations, see WebFOCUS Font Map File Entries for PDF on page 449 and WebFOCUS Font Map File Entries for PS on page 451.

Through a simple process, you can customize your environment to take advantage of these fonts.

- First, you copy the font files (AFM and PFB or PFA) to a location that is accessible by the WebFOCUS Reporting Server. (The location that contains the default fonts shipped with WebFOCUS is a good choice: drive:/srv76/home/nls.)

- Next, you update the WebFOCUS font map files (PDF.FMP and/or PSCRIPT.FMP). These configuration files already exist in the correct location, with specifications for the fonts that are distributed with Adobe Acrobat Reader. The FMP files map the name of a font to its actual font file information. You will add the new font definitions to these files.

  After this step is completed, you can begin using the new font in your StyleSheet declarations.

  Note that if you specify a style (such as bold or italic) that is not supported for the font you are using, the specified font will display without the style.

This procedure varies for Windows/UNIX platforms and z/OS.

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

**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 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 and PPT 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 and PPT 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.

**Procedure:** How to Configure Type 1 Postscript Fonts on the Windows and UNIX Platforms

Locate the necessary font files (AFM and PFB or PFA). These file 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. WebFOCUS will embed the fonts in a PDF file, making it possible to display the PDF file on any machine that has Adobe Acrobat or Acrobat Reader.)

- If you are working in a UNIX environment, note that PFB files are binary, so if they are FTPed from another machine, they must be FTPed in BINARY mode.

- For PostScript, the fonts you configure must be installed on the printer(s) you intend to use.

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, copy both the AFM and PFB (or PFA) files to a location known to the WebFOCUS Reporting Server. The suggested location on a Windows platform is

   `<drive>:ibi\srv76\home\nls`

   where the default font files shipped with WebFOCUS are installed.

   If you are working in a UNIX environment, copy the files to the corresponding location.

   After you copy these files, you can rename them to any descriptive name.
2. Open the font map file (pdf.fmp or pscript.fmp) in a text editor. These files provide the references to font information that are used in the WebFOCUS StyleSheet for PDF and PS formats, respectively. (They contain the mapping information for the default fonts that are delivered with Acrobat Reader.)

The font map files are already located in

<drive>:ibi\srv76\home\nls

3. Add a separate declaration for each font and style. A line in the WebFOCUS font map file may not exceed 256 characters. You can break the line after any comma.

The syntax is

```
font=fontname, style=style, metricsfile=METRICSFILENAME AFM *,
fontfile=FONTFILENAME PFB *, $
```

where:

- **fontname**
  - Is the name of a Type 1 font. This name is used to reference the font in the WebFOCUS StyleSheet.

- **style**
  - Is a font style. Normal is the default value. This entry is used to reference the font style in the WebFOCUS StyleSheet.

  If you wish to specify font style variations (for example, bold, italic, bold-italic for a particular font), you must purchase the font files for those styles and create a separate entry for each one in the WebFOCUS font map file.

- **METRICSFILENAME**
  - Maps the font to the name of the font metrics file. The name of the metrics file must be followed by a space, followed by AFM, followed by an '*'.

  This entry must be in upper case and may not exceed 8 characters. (On UNIX systems, the actual file should have a lower case name, but the WebFOCUS font map entries should be upper case.)

- **FONTFILENAME**
  - Maps the font to the name of the font file. The name of the font file must be followed by a space, followed by PFB (or PFA), followed by an '*'.

  The entry must be in uppercase and may not exceed 8 characters. (On UNIX systems, the actual file should have a lower case name, but the WebFOCUS font map entries should be upper case.)

For illustrations, see WebFOCUS Font Map File Entries for PDF on page 449 and WebFOCUS Font Map File Entries for PS on page 451.
**Tip:** If you are familiar with WebFOCUS StyleSheets, notice that the syntax of these files is similar. Lines can be continued after a comma and statements are terminated by a dollar sign ($).

**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 set up WebFOCUS to use one or more Type 1 fonts.

1. Copy the AFM file into the EDAHETC data set (typically named PREFIX.P.HOME.ETC), or into a PDS with the same DCB attributes that is concatenated to it).

   You can copy this file using FTP from Windows (in standard ASCII mode). The member name in the PDS should match the METRICSFILE name in the font map file.

   Referring to the example, WebFOCUS Font Map File Entries for PDF on page 449, you would place the AFM file in PREFIX.P.HOME.ETC(OCRA).

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, via FTP in BINARY mode), and allocate this 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 are using PFA (ASCII) font files, create a PDS, put the PFA file in it (for example, using FTP in regular, ASCII mode), and allocate this 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 name in the font map file.

   Note that you can use PFB and PFA files simultaneously. The three character file type in the font map file (PFB or PFA) tells WebFOCUS which PDS to search for the specified member name.
3. Open the WebFOCUS font map file (members PDF or PSCRIPT in the PDS allocated to DDNAME EDAERR) in a text editor. (This is usually the same data set that contains the AFM files.)

**Note:** If the Windows font file names contain underscore characters, you must rename them, since underscore characters are not valid in z/OS member names. For example, if the Windows designation is font=garamond_light, z/OS references would be as follows:

```
FONTFILENAME=garalt pdb *,
METRICFILENAME= garalt afm*, $
```

4. Add a separate declaration for each font and style. A line in the WebFOCUS font map file may not exceed 80 characters. You can break the line after any comma.

The syntax is

```
font=fontname, style=style, metricsfile=METRICSFILENAME AFM *,
fontfile=FONTFILENAME PFB *, $
```

where:

- **fontname**
  
  Is the name of a Type 1 font. This name is used to reference the font in the WebFOCUS StyleSheet.

- **style**
  
  Is a font style. Normal is the default value. This entry is used to reference the font style in the WebFOCUS StyleSheet.

  If you wish to specify font style variations (for example, bold, italic, bold-italic for a particular font), you must purchase the font files for those styles and create a separate entry for each one in the WebFOCUS font map file.

- **METRICSFILENAME**
  
  Maps the font to the name of the font metrics file. The name of the metrics file must be followed by a space, followed by AFM, followed by a *.

  This entry must be in upper case and may not exceed 8 characters.

- **FONTFILENAME**
  
  Maps the font to the name of the font file. The name of the font file must be followed by a space, followed by PFB (or PFA), followed by a *.

  The entry must be in upper case and may not exceed 8 characters.

For illustrations, see *WebFOCUS Font Map File Entries for PDF* on page 449 and *WebFOCUS Font Map File Entries for PS* on page 451.
Tip: If you are familiar with WebFOCUS StyleSheets, notice that the syntax of these files is similar. Lines can be continued after a comma and statements are terminated by a dollar sign ($).

Example: WebFOCUS Font Map File Entries for PDF

Initially, this file contains mappings for the default fonts delivered with Adobe Acrobat Reader. These core Type 1 fonts are mapped to the appropriate font files. Each "style" of the font requires a separate font file, as shown for the core fonts. Notice that each one has four different sets of files for normal, bold, italic, and bold plus italic. You will need to follow the same model when you add a new Type 1 font to the file. If you want to add the normal and the bold versions of a new Type 1 font, you must make both sets of files available to the Reporting Server and map then in the font map file.

Tip: The file also contains a series of aliases (not shown) that map common Windows True Type fonts to existing Type 1 fonts (for example, Arial to Helvetica) along with a series of fonts used for various languages that WebFOCUS supports. You can set up your own aliases to ensure proper mapping and interpretation of Windows True Type Fonts.

Font metrics are already defined for the default fonts so there is not need to specify the PFB file name in the map file. However, you will need to specify that file, along with the AFM file for each new font.

$ PDF.FMP: StyleSheets Font Map file for PDF Driver $
$ Version 1: All fonts mapped into Courier, Helvetica or Times. $ "Native" Acrobat fonts:

```
font=Courier, style=normal, metricsfile=PSCOUR AFM *, $  
font=Courier, style=bold, metricsfile=PSCOURB AFM *, $  
font=Courier, style=italic, metricsfile=PSCOURI AFM *, $  
font=Courier, style=bold+italic, metricsfile=PSCOURBI AFM *, $  
font=Helvetica, style=normal, metricsfile=PDHELV AFM *, $  
font=Helvetica, style=bold, metricsfile=PDHELVB AFM *, $  
font=Helvetica, style=italic, metricsfile=PDHELVI AFM *, $  
font=Helvetica, style=bold+italic, metricsfile=PDHELVBI AFM *, $  
font=Times, style=normal, metricsfile=PDTIME AFM *, $  
font=Times, style=bold, metricsfile=PDTIMEB AFM *, $  
font=Times, style=italic, metricsfile=PDTIMEI AFM *, $  
font=Times, style=bold+italic, metricsfile=PDTIMEBI AFM *, $  
```

/* Following is an entry for a new Type 1 Font: OCRA */
/* A quick way to add an entry is to copy and modify an existing one. */

```
font=OCRA, style=normal, metricsfile=OCRA AFM *, fontfile=OCRA PFB *, $  
font=OCRA, style=bold, metricsfile=OCRAB AFM *, fontfile=OCRAB PFB *, $  
```
**WebFOCUS StyleSheet Declaration**

The following is a WebFOCUS StyleSheet declaration for the new Type1 font, OCRA.

```
ON TABLE SET STYLE *
TYPE=REPORT, FONT=OCRA, $
ENDSTYLE
```

The STYLE attribute has been omitted in the WebFOCUS StyleSheet. It is not required since STYLE defaults to NORMAL, the desired effect in this instance.
**Example:** WebFOCUS Font Map File Entries for PS

The WebFOCUS font map file for PS includes a wide range of fonts that are typically supported by Level 2 Printers. To take advantage of a listed font, it must be installed on your printer.

These core Type 1 fonts are mapped to the appropriate font files. Each "style" of the font requires a separate font file, as shown for the core fonts. Notice that each one has four different sets of files for normal, bold, italic, and bold plus italic. You will need to follow the same model when you add a new Type 1 font to the file. If you want to add the normal and the bold versions of a new Type 1 font, you must make both sets of files available to the Reporting Server and map then in the font map file.

```plaintext
font=Arial, style=normal, metricsfile=PSHELV AFM *, $
font=Arial, style=bold, metricsfile=PSHELVB AFM *, $
font=Arial, style=italic, metricsfile=PSHELVl AFM *, $
font=Arial, style=bold+italic, metricsfile=PSHELVBI AFM *, $
font=Avant Garde Gothic, style=normal, metricsfile=PSAVAN AFM *, $
font=Avant Garde Gothic, style=bold, metricsfile=PSAVANB AFM *, $
font=Avant Garde Gothic, style=italic, metricsfile=PSAVANI AFM *, $
font=Avant Garde Gothic, style=bold+italic, metricsfile=PSAVANBI AFM *, $
font=Bookman, style=normal, metricsfile=PSBOOK AFM *, $
font=Bookman, style=bold, metricsfile=PSBOOKB AFM *, $
font=Bookman, style=italic, metricsfile=PSBOOKI AFM *, $
font=Bookman, style=bold+italic, metricsfile=PSBOOKBI AFM *, $
font=Courier, style=normal, metricsfile=PSCOUR AFM *, $
font=Courier, style=bold, metricsfile=PSCOURB AFM *, $
font=Courier, style=italic, metricsfile=PSCOURI AFM *, $
font=Courier, style=bold+italic, metricsfile=PSCOURBI AFM *, $
font=Courier New, style=normal, metricsfile=PSCOUR AFM *, $
font=Courier New, style=bold, metricsfile=PSCOURB AFM *, $
font=Courier New, style=italic, metricsfile=PSCOURI AFM *, $
font=Courier New, style=bold+italic, metricsfile=PSCOURBI AFM *, $
font=Helvetica, style=normal, metricsfile=PSHELV AFM *, $
font=Helvetica, style=bold, metricsfile=PSHELVB AFM *, $
font=Helvetica, style=italic, metricsfile=PSHELVl AFM *, $
font=Helvetica, style=bold+italic, metricsfile=PSHELVBI AFM *, $
font=Helvetica Narrow, style=normal, metricsfile=PSNHLE AFM *, $
font=Helvetica Narrow, style=bold, metricsfile=PSNHLEB AFM *, $
font=Helvetica Narrow, style=italic, metricsfile=PSNHLEI AFM *, $
font=Helvetica Narrow, style=bold+italic, metricsfile=PSNHLEBI AFM *, $
```
Creating PDF Files for Use With UNIX Systems

PDF files created with HOLD FORMAT PDF present a challenge if you work in a z/OS or VM 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 or VM 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:

```plaintext
SET PDFLINETERM={STANDARD|SPACE}
```

In a TABLE request, issue the following command

```plaintext
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:

<table>
<thead>
<tr>
<th>Transferring from z/OS or VM to:</th>
<th>SET PDFLINETERM=</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBCDIC UNIX (text transfer)</td>
<td>SPACE</td>
</tr>
<tr>
<td>ASCII UNIX (text transfer)</td>
<td>SPACE</td>
</tr>
<tr>
<td>ASCII UNIX (text); then to Windows (binary)</td>
<td>SPACE</td>
</tr>
<tr>
<td>UNIX (text); then to Windows (text)</td>
<td>STANDARD</td>
</tr>
<tr>
<td>Directly to Windows (text)</td>
<td>STANDARD</td>
</tr>
</tbody>
</table>

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 431.
You can display a WebFOCUS report as one of several kinds of Microsoft Excel spreadsheets. The report opens in Excel within a Web browser.
Using Excel Display Formats: Excel 2000, Excel 97, Excel

You can display a report as an:

- **Excel 2000 Worksheet** (format EXL2K). The Excel 2000 format supports most StyleSheet attributes, allowing for full report formatting. The computer on which the report is being displayed must have Microsoft Excel 2000 or higher installed. 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 433.

In addition, WebFOCUS supports two Excel 2000 variations: EXL2K FORMULA and EXL2K PIVOT. When either of these formats is specified additional processing is done.

- **Excel 2000 FORMULA** (EXL2K FORMULA). If you display a report using the format variation EXL2K FORMULA, the resulting spreadsheet will contain Excel formulas that calculate and display the results of any type of summed information (such as column totals, row totals, and sub-totals).

- **Excel 2000 PivotTable** (format EXL2K PIVOT). Generates a fully functional Excel PivotTable, an Excel tool for analyzing complex data, much like WebFOCUS OLAP.

- **Excel 97 spreadsheet** (format EXL97). Excel 97 is an HTML-based display format that opens in Excel 97 or higher and supports report formatting and drill-downs. The computer on which the report is being displayed must have Microsoft Excel 97 or higher installed.

- **Excel spreadsheet** (format EXCEL). Excel is a binary display format with limited formatting support. (Although it does not support text fields, Excel 2000 format does.) The computer on which the report is being displayed must have Microsoft Excel installed.

You can specify that a report display as an Excel spreadsheet when you run the report in WebFOCUS. You can use either:

- **PCHOLD command**. For more information, see *How to Choose a Display Format Using PCHOLD* on page 431.

- **ONLINE-FMT parameter** of the SET command. This is supported for the Excel 2000 and Excel 97 formats; for Excel 2000 PivotTable, Excel 2000 FORMULA, and Excel use the PCHOLD command. For more information, see *How to Choose a Display Format Using SET ONLINE-FMT* on page 432.

You can also view Excel reports outside of the browser in a standalone Excel application window. Under Windows, you must uncheck the *Browse in same window* option for the .xls file type. 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 standalone Excel application window.
Reference: Usage Notes for EXL2K Output Files

Certain EXL2K format reports, such as Pivot Tables, generate multiple output files. The main document is returned to the browser (which opens it in Excel), and the secondary file (such as the pivot cache file in a Pivot Table report) is linked through a URL in the main file, to be retrieved by an HTTP request that Excel makes to the WebFOCUS servlet. When used with HOLD instead of PCHOLD, the output is a single web archive file with the same .xht suffix as a normal EXL2K HOLD file. In Excel, this file format is designated in the Save dialog as Single File Web Page.

- If you are using Excel 2002 or later, with WebFOCUS Version 7 Release 6.9 or earlier, you can avoid having to issue this second call by setting WEBARCHIVE=ON in any of the supported profiles or in the procedure (fex). Certain types of EXL2K reports, such as compound reports and reports with a table of contents (TOC), rely on the web archive format and automatically turn WEBARCHIVE ON.

- If you are using Excel 2002 or later, with WebFOCUS Version 7 Release 6.10 or higher, by default, the WEBARCHIVE setting is set to ON.

Note: WEBARCHIVE=ON uses a file format that packages all of the documents into a single file.

For additional information about redirection settings for EXCEL formats with SET WEBARCHIVE=ON, see Microsoft Excel Report Output Considerations in the WebFOCUS Security and Administration Manual.

Assigning Excel 2000 Format to Your Report Output

Reference:

Font Support for Excel 2000 Format

The command ON TABLE PCHOLD FORMAT EXL2K generates a fully styled Excel report in your browser, with conditional styling and drill-down capability.

The EXL2K (Excel 2000) format is a full StyleSheet driver that renders all report elements (for example, headings and subtotals) as well as StyleSheet syntax (such as, conditional styling and drill-downs). If your Excel report contains a drill-down to a procedure and uses the TOC feature, see Drilling Down to a Procedure in Excel 2000 With a Table of Contents on page 459.

In addition, the format variation EXL2K FORMULA enables you to convert summed information (such as column totals, row totals, subtotals) and calculated values to interactive formulas in an Excel 2000 spreadsheet. See Generating Native Excel Formulas in Excel 2000 on page 474. Another format variation, EXL2K PIVOT, enables you to analyze different views of your data. See Using PivotTables in Excel 2000 on page 480.

EXL2K is supported only in Excel 2000 or higher. It does not work with any previous releases of Excel. You can invoke format EXL2K reports using any browser supported by WebFOCUS. WebFOCUS can also run directly from Excel without using a browser by creating a hyperlink within Excel that calls the WebFOCUS client with the necessary parameters. For details, see Embedding a WebFOCUS Report in an Excel 2000 Spreadsheet on page 501.

By default, when you choose EXL2K as your display format, the report opens in an Excel 2000 worksheet, identified in a tab at the bottom of the spreadsheet as Sheet 1, Sheet2, and so on. You can change the name of a Sheet tab to make it more descriptive of your report's content. For an illustration, see Creating an Excel 2000 Report in Internet Explorer on page 458.

**Tip:** If an Excel report is launched within a frame, the Excel toolbar options are not available. To retain access to the toolbar when working with frames, launch the report outside of the frameset, in a new window. For details on using target frames, see Linking a Report to Other Resources on page 519.

### Example: Creating an Excel 2000 Report in Internet Explorer

The following example illustrates how to create a report in EXL2K (Excel 2000) format with a styled heading, conditional styling on SALES, and drill-downs on COUNTRY:

```plaintext
TABLE FILE CAR
HEADING
"SALES BY COUNTRY"
SUM SALES BY COUNTRY BY CAR BY MODEL
WHERE (COUNTRY EQ 'ENGLAND') OR (COUNTRY EQ 'ITALY') OR (COUNTRY EQ 'FRANCE');
ON TABLE PCHOLD FORMAT EXL2K
ON TABLE SET STYLE *
  TYPE=REPORT, FONT=ARIAL, TITLETEXT=SALES REPORT, $
  TYPE=DATA, COLUMN=COUNTRY, FOCEXEC=DRILLFEX (PARAM1=COUNTRY PARAM2=CAR),$
  TYPE=DATA, COLUMN=SALES, COLOR=RED, BACKCOLOR=YELLOW, WHEN=SALES GT 10000,$
  TYPE=DATA, COLUMN=MODEL, BACKCOLOR=YELLOW, WHEN=SALES GT 10000,$
  TYPE=HEADING, FONT=ARIAL BLACK, COLOR=RED, BACKCOLOR=SILVER, SIZE=16, $
  TYPE=TITLE, FONT=ARIAL, SIZE=12, $
ENDSTYLE
END
```
Notice that the tab name has been changed from the Excel default, Sheet1, to a more descriptive name, which is specified in the TITLETEXT attribute of the StyleSheet. For details on the TITLETEXT attribute see *Using Headings, Footings, Titles, and Labels* on page 1053.

**Reference: Font Support for Excel 2000 Format**

The EXL2K (Excel 2000) format supports fonts that are available on the WebFOCUS client. Be sure to verify the existence of a particular font on the client before displaying or saving a report in the EXL2K format.

**Drilling Down to a Procedure in Excel 2000 With a Table of Contents**

**How to:**

Drill Down to a Procedure in Excel 2000

If an EXL2K (Excel 2000) report utilizes the Table of Contents feature and contains a drill-down to a procedure, the drill-down must reference an absolute URL. Excel incorrectly evaluates relative URL strings when drilling down to a procedure. EXL2K reports with drill-downs to absolute URLs are properly executed.

To drill down to a procedure in Excel 2000, perform the following procedure which enables WebFOCUS to generate only absolute URLs. This procedure is necessary for Excel 2000 and prior versions only. If your WebFOCUS environment uses Proxy servers, you may not be able to implement this procedure.
**Procedure:** How to Drill Down to a Procedure in Excel 2000

1. Identify the type of WebFOCUS access method your installation is using (CGI, ISAPI or Servlet).

2. Locate the default.wfs file.

   For WebFOCUS:
   ```
   install_drive or directory\ibi\client76\wfc\etc\default.wfs
   ```

   For Developer Studio:
   ```
   install_drive or directory\ibi\srv76\wfs\etc\default.wfs
   ```

3. Edit the default.wfs file to update the following string depending on the access method your installation is using:

   For Servlet:
   ```
   \n-SET &FOCEXURL='http://ip_address/ibi_apps/WFServlet?IBIF_webapp=&IBIF_webapp' | '&&'; <\=>
   ```

   For CGI:
   ```
   \n-SET &FOCEXURL='http://ip_address/cgi-bin/ibi.cgi/ibiweb.exe?IBIF_webapp=&IBIF_webapp' | '&&'; <\=>
   ```

   For ISAPI:
   ```
   \n-SET &FOCEXURL='http://ip_address/cgi-bin/ibi_cgi/webapi.dll?IBIF_webapp=&IBIF_webapp' | '&&'; <\=>
   ```

   where:

   *ip_address*  
   Is the IP address of your Web Server.

4. Edit the default.wfs file to update the following string depending on the access method your installation is using:

   For Servlet:
   ```
   FOCEXURL=CGI_PROG?IBIF_webapp=IBIF_webapp&&IBIC_server=_EDA_NODE&
   ```

   For CGI:
   ```
   FOCEXURL=http://ip_address/ibi_apps/WFServlet?IBIF_webapp=&IBIF_webapp&&IBIC_server=_EDA_NODE&
   ```
For CGI:

FOCEXURL=http://ip_address/cgi-bin/ibi_cgi/ibiweb.exe?IBIF_webapp=&IBIF_webapp&&IBIC_server=&_EDA_NODE&

For ISAPI:

FOCEXURL=http://ip_address/cgi-bin/ibi_cgi/webapi.dll?IBIF_webapp=&IBIF_webapp&&IBIC_server=&_EDA_NODE&

where:

\textit{ip\_address}

Is the IP address of your Web Server.

5. Restart your Web and Application servers.

**Identifying Null Values in Excel 2000**

**How to:**

Identify Null Values in Excel 2000

When an Excel 2000 report is run, and null values are retrieved for one or more fields, blank spaces are displayed by default in each cell of the report output for the empty (null) fields. This behavior is the result of SET EMPTYCELLS ON being set by default in the background of all Excel 2000 reports. If you want to identify null values with something other than blank spaces, a character string can be used to populate all empty fields in a report.

**Syntax:**  

**How to Identify Null Values in Excel 2000**

To identify null values in Excel 2000, use

\begin{verbatim}
ON TABLE SET NODATA character_string
ON TABLE SET EMPTYCELLS \texttt{[ON|OFF]}
\end{verbatim}

where:

\textit{character\_string}

Is the string of characters displayed in the cells of the report for each field where null values are retrieved from the database. The maximum number of characters is 11. If the number of characters in the string exceeds the length of the output field, the additional characters will not be displayed. If special characters are used, the string must be enclosed in single quotes. SET EMPTYCELLS OFF must also be specified to make the SET NODATA command effective.
ON

Indicates that empty spaces are displayed in the cells of the report for each field where null values are retrieved from the database. ON is the default.

OFF

Indicates that zeros, or the character string specified with the SET NODATA command, will be displayed in the cells of the report for each field where null values are retrieved from the database. OFF must be specified when using SET NODATA.

Example: Identifying Null Values in Excel 2000

The following syntax utilizes the default behavior of ON TABLE SET EMPTYCELLS ON, which is set in the background:

```
TABLE FILE CAR
  SUM SALES BY COUNTRY ACROSS SEATS
ON TABLE PCHOLD FORMAT EXL2K
END
```

The following output displays empty spaces in the cells of the report for each field where null values are retrieved from the database:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>SEATS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>COUNTRY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ENGLAND</td>
<td>0</td>
<td>0</td>
<td>12000</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>FRANCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>ITALY</td>
<td>25400</td>
<td>4800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>JAPAN</td>
<td></td>
<td>78030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>W GERMANY</td>
<td>8900</td>
<td>79290</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following syntax utilizes the SET NODATA command:

```
TABLE FILE CAR
  SUM SALES BY COUNTRY ACROSS SEATS
  ON TABLE SET NODATA 'n/a'
  ON TABLE SET EMPTYCELLS OFF
  ON TABLE PCHOLD FORMAT EXL2K
END
```

**Note:** If you do not add SET EMPTYCELLS OFF, the SET NODATA command will be ignored.
The following output displays 'n/a' in the cells of the report for each field where null values are retrieved from the database:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SEATS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>COUNTRY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ENGLAND</td>
<td>0</td>
<td>0</td>
<td>12000</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>FRANCE</td>
<td>n/a</td>
<td>n/a</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>ITALY</td>
<td>25400</td>
<td>4800</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>JAPAN</td>
<td>n/a</td>
<td>78030</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>W GERMANY</td>
<td>n/a</td>
<td>8900</td>
<td>79290</td>
<td></td>
</tr>
</tbody>
</table>

The following syntax turns off the default SET EMPTYCELLS behavior and does not use SET NODATA, which makes it impossible to distinguish null values from zero quantities:

```plaintext
TABLE FILE CAR
    SUM SALES BY COUNTRY ACROSS SEATS
    ON TABLE SET EMPTYCELLS OFF
    ON TABLE PCHOLD FORMAT EXL2K
END
```

The following output displays zeros in the cells of the report for each field where either null values are retrieved from the database or the quantity is zero:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SEATS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>COUNTRY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ENGLAND</td>
<td>0</td>
<td>0</td>
<td>12000</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>FRANCE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>ITALY</td>
<td>25400</td>
<td>4800</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>JAPAN</td>
<td>0</td>
<td>78030</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>W GERMANY</td>
<td>0</td>
<td>8900</td>
<td>79290</td>
<td></td>
</tr>
</tbody>
</table>

Creating Reports With WebFOCUS Language
Displaying Formatted Dates and Numeric Values in Excel 2000

**Reference:**
- Usage Notes for Date and Numeric Formats
- Using Date Separators in Excel

Excel 2000 spreadsheets generated by WebFOCUS contain the same numeric formatting as specified in the data source's Master File or as specified in a temporary field. WebFOCUS numeric values and date formats (such as currency and Smart Dates) are translated to supported Excel formats and display properly in Excel 2000.

When translating numeric and date formats from WebFOCUS to Excel, there must be a corresponding Excel format to translate to. If there is no corresponding format, then the value will be formatted in the closest matching Excel format or in Excel General format. For details, see *Usage Notes for Date and Numeric Formats* on page 466.

**Example: Displaying Formatted Numeric Data in Excel 2000**

The following example illustrates how formatted numeric data displays in a spreadsheet when using the EXL2K format. Note that the format for the Fund Balance field (D16.2M—which represents floating point double-precision with two decimal places and a floating dollar sign) is translated to the corresponding Excel format.

```
DEFINE FILE SHORT
NEWBLNC/D16.2M = BALANCE;
END
SET PAGE-NUM=OFF

TABLE FILE SHORT
"Short Term Investments"
"Excel 2000 Spreadsheet"
""
SUM NEWBLNC AS 'Fund Balance'
BY MANAGER_ID AS 'Fund Manager'
BY TYPE
BY HOLDER
ON TABLE SET BYDISPLAY ON
ON TABLE PCHOLD FORMAT EXL2K
ON TABLE SET STYLE *
  TYPE=REPORT, GRID=OFF, FONT=TAHOMA, $
  TYPE=HEADING, SIZE=14, COLOR=NAVY, $
  TYPE=HEADING, LINE=2, SIZE=12, COLOR=RED, $
  TYPE=TITLE, JUSTIFY=CENTER, STYLE=BOLD, $
  TYPE=DATA, JUSTIFY=CENTER, $
  TYPE=DATA, COLUMN=NEWBLNC, JUSTIFY=RIGHT, $
END
```
Notice that the values of the sort fields are repeated in the output; this presentation, which is particularly desirable in a spreadsheet, is controlled by the command ON TABLE SET BYDISPLAY ON. For details, see Sorting Tabular Reports on page 81.

Example: Displaying Formatted Dates in Excel 2000

The following example illustrates how customized dates display in a spreadsheet when using the EXL2K format.

- The format for Month Hired is defined in the request as MtYY (the month is represented as a 3-character abbreviation with an initial capital letter followed by a four-digit year).
- The format for Years of Service is defined as I4C, a four-digit integer with a comma if required. Both formats are properly displayed as defined the spreadsheet.

```
SET PAGE-NUM=OFF
DEFINE FILE EMPLOYEE
  YRHIRED/YY = HIRE_DATE;
  MHIRED/MtYY = HIRE_DATE;
  TOTSVC/I4C = 2002 - YRHIRED;
END
```
TABLE FILE EMPLOYEE
"Employee Service Report for 2002"
"Excel 2000 Spreadsheet"

PRINT FIRST_NAME AS 'First Name'
MHIRED AS 'Month Hired'
TOTSVC AS 'Years of Service'

BY LAST_NAME AS 'Last Name'

ON TABLE SET BYDISPLAY ON
ON TABLE PCHOLD FORMAT EXL2K
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, FONT=TAHOMA, $
TYPE=HEADING, SIZE=14, COLOR=NAVY, $
TYPE=HEADING, LINE=2, SIZE=12, COLOR=RED, $
TYPE=TITLE, JUSTIFY=CENTER, STYLE=BOLD, $
TYPE=DATA, JUSTIFY=CENTER, $
TYPE=DATA, COLUMN=TOTSVC, COLOR=BLUE, WHEN=TOTSVC GT 20, $
END

The output is:

The command ON TABLE SET BYDISPLAY ON ensures that sort fields are repeated in each spreadsheet cell. For details, see Sorting Tabular Reports on page 81.

Reference: Usage Notes for Date and Numeric Formats

The following formats are not supported in EXL2K. They will translate into Excel General format and possibly produce unpredictable results:
9. Choosing a Display Format

- YY, Y, M, D, JUL, and I2MT.
- Any date format with a Q (quarter).
- Any packed-decimal (P) date formats.
- Any alphanumeric (A) date formats.
- Fixed Dollar (N) formats.
- Multiple format options. Only single format options are supported when using FORMAT EXL2K. For example, the formats I9C and I9B are supported, but I9BC is not.

The following applies to headings and footings with embedded numeric fields:

- If you embed a numeric field in a heading, subheading, footing, or subfooting of a report in Excel 2000 or higher format, the numeric field displays in Excel general format (text). To display a numeric field in Excel number format, you must set HEADALIGN=BODY in the StyleSheet.

Reference: Using Date Separators in Excel

In order to use a "-" as a separator between month, day, and year in Excel, you must change the default date separator for Windows. This setting can be located under Regional Options in the Control Panel.

Controlling Column Width and Wrapping in Excel 2000

**How to:**
- Wrap Data in Excel 2000
- Set Column Width in Excel 2000

Data wrapping, column width, and the scroll area can be controlled in an Excel 2000 spreadsheet when using FORMAT EXL2K. You can:

- Turn data wrapping on. The default behavior is for all data to wrap according to a default column width that is determined by Excel.
- Turn data wrapping off. This setting allows columns to expand to the length of the data value. The column width is determined by Excel, but should be wide enough to fit the longest data value in the column. If a portion of the data is hidden, you can adjust the column width in Excel after the spreadsheet has been generated.
- Turn data wrapping off and set the column width at the same time. If a data value is wider than the specified width of the column, a portion of the data will be hidden from view. You can adjust the column width in Excel after the spreadsheet has been generated.
Specify the exact width of a column with data wrapping on.

For details on WRAP and SQUEEZE syntax, see *Formatting Report Data* on page 1205.

**Syntax:**

**How to Wrap Data in Excel 2000**

```plaintext
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’s height expands to meet the new height of the wrapped data.
    - **OFF**
      - Turns off data wrapping. This setting adjusts the column width of the largest data value in the column. 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.

**Syntax:**

**How to Set Column Width in Excel 2000**

```plaintext
TYPE=REPORT, [COLUMN=column,] SQUEEZE={ON|OFF|n}, $
```

**where:**

- **column**
  - Identifies a particular column. If COLUMN is not included in the declaration, default SQUEEZE behavior is applied to the entire report.

- **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 is the most commonly used SQUEEZE setting in an Excel 2000 report.

**Note:**

- ON/OFF settings for SQUEEZE are not meaningful for EXL2K, and both produce the default behavior.
- SQUEEZE = n turns off data wrapping. If a data value is wider than the specified width of the column, it is hidden from view. You can adjust column width in Excel after the spreadsheet has been generated.
- SQUEEZE is not supported for columns created with the OVER phrase.

**Example: Controlling Column Width and Wrapping in Excel 2000**

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 CAR
   MYDATE/MDY='10/22/60';
   RCD/D14.3=RETAIL_COST;
   VERYLON/A80='Subtract dealer cost from retail cost to calculate profit.';
END

TABLE FILE CAR
PRINT MYDATE RCD
   VERYLONG AS 'Default' VERYLONG AS 'WRAP=OFF'
   VERYLONG AS 'WRAP=4.1' VERYLONG AS 'WRAP=2'
   VERYLONG AS 'SQUEEZE=2' SALES
BY COUNTRY
ON TABLE PCHOLD FORMAT EXL2K
ON TABLE SET STYLE *
   TYPE=DATA, COLUMN=MYDATE, JUSTIFY=CENTER, $
1. TYPE=REPORT, COLUMN=VERYLONG(2), WRAP=OFF, $
2. TYPE=REPORT, COLUMN=VERYLONG(3), WRAP=4.1, $
3. TYPE=REPORT, COLUMN=VERYLONG(4), WRAP=2, $
4. TYPE=REPORT, COLUMN=VERYLONG(5), SQUEEZE=2, $
END
```

where:

1. Identifies the column titled "WRAP=OFF" and turns off data wrapping for that column.
2. Identifies the column titled "WRAP=4.1" and sets the column width to 4.1 inches with data wrapping on.
3. Identifies the column titled "WRAP=2" and sets the column width to 2 inches with data wrapping on.
4. Identifies the column titled "SQUEEZE=2" and sets the column width to 2 inches with data wrapping off.

**Note:** The column titled "Default" illustrates the default column width and wrapping behavior.

Since the output is wider than this page, it is shown in two sections. The following output displays the "Default", "WRAP=OFF", and "WRAP=4.1" columns:

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>WRAP=OFF</th>
<th>WRAP=4.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>3</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>4</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>5</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>6</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>7</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>8</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>9</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>10</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
</tbody>
</table>

The following output displays the "WRAP=2", and "SQUEEZE=2" columns:

<table>
<thead>
<tr>
<th></th>
<th>WRAP=4.1</th>
<th>WRAP=2</th>
<th>SQUEEZE=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>3</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>4</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>5</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>6</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>7</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>8</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>9</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
<tr>
<td>10</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
<td>Subtract dealer cost from retail cost to calculate profit.</td>
</tr>
</tbody>
</table>
Locking Columns in Excel Report Output

Using StyleSheet attributes, you can lock Excel spreadsheet values so they are read-only. These attributes apply to all Excel formats including EXL2K, EXL2K PIVOT, and EXL2K FORMULA.

**Syntax:**

**How to Enable Spreadsheet Locking**

To enable locking, use the following attributes:

`TYPE=REPORT, PROTECTED={ON|OFF}, [LOCKED={ON|OFF}],`$

where:

`TYPE=REPORT, PROTECTED=ON`

Is necessary to enable spreadsheet locking. PROTECTED=OFF is the default. If you omit the LOCKED=OFF attribute, the entire spreadsheet is locked.

`LOCKED=ON`

Locks the entire spreadsheet. ON is the default value.

`LOCKED=OFF`

Unlocks the spreadsheet as a whole, but enables you to lock or unlock specific cells or groups of cells.

**Syntax:**

**How to Lock Specific Cells Within a Spreadsheet**

Once you include the following declaration in your StyleSheet, you can specify the LOCKED attribute for specific cells or groups of cells:

`TYPE=REPORT, PROTECTED=ON, LOCKED=OFF,`$

To lock specific parts of the spreadsheet, add the LOCKED=ON attribute to the StyleSheet declaration for the cells you want to lock.

`TYPE=type, [ COLUMN=columnspec ] ,LOCKED={ON|OFF},`$

where:

`type`

Is the type of element that describes the cells to be locked.
columnspec

Is a valid column specification.

Example:  Locking an Entire Excel Spreadsheet

The following request locks the entire spreadsheet because the StyleSheet declarations include the following declaration:

```
TYPE=REPORT, PROTECTED=ON, $
```

The request is:

```
TABLE FILE CAR
HEADING
"Profit By Car "
"
SUM RETAIL_COST AND DEALER_COST AND
COMPUTE PROFIT/D12.2 = RETAIL_COST - DEALER_COST;
BY CAR
ON TABLE SET PAGE-NUM OFF
ON TABLE PCHOLD AS EXLFORM1 FORMAT EXL2K
ON TABLE SET STYLE *
TYPE=REPORT, COLOR=BLUE, BACKCOLOR=SILVER, SIZE=9,$
TYPE=REPORT, PROTECTED=ON, $
TYPE=HEADING, STYLE=BOLD, SIZE=14, $
TYPE=TITLE, STYLE=BOLD, SIZE=11,$
ENDSTYLE
END
```

You cannot edit any value on the spreadsheet. Any attempt to do so displays a message that the sheet is protected:

---

**Profit By Car**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CAR</td>
<td>RETAIL_COST</td>
<td>DEALER_COST</td>
<td>PROFIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>19,566</td>
<td>19,235</td>
<td>330.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ALFA ROMEO</td>
<td>5,970</td>
<td>5,083</td>
<td>907.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DATSUN</td>
<td>10,355</td>
<td>9,750</td>
<td>605.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>JAGUAR</td>
<td>12,500</td>
<td>11,800</td>
<td>700.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>JENSEN</td>
<td>14,000</td>
<td>13,200</td>
<td>800.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>MASERATI</td>
<td>16,000</td>
<td>15,200</td>
<td>1,000.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>PEUGEOT</td>
<td>18,000</td>
<td>17,200</td>
<td>1,800.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>TOYOTA</td>
<td>20,000</td>
<td>19,400</td>
<td>600.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TRIUMPH</td>
<td>22,000</td>
<td>21,200</td>
<td>800.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: Locking a Single Column on an Excel Spreadsheet

The following request locks the second column (RETAIL_COST) because the StyleSheet declarations include the following declarations:

```
TYPE=REPORT, PROTECTED=ON, LOCKED=OFF, $
TYPE=DATA, COLUMN=2, LOCKED=ON,$
```

The request is:

```
TABLE FILE CAR
HEADING
"Profit By Car "
" "
SUM RETAIL_COST AND DEALER_COST AND
COMPUTE PROFIT/D12.2 = RETAIL_COST - DEALER_COST;
BY CAR
ON TABLE SET PAGE-NUM OFF
ON TABLE PCHOLD AS EXLFORM2 FORMAT EXL2K
ON TABLE SET STYLE *
TYPE=REPORT, COLOR=BLUE, BACKCOLOR=SILVER, SIZE=9,$
TYPE=REPORT, PROTECTED=ON, LOCKED=OFF,$
TYPE=HEADING, STYLE=BOLD, SIZE=14, $
TYPE=TITLE, STYLE=BOLD, SIZE=11,$
TYPE=DATA, COLUMN=2, LOCKED=ON,$
ENDSTYLE
END
```

You cannot edit any value in column 2, although you can edit values in other columns. Any attempt to edit a value in column 2 displays a message that the cells are protected:
Generating Native Excel Formulas in Excel 2000

How to:
Save Reports as FORMAT EXL2K FORMULA

Reference:
Translation Support for FORMAT EXL2K FORMULA

When you display or save a tabular report request using EXL2K FORMULA, the resulting spreadsheet 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. Spreadsheets saved using the EXL2K FORMULA format are interactive, allowing for "what if" scenarios that immediately reflect any additions or modifications made to the data.

The EXL2K FORMULA format is supported for the WebFOCUS TABLE commands: ROW-TOTAL, COLUMN-TOTAL, SUB-TOTAL, SUBTOTAL, SUMMARIZE, RECOMPUTE, and COMPUTE, and for calculations performed by functions. See Translation Support for FORMAT EXL2K FORMULA on page 479.

EXL2K FORMULA is not supported with PivotTables (EXL2K PIVOT), with Excel 97 (EXL97), or with financial reports created with the Financial Report Painter or the underlying Financial Modeling Language (FML).

Syntax:  How to Save Reports as FORMAT EXL2K FORMULA

Add the following syntax to your request to take advantage of Excel formulas in your spreadsheet:

ON TABLE {PCHOLD|HOLD} FORMAT EXL2K FORMULA

where:

PCHOLD
Displays the output in an Excel 2000 spreadsheet.

HOLD
Saves the output for reuse in an Excel 2000 spreadsheet. For details, see Saving and Reusing Your Report Output on page 351.
**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 format EXL2K FORMULA. Notice that the formatting of the column total (TYPE=GRANDTOTAL) is retained in the Excel 2000 spreadsheet.

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 SET STYLE *
TYPE=REPORT, COLOR=BLUE, BACKCOLOR=SILVER, SIZE=9, $
TYPE=HEADING, STYLE=BOLD, SIZE=14, $
TYPE=TITLE, STYLE=BOLD+UNDERLINE, SIZE=10,$
TYPE=GRANDTOTAL, STYLE=BOLD, $
ENDSTYLE

ON TABLE PCHOLD FORMAT EXL2K FORMULA
END
```

The output is:

![Excel spreadsheet showing column totals](image)

You can translate any total (subtotal, row total, or column total) to an Excel formula. For related information, see *Translation Support for FORMAT EXL2K FORMULA* on page 479.
**Example:** **Generating Native Excel Formulas for Row Totals**

This request calculates totals for returns and balances across continents. The row totals are represented as sums of cell ranges.

```sql
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 SET STYLE *
TYPE=REPORT, COLOR=BLUE, BACKCOLOR=SILVER, SIZE=9,$
TYPE=HEADING, STYLE=BOLD, SIZE=14, $
TYPE=TITLE, STYLE=BOLD, SIZE=11,$
TYPE=SUBTOTAL,STYLE=BOLD, $
TYPE=GRANDTOTAL,STYLE=BOLD,SIZE=11,$
TYPE=ACROSSTITLE,STYLE=BOLD,SIZE=11,JUSTIFY=LEFT,$
TYPE=ACROSSVALUE,STYLE=BOLD,SIZE=10,JUSTIFY=CENTER,$
ENDSTYLE
ON TABLE PCHOLD FORMAT EXL2K FORMULA
END
```

The following output highlights the formula that calculates the row total in cell I12=C12+E12+G12:

```
I12  =C12+E12+G12
```

<table>
<thead>
<tr>
<th>CONTINENT</th>
<th>AMERICAS</th>
<th>ASIA</th>
<th>EUROPE</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGION</td>
<td>RETURN</td>
<td>BALANCE</td>
<td>RETURN</td>
<td>BALANCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMERICAS</td>
<td>36,380</td>
<td>466,472,731</td>
<td>17,089</td>
<td>247,320,813</td>
</tr>
<tr>
<td></td>
<td>49,200,667</td>
<td></td>
<td>12,377,597</td>
<td></td>
</tr>
<tr>
<td>EASTERN EUROPE</td>
<td>9,100</td>
<td>202,580,842</td>
<td>4,900</td>
<td>320,640,842</td>
</tr>
<tr>
<td></td>
<td>16,159</td>
<td>4,900</td>
<td>44,653,771</td>
<td>7,500</td>
</tr>
<tr>
<td>MIDDLE EAST</td>
<td>7,300</td>
<td>44,653,771</td>
<td>7,500</td>
<td>44,550,771</td>
</tr>
<tr>
<td>NORTH AMERICA</td>
<td>12,460</td>
<td>307,690,802</td>
<td>12,460</td>
<td>307,690,802</td>
</tr>
<tr>
<td>SOUTH AMERICA</td>
<td>6,900</td>
<td>67,961,261</td>
<td>8,900</td>
<td>87,961,261</td>
</tr>
<tr>
<td>WESTERN EUROPE</td>
<td>7,989</td>
<td>245,990,274</td>
<td>7,989</td>
<td>245,990,274</td>
</tr>
</tbody>
</table>
```

Using Excel Display Formats: Excel 2000, Excel 97, Excel
Example:  Generating Native Excel Formulas for Calculated Values

This request totals the columns for retail cost and dealer cost and calculates the value of a field called PROFIT by subtracting the dealer cost from the retail cost.

The formula for the calculated values is generated by translating the internal form of the WebFOCUS expression (COMPUTE PROFIT/D12.2 = RC - DC;) into an Excel formula. In this example, the formulas appear in cells B14, C14, and D14.

```
TABLE FILE CAR
ON TABLE SET PAGE-NUM OFF
SUM RC AND DC AND
COMPUTE PROFIT/D12.2 = RC - DC;
BY CAR
HEADING
"Profit By Car"
""
ON TABLE COLUMN-TOTAL
ON TABLE SET STYLE *
TYPE=REPORT, COLOR=BLUE, BACKCOLOR=SILVER, SIZE=9, $
TYPE=HEADING, STYLE=BOLD, SIZE=14, $
TYPE=TITLE, STYLE=BOLD, SIZE=11,$
TYPE=GRANDTOTAL, STYLE=BOLD, SIZE=11,$
ENDSTYLE
ON TABLE PCHOLD FORMAT EXL2K FORMULA
END
```

The following output highlights the formula that calculates for the column total of PROFIT:

\[ D14 = \text{SUM}(D4:D13) \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Profit By Car</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>CAR</strong></td>
<td><strong>RETAIL_COST</strong></td>
<td><strong>DEALER_COST</strong></td>
<td><strong>PROFIT</strong></td>
</tr>
<tr>
<td>4</td>
<td>ALFA ROMEO</td>
<td>19,565</td>
<td>16,235</td>
<td>3,330.00</td>
</tr>
<tr>
<td>5</td>
<td>AUDI</td>
<td>5,970</td>
<td>5,063</td>
<td>907.00</td>
</tr>
<tr>
<td>6</td>
<td>BMW</td>
<td>50,762</td>
<td>49,500</td>
<td>9,262.00</td>
</tr>
<tr>
<td>7</td>
<td>DATSUN</td>
<td>3,139</td>
<td>2,626</td>
<td>513.00</td>
</tr>
<tr>
<td>8</td>
<td>JAGUAR</td>
<td>22,368</td>
<td>18,621</td>
<td>3,748.00</td>
</tr>
<tr>
<td>9</td>
<td>JENSEN</td>
<td>17,850</td>
<td>14,940</td>
<td>2,910.00</td>
</tr>
<tr>
<td>10</td>
<td>MASERATI</td>
<td>31,500</td>
<td>25,000</td>
<td>6,500.00</td>
</tr>
<tr>
<td>11</td>
<td>PEUGEOT</td>
<td>5,610</td>
<td>4,831</td>
<td>979.00</td>
</tr>
<tr>
<td>12</td>
<td>TOYOTA</td>
<td>3,339</td>
<td>2,886</td>
<td>453.00</td>
</tr>
<tr>
<td>13</td>
<td>TRIUMPH</td>
<td>5,100</td>
<td>4,292</td>
<td>808.00</td>
</tr>
<tr>
<td>14</td>
<td>TOTAL</td>
<td>173,204</td>
<td>143,794</td>
<td>29,410.00</td>
</tr>
</tbody>
</table>
Example: Generating a Native Excel Formula for a Function

The following illustrates how functions are translated to Excel 2000 reports. The function DMOD divides ACCTNUMBER by 1000 and returns the remainder to LAST3_ACCT. The Excel formula corresponds to this, =(MOD($C3,1000)).

```
TABLE FILE EMPLOYEE
PRINT ACCTNUMBER AS 'Account Number' AND COMPUTE
LAST3_ACCT/I3L = DMOD(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 EXL2K FORMULA
ON TABLE SET STYLE *
   TYPE=TITLE, SIZE=12, STYLE=BOLD
$ END
```

The output is:

```
<table>
<thead>
<tr>
<th>Last Name</th>
<th>First Name</th>
<th>Account Number</th>
<th>LAST3_ACCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>122360108</td>
<td>108</td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>153800144</td>
<td>144</td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
<td>150150302</td>
<td>302</td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td>040950036</td>
<td>036</td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td>109200096</td>
<td>096</td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td>0273001024</td>
<td>024</td>
</tr>
</tbody>
</table>
```

![Excel Output Screenshot](image-url)
Reference: Translation Support for FORMAT EXL2K FORMULA

- All standard operators are supported. These include arithmetic operators, relational operators, string operators, IF/THEN/ELSE, and logical operators. However, prefix operators are not supported.

The IS-PRESENT and IS-MISSING operators are not supported. In addition the logical operators (AND, OR) are not supported within IF/THEN/ELSE statements.

- The following functions are supported:
  
  - ABS, ARGLEN, ATODBL, BYTVAL, CHARGET, CTRAN, CTRFLD, DECODE DMOD, DOWK, DOWKI, DOWKL, DOWKLI, EDIT (1 argument variant only), EXP, EXPN, FMOD, HEXBYT, HHMMSS, IMOD, INT, LCWORD, LJJUST, LOCASE, LOG, MAX, MIN, OVRLAY, POSIT, RDUNIF, RANDOM, RJJUST, SQRT, SUBSTR, TODAY, TODAYI, UPCASE, YM.

  If you use the COMPUTE command with an unsupported function, an error message is displayed.

- EXL2K FORMULA is not supported with the following WebFOCUS commands and phrases:
  
  - DEFINE.
  - OVER.
  - NOPRINT. If your report contains a calculated value (generated by the COMPUTE command), all of the fields referenced by the calculated value must be displayed in the report.

  - Multiple display (PRINT, LIST, SUM, and COUNT) commands.
  - SEQUENCE.
  - RECAP.
  - Prefix operators.

- Formulas for ROW-TOTALs and COLUMN-TOTALs are represented as sums of cell ranges. For example, \(=\text{SUM}(G2:G10)\).

- A formula for a calculated value is generated by translating the internal form of the WebFOCUS expression into an Excel formula.

- Excel formulas are limited to 1024 characters. Therefore, the translation to an Excel formula will fail if a WebFOCUS total or the result of an expression requires more than 1024 bytes of Excel formula code and an error message will be generated.

- Conditional styling is based on the values in the original report. If the spreadsheet values are changed and the formulas are recomputed, the styling will not reflect the updated information.
Using PivotTables in Excel 2000

How to:
View the PivotTable Toolbar

Reference:
How TABLE Elements Appear in a Pivot Table
Effect of TABLE Syntax Elements on PivotTables

The power of EXL2K format derives in large measure from its ability to take advantage of PivotTables. The PivotTable is a tool used in Microsoft Excel to analyze complex data much like the OLAP tool in WebFOCUS. It allows you to drag and drop data fields within a PivotTable, providing different views of the data, such as sorting across rows or columns. You can also create dimensional hierarchies, similar to those created using WITHIN syntax, by using the PAGEFIELDS command. For information about WITHIN syntax, see the Describing Data With WebFOCUS Language manual.

Report requests can be created in WebFOCUS and sent as output to a fully formatted Excel PivotTable. The ON TABLE PCHOLD FORMAT EXL2K PIVOT command will generate an Excel PivotTable in your browser.
When FORMAT EXL2K PIVOT is enabled, two data streams are created:

- The first data stream is the PivotTable file.
  
  The PivotTable file (.xht) is an HTML file with embedded XML. The HTML file contains all the information that is displayed in your browser.

- The second data stream is the PivotTable cache file (.xml).

  The PivotTable cache file is a metadata type of file. It contains all the fields specified in the procedure and links internally to the PivotTable file. The PivotTable cache file can contain data fields called CACHEFIELDS, which populate the PivotTable toolbar, but do not initially display in the report. CACHEFIELDS can be dragged and dropped from the PivotTable toolbar into the PivotTable when required for analysis. Note that when the WEBARCHIVE parameter is set to ON (the default), both data streams are packaged into one output file.

For details on the WEBARCHIVE parameter, see the Developing Reporting Applications manual.

**Procedure: How to View the PivotTable Toolbar**

The PivotTable toolbar may not automatically display when a PivotTable is created. To display the PivotTable toolbar, use the following procedure:

1. Click View in the Excel toolbar.
2. Highlight Toolbars.
3. Click PivotTable.

Excel displays the PivotTable toolbar, listing the fields available to be dragged into the body of the PivotTable report. A cell within the PivotTable must be selected for the Pivot toolbar to display all fields.
Reference: How TABLE Elements Appear in a Pivot Table

The PivotTable is generated by the PRINT command in combination with the BY, ACROSS, PAGEFIELDS and CACHEFIELDS phrases. It contains all options used to design and format the report, as well as fields specified in the PIVOT request. Fields can be dragged into the report from the toolbar. The following graphic depicts PivotTable output with the major elements identified.

The following summary table shows PivotTable elements and the associated FOCUS syntax.

<table>
<thead>
<tr>
<th>PivotTable Element</th>
<th>Contains...</th>
<th>Function</th>
<th>Generating Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page field</td>
<td>Field that controls view of the entire page (worksheet).</td>
<td>A filtering mechanism to conduct a high level sort.</td>
<td>PAGEFIELDS phrase</td>
</tr>
<tr>
<td>Page field item</td>
<td>The value for a page field item displays in a drop-down list.</td>
<td>Selecting a page field item summarizes data for the entire report.</td>
<td>PAGEFIELDS phrase</td>
</tr>
</tbody>
</table>
Generating Syntax Function Contains... Generating Syntax

Data field Numeric data that is available to be summarized. PRINT command

Column field Horizontal sort data. ACROSS command

Row field Vertical sort data. BY command

Reference: Effect of TABLE Syntax Elements on PivotTables

The following table summarizes TABLE syntax elements that are supported in EXL2K PIVOT. The effect of each command on your PivotTable request is listed, along with required usage.

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Usage</th>
<th>Effect on PivotTable</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT</td>
<td>Required</td>
<td>Designates the data field in a PivotTable.</td>
</tr>
<tr>
<td>BY</td>
<td>Optional. *</td>
<td>Designates row field in a PivotTable.</td>
</tr>
<tr>
<td>ACROSS</td>
<td>Optional. *</td>
<td>Designates a column field in a PivotTable.</td>
</tr>
<tr>
<td>CACHEFIELDS</td>
<td>Optional. *</td>
<td>Places fields in the Pivot cache file and makes them available from the Pivot toolbar.</td>
</tr>
<tr>
<td>PAGEFIELDS</td>
<td>Optional. *</td>
<td>Designates a Page field in a PivotTable.</td>
</tr>
</tbody>
</table>

* You need at least one sort field or a PAGEFIELD for a valid Pivot Table.
Enabling PivotTables

How to:
Enable the PivotTable

Reference:
Usage Notes for PivotTable Requests

The standard HOLD and SAVE syntax for storing the output file on disk is supported for the EXL2K PIVOT file format. When FORMAT EXL2K PIVOT is enabled, two files are created, the PivotTable file (.xht) and a PivotTable cache file (.xml). The PivotTable cache file contains all the fields specified in the procedure and links internally to the PivotTable file. All available fields can be viewed in the PivotTable toolbar.

You can include the CACHEFIELDS phrase in a request to add fields to the pivot cache that are not initially displayed in the report. The cache file enables you to add available fields from the PivotTable toolbar into the body of the PivotTable by dragging and dropping. You can remove fields from the PivotTable by dragging and dropping them anywhere outside the report. Using these tools, you can quickly vary data views.

Syntax: How to Enable the PivotTable

The following are syntax variations that you can use to enable FORMAT EXL2K PIVOT.

- To view the output immediately in Excel 2000:
  ```sql
  ON TABLE PCHOLD FORMAT EXL2K PIVOT
  ```

- To store output on disk for later viewing
  ```sql
  ON TABLE HOLD FORMAT EXL2K PIVOT AS mypivot
  ```
  where:
  ```sql
  mypivot
  ```
  Is a name you assign to the HOLD file.

Two files are generated with this syntax:

- MYPIVOT.XHT is the main file that is displayed in the browser or Excel window.
- MYPIVO$.XML is the Pivot Cache file.
Usage Notes for PivotTable Requests

You should consider the following when writing requests for output to a PivotTable:

- Requests must use the display command PRINT. This is necessary to extract all data (each individual record) into the pivot cache file. The field that follows the PRINT command is designated as the PivotTable data field and must be numeric.

  Note that the SUM and COUNT commands are not supported for EXL2K PIVOT. Nevertheless, you can use this native Pivot Table behavior in Excel after the data has been output to an Excel PivotTable.

- A field that is used in a PivotTable cannot be listed more than once in the report request. Each field can function in only one role in a FORMAT EXL2K PIVOT request. For example, a Page field cannot display simultaneously as a column and a row field.

- Styling is based on the initial state of your report and retained only on a limited basis when you pivot the output.

- NODATA fields will display as blank cells in the PivotTable.

- Text fields and Timestamp fields may not be used in Excel PivotTables. In addition, ensure that there is no missing data for any fields.

- You can include column, row, grand totals, and subtotals in a PivotTable. For details and examples, see Viewing a Report in Excel 97 Format on page 490. If no totals are specified in a request, no totals will display in Excel 2000.
Example: Populating a PivotTable

The following example illustrates how to generate a PivotTable:

```
TABLE FILE CAR
HEADING
"CAR File PivotTable"
"Sum of Sales by Car Across Model"
PRINT SALES
BY CAR
ACROSS MODEL
ON TABLE COLUMN-TOTAL
ON TABLE SET STYLE *
TYPE=HEADING, LINE=1, FONT='ARIAL', COLOR=PURPLE, SIZE=16, STYLE=BOLD,$
TYPE=HEADING, LINE=2, FONT='ARIAL', COLOR=PURPLE, SIZE=12, STYLE=BOLD,$
TYPE=DATA, FONT='ARIAL', COLOR=PURPLE,$
TYPE=GRANDTOTAL, FONT='ARIAL', COLOR=PURPLE, SIZE=16, STYLE=BOLD,$
ENDSTYLE
ON TABLE PCHOLD AS CARPIVOT FORMAT EXL2K PIVOT
PAGEFIELDS COUNTRY
CACHEFIELDS RETAIL_COST TYPE
END
```

The command ON TABLE PCHOLD FORMAT EXL2K PIVOT displays the PivotTable in Excel 2000. All fields (including PRINT, BY, and ACROSS fields, and PAGEFIELDS and CACHEFIELDS) appear in the toolbar.
Designating CACHEFIELDS in PivotTables

Reference:
Usage Notes for Specifying CACHEFIELDS

You can specify multiple fields as CACHEFIELDS. These fields will not initially display in the report, but will be available in the pivot cache file if you wish to include them in the report at a later time.

You can then drag these fields in and out of the PivotTable as desired. CACHEFIELDS is an optional phrase; you can generate a PivotTable without a CACHEFIELD.

Reference: Usage Notes for Specifying CACHEFIELDS

Fields designated as CACHEFIELDS must be placed immediately after the PIVOT keyword in the ON TABLE PCHOLD FORMAT EXL2K PIVOT syntax or after a PAGEFIELDS phrase. A field specified as a CACHEFIELD cannot be designated anywhere else in the request. A list of CACHEFIELDS is terminated by the same keywords that terminate a normal report request, such as END or another ON phrase.

Example: Using EXL2K PIVOT With CACHEFIELDS

The following example illustrates how to specify CACHEFIELDS to populate the PivotTable toolbar.

```
TABLE FILE CAR
HEADING
"My PivotTable"
PRINT SALES
BY COUNTRY BY CAR
ON COUNTRY SUB-TOTAL
ON TABLE SET STYLE *
TYPE=DATA, COLUMN=COUNTRY, COLOR=RED, $
type=headings, color=blue, style=bold, size=14,
ENDSTYLE
ON TABLE PCHOLD FORMAT EXL2K PIVOT
CACHEFIELDS MODEL TYPE
END
```
Designating PAGEFIELDS in PivotTables

You can specify a field in the procedure as an Excel 2000 page field. The page field filters the data for the specified field. Then, using PivotTable functionality, you can choose a single value from the page field drop-down menu (also called a page field item) and immediately see only the data associated with that selection. For example, in a report that shows international sales data for cars, if you specify COUNTRY as your page field and select JAPAN as your page field item, you will see only sales data for TOYOTA or NISSAN. If you then select ENGLAND, you will see the data for JAGUAR and TRIUMPH.

A page field can act as the sort field. A valid PivotTable can be generated without specifying a PAGEFIELD if sorting is handled by either a BY or ACROSS phrase. However, if the PivotTable request contains neither a BY or ACROSS phrase, a PAGEFIELD phrase must be included.

**Note:**

- A field specified as a PAGEFIELD cannot be designated anywhere else in the request.
- Because Excel is case insensitive, the values of the PAGEFIELD must not contain duplicate values in different cases. For example, "JONES" and "jones" are considered equal in Excel. Use the UPCASE and LOCASE functions to convert mixed case values.
Example: Using FORMAT EXL2K PIVOT With PAGEFIELDS

This example illustrates the use of PAGEFIELDS syntax to make three fields available in the PivotTable toolbar.

TABLE FILE CAR
HEADING
"My Pivot Table"
PRINT SALES RETAIL_COST
ON TABLE SET STYLE *
type=data, column=country, color=red, $
 type=heading, color=blue, style=bold, size=14,$
ENDSTYLE
ON TABLE PCHOLD FORMAT EXL2K PIVOT
PAGEFIELDS COUNTRY CAR MODEL
CACHEFIELDS TYPE
END

This output is:
Viewing a Report in Excel 97 Format

The EXL97 format allows you to view reports in Excel 97, which supports full styling and drill-down capability.

When you specify EXL97 format, an HTML-based file is generated with an extension of .e97. The appropriate MIME type is automatically assigned to designate Excel as the active application for this file type.

When using the Microsoft Internet Explorer browser and Excel 97, the Excel client opens in the background and the report launches in your browser. You may see the Excel application open and minimize while viewing your report; leave Excel open when viewing the spreadsheet.

After the report is generated, you can save the document as an Excel spreadsheet locally on your PC. In Excel 97, you will be prompted to save the document as a Microsoft Excel Workbook with an .xls extension. This will save the file as a binary Excel document.

WebFOCUS can also run directly from Excel without using a browser by creating a hyperlink within Excel that calls the WebFOCUS client with the necessary parameters. For details, see *Embedding a WebFOCUS Report in an Excel 2000 Spreadsheet* on page 501.


**Note:** Although format EXL97 is available, there are some limitations when compared to the Excel 2000 (EXL2K) formats. See *Limitations for FORMAT EXL97* on page 492. Future enhancements in the area of Excel integration will primarily be made to the EXL2K formats. We recommend upgrading to Excel 2000 or higher so you can take full advantage of our Excel integration, as well as all future enhancements.
**Example:** Viewing a Report in EXL97 Format

The following example illustrates how a report displays when using PCHOLD FORMAT EXL97. The request includes drill-downs to another procedure and formatting options, including conditional styling.

```
TABLE FILE CENTORD
HEADING
"Order Revenue"
"Styled Report in Excel 97"
""
SUM ORDER_DATE LINEPRICE AS 'Order,Total:'
BY HIGHEST 10 ORDER_NUM
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ACCESSIBLE 508
ON TABLE PCHOLD FORMAT EXL97
ON TABLE SET STYLE *
TYPE=HEADING, COLOR=NAVY, SIZE=14, $
TYPE=HEADING, LINE=2, COLOR=RED, $
TYPE=DATA, BACKCOLOR=AQUA, STYLE=BOLD, WHEN=LINEPRICE GT 200000, $
TYPE=TITLE, STYLE=BOLD, $
TYPE=DATA, COLUMN=ORDER_NUM, FOCEXEC=DETAILS, $
END
```

The output is:

```
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Order Number</td>
<td>Date Of Order</td>
<td>Order Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>--------------</td>
<td>---------------</td>
<td>--------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5272</td>
<td>4-4-1995</td>
<td>$285,211.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6574</td>
<td>5-8-1995</td>
<td>$171,643.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2868</td>
<td>6-8-1995</td>
<td>$150,300.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5272</td>
<td>4-4-1995</td>
<td>$132,427.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5272</td>
<td>4-4-1995</td>
<td>$132,427.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5272</td>
<td>4-4-1995</td>
<td>$132,427.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>5272</td>
<td>4-4-1995</td>
<td>$132,427.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5272</td>
<td>4-4-1995</td>
<td>$132,427.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>5272</td>
<td>4-4-1995</td>
<td>$132,427.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5272</td>
<td>4-4-1995</td>
<td>$132,427.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>5272</td>
<td>4-4-1995</td>
<td>$132,427.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>5272</td>
<td>4-4-1995</td>
<td>$132,427.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>5272</td>
<td>4-4-1995</td>
<td>$132,427.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>5272</td>
<td>4-4-1995</td>
<td>$132,427.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>5272</td>
<td>4-4-1995</td>
<td>$132,427.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>5272</td>
<td>4-4-1995</td>
<td>$132,427.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Creating Reports With WebFOCUS Language 491
Reference: Limitations for FORMAT EXL97

- This format is only compatible with Excel 97 or higher. This format is not compatible with Excel 95 or any versions of Excel prior to Excel 95.

- Excel 97 does not support Cascading Style Sheets (CSS). Any styling specified in a report that uses CSS (SET HTMLCSS=ON) will not be respected in Excel 97. The WRAP=n feature is not supported with EXL97 since this feature requires CSS.

- WebFOCUS uses nested HTML tables when generating a heading, which may cause a problem in EXL97. As an alternative, specify HEADALIGN=BODY in your StyleSheets for all HEADING objects including FOOTING, SUBHEAD, and SUBFOOT commands. This creates the heading as a separate table.

- Page numbers display by default and are placed in the heading. We recommend that you turn page numbering off (SET PAGE-NUM=OFF).

- All numeric and date formatting options that are not supported for EXL2K are also not supported for EXL97. In addition, negative numbers displayed with brackets, trailing zeroes after the decimal, and leading zeros will not display with EXL97 reports. Note that dates are typically translated into Excel's General format, which can cause problems with sorting and other Excel features.

- PivotTables are not supported with EXL97. (EXL2K PIVOT is the only valid PivotTable format.)

Embedding a WebFOCUS Report in an Excel 97 Spreadsheet

**How to:**

Create a Web Query (IQY) File

WebFOCUS can integrate with Excel 97 to give you a completely customizable spreadsheet with styling, drill-downs, and formulas.

In an effort to integrate Microsoft Excel with the web, Microsoft introduced a host of new capabilities in Excel 97, including the ability to import HTML directly into Excel spreadsheets with formulas, styling, and AutoFilters. In addition, Excel 97 offers an optional feature of Excel, Web Query, which enables users to query specific web pages or servers in order to pull live data from the web into their spreadsheets while maintaining worksheet layout and formulas. A query can be set to automatically refresh and even prompt the user for parameters.

In Excel 97, web queries are text files that are created in an editor and saved in the appropriate directory with the extension .iqy. (Unlike Excel 2000, there is no graphical environment in which to create queries. For details, see Embedding a WebFOCUS Report in an Excel 97 Spreadsheet on page 494.)
A web query consists of three or four lines of text separated by carriage returns. The IQY file is only needed the first time a query is executed to establish the data location and parameters. After a query is run inside a spreadsheet and the spreadsheet is saved, the query information is always present and can be refreshed at any time.

The process of embedding a WebFOCUS report into an Excel 97 spreadsheet has three parts:

- Create the WebFOCUS report.
- Create the Web Query file.
- Run the Web Query utility to pull the report into the spreadsheet.

**Syntax:** **How to Create a Web Query (IQY) File**

An IQY file is constructed as follows

```
Type_of_query   [CR]
Version_of_query [CR]
URL             [CR]
{POST|GET} parameters  [CR]
```

where:

- **Type_of_query**
  
  Is Web. Since Web is the default value, this is an optional entry.

- **Version_of_query**
  
  Is the current version of the Web Query software. 1 is the default value. This entry is optional.

- **URL**
  
  Is a required entry that points to the WebFOCUS server. The URL needs to be constructed as a call to the WebFOCUS CGI or Servlet and contain any appropriate parameters that need to be passed.

- **POST parameters**
  
  Is the preferred method of retrieving large amounts of data and sending it data back as a separate line of text. For example:

  `http://WFserver/ibi_apps/WFServlet[CR]
  PARAM1=value & PARAM2=value`

- **GET parameters**
  
  Is the preferred method of retrieving smaller amounts of data and appending the data to the URL. For example:

  `http://server/ibi_apps/WFServlet?PARAM1=value&PARAM2=value`
Designates required carriage returns.

**Example:**  Embedding a WebFOCUS Report in an Excel 97 Spreadsheet

**Step 1: Create the WebFOCUS Report**

Create and save the following fully styled WebFOCUS report as webq97. You will embed this report in an Excel 97 spreadsheet. (For proper styling and execution of the request in Excel 97, ensure that Cascading Style Sheets are not in effect; SET HTMLCSS=OFF is the default value.)

The numbers to the left of each line of code correspond to annotations that follow the request.

1. `TABLE FILE CENTORD`
2. `SUM LINEPRICE AS 'Sales'`
3. `BY YEAR AS 'Year'`
4. `BY PRODCAT AS 'Category'`
5. `BY PRODNAME AS 'Product'`
6. `WHERE YEAR EQ '&YEAR';`
7. `ON TABLE SET BYDISPLAY ON`
8. `ON TABLE PCHOLD FORMAT HTML`
9. `ON TABLE SET PAGE-NUM TOP`
10. `ON TABLE SET STYLE *`
11. `TYPE=REPORT, FONT=ARIAL, SIZE=10, GRID=OFF, BACKCOLOR=yellow, $`
12. `TYPE=TITLE, SIZE=11, STYLE=BOLD, $`
13. `TYPE=DATA, COLOR=RED, WHEN=LINEPRICE LT 15000000, $`
14. `TYPE=DATA, COLUMN=PRODCAT, FOCEXEC=webq2(PRODCAT=N1), $`
15. `ENDSTYLE`
16. `END`

**Line 1** identifies the sample data source, CENTORD, which contains sales data for the Century Corporation.

**Lines 2-5** define display and sorting requirements for the report.

**Line 6** specifies selection criteria using a variable, &YEAR.

**Line 7** sets BYDISPLAY to ON. This setting ensures that repeated sort values in the report are populated with data.

By default, repeated sort values in vertical columns (or BY fields) are suppressed in a WebFOCUS report, leaving blank fields in the report. This behavior is not desirable for Excel reports, which are designed to work with data that is repeated in every row to which it applies. A blank column or row can produce misleading results when sorting data.

**Lines 8** defines the output format as HTML.
Important: The report must be in HTML format and not EXL97. This is because Microsoft Query imports fully formatted HTML pages or individual HTML tables into a spreadsheet. Executing an EXL97 request from a query will not return any data to the spreadsheet.

Line 9 turns off page numbering and removes any extra lines generated above the column titles.

Lines 10-15 specify styling attributes for the report: background color, conditional styling, a drill-down to another procedure, and some other miscellaneous styling have been designated. Styling, including drill-downs, carries over into Excel.

Step 2: Create a Web Query (IQY) File

Open a text editor and create the following web query file

```
WEB
1
http://WFserver/ibi_apps/WFServlet?IBIF_ex=webq97
&YEAR=["YEAR","Select Year"]
```

where:

Line 1 identifies this as a Web query.

Line 2 identifies the current version of Microsoft Query.

Line 3 is a typical WebFOCUS CGI request using the GET method. The name of the procedure is webq97. The URL line passes a parameter of YEAR into the procedure. The user will be prompted to supply a value for YEAR. Of course, passing parameters is optional. In the example above, we have a dynamic parameter, but the parameters can be static as well. In other words, hard coded in the IQY file.

Note that carriage returns are required between lines in this file.

After you have constructed the query, save the document with an .iqy extension in the Queries directory under Microsoft Office. For example,

```
<drive>:\Program Files\Microsoft Office\Queries\webq97.iqy
```

Now that the IQY file is ready, you can create the spreadsheet and embed the query.

Step 3: Embed the Query in an Excel 97 Spreadsheet

This example starts with a blank spreadsheet, embeds the query, and then adds the bells and whistles.

1. Open a blank spreadsheet. From the Data menu, select Get External Data, followed by Run Web Query. The Run Query dialog box opens.
2. Locate the IQY file, webq97.icy, and click the Get Data button. The Run Query dialog box opens.

![Run Query Dialog Box](image)

3. Specify where the data will go in your spreadsheet. You can choose a range in the current spreadsheet or you can choose to put the data into a new worksheet. In this case, specify the range as indicated:

![Returning External Data to Microsoft Excel](image)

There’s no need to set the range in the case of a blank worksheet, however, you may want to output the data in a new worksheet and use formula references to work with the data.
4. Click the *Properties* button in the Returning External Data dialog box. The External Data Range Properties dialog box opens. Name the query and set its behavior. For example, you can set the query to automatically refresh every time the worksheet opens. (Note that you can also set these properties after the initial execution.)

![External Data Range Properties dialog box](image)

Click **OK**.
5. Click the **Parameters** button in the Returning External Data dialog box to define the behavior for any parameters you may have in your query. You can prompt the user, set a static value, or use a value from a cell in the worksheet. In this example, you will prompt the user for the **YEAR**.

Enter the value 1999.

![Parameter Value dialog box](image)

6. After you select the range and set all of the query properties, click the **OK** button.

The query executes and imports the data. The following message is displayed in the spreadsheet:

```
webq97: Getting Data...
```
The data appears in the spreadsheet:
7. Now that the data has been pulled into the spreadsheet, you can add features such as formulas, images, charts, macros, etc.

You can now save this spreadsheet locally on your PC, save it to a shared network drive where multiple users can access the information, or deploy it within your web application. See *Deploying a Spreadsheet With a WebFOCUS Web Query to a Web Server* on page 509.
Embedding a WebFOCUS Report in an Excel 2000 Spreadsheet

Using an optional feature of Excel called Microsoft Web Query, you can embed a WebFOCUS report in a customized Excel spreadsheet that already contains styling, formulas, and macros.

Microsoft Web Query uses a URL to embed external data or an HTML page into the spreadsheet. Since you can easily construct a URL that calls the WebFOCUS engine, you can use this URL as the source of your query. Then, instead of following the typical reporting path in which you execute a request from the browser and display the output in an Excel spreadsheet that conforms to the formatting of the report request, you can execute a request that returns HTML from a query inside an existing Excel spreadsheet whose content and layout you can control in Excel. Said another way, instead of having WebFOCUS push the report into the spreadsheet, you can have Excel pull the report into the spreadsheet. This "pull" technique supports the delivery of real time data directly into your own customized spreadsheets.

The process has two parts:

- Create the WebFOCUS report, if you have not already done so.
- Designate the area in your spreadsheet into which Microsoft Web Query will pull the report, and provide Query with specifications for locating and embedding it.

You must have a web browser installed on your PC to take advantage of this technique.
The following is a customized spreadsheet that was created using this technique. Notice that it includes logos (images), a report, formulas, and an Excel graph. There are two formulas added outside of the query range that sum and average the data brought back by the query. The report is generated as the result of a web query in the area of the spreadsheet that has been designated to contain it. The graph is also based on the results of this query. Each time the query is refreshed, the formulas and the graph update as well.

After you understand the rudiments of this technique, you will be able to adapt it to suit your preferred way of working with Excel. For example, you can:

- Pull a WebFOCUS report into an Excel spreadsheet that you have already customized with logos (images), a report, formulas, or an Excel graph.

- Open a blank spreadsheet, pull in a WebFOCUS report, and then customize the spreadsheet as you like.
You can implement this technique, with some variations, in Excel 2000 and Excel 97. For illustrations, see Embedding a WebFOCUS Report in a Customized Excel 2000 Spreadsheet on page 503 and Embedding a WebFOCUS Report in an Excel 97 Spreadsheet on page 494. For detail on how to set up a web query in Excel 2000 or Excel 97, see your Microsoft Excel documentation.

Note: Microsoft Web Query may not have been installed by default. If you have not already installed it, you will be prompted to install Web Query the first time you attempt to access the feature. Just follow the install instructions provided.

Example: Embedding a WebFOCUS Report in a Customized Excel 2000 Spreadsheet

This example creates the HTML report, then uses Web Query to set up the customized spreadsheet to receive it.

Step 1: Create the WebFOCUS report

Create and save the following request, webq.fex, which you will embed in a customized Excel 2000 spreadsheet: The numbers to the left of each line of code correspond to annotations that follow the request.

1. TABLE FILE CENTORD
2. SUM LINEPRICE AS 'Sales'
3. BY PRODCAT
4. BY PRODNAME
5. ON TABLE PCHOLD FORMAT HTML
6. ON TABLE SET HTMLCSS ON
7. ON TABLE SET BYDISPLAY ON
8. ON TABLE SET PAGE-NUM TOP
9. ON TABLE SET STYLESHEET *
10. TYPE=REPORT, FONT=ARIAL, GRID=OFF, BACKCOLOR=YELLOW, $
11. TYPE=TITLE, STYLE=BOLD, $
12. TYPE=DATA, COLOR=RED, WHEN=LINEPRICE LT 15000000, $
13. ENDSTYLE
14. END

Line 1 identifies the sample data source, CENTORD, which contains sales data for the Century corporation.

Lines 2-4 define display and sorting requirements for the report.

Lines 5-6 define the output format as HTML and turn on Cascading Style Sheets, a feature that increases the efficiency and overall styling capabilities of HTML.

Important: The report must be in HTML format and not EXL2K. This is because Web Query imports fully formatted HTML pages or individual HTML tables into a spreadsheet. Executing an EXL2K request from a query will not return any data to the spreadsheet.
**Line 7** sets BYDISPLAY to ON. This optional setting ensures that repeated sort values in the report are populated with data.

By default, repeated sort values in vertical columns (or BY fields) are suppressed in a WebFOCUS report, leaving blank fields in the report. This behavior is not desirable for Excel reports, which are designed to work with data that is repeated in every row to which it applies. A blank column or row can produce misleading results when sorting data.

**Line 8** turns off page numbering and removes any extra lines generated above the column titles.

**Lines 9-12** specify styling attributes for the report: background color, conditional styling, and some other miscellaneous styling have been designated. Styling, including drill-downs, carry over into Excel.

**Step 2: Designate the area in the spreadsheet into which Web Query will pull the report**

In your spreadsheet, you designate the area where you want your WebFOCUS report to be displayed. You can designate a cell and let Excel determine how much space to allot to the report or you can specify a range of cells to limit the placement. For some techniques that may help you evaluate and control space requirements, see *Tips for Populating Excel Spreadsheets With WebFOCUS Reports* on page 508.

1. Open a spreadsheet and highlight the cell or range.
2. From the Data menu, select *Get External Data*, then select *New Web Query*. 
The New Web Query dialog box opens.

3. In this dialog box:

   a. Specify a URL that points to the WebFOCUS server. The URL needs to be constructed as a call to the WebFOCUS Servlet and contain any appropriate parameters that need to be passed.

   In the illustration, the code

   ```
   http://servername/ibi_apps/WFServlet?IBIF_EX=webq
   ```

   calls the WebFOCUS Servlet and requests the procedure called webq.fex. This is the report created in Step 1.

   Keep in mind that you may also need to pass a user ID and password to get to a secure server. You must add parameters to the query manually. To do this, save the query as an .iqy file and manually add parameters to the URL request. For details, see How to Create a Web Query (IQY) File on page 493.
b. Specify whether the entire page or only the tables should be extracted from the requested file. In this example, select the entire page.

Since WebFOCUS HTML reports are generated using HTML tables, you can select either *The entire page* or *Only the tables*. (At the present time, you cannot identify individual tables in a WebFOCUS HTML report.)

c. Select *Full HTML formatting* to ensure that all styling transfers to the spreadsheet. This step is important if you want a styled report to be carried over to Excel.

d. Click OK.

4. The Returning External Data dialog box opens. Indicate the range of cells into which you want the data to be returned in your spreadsheet.
5. Click the Properties button in the Returning External Data dialog box. The External Data Range Properties dialog box opens.

Name the query and set its refresh behavior. For example, you can set the query to automatically refresh every time the worksheet opens. (Note that you can also set these properties after the initial execution.)

Click OK to run the web query. The report is pulled into the area of the spreadsheet that has been designated to contain it.

6. Save the spreadsheet with the embedded query. (Note that if you click the Save Query button, the query information will be saved as an .iqy file that you can use again with other worksheets.)

7. After the query is created and executed, you can customize your spreadsheet as much as you like.
Note: You only need to run Web Query once to embed a query in a worksheet. After it is executed, the static data stays in the spreadsheet as a placeholder for the query. However, as long as you are connected to your network, the internet, or whatever other connection mechanism is in affect for your application, when you open your local spreadsheet, WebFOCUS delivers the latest information directly into your own customized spreadsheet. You can also click the Refresh Data option at anytime to update the information.

If you wish to delete a query, delete all the query data in the spreadsheet.

For an illustration of a customized spreadsheet that contains the report created in step 1, see Embedding a WebFOCUS Report in an Excel 2000 Spreadsheet on page 501. The spreadsheet includes logos (images), a report, formulas, and an Excel graph. The graph is also based on the results of this query. Each time the query is refreshed, the formulas and the graph update as well.

Reference: Tips for Populating Excel Spreadsheets With WebFOCUS Reports

In addition to the method described in the example Embedding a WebFOCUS Report in an Excel 2000 Spreadsheet on page 501, you can employ other techniques to take advantage of Web Query in your spreadsheet. For example, if you have an existing spreadsheet with a lot of pre-defined elements, you may need to run the query in a separate worksheet and see how much space it will take up before adding it to the main "working area" of the spreadsheet. It is a good idea to designate one particular worksheet for your query and have everything else refer to the data via formulas on another worksheet.

Another approach is to include everything in the same worksheet. Other surrounding elements in the spreadsheet should automatically reposition when the data comes back, but you may want to leave a buffer area around the query to ensure sufficient space for the report output.

Deploying a Spreadsheet With a WebFOCUS Web QUERY

In both Excel 97 and Excel 2000, you can deploy a customized spreadsheet containing a WebFOCUS Web Query to the web server and access it via a hyperlink. This technique enables you to deploy an Excel "template" that is pumped with live data each time it is accessed by a user. It is intended to work with Excel 97 versions and higher.
**Example:** Deploying a Spreadsheet With a WebFOCUS Web Query to a Web Server

You must create a spreadsheet and enable the query to be refreshed when the file opens. Select *Refresh data on file open* under the Refresh Control in the External Data Range Properties dialog box.

To ensure that the query is embedded in the worksheet, execute the Query and save the worksheet before deploying.
To set up the launch page, create a link that calls the spreadsheet. For example,

```html
<a href=http://server/ibi_html/webq97.xls>Excel Sales Report</a>
```

Now, multiple users can benefit from a deployed WebFOCUS/Excel Web Query if running Excel 97 or higher.

**Note:** To take advantage of this feature, Excel (with Query) needs to be installed on the end-user (client) machine.
Creating Excel Templates

How to:
Create Excel Templates

Reference:
Support for Excel Templates

Excel templates enable users to create WebFOCUS reports with any new or existing Excel spreadsheet residing on the WebFOCUS Reporting Server. WebFOCUS can integrate with complex Excel workbooks containing macros, graphs, or Visual Basic applications with the use of Excel templates. Users also have access to more advanced Excel functionality including filters, subtotals, page and print settings, and advanced report styling.

To use an Excel template, open a new or existing Excel workbook and designate one worksheet to be replaced with a WebFOCUS report. The worksheet being replaced must be visible (not hidden). Each worksheet in the workbook must not be empty (populated with at least one blank in one cell). A workbook utilizing Excel templates must be saved as a Single File Web Page (Web Archive) with an extension of .mht and stored in EDAPATH or APP PATH.
After saving, if you do not see the .mht file extension for the Excel template file you created, uncheck the *Hide extensions for known file types* option in Windows Folder Options View tab dialog:

**Reference:** Support for Excel Templates

EXL2K and EXL2K FORMULA are the only formats supported and Excel templates will only work with Excel 2002 and higher versions.

The following are not supported with Excel templates: EXL2K PIVOT, EXL2K BYTOC reports, EXL97, EXCEL, and compound Excel reports.
**Syntax:**

**How to Create Excel Templates**

To create an Excel template, specify the following:

```
ON TABLE PCHOLD FORMAT EXL2K TEMPLATE template_name SHEETNUMBER #
```

where:

- `template_name`
  - The name of the Excel template file (workbook) residing on the WebFOCUS Reporting Server.
- `#`
  - The number of the Excel worksheet being replaced in the template file (workbook).

### Utilizing Excel Named Ranges

**How to:**

Use Excel Named Ranges

**Reference:**

- Rules for Excel Named Ranges
- Support for Excel Named Ranges

An Excel Named Range is a name assigned to a specific group of cells within an Excel worksheet that can be easily referenced by WebFOCUS applications. WebFOCUS StyleSheet language facilitates the generation of Named Ranges.

The use of Excel Named Ranges provides many benefits, including the following:

- Provides advantages over static cell references, including the ability of named range data areas to expand to include new data added during scheduled workbook updates.
- Enables easy setup of Excel worksheets, created by WebFOCUS applications, as an ODBC (Open Database Connectivity) data source.
- Provides accurate, consistent data feeds to advanced Excel worksheet applications, which eliminates manual activities that tend to result in errors.
- Simplifies the process of referencing data in multiple worksheets. This is especially useful when named ranges are added to the output of an Excel Template report.
**Syntax:**  

How to Use Excel Named Ranges

To create Excel Named Ranges, use

\[ \text{TYPE}=\text{type}, \ \text{IN-RANGES}=\text{rangename}, \ \$ \]

where:

- \text{type}
  
  Identifies the WebFOCUS report component to be included in the range. Normally, both of the following are used together:

  - \text{DATA} adds the DATA element of the report to the named range (excludes heading, footing, and column titles).
  - \text{TITLE} adds the TITLE element of the report to the named range (includes all column titles).

- \text{Note:} Multiple elements can be added to the same named range.

- \text{rangename}
  
  Is the name assigned to the output in the Excel workbook your application is creating, and is also the name that will be referenced by other WebFOCUS applications.

**Example:**  

Using Excel Named Ranges

This example creates one report in one worksheet of an Excel workbook. The code specific to Excel Named Ranges appears in bold in the following syntax.

```
TABLE FILE GGSALES
PRINT
    PRODUCT
    DATE
    UNITS
BY REGION
BY DOLLARS
ON TABLE SET PAGE-NUM OFF
ON TABLE SET BYDISPLAY ON
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT EXL2K
ON TABLE SET STYLE *
    UNITS=IN, SQUEEZE=ON, ORIENTATION=PORTRAIT, $
    TYPE=REPORT, FONT='ARIAL', SIZE=9, COLOR='BLACK', BACKCOLOR='NONE',
    STYLE=NORMAL, $
    TYPE=DATA, IN-RANGES='RegionalSales', $
    TYPE=TITLE, STYLE=BOLD, IN-RANGES='RegionalSales', $
ENDSTYLE
END
```
The Excel output is:

<table>
<thead>
<tr>
<th>Region</th>
<th>Dollar Sales</th>
<th>Product</th>
<th>Date</th>
<th>Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midwest</td>
<td>748</td>
<td>Thermos</td>
<td>1996/08/01</td>
<td>68</td>
</tr>
<tr>
<td>Midwest</td>
<td>944</td>
<td>Mug</td>
<td>1997/08/01</td>
<td>86</td>
</tr>
<tr>
<td>Midwest</td>
<td>980</td>
<td>Thermos</td>
<td>1996/07/01</td>
<td>89</td>
</tr>
<tr>
<td>Midwest</td>
<td>1010</td>
<td>Scone</td>
<td>1996/09/01</td>
<td>101</td>
</tr>
<tr>
<td>Midwest</td>
<td>1060</td>
<td>Thermos</td>
<td>1996/02/01</td>
<td>96</td>
</tr>
<tr>
<td>Midwest</td>
<td>1069</td>
<td>Mug</td>
<td>1996/11/01</td>
<td>89</td>
</tr>
<tr>
<td>Midwest</td>
<td>1100</td>
<td>Croissant</td>
<td>1996/11/01</td>
<td>100</td>
</tr>
<tr>
<td>Midwest</td>
<td>1111</td>
<td>Espresso</td>
<td>1997/08/01</td>
<td>101</td>
</tr>
<tr>
<td>Midwest</td>
<td>1111</td>
<td>Latte</td>
<td>1996/11/01</td>
<td>101</td>
</tr>
<tr>
<td>Midwest</td>
<td>1230</td>
<td>Espresso</td>
<td>1996/07/01</td>
<td>123</td>
</tr>
<tr>
<td>Midwest</td>
<td>1247</td>
<td>Coffee Grinder</td>
<td>1996/06/01</td>
<td>125</td>
</tr>
</tbody>
</table>

The name assigned to this Excel Named Range is RegionalSales. If additional rows of data are added, or columns of data are inserted, the named range will stretch to contain both new and existing data.

**Reference: Rules for Excel Named Ranges**

- The Excel data area associated with a named range must be continuous and cannot contain any breaks in the data. Examples of report components containing breaks in the data that cannot be part of a named range include SUBHEAD and SUBFOOT.

- It is recommended that you use ON TABLE SET BYDISPLAY ON. This activates the option to display repeated sort values, which produces continuous output with no breaks in the data.

- Two different worksheets from the same workbook cannot have the same range name.

- When creating Compound Excel reports (multiple TABLE requests output to the same Excel workbook), each report must have a unique range name that is stored at the workbook level.
**Reference: Support for Excel Named Ranges**

Excel Named Ranges are supported for the following Excel formats:

`EXL2K, EXL2K FORMULA, EXL2 TEMPLATE`

Excel Named Ranges are not supported for the following Excel formats:

`EXL2K BYTOC, EXCEL PIVOT`

Excel Named Ranges are not supported with any report syntax that produces discontinuous data or uses columnar references that span multiple columns, which includes the following:

`ACROSSCOLUMN, RECAP, RECOMPUTE, SUBHEAD, SUBFOOT, SUBTOTAL, SUB-TOTAL`

**Creating Excel Tables of Contents Reports**

**How to:**

Use the Excel Table of Contents Feature

**Reference:**

How to Name Worksheets

Limitations of TOC Reports

Excel Table of Contents (TOC) enables you to generate a multiple worksheet report in which a separate worksheet is generated for each value of the first BY field in the FOCUS report.

**Note:** This feature can be used only with EXCEL 2002 or higher releases because it requires the Web Archive file format, which was not available in Excel 2000 and earlier releases.

**Syntax:**

**How to Use the Excel Table of Contents Feature**

The syntax is identical to that of HTML TOC reports, except that only a single BY field is allowed in EXL2K, while multiple BY fields are allowed in HTML.

```
ON TABLE {HOLD|PCHOLD} FORMAT EXL2K BYTOC
```

Since only one level of TOC is allowed for EXL2K reports, the optional number following the BYTOC keyword can only be 1.

As with HTML, the SET COMPOUND syntax, which precedes the TABLE command, may also be used to specify that a TOC be created:

```
SET COMPOUND=BYTOC
```

Since a TOC report is burst into worksheets according to the value of the first BY field in the report, the report must contain at least one BY field. The bursting field may be a NOPRINT field.
**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.

**Reference:** Limitations of TOC Reports

- A TOC report cannot be embedded in a compound report.

- A TOC report cannot be a pivot table report.

**Note:** FORMULA is not supported with bursting.

**Example:** Creating a Simple TOC Report

```plaintext
SET COMPOUND=BYTOC
TABLE FILE CAR
PRINT SALES BY COUNTRY NOPRINT BY CAR
ON TABLE PCHOLD FORMAT EXL2K
ON TABLE SET STYLE *
type=report, style=bold, color=yellow, backcolor=black, $
type=data, backcolor=red, $
type=data, column=car, color=blue, backcolor=yellow, $
END
```

The output is:
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 Web pages, Web sites, CGI or Servlet programs, or non-World Wide Web resources, such as an e-mail 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 1245.

**Topics:**

- Linking Using StyleSheets
- Linking to Another Report
- Linking to a URL
- Linking to a JavaScript Function
- Linking to a WebFOCUS Maintain Procedure
- Creating Multiple Drill-Down Links
- Creating Parameters
- Linking With Conditions
- Linking From a Graphic Image
- Specifying a Base URL
- Specifying a Target Frame
- Creating a Compound Report
- Creating a PDF Compound Report With Drill Through Links
Linking Using StyleSheets

**How to:**
Create Links 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 544.

**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

**How to:**
Determine a WebFOCUS File Name in Managed Reporting

Link to Reports and Procedures

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 544.
Syntax: How to Link to Reports and Procedures

\[
\text{TYPE} = \text{type}, [\text{subtype}], \text{FOCEXEC} = \text{fex}[(\text{parameters} ...)], [\text{TARGET} = \text{frame}], [\text{ALT} = '\text{description}',] \$
\]

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 849.

**fex**
Identifies the file name of the linked procedure to run when you select the report component.

To determine the file name in WebFOCUS, see *How to Determine a WebFOCUS File Name in Managed Reporting* on page 522. The file name in Developer Studio or in deployed self-service applications is the name of the procedure.

The maximum length of a \text{FOCEXEC}=\text{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 807.

**parameters**
Values that are passed to the report, URL, or JavaScript function. For details, see *Creating Parameters* on page 544.

**frame**
Identifies the target frame in the Web page in which the output from the drill-down link is displayed. For details, see *Specifying a Target Frame* on page 562.

**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.
Procedure: How to Determine a WebFOCUS File Name in Managed Reporting

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:

```plaintext
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:

```plaintext
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:

<table>
<thead>
<tr>
<th>Region</th>
<th>State</th>
<th>Store ID</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midwest</td>
<td>IL</td>
<td>R1020</td>
<td>$3,924,401</td>
</tr>
</tbody>
</table>

View Store R1020 Sales By Product

View Store R1020 Sales By Date

<table>
<thead>
<tr>
<th>Region</th>
<th>Store ID</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>MO</td>
<td>R1250</td>
<td>$3,761,286</td>
</tr>
</tbody>
</table>

View Store R1250 Sales By Product

View Store R1250 Sales By Date

<table>
<thead>
<tr>
<th>Region</th>
<th>Store ID</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX</td>
<td>R1019</td>
<td>$3,714,978</td>
</tr>
</tbody>
</table>

View Store R1019 Sales By Product

View Store R1019 Sales By Date
If you click the Sales By Product link for store R1020, the output is:

<table>
<thead>
<tr>
<th>Store ID</th>
<th>Product</th>
<th>Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1020</td>
<td>Biscotti</td>
<td>29413</td>
</tr>
<tr>
<td></td>
<td>Coffee Grinder</td>
<td>19339</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>15785</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>43300</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>32237</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>77344</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>30157</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>45355</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>14651</td>
</tr>
</tbody>
</table>

### Linking to a URL

**How to:**

Link to a URL

You can define a link from any report component to any URL including Web pages, Web sites, CGI or Servlet programs, or non-World Wide Web resources, such as an e-mail 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 544.

**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 849.
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 807.

  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 544.

**frame**

Identifies the target frame in the Web page in which the output from the drill-down link is displayed. For details, see *Specifying a Target Frame* on page 562.

**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 http://www.ibi.com. The relevant StyleSheet declarations are highlighted in the request.
Note that *webserver* indicates the name of the Web server that runs WebFOCUS and Developer Studio.

```
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.*

<table>
<thead>
<tr>
<th>Category</th>
<th>Region</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Midwest</td>
<td>322777</td>
<td>4178513</td>
</tr>
<tr>
<td></td>
<td>Northeast</td>
<td>335778</td>
<td>4164017</td>
</tr>
<tr>
<td></td>
<td>Southeast</td>
<td>350948</td>
<td>4415408</td>
</tr>
<tr>
<td></td>
<td>West</td>
<td>356763</td>
<td>4473517</td>
</tr>
<tr>
<td>Food</td>
<td>Midwest</td>
<td>341414</td>
<td>4338271</td>
</tr>
<tr>
<td></td>
<td>Northeast</td>
<td>353368</td>
<td>4379994</td>
</tr>
<tr>
<td></td>
<td>Southeast</td>
<td>349829</td>
<td>4308731</td>
</tr>
<tr>
<td></td>
<td>West</td>
<td>340234</td>
<td>4202337</td>
</tr>
<tr>
<td>Gifts</td>
<td>Midwest</td>
<td>230854</td>
<td>2883881</td>
</tr>
<tr>
<td></td>
<td>Northeast</td>
<td>227529</td>
<td>2848289</td>
</tr>
<tr>
<td></td>
<td>Southeast</td>
<td>234455</td>
<td>2986240</td>
</tr>
<tr>
<td></td>
<td>West</td>
<td>235042</td>
<td>2977092</td>
</tr>
</tbody>
</table>

When you click the link the site displays in your browser.
Linking to a JavaScript Function

How to: Link 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 544.

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 849 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 544.

**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
<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.

<table>
<thead>
<tr>
<th>Store Code</th>
<th>Product Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1250</td>
<td>F102</td>
</tr>
<tr>
<td></td>
<td>G110</td>
</tr>
<tr>
<td></td>
<td>G121</td>
</tr>
<tr>
<td></td>
<td>F103</td>
</tr>
<tr>
<td></td>
<td>B142</td>
</tr>
<tr>
<td></td>
<td>B141</td>
</tr>
<tr>
<td></td>
<td>G100</td>
</tr>
<tr>
<td></td>
<td>F101</td>
</tr>
<tr>
<td></td>
<td>G104</td>
</tr>
</tbody>
</table>

**Product Description:**

Thermos

**Linking to a WebFOCUS Maintain Procedure**

**How to:**

Link to a WebFOCUS Maintain Procedure

You can provide update capabilities directly from your report by linking it to a WebFOCUS Maintain procedure. The link is a URL for the WebFOCUS CGI and includes the IBIF_cmd command with the MNTCON RUN or MNTCON EX syntax to invoke an existing WebFOCUS Maintain form procedure. The link can pass control to a Maintain form, or run a batch mode Maintain procedure that doesn’t display a user interface.

**Note:** Your reporting procedures can be run from the WebFOCUS Servlet or CGI. All drill down links to Maintain should use a path to the WebFOCUS CGI.
How to Link to a WebFOCUS Maintain Procedure

Syntax:

```
TYPE=type, [subtype, ] URL=/cgi-bin/ibi_cgi/ibiweb.exe? 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 849 for information on identifying report components.

- **procname**: Is the name of the WebFOCUS Maintain procedure.

- **parameters**: Values that are passed to the WebFOCUS Maintain procedure. For details, see Creating Parameters on page 544.

Example: Linking to a WebFOCUS Maintain 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_cgi/ibiweb.exe?(PRODUCT_ID=N1 IBIF_cmd='MNTCON RUN GGUPD1'
    'on' IBIS_passthru='on' IBIS_connect='on'), $
ENDSTYLE
END
```
The WebFOCUS Maintain procedure (ggupd1) is:

```
MAINTAIN FILE ggprods
module import(mntuws FOCCOMP)
$$Declarations
Declare pcode/a4;
Case Top
compute timechk/a0=HHMMSS();
document.referer='/cgi-bin/ibi_cgi/ibiweb.exe?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 Updte1
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 Developer Studio.

The report is:

<table>
<thead>
<tr>
<th>Product Code</th>
<th>Product</th>
<th>Vendor ID</th>
<th>Vendor Name</th>
<th>Unit Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>B141</td>
<td>Hazelnut</td>
<td>V082</td>
<td>Coffee Connection</td>
<td>59.00</td>
</tr>
<tr>
<td>B142</td>
<td>French Roast</td>
<td>V083</td>
<td>European Specialities</td>
<td>81.00</td>
</tr>
<tr>
<td>B144</td>
<td>Kona</td>
<td>V081</td>
<td>Evelina Imports, Ltd</td>
<td>76.00</td>
</tr>
<tr>
<td>F101</td>
<td>Scone</td>
<td>V102</td>
<td>Ridgewood Bakeries</td>
<td>13.00</td>
</tr>
<tr>
<td>F102</td>
<td>Biscotti</td>
<td>V100</td>
<td>Delancey Bakeries</td>
<td>17.00</td>
</tr>
<tr>
<td>F103</td>
<td>Croissant</td>
<td>V103</td>
<td>West Side Bakers</td>
<td>30.00</td>
</tr>
<tr>
<td>G100</td>
<td>Mug</td>
<td>V305</td>
<td>NY Ceramic Supply</td>
<td>26.00</td>
</tr>
<tr>
<td>G104</td>
<td>Thermos</td>
<td>V305</td>
<td>ThermoTech, Inc</td>
<td>97.00</td>
</tr>
<tr>
<td>G110</td>
<td>Coffee Grinder</td>
<td>V303</td>
<td>Appliance Craft</td>
<td>125.00</td>
</tr>
<tr>
<td>G121</td>
<td>Coffee Pot</td>
<td>V304</td>
<td>Appliance Craft</td>
<td>140.00</td>
</tr>
</tbody>
</table>

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 WebFOCUS Maintain procedure ggupd1 opens and you can update the unit price for that product:

<table>
<thead>
<tr>
<th>Product Id</th>
<th>Product Description</th>
<th>Vendor Code</th>
<th>Vendor Name</th>
<th>Unit Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Croissant</td>
<td>V103</td>
<td>West Side Bakers</td>
<td>28.00</td>
</tr>
</tbody>
</table>

**Creating Multiple Drill-Down Links**

In this section:
- Applying Conditional Styling
- Saving a Drill-Down Report to a Server With HTMTABLE

How to:
- Create Multiple Drill-Down Links

Reference:
- Syntax Guidelines
- Summary of Drill-Down Links

You can create multiple drill-down links from a component on a summary report, chosen from any combination of supported actions. For example, you can create links to a detail report or Maintain procedure, a JavaScript function, and a URL. When you click a component in the summary report, a pop-up menu appears listing the drill-down options.

This feature is available for HTML reports and the HTMTABLE format. It applies to data fields in the body of the report, not to headings, subheadings, footings, or subfootings. When the HTMTABLE format is used, special JavaScript coding is required. For more information, see Saving a Drill-Down Report to a Server With HTMTABLE on page 542.
Developer Studio has an interface that automatically generates the code. For details, see the topic on creating a multiple drill-down procedure in the *Creating Reporting Applications With Developer Studio* manual.

**Syntax:**

**How to Create Multiple Drill-Down Links**

```plaintext
TYPE=type, [subtype], DRILLMENUITEM='description'|'DrillDown n',
    type_of_link
```

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 849, for information on identifying report components.

- **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.

- **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 536.

**Reference:**

**Syntax Guidelines**

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_html/javaassist/ibi/html/js/multidrill.css`, where xx is the version and major release number of WebFOCUS.

You can make changes to this file to affect the font, size, and color of the DRILLMENUITEM menu. **Tip:** Make a backup of this file before modifying it.
Reference: Summary of Drill-Down Links

You can link to:

- Another report. The StyleSheet attribute is FOCEXEC. For details on the syntax, see Linking to Another Report on page 520.

- A URL. The StyleSheet attribute is URL. You pass a valid URL. For details on the syntax, see Linking to a URL on page 525.
  
  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.

- 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 525.

- A JavaScript function. The StyleSheet attribute is JAVASCRIPT. For details on the syntax, see Linking to a JavaScript Function on page 528.

- A WebFOCUS Maintain procedure. The StyleSheet attribute is URL with the keyword MNTCON EX. For details on the syntax, see Linking to a WebFOCUS Maintain Procedure on page 531.

- A WebFOCUS compiled Maintain procedure. The StyleSheet attribute is URL with the keyword MNTCON RUN. For details on the syntax, see Linking to a WebFOCUS Maintain Procedure on page 531.

Example: Creating Multiple Drill-Down Links

In this example, the Developer Studio Report Painter was used to create two procedures: a summary report and a detail report. The summary report uses the DRILLMENUITEM and URL attributes to create a link to a Web site; and the DRILLMENUITEM and FOCEXEC attributes to create a link to the detail report, as shown in boldface in the code.

Notice that any StyleSheet syntax that applies to a report component as a whole precedes the first instance of DRILLMENUITEM. For example, COLOR=’BLUE’ and STYLE=UNDERLINE apply to the color and underlining of text in the component, not to a particular drill-down action. For information about editing the menu styles, see Syntax Guidelines on page 535.

For information on creating a multiple drill-down report using Developer Studio, see the Creating Reporting Applications With Developer Studio manual.
Summary Report:

TABLE FILE GGSALES
SUM
   DOLLARS/D12.2CM
   BUDDOLLARS/D12.2CM
BY ST
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
   UNITS=IN,
   PAGESIZE='SCREEN',
   LEFTMARGIN=0.000000,
   RIGHTMARGIN=0.000000,
   TOPMARGIN=0.000000,
   BOTTOMMARGIN=0.000000,
   SQUEEZE=ON,
   ORIENTATION=PORTRAIT,$

TYPE=REPORT,
   GRID=OFF,
   FONT='ARIAL',
   SIZE=9,
   COLOR='BLACK',
   BACKCOLOR='NONE',
   STYLE=NORMAL,
   TOPGAP=0.013889,
   BOTTOMGAP=0.027778,$

TYPE=DATA,
   COLUMN=N1,
   COLOR='BLUE',
   STYLE=UNDERLINE,
   DRILLMENUITEM='DrillDown 1',
      URL=http://www.informationbuilders.com,
   DRILLMENUITEM='DrillDown 2',
      FOCEXEC=DETAILREPORT(PARAMETER=N1),$

TYPE=TITLE,
   STYLE=BOLD,$
Creating Multiple Drill-Down Links

TYPE=TITLE,
   COLUMN=N1,
   COLOR='BLUE',
   STYLE=BOLD+UNDERLINE,
   DRILLMENUITEM='DrillDown 1',
   DRILLMENUITEM='DrillDown 2',
   FOCEXEC=DETAILREPORT(PARAMETER=N1),$

TYPE=TABHEADING,
   SIZE=12,
   STYLE=BOLD,$

TYPE=TABFOOTING,
   SIZE=12,
   STYLE=BOLD,$

TYPE=HEADING,
   SIZE=12,
   STYLE=BOLD,$

TYPE=FOOTING,
   SIZE=12,
   STYLE=BOLD,$

TYPE=SUBHEAD,
   SIZE=10,
   STYLE=BOLD,$

TYPE=SUBFOOT,
   SIZE=10,
   STYLE=BOLD,$

TYPE=SUBLTOTAL,
   BACKCOLOR=RGB(210 210 210),$

TYPE=ACROSSVALUE,
   SIZE=9,$

TYPE=ACROSSTITLE,
   STYLE=BOLD,$

TYPE=GRANDTOTAL,
   BACKCOLOR=RGB(210 210 210),
   STYLE=BOLD,$

ENDSTYLE
END
**Detail Report:**

TABLE FILE GGSALES
PRINT

DOLLARS/D12.2CM
BUDDOLLARS/D12.2CM

BY ST
WHERE (ST EQ '&PARAMETER');

ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *

UNITS=IN,
PAGESIZE='SCREEN',
LEFTMARGIN=0.000000,
RIGHTMARGIN=0.000000,
TOPMARGIN=0.000000,
BOTTOMMARGIN=0.000000,
SQUEEZE=ON,
ORIENTATION=PORTRAIT,$

TYPE=REPORT,
GRID=OFF,
FONT='ARIAL',
SIZE=9,
COLOR='BLACK',
BACKCOLOR='NONE',
STYLE=NORMAL,
TOPGAP=0.013889,
BOTTOMGAP=0.027778,$

TYPE=TITLE,
STYLE=BOLD,$

TYPE=TABHEADING,
SIZE=12,
STYLE=BOLD,$

TYPE=TABFOOTING,
SIZE=12,
STYLE=BOLD,$

TYPE=HEADING,
SIZE=12,
STYLE=BOLD,$

TYPE=FOOTING,
SIZE=12,
STYLE=BOLD,$
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 554.

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 536. 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.

Sample output is:

<table>
<thead>
<tr>
<th>State</th>
<th>Dollar Sales</th>
<th>Budget Dollars</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>$7,642,261.00</td>
<td>$7,586,347.00</td>
</tr>
<tr>
<td>CT</td>
<td>$3,782,049.00</td>
<td>$3,832,202.00</td>
</tr>
<tr>
<td>FL</td>
<td>$3,923,215.00</td>
<td>$3,870,405.00</td>
</tr>
<tr>
<td>GA</td>
<td>$4,100,107.00</td>
<td>$4,247,587.00</td>
</tr>
<tr>
<td>MA</td>
<td>$3,707,986.00</td>
<td>$3,818,397.00</td>
</tr>
<tr>
<td>MO</td>
<td>$3,761,286.00</td>
<td>$3,646,838.00</td>
</tr>
<tr>
<td>NY</td>
<td>$3,902,265.00</td>
<td>$3,926,322.00</td>
</tr>
<tr>
<td>TN</td>
<td>$3,687,057.00</td>
<td>$3,689,979.00</td>
</tr>
<tr>
<td>TX</td>
<td>$3,714,978.00</td>
<td>$3,680,679.00</td>
</tr>
<tr>
<td>WA</td>
<td>$4,010,685.00</td>
<td>$4,055,166.00</td>
</tr>
</tbody>
</table>

`Sample output is:  

<table>
<thead>
<tr>
<th>State</th>
<th>Dollar Sales</th>
<th>Budget Dollars</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>$7,642,261.00</td>
<td>$7,586,347.00</td>
</tr>
<tr>
<td>CT</td>
<td>$3,782,049.00</td>
<td>$3,832,202.00</td>
</tr>
<tr>
<td>FL</td>
<td>$3,923,215.00</td>
<td>$3,870,405.00</td>
</tr>
<tr>
<td>GA</td>
<td>$4,100,107.00</td>
<td>$4,247,587.00</td>
</tr>
<tr>
<td>MA</td>
<td>$3,707,986.00</td>
<td>$3,818,397.00</td>
</tr>
<tr>
<td>MO</td>
<td>$3,761,286.00</td>
<td>$3,646,838.00</td>
</tr>
<tr>
<td>NY</td>
<td>$3,902,265.00</td>
<td>$3,926,322.00</td>
</tr>
<tr>
<td>TN</td>
<td>$3,687,057.00</td>
<td>$3,689,979.00</td>
</tr>
<tr>
<td>TX</td>
<td>$3,714,978.00</td>
<td>$3,680,679.00</td>
</tr>
</tbody>
</table>
| WA    | $4,010,685.00 | $4,055,166.00  |`
Saving a Drill-Down Report to a Server With HTMTABLE

How to:
Save a Drill-Down Report to a Server With HTMTABLE

Instead of displaying report output immediately to your browser, you can save a report to a server and use it to perform additional processing. When you specify the HTMTABLE format to save a drill-down report to a reporting server, JavaScript coding is required. Specific JavaScript code must be included in the output file to identify the server location where the HTMTABLE is saved.

Syntax: How to Save a Drill-Down Report to a Server With HTMTABLE

It is necessary to add JavaScript code to the output file created with the -HTMLFORM command when both the DRILLMENUITEM attribute and the HTMTABLE format are used to create a report containing one or more drill-down links.

In the output file, add the appropriate paths to the following attributes.

```
BASE HREF = http://localhost:port/approot/appname/
var focexurl = dir?var&IBIC_server=reportserv&IBIAPP_app=appname
```

where:

- **localhost**
  - Is the Web Server.
- **port**
  - Is the Web Server port number.
- **approot**
  - Is the hard-coded default value for application root.
- **appname**
  - Is the application name.
- **dir**
  - Is the CGI, ISAPI or WF Servlet relative directory:
    - For CGI use: /cgi-bin/ibi_cgi/ibiweb.exe
    - For ISAPI use: /cgi-bin/ibi_cgi/webapi.dll
    - For WF Servlet use: /ibi_apps/WFServlet
- **?**
  - Is a hard-coded delimiter.
var
   Use the following hard-coded default variable:
   IBIF_webapp=/ibi_apps

&IBIC_server=
   Is the hard-coded variable name placed before the reporting server node.

reportserv
   Is the reporting server node.

&IBIAPP_app=
   Is the hard-coded variable name placed before the application name.

appname
   Is the application name.

&
   Is a hard-coded delimiter.

Example:  Saving a Drill-Down Report to a Server With HTMTABLE

TABLE FILE GGSALES
  .
  .
  .
ON TABLE HOLD FORMAT HTMTABLE AS REPORT1
  .
  .
  .
  TYPE=DATA,
   COLUMN=N1,
   COLOR='BLUE',
   STYLE=UNDERLINE,
   DRILLMENUEM='DrillDown 1',
   URL=http://www.informationbuilders.com?,
   DRILLMENUEM='DrillDown 2',
   FOCEXEC=DETAILREPORT(PARAMETER=N1),$
  .
  .
END
-HTMLFORM output.HTM
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 807.

**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 849.

- `'^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:

```sql
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):

```sql
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:

<table>
<thead>
<tr>
<th>Date of Statement</th>
<th>Region</th>
<th>Projected Annualized Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>06/29/1998</td>
<td>CENTRAL AMERICA</td>
<td>1.360</td>
</tr>
<tr>
<td></td>
<td>EASTERN EUROPE</td>
<td>2.300</td>
</tr>
<tr>
<td></td>
<td>FAR EAST</td>
<td>1.300</td>
</tr>
<tr>
<td></td>
<td>MIDDLE EAST</td>
<td>1.140</td>
</tr>
<tr>
<td></td>
<td>NORTH AMERICA</td>
<td>1.780</td>
</tr>
<tr>
<td></td>
<td>SOUTH AMERICA</td>
<td>1.200</td>
</tr>
<tr>
<td></td>
<td>WESTERN EUROPE</td>
<td>1.140</td>
</tr>
<tr>
<td>06/30/1998</td>
<td>CENTRAL AMERICA</td>
<td>1.360</td>
</tr>
<tr>
<td></td>
<td>EASTERN EUROPE</td>
<td>2.350</td>
</tr>
<tr>
<td></td>
<td>FAR EAST</td>
<td>1.300</td>
</tr>
<tr>
<td></td>
<td>MIDDLE EAST</td>
<td>1.140</td>
</tr>
<tr>
<td></td>
<td>NORTH AMERICA</td>
<td>1.780</td>
</tr>
<tr>
<td></td>
<td>SOUTH AMERICA</td>
<td>1.200</td>
</tr>
<tr>
<td></td>
<td>WESTERN EUROPE</td>
<td>1.140</td>
</tr>
<tr>
<td>07/01/1998</td>
<td>CENTRAL AMERICA</td>
<td>1.360</td>
</tr>
<tr>
<td></td>
<td>EASTERN EUROPE</td>
<td>2.300</td>
</tr>
<tr>
<td></td>
<td>FAR EAST</td>
<td>1.300</td>
</tr>
<tr>
<td></td>
<td>MIDDLE EAST</td>
<td>1.140</td>
</tr>
<tr>
<td></td>
<td>NORTH AMERICA</td>
<td>1.780</td>
</tr>
<tr>
<td></td>
<td>SOUTH AMERICA</td>
<td>1.200</td>
</tr>
<tr>
<td></td>
<td>WESTERN EUROPE</td>
<td>1.140</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Category</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1376266</td>
<td>17231455</td>
</tr>
<tr>
<td>Food</td>
<td>1384845</td>
<td>17229333</td>
</tr>
<tr>
<td>Gifts</td>
<td>927880</td>
<td>11695502</td>
</tr>
</tbody>
</table>

This report was created on 05/11/01.
Creating Parameters and the product and regional breakdown for Coffee displays:

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Midwest Unit Sales</th>
<th>Northeast Unit Sales</th>
<th>Southeast Unit Sales</th>
<th>West Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coffee</td>
<td>.</td>
<td>44785</td>
<td>542095</td>
<td>844000</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>101154</td>
<td>1294947</td>
<td>68127</td>
<td>850107</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>231623</td>
<td>2883566</td>
<td>222866</td>
<td>2771815</td>
</tr>
</tbody>
</table>

**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:

![Prompt](image1)

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*
Creating Parameters

When you click a bar on the graph, the output is:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>MARY</td>
<td>SMITH</td>
</tr>
<tr>
<td></td>
<td>DIANE</td>
<td>JONES</td>
</tr>
<tr>
<td></td>
<td>JOHN</td>
<td>MCCOY</td>
</tr>
<tr>
<td></td>
<td>ROSEMARIE</td>
<td>BLACKWOOD</td>
</tr>
<tr>
<td></td>
<td>MARY</td>
<td>GREENSPAN</td>
</tr>
<tr>
<td></td>
<td>BARBARA</td>
<td>CROSS</td>
</tr>
</tbody>
</table>

**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:

```plaintext
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):

```plaintext
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:

Enter MIS and click Submit. The output is:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>FIRST NAME</th>
<th>LAST NAME</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>MARY</td>
<td>SMITH</td>
<td>$13,200.00</td>
</tr>
</tbody>
</table>

When you click the MIS bar, the output is:
Linking With Conditions

How to:
Link 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 825.

**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 849.

- **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 520.
**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 525.

**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 528.

**parameters**

Values that are passed to the report, URL, or JavaScript function. For details, see *Creating Parameters* on page 544.

**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 Web page in which the output from the drill-down link is displayed. For details, see *Specifying a Target Frame* on page 562.

**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:**

```plaintext
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):

```sql
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:

<table>
<thead>
<tr>
<th>Department</th>
<th>Salaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>$108,002.00</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>$114,282.00</td>
</tr>
</tbody>
</table>

When you click MIS, the following output displays:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>LAST_NAME</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>BLACKWOOD</td>
<td>$21,780.00</td>
</tr>
<tr>
<td></td>
<td>CROSS</td>
<td>$27,062.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$25,755.00</td>
</tr>
<tr>
<td></td>
<td>GREENSPAN</td>
<td>$9,000.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$8,650.00</td>
</tr>
<tr>
<td></td>
<td>JONES</td>
<td>$18,480.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$17,750.00</td>
</tr>
<tr>
<td></td>
<td>MCCOY</td>
<td>$18,480.00</td>
</tr>
<tr>
<td></td>
<td>SMITH</td>
<td>$13,200.00</td>
</tr>
</tbody>
</table>

**Linking From a Graphic Image**

**How to:**

Specify Links 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 849.

- **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 849 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 Web server, such as, /approot/ibinccen/graphic.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 929.

**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 520.

**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 525.

**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 528.

**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 544.

**frame**

Identifies the target frame in the Web page in which the output from the drill-down link is displayed. For details, see *Specifying a Target Frame* on page 562.

**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\WEBFOCUS76\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:

![Century Corp.]

**List Of Employees By Employee ID**

<table>
<thead>
<tr>
<th>EMP_ID</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>071382660</td>
<td>STEVENS</td>
</tr>
<tr>
<td>112847612</td>
<td>SMITH</td>
</tr>
<tr>
<td>117593129</td>
<td>JONES</td>
</tr>
<tr>
<td>119265415</td>
<td>SMITH</td>
</tr>
<tr>
<td>119291444</td>
<td>BANNING</td>
</tr>
<tr>
<td>123764317</td>
<td>IRVING</td>
</tr>
<tr>
<td>126724188</td>
<td>ROMANS</td>
</tr>
<tr>
<td>219984371</td>
<td>MCCOY</td>
</tr>
<tr>
<td>326173357</td>
<td>BLACKWOOD</td>
</tr>
<tr>
<td>451123478</td>
<td>MCKNIGHT</td>
</tr>
<tr>
<td>543729165</td>
<td>GREENSPAN</td>
</tr>
<tr>
<td>818692173</td>
<td>CROSS</td>
</tr>
</tbody>
</table>

When you click the graphic, the output is:

<table>
<thead>
<tr>
<th>DIV</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>$62,500.00</td>
</tr>
<tr>
<td></td>
<td>$54,100.00</td>
</tr>
<tr>
<td></td>
<td>$25,400.00</td>
</tr>
<tr>
<td></td>
<td>$115,000.00</td>
</tr>
<tr>
<td></td>
<td>$33,300.00</td>
</tr>
<tr>
<td></td>
<td>$25,000.00</td>
</tr>
<tr>
<td></td>
<td>$49,000.00</td>
</tr>
<tr>
<td></td>
<td>$40,900.00</td>
</tr>
<tr>
<td></td>
<td>$43,000.00</td>
</tr>
<tr>
<td></td>
<td>$45,000.00</td>
</tr>
<tr>
<td>CORP</td>
<td>$55,500.00</td>
</tr>
<tr>
<td></td>
<td>$83,000.00</td>
</tr>
<tr>
<td></td>
<td>$32,000.00</td>
</tr>
<tr>
<td></td>
<td>$62,500.00</td>
</tr>
<tr>
<td></td>
<td>$79,000.00</td>
</tr>
<tr>
<td></td>
<td>$35,200.00</td>
</tr>
<tr>
<td></td>
<td>$62,500.00</td>
</tr>
<tr>
<td></td>
<td>$26,400.00</td>
</tr>
</tbody>
</table>
Specifying a Base URL

How to:
Specify 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 Web page.

For more details on specifying URLs, see Navigating Within an HTML Report on page 639.

Syntax: How to Specify a Base URL

SET BASEURL=url

where:

url

Is the fully-qualified directory in which additional HTML files, graphics files, and Java applet CLASS files reside. 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://www.informationbuilders.com/ibi_html/newcentcorp/images/

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=leftlogo.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

**How to:**
- Specify a Target Frame
- Specify a Default 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 Web page.

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 Web page 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 Painter, manually added commands in the StyleSheet are not recognized. The Report Painter 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 Web page.

**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 849, 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 520.

**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 525.

**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 544.

**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.
**Note:** When specifying a target frame from the Report Painter, manually added commands in the StyleSheet are not recognized. For example, entering TARGET=NEW is removed when the procedure is run in the Report Painter. The Report Painter removes commands that it does not generate itself. For this example, set the target _BLANK to open a drill down report in a new window.

**Syntax:**  
How to Specify a Default Target Frame

```
SET TARGETFRAME=frame
```

where:

`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.

**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**

In this section:

Creating a Compound Layout Report With Document Composer Syntax
Creating a Compound PDF or PS Report
Creating a Compound Excel Report

Compound reports combine multiple reports into a single file. This enables you to concatenate reports with styled formats (such as PDF, HTML, PS, or EXL2K). 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 604 and *Creating a Compound Excel Report* on page 612.

- **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 Composer Syntax* on page 566.

- **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 590.

For information about creating PDF Compound Reports with Drill Through links, see *Creating a PDF Compound Report With Drill Through Links* on page 623.
Creating a Compound Layout Report With Document Composer Syntax

Typically, you create a compound layout report by using the options in the Document Composer tool. Alternatively, you may create a compound layout report by modifying the syntax in the Text Editor tab of the Document Composer or in any text editor.

**Note:** For details about creating a compound layout report using the Document Composer tool, see *Creating a Compound Layout Report* in the *Creating Compound Reports With Document Composer* manual.

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.

**Tip:** For details about StyleSheet syntax, see *Creating and Managing a WebFOCUS StyleSheet* on page 807.
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 FORMAT format SECTION
   PAGELAYOUT
      COMPONENT
      COMPONENT
      ...
   PAGELAYOUT
      COMPONENT
      COMPONENT
      ...
   ...
END
...
COMPOUND END
```

**Note:** The available compound layout output formats are PDF, DHTML, PowerPoint, AHTML, Excel, Active Flex (FLEX), and Active PDF (APDF). The selected compound layout format will override any report output format from the COMPONENT declaration.

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. For an example of how this syntax appears, see *Creating a Compound Layout Report* in the *Creating Compound Reports With Document Composer* manual.

This topic describes the structure of the compound layout block syntax and some examples.

**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=Style Sheet selections,]
   [ORIENTATION=PORTRAIT|LANDSCAPE,]
   [LEFTMARGIN=m,] [RIGHTMARGIN=m,]
   [TOPMARGIN=m,] [BOTTOMMARGIN=m,] $
```
Creating a Compound Report

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}, $

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.

```
PAGELAYOUT=ALL, $```

**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.

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 + or - 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).

*OVERFLOW-POSITION=(x y)* and *OVERFLOW-DIMENSION=(xsize ysize)*

These optional items specify the position and dimension of the 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 Creating a PDF Compound Report With Drill Through Links on page 623.

**Note:** The double parentheses are required.

**Example:** Creating a Compound Layout Report With Document Composer 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 tab of the Document Composer.

```
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, $
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 $R2$.

```webfocus
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:

![Image of report output]

The table presents sales data for the Midwest region, including state, city, dollar sales, and product details with their respective unit sales.
**Example: Creating a Coordinated Graph With Document Composer 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 tab of the Document Composer.

```
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(getO1MajorGrid(),false);
setTransparentBorderColor(getFrame(),true);
setDisplay(getDataText(),true);
setFontFormatPreset(getDataText(),28);
setFontSizeAbsolute(getDataText(),true);
setFontSizeInPoints(getDataText(),9);
setPlaceResize(getDataText(),0);
setFontStyle(getDataText(),0);
setTransparentBorderColor(getSeries(0),true);
setDisplay(getO1AxisLine(),false);
setFontSizeAbsolute(getO1Label(),true);
setFontSizeInPoints(getO1Label(),9);
setPlaceResize(getO1Label(),0);
setFontSizeAbsolute(getY1Label(),true);
setFontSizeInPoints(getY1Label(),9);
setPlaceResize(getY1Label(),0);
setTextFormatPreset(getY1Label(),28);
setGridStyle(getY1MajorGrid(),3);
setDisplay(getY1AxisLine(),false);
setDisplay(getY1MajorGrid(),false);
setDisplay(getY1Label(),false);
setDisplay(getO1Title(),false);
setTextString(getO1Title(),"")
setTextString(getY1Title(),"")
setFontStyle(getTitle(),0);
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, $
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, $
END
COMPOUND END
**Example:** Creating Multi-Page Layouts With Document Composer 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:

```plaintext
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, $
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, $
END
```

From the Text Editor tab of the Document Composer, 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
```
10. Linking a Report to Other Resources

Page 1 of the output is:

<table>
<thead>
<tr>
<th>State</th>
<th>City</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>IL</td>
<td>Chicago</td>
<td>$3,924,401</td>
</tr>
<tr>
<td>MO</td>
<td>St. Louis</td>
<td>$3,761,286</td>
</tr>
<tr>
<td>TX</td>
<td>Houston</td>
<td>$3,714,978</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product</th>
<th>Number of units sold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscotti</td>
<td>89</td>
</tr>
<tr>
<td>Coffee Grinder</td>
<td>71</td>
</tr>
<tr>
<td>Coffee Pot</td>
<td>72</td>
</tr>
<tr>
<td>Croissant</td>
<td>144</td>
</tr>
<tr>
<td>Espresso</td>
<td>117</td>
</tr>
<tr>
<td>Latte</td>
<td>243</td>
</tr>
<tr>
<td>Mug</td>
<td>144</td>
</tr>
<tr>
<td>Scone</td>
<td>127</td>
</tr>
<tr>
<td>Thermos</td>
<td>72</td>
</tr>
</tbody>
</table>

Page 2 of the output is:

<table>
<thead>
<tr>
<th>State</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>IL</td>
<td>3924401</td>
</tr>
<tr>
<td>MO</td>
<td>3761286</td>
</tr>
<tr>
<td>TX</td>
<td>3714978</td>
</tr>
</tbody>
</table>
Creating a Compound Report

Example: Creating Page Overflow With Document Composer 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 tab of the Document Composer.

```
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 BUDUNITS DOLLARS BUDDOLLARS BY CATEGORY
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, SQUEEZE=ON, $
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, $
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), $
END
Creating a Compound Report

The first page of output is:

Gotham Grinds sales to Information Builders, October 1997

Report 1
Sales Summary by Category

<table>
<thead>
<tr>
<th>Category</th>
<th>Unit Sales</th>
<th>Budget Units</th>
<th>Dollar Sales</th>
<th>Budget Dollars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1376266</td>
<td>1385823</td>
<td>17231455</td>
<td>17293856</td>
</tr>
<tr>
<td>Food</td>
<td>1384845</td>
<td>1377764</td>
<td>17226333</td>
<td>17287160</td>
</tr>
<tr>
<td>Gifts</td>
<td>927680</td>
<td>931007</td>
<td>11695502</td>
<td>11659732</td>
</tr>
</tbody>
</table>

Report 2
Sales Detail Report

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Region</th>
<th>Unit Sales</th>
<th>Budget Units</th>
<th>Dollar Sales</th>
<th>Budget Dollars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>Northeast</td>
<td>44785</td>
<td>44432</td>
<td>542095</td>
<td>561491</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Southeast</td>
<td>73264</td>
<td>75353</td>
<td>944020</td>
<td>956661</td>
</tr>
<tr>
<td></td>
<td></td>
<td>West</td>
<td>71168</td>
<td>70565</td>
<td>895496</td>
<td>877304</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*TOTAL PRODUCT Capuccino</td>
<td></td>
<td></td>
<td>169217</td>
<td>190370</td>
<td>2381590</td>
<td>2395456</td>
</tr>
<tr>
<td>Espresso</td>
<td></td>
<td>Midwest</td>
<td>101154</td>
<td>101869</td>
<td>1294947</td>
<td>1258232</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Northeast</td>
<td>68127</td>
<td>69776</td>
<td>850107</td>
<td>872902</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Southeast</td>
<td>68030</td>
<td>66785</td>
<td>853572</td>
<td>849465</td>
</tr>
<tr>
<td></td>
<td></td>
<td>West</td>
<td>71675</td>
<td>72927</td>
<td>907617</td>
<td>923941</td>
</tr>
<tr>
<td>*TOTAL PRODUCT Espresso</td>
<td></td>
<td></td>
<td>305996</td>
<td>311357</td>
<td>3906243</td>
<td>3904540</td>
</tr>
<tr>
<td>Latte</td>
<td></td>
<td>Midwest</td>
<td>231623</td>
<td>2336557</td>
<td>2853565</td>
<td>2627800</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Northeast</td>
<td>222866</td>
<td>221712</td>
<td>2771815</td>
<td>2618069</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Southeast</td>
<td>209654</td>
<td>213585</td>
<td>2617836</td>
<td>2625303</td>
</tr>
<tr>
<td></td>
<td></td>
<td>West</td>
<td>213920</td>
<td>215272</td>
<td>2670405</td>
<td>2722718</td>
</tr>
<tr>
<td>*TOTAL PRODUCT Latte</td>
<td></td>
<td></td>
<td>878063</td>
<td>884196</td>
<td>10943622</td>
<td>10993980</td>
</tr>
<tr>
<td>*TOTAL CATEGORY Coffee</td>
<td></td>
<td></td>
<td>1376266</td>
<td>1385923</td>
<td>17231455</td>
<td>17293856</td>
</tr>
</tbody>
</table>
How to Draw Objects With Document Composer 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 \((x_1 y_1)\) to point \((x_2 y_2)\), enter the following syntax:

  \[
  \text{OBJECT=LINE, POSITION=(}x_1 y_1\text{), ENDPOINT=(}x_2 y_2\text{), } [\text{BORDER=b,}] [\text{BORDER-COLOR=c,}] [\text{BORDER-STYLE=s,}]$
  \]
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 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, or PPT 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 586.

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=(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 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 + or - 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=(xdim ydim)**

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|1.5LINES|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:

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULTIPLE</td>
<td>The percentage by which to increase or decrease the line space.</td>
<td>LINESPACING=MULTIPLE(1.2) increases line space by 20 percent.</td>
</tr>
<tr>
<td>MIN</td>
<td>The minimum line space (in the unit specified by UNITS parameter) needed to fit the largest font on the line.</td>
<td>LINESPACING=MIN(0.5) provides a minimum line space of 0.5 inch when UNITS=IN.</td>
</tr>
<tr>
<td>EXACT</td>
<td>The fixed line space (in the unit specified by UNITS parameter) that WebFOCUS does not adjust.</td>
<td>LINESPACING=EXACT(.3) provides a fixed line space of 0.3 inch when UNITS=IN.</td>
</tr>
</tbody>
</table>
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.

The following example shows how drawing objects can be placed inside the COMPOUND LAYOUT syntax using the Car 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=Fuel, TYPE=REPORT, POSITION=(6.25 1), DIMENSION=(4 4), $
    OBJECT=IMAGE, IMAGE=iwcpower.gif, POSITION=(1 4.5), DIMENSION=(2 .5), $
    OBJECT=BOX, POSITION=(1 1), DIMENSION=(5 3), BACKCOLOR=LIME, $
END

SET COMPONENT=Sales
TABLE FILE CAR
"Sales report for <COUNTRY"
" "
SUM SALES
BY COUNTRY NOPRINT
BY CAR BY MODEL
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
    TYPE=REPORT, FONT=HELVETICA, COLOR=RED, $
END
```
Creating a Compound Report

```
SET COMPONENT=Fuel
TABLE FILE CAR
"Fuel efficiency report for <COUNTRY"
"
BY COUNTRY NOPRINT
BY HIGHEST MPG
BY CAR BY MODEL
ON TABLE HOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, FONT=HELVETICA, COLOR=BLUE, $
END
COMPOUND END

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 Composer 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:

- **Left Justification**: `<left>text</left>`
- **Right Justification**: `<right>text</right>`
- **Center Justification**: `<center>text</center>`
- **Full Justification**: `<full>text</full>`

**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>
  <li>list item</li>
</ul>
<ul type=circle>
  <li>list item</li>
</ul>
<ul type=square>
  <li>list item</li>
</ul>
```
**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>`

**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 following text. To resolve the space issue, do not insert markup tags in the text object that has the page numbering tags and turn `MARKUP=OFF`.

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:

```
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:

  `<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>`
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.
Compound reports with empty component reports and coordinated compound reports with missing components do not support relative horizontal positioning. 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. For flowing reports where components may be empty, you can anchor the left margin (X axis) rather than make it relative. Define a fixed value (without + or -) to anchor the axis and provide the desired horizontal positioning. For example, POSITION=(0.490 +0.382) anchors the left margin while letting the vertical position flow relative to the previous report.

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:

<table>
<thead>
<tr>
<th>Data File Created</th>
<th>Type of Information Included</th>
<th>Stores included</th>
</tr>
</thead>
<tbody>
<tr>
<td>GGHDR</td>
<td>Store information</td>
<td>R1019, R1020, R1040, R1041</td>
</tr>
<tr>
<td>GG1</td>
<td>Order transactions for Coffee</td>
<td>R1019, R1040, R1088</td>
</tr>
<tr>
<td>GG2</td>
<td>Order transactions for Food</td>
<td>R1019, R1020, R1041, R1088</td>
</tr>
</tbody>
</table>
The APP HOLD, JOIN, and DEFINE commands are:

```sql
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:

```sql
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','R1088');
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
```
Creating a Compound Report

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
"<PRODUCTCATEGORY"
""
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:

<table>
<thead>
<tr>
<th>Coffee</th>
<th>Product</th>
<th>Ordered Units</th>
<th>Unit Price</th>
<th>Package</th>
<th>Size</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1019</td>
<td>French Roast</td>
<td>4627</td>
<td>1.206</td>
<td>Pounds</td>
<td>12</td>
<td>European Specialties,</td>
</tr>
<tr>
<td></td>
<td>Hazelnut</td>
<td>1002</td>
<td>232.00</td>
<td>Pounds</td>
<td>16</td>
<td>Coffee Connection</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Food</th>
<th>Store Code</th>
<th>Product</th>
<th>Units</th>
<th>Package</th>
<th>Size</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1019</td>
<td></td>
<td>Broccoli</td>
<td>3927</td>
<td>Case</td>
<td>24</td>
<td>Delancy Bakers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Croissant</td>
<td>324</td>
<td>Case</td>
<td>20</td>
<td>West Side Bakers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Googe</td>
<td>1004</td>
<td>Case</td>
<td>20</td>
<td>Ridgewood Bakers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gifts</th>
<th>Store Code</th>
<th>Product</th>
<th>Units</th>
<th>Unit Price</th>
<th>Package</th>
<th>Size</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1019</td>
<td></td>
<td>Coffee Grinder</td>
<td>738</td>
<td>500.00</td>
<td>Case</td>
<td>12</td>
<td>Appliance Craft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coffee Pot</td>
<td>902</td>
<td>500.00</td>
<td>Case</td>
<td>8</td>
<td>Appliance Craft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mug</td>
<td>2172</td>
<td>238.00</td>
<td>Case</td>
<td>24</td>
<td>NY Ceramic Supply</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Thermos</td>
<td>510</td>
<td>384.00</td>
<td>Case</td>
<td>10</td>
<td>ThermoTech, Inc</td>
</tr>
</tbody>
</table>
Creating a Compound Report

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:

![Image of a table with data]

### Coffee

<table>
<thead>
<tr>
<th>Store Code</th>
<th>Product</th>
<th>Ordered Units</th>
<th>Unit Price</th>
<th>Package</th>
<th>Size</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1019</td>
<td>French Roast</td>
<td>4827</td>
<td>1,296.00</td>
<td>Pounds</td>
<td>12</td>
<td>European Specialties,</td>
</tr>
<tr>
<td></td>
<td>Hazelnut</td>
<td>1002</td>
<td>232.00</td>
<td>Pounds</td>
<td>16</td>
<td>Coffee Connection</td>
</tr>
</tbody>
</table>

### Food

<table>
<thead>
<tr>
<th>Store Code</th>
<th>Product</th>
<th>Ordered Units</th>
<th>Unit Price</th>
<th>Package</th>
<th>Size</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1019</td>
<td>Biscotti</td>
<td>2057</td>
<td>136.00</td>
<td>Case</td>
<td>24</td>
<td>Delancey Bakeries</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>3244</td>
<td>224.00</td>
<td>Case</td>
<td>20</td>
<td>West Side Bakers</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>1304</td>
<td>52.00</td>
<td>Case</td>
<td>20</td>
<td>Ridgewood Bakeries</td>
</tr>
</tbody>
</table>

### Gifts

<table>
<thead>
<tr>
<th>Store Code</th>
<th>Product</th>
<th>Ordered Units</th>
<th>Unit Price</th>
<th>Package</th>
<th>Size</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1019</td>
<td>Coffee Grinder</td>
<td>738</td>
<td>500.00</td>
<td>Case</td>
<td>12</td>
<td>Appliance Craft</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>902</td>
<td>580.00</td>
<td>Case</td>
<td>8</td>
<td>Appliance Craft</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>2121</td>
<td>238.00</td>
<td>Case</td>
<td>24</td>
<td>NY Ceramic Supply</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>815</td>
<td>384.00</td>
<td>Case</td>
<td>16</td>
<td>ThermoTech, Inc</td>
</tr>
</tbody>
</table>
Creating a Compound Report

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:

```
GOTHAM GRINDS #1020
MARY MULLER
79 RICHMOND LANE
CHICAGO, IL 60601

Food

<table>
<thead>
<tr>
<th>Store Code</th>
<th>Product</th>
<th>Ordered Units</th>
<th>Unit Price</th>
<th>Package</th>
<th>Size</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1020</td>
<td>Biscotti</td>
<td>2475</td>
<td>135.00</td>
<td>Case</td>
<td>24</td>
<td>Delancey Bakeries</td>
</tr>
<tr>
<td></td>
<td>Crossant</td>
<td>2417</td>
<td>224.00</td>
<td>Case</td>
<td>20</td>
<td>West Side Bakers</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>1090</td>
<td>52.00</td>
<td>Case</td>
<td>20</td>
<td>Ridgewood Bakeries</td>
</tr>
</tbody>
</table>

Gifts

<table>
<thead>
<tr>
<th>Store Code</th>
<th>Product</th>
<th>Ordered Units</th>
<th>Unit Price</th>
<th>Package</th>
<th>Size</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1020</td>
<td>Coffee Grinder</td>
<td>700</td>
<td>500.00</td>
<td>Case</td>
<td>12</td>
<td>Appliance Craft</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>663</td>
<td>580.00</td>
<td>Case</td>
<td>8</td>
<td>Appliance Craft</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>1908</td>
<td>238.00</td>
<td>Case</td>
<td>24</td>
<td>NY Ceramic Supply</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>1043</td>
<td>384.00</td>
<td>Case</td>
<td>16</td>
<td>ThermoTech, Inc</td>
</tr>
</tbody>
</table>
```
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:

### Coffee

<table>
<thead>
<tr>
<th>Store Code</th>
<th>Product</th>
<th>Ordered Units</th>
<th>Unit Price</th>
<th>Package</th>
<th>Size</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1088</td>
<td>French Roast</td>
<td>4535</td>
<td>972.00</td>
<td>Pounds</td>
<td>12</td>
<td>European Specialties, Inc.</td>
</tr>
<tr>
<td></td>
<td>Hazelnut</td>
<td>1147</td>
<td>233.00</td>
<td>Pounds</td>
<td>16</td>
<td>Coffee Connection</td>
</tr>
<tr>
<td></td>
<td>Kona</td>
<td>1316</td>
<td>304.00</td>
<td>Pounds</td>
<td>12</td>
<td>Evelina Imports, Ltd</td>
</tr>
</tbody>
</table>

### Food

<table>
<thead>
<tr>
<th>Store Code</th>
<th>Product</th>
<th>Ordered Units</th>
<th>Unit Price</th>
<th>Package</th>
<th>Size</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1088</td>
<td>Biscotti</td>
<td>615</td>
<td>68.00</td>
<td>Case</td>
<td>24</td>
<td>Delancey Bakeries</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>4917</td>
<td>338.00</td>
<td>Case</td>
<td>20</td>
<td>West Side Bakers</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>1200</td>
<td>22.00</td>
<td>Case</td>
<td>20</td>
<td>Ridgewood Bakeries</td>
</tr>
</tbody>
</table>

### Gifts

<table>
<thead>
<tr>
<th>Store Code</th>
<th>Product</th>
<th>Ordered Units</th>
<th>Unit Price</th>
<th>Package</th>
<th>Size</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1088</td>
<td>Coffee Grinder</td>
<td>648</td>
<td>500.00</td>
<td>Case</td>
<td>12</td>
<td>Appliance Craft</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>1354</td>
<td>560.00</td>
<td>Case</td>
<td>8</td>
<td>Appliance Craft</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>1281</td>
<td>289.00</td>
<td>Case</td>
<td>24</td>
<td>NY Ceramic Supply</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>915</td>
<td>384.00</td>
<td>Case</td>
<td>16</td>
<td>ThermoTech, Inc.</td>
</tr>
</tbody>
</table>
Creating a Compound PDF or PS Report

**How to:**

Display Compound Reports

Embed Graphics in a Compound 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 Acrobat Reader or sends the compound PS report directly to a printer. See the ReportCaster documentation for details about this product.

For information about creating Drill Through PDF Compound Reports, see *Creating a PDF Compound Report With Drill Through Links* on page 623. For information about creating Excel Compound Reports, see *Creating a Compound Excel Report* on page 612.

**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 351.
- 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 specifies 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 607.
Creating a Compound Report

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**

<table>
<thead>
<tr>
<th>PRODCAT</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD Players</td>
<td>$40,694,863.53</td>
</tr>
<tr>
<td>Camcorders</td>
<td>$333,884,927.52</td>
</tr>
<tr>
<td>Cameras</td>
<td>$9,718,752.55</td>
</tr>
<tr>
<td>DVD</td>
<td>$53,182,674.89</td>
</tr>
<tr>
<td>Digital Tape Recorders</td>
<td>$38,773,229.69</td>
</tr>
<tr>
<td>PDA Devices</td>
<td>$233,967,566.49</td>
</tr>
<tr>
<td>VCRs</td>
<td>$23,801,009.34</td>
</tr>
</tbody>
</table>

**Inventory Report**

<table>
<thead>
<tr>
<th>PRODCAT</th>
<th>Quantity:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD Players</td>
<td>308,274</td>
</tr>
<tr>
<td>Camcorders</td>
<td>950,700</td>
</tr>
<tr>
<td>Cameras</td>
<td>98,141</td>
</tr>
<tr>
<td>DVD</td>
<td>241,287</td>
</tr>
<tr>
<td>Digital Tape Recorders</td>
<td>524,027</td>
</tr>
<tr>
<td>PDA Devices</td>
<td>610,844</td>
</tr>
<tr>
<td>VCRs</td>
<td>170,977</td>
</tr>
</tbody>
</table>

**Cost of Goods Sold Report**

<table>
<thead>
<tr>
<th>PRODCAT</th>
<th>Line Cost of Goods Sold</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD Players</td>
<td>30,519,126.00</td>
</tr>
<tr>
<td>Camcorders</td>
<td>299,487,742.00</td>
</tr>
<tr>
<td>Cameras</td>
<td>8,944,891.00</td>
</tr>
<tr>
<td>DVD</td>
<td>47,275,593.00</td>
</tr>
<tr>
<td>Digital Tape Recorders</td>
<td>36,157,863.00</td>
</tr>
<tr>
<td>PDA Devices</td>
<td>202,123,656.00</td>
</tr>
<tr>
<td>VCRs</td>
<td>22,056,033.00</td>
</tr>
</tbody>
</table>

**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 1245.
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 1016.

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
Creating a Compound Report

**Report 3:**

```
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
```

10. Linking a Report to Other Resources

The output is:

**Investment Report**

<table>
<thead>
<tr>
<th>Continent</th>
<th>Instrument Holder</th>
<th>Return on Investment</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMERICAS</td>
<td>COMM</td>
<td>14.98%</td>
</tr>
<tr>
<td></td>
<td>GOVT</td>
<td>15.40%</td>
</tr>
<tr>
<td>ASIA</td>
<td>COMM</td>
<td>8.43%</td>
</tr>
<tr>
<td></td>
<td>GOVT</td>
<td>8.85%</td>
</tr>
<tr>
<td>EUROPE</td>
<td>COMM</td>
<td>9.82%</td>
</tr>
<tr>
<td></td>
<td>GOVT</td>
<td>14.28%</td>
</tr>
</tbody>
</table>

**Trades Report**

<table>
<thead>
<tr>
<th>Continent</th>
<th>Region</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMERICAS</td>
<td>CENTRAL AMERICA</td>
<td>451,333,300</td>
</tr>
<tr>
<td></td>
<td>NORTH AMERICA</td>
<td>1,370,452,600</td>
</tr>
<tr>
<td>ASIA</td>
<td>SOUTH AMERICA</td>
<td>454,354,100</td>
</tr>
<tr>
<td></td>
<td>FAR EAST</td>
<td>364,381,400</td>
</tr>
<tr>
<td></td>
<td>MIDEAST</td>
<td>229,458,500</td>
</tr>
<tr>
<td>EUROPE</td>
<td>EAST EUROPE</td>
<td>456,090,610</td>
</tr>
<tr>
<td></td>
<td>WEST EUROPE</td>
<td>1,042,659,900</td>
</tr>
</tbody>
</table>

**Balance by Region**

<table>
<thead>
<tr>
<th>Continent</th>
<th>Region</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMERICAS</td>
<td>CENTRAL AMERICA</td>
<td>40,931,317</td>
</tr>
<tr>
<td></td>
<td>NORTH AMERICA</td>
<td>267,810,693</td>
</tr>
<tr>
<td>ASIA</td>
<td>SOUTH AMERICA</td>
<td>3,040,321</td>
</tr>
<tr>
<td></td>
<td>FAR EAST</td>
<td>272,990,802</td>
</tr>
<tr>
<td></td>
<td>MIDEAST</td>
<td>44,703,771</td>
</tr>
<tr>
<td>EUROPE</td>
<td>EASTERN EUROPE</td>
<td>121,977,397</td>
</tr>
<tr>
<td></td>
<td>WESTERN EUROPE</td>
<td>210,580,237</td>
</tr>
</tbody>
</table>
Creating a Compound Excel Report

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

Guidelines for Producing Excel Compound Reports

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

- **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 HOLD 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 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 FORMAT EXL2K CLOSE
END
```
The output for each tab in the Excel worksheet is:

### Sales Report

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNTRY</td>
<td>SALES</td>
<td></td>
</tr>
<tr>
<td>ENGLAND</td>
<td>12000</td>
<td></td>
</tr>
<tr>
<td>FRANCE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ITALY</td>
<td>30200</td>
<td></td>
</tr>
<tr>
<td>JAPAN</td>
<td>78030</td>
<td></td>
</tr>
<tr>
<td>WEST GERMANY</td>
<td>88190</td>
<td></td>
</tr>
</tbody>
</table>

### Inventory Report

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNTRY</td>
<td>RETAIL_COST</td>
<td></td>
</tr>
<tr>
<td>ENGLAND</td>
<td>45,319</td>
<td></td>
</tr>
<tr>
<td>FRANCE</td>
<td>5,610</td>
<td></td>
</tr>
<tr>
<td>ITALY</td>
<td>51,066</td>
<td></td>
</tr>
<tr>
<td>JAPAN</td>
<td>5,478</td>
<td></td>
</tr>
<tr>
<td>WEST GERMANY</td>
<td>64,732</td>
<td></td>
</tr>
</tbody>
</table>
Creating a Compound Report

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 HOLD 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
Creating a Compound Report

The output for each tab in the Excel worksheet is:

### Sales Report

<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>RETAIL_COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGLAND</td>
<td>8,678</td>
</tr>
<tr>
<td></td>
<td>13,491</td>
</tr>
<tr>
<td></td>
<td>17,850</td>
</tr>
<tr>
<td></td>
<td>5,100</td>
</tr>
<tr>
<td>FRANCE</td>
<td>5,610</td>
</tr>
<tr>
<td>ITALY</td>
<td>5,925</td>
</tr>
<tr>
<td></td>
<td>6,820</td>
</tr>
<tr>
<td></td>
<td>6,820</td>
</tr>
<tr>
<td></td>
<td>31,800</td>
</tr>
<tr>
<td>JAPAN</td>
<td>3,139</td>
</tr>
<tr>
<td></td>
<td>3,339</td>
</tr>
<tr>
<td>W GERMANY</td>
<td>5,570</td>
</tr>
<tr>
<td></td>
<td>5,940</td>
</tr>
<tr>
<td></td>
<td>6,355</td>
</tr>
<tr>
<td></td>
<td>13,752</td>
</tr>
<tr>
<td></td>
<td>14,123</td>
</tr>
<tr>
<td></td>
<td>9,087</td>
</tr>
<tr>
<td></td>
<td>9,486</td>
</tr>
</tbody>
</table>

### Inventory Report

<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGLAND</td>
<td>12000</td>
</tr>
<tr>
<td>FRANCE</td>
<td>0</td>
</tr>
<tr>
<td>ITALY</td>
<td>30200</td>
</tr>
<tr>
<td>JAPAN</td>
<td>76000</td>
</tr>
<tr>
<td>W GERMANY</td>
<td>88190</td>
</tr>
</tbody>
</table>

-造者: WebFOCUS
10. Linking a Report to Other Resources

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNTRY</td>
<td>CAR</td>
<td>MODEL</td>
<td>BODYTYPE</td>
<td>SEATS</td>
<td>RETAIL_COST</td>
<td></td>
</tr>
<tr>
<td>ENGLAND</td>
<td>JAGUAR</td>
<td>XJ12L AUTO</td>
<td>CONVERTIBLE</td>
<td>2</td>
<td>8,878</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>TOTAL SEATS 2</td>
<td>8,878</td>
<td>3</td>
<td>TOTAL BODYTYPE CONVERTIBLE</td>
<td>8,878</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>TOTAL MODEL XJ12L AUTO</td>
<td>8,878</td>
<td>5</td>
<td>TOTAL CAR JAGUAR</td>
<td>22,369</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>JENSEN</td>
<td>INTERCEPTOR III</td>
<td>SEDAN</td>
<td>4</td>
<td>17,839</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>TOTAL SEATS 4</td>
<td>17,839</td>
<td>13</td>
<td>TOTAL BODYTYPE SEDAN</td>
<td>17,839</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>TOTAL MODEL INTERCEPTOR III</td>
<td>17,839</td>
<td>16</td>
<td>TOTAL CAR JENSEN</td>
<td>17,839</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>TRIO</td>
<td>TR7</td>
<td>HARDTOP</td>
<td>2</td>
<td>5,100</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>TOTAL SEATS 2</td>
<td>5,100</td>
<td>19</td>
<td>TOTAL BODYTYPE HARDTOP</td>
<td>5,100</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>TOTAL MODEL TR7</td>
<td>5,100</td>
<td>21</td>
<td>TOTAL COUNTRY ENGLAND</td>
<td>45,319</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>FRANCE</td>
<td>PEUGEOT</td>
<td>504 4 DOOR</td>
<td>SEDAN</td>
<td>5</td>
<td>5,810</td>
</tr>
<tr>
<td>24</td>
<td>TOTAL SEATS 5</td>
<td>5,810</td>
<td>25</td>
<td>TOTAL BODYTYPE SEDAN</td>
<td>5,810</td>
<td></td>
</tr>
</tbody>
</table>

Cost of Goods Sold Report

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>COST OF GOODS SOLD REPORT</td>
<td></td>
</tr>
<tr>
<td>Sum of DEALER_COST</td>
<td></td>
</tr>
<tr>
<td>COUNTRY</td>
<td>Total</td>
</tr>
<tr>
<td>ENGLAND</td>
<td>37,853</td>
</tr>
<tr>
<td>FRANCE</td>
<td>4,831</td>
</tr>
<tr>
<td>ITALY</td>
<td>41,236</td>
</tr>
<tr>
<td>JAPAN</td>
<td>5,512</td>
</tr>
<tr>
<td>W GERMANY</td>
<td>54,863</td>
</tr>
</tbody>
</table>

Creating Reports With WebFOCUS Language 619
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 CAR
HEADING
"Report 1: England"
PRINT DCOST BY COUNTRY BY CAR
IF COUNTRY EQ ENGLAND
ON TABLE HOLD FORMAT EXL2K OPEN NOBREAK
ON TABLE SET STYLE *
  type=report, color=red, $
  type=data, backcolor=yellow, $
  type=heading, color=blue, $
END

TABLE FILE CAR
HEADING
  
  
  
"Report 2: France"
PRINT RCOST BY COUNTRY
IF COUNTRY EQ FRANCE
ACROSS SEATS
ON TABLE PCHOLD FORMAT EXL2K
ON TABLE SET STYLE *
  type=report, color=lime, backcolor=fuschia, style=bold, $
  type=title, color=black, style=bold+italic, $
  type=heading, backcolor=black, $
END

TABLE FILE CAR
FOOTING
"Report 3 - All"
PRINT SALES BY COUNTRY BY CAR
ON TABLE PCHOLD FORMAT EXL2K NOBREAK
ON TABLE SET STYLE *
  type=report, backcolor=yellow, style=bold, $
  type=title, color=blue, $
  type=footing, backcolor=aqua, $
END
TABLE FILE CAR
HEADING
" "
" "
"Report 4"
PRINT SALES BY COUNTRY BY CAR
ON TABLE PCHOLD FORMAT EXL2K CLOSE
ON TABLE SET STYLE *
type=report, color=yellow, backcolor=black, style=bold, $
END

The output is:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>A1</strong></td>
<td><strong>Report 1: England</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Report 1: England</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUNTRY</td>
<td>CAR</td>
<td>DEALER_COST</td>
<td></td>
</tr>
<tr>
<td>ENGLAND</td>
<td>JAGUAR</td>
<td>7,427</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>11,194</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JENSEN</td>
<td>14,940</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRIUMPH</td>
<td>4,292</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>A1</strong></td>
<td><strong>Report 2: France</strong></td>
</tr>
<tr>
<td><strong>Report 2: France</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUNTRY</td>
<td>SEATS</td>
<td></td>
</tr>
<tr>
<td>ENGLAND</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FRANCE</td>
<td>3,610</td>
</tr>
</tbody>
</table>
## Creating a Compound Report

<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>CAR</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGLAND</td>
<td>JAGUAR</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>JENSEN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>TRIUMPH</td>
<td>0</td>
</tr>
<tr>
<td>FRANCE</td>
<td>PEUGEOT</td>
<td>0</td>
</tr>
<tr>
<td>ITALY</td>
<td>ALFA ROMEO</td>
<td>4800</td>
</tr>
<tr>
<td></td>
<td>MASEIRATI</td>
<td>0</td>
</tr>
<tr>
<td>JAPAN</td>
<td>DATSUN</td>
<td>43000</td>
</tr>
<tr>
<td></td>
<td>TOYOTA</td>
<td>35030</td>
</tr>
<tr>
<td>W GERMANY</td>
<td>AUDI</td>
<td>7800</td>
</tr>
<tr>
<td></td>
<td>BMW</td>
<td>8950</td>
</tr>
<tr>
<td></td>
<td>8900</td>
<td>14000</td>
</tr>
<tr>
<td></td>
<td>18940</td>
<td>14000</td>
</tr>
<tr>
<td></td>
<td>15600</td>
<td></td>
</tr>
</tbody>
</table>

---

### Report 3 - All

<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>CAR</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGLAND</td>
<td>JAGUAR</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>JENSEN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>TRIUMPH</td>
<td>0</td>
</tr>
<tr>
<td>FRANCE</td>
<td>PEUGEOT</td>
<td>0</td>
</tr>
<tr>
<td>ITALY</td>
<td>ALFA ROMEO</td>
<td>4800</td>
</tr>
<tr>
<td></td>
<td>MASEIRATI</td>
<td>0</td>
</tr>
<tr>
<td>JAPAN</td>
<td>DATSUN</td>
<td>43000</td>
</tr>
<tr>
<td></td>
<td>TOYOTA</td>
<td>35030</td>
</tr>
<tr>
<td>W GERMANY</td>
<td>AUDI</td>
<td>7800</td>
</tr>
<tr>
<td></td>
<td>BMW</td>
<td>8950</td>
</tr>
<tr>
<td></td>
<td>8900</td>
<td>14000</td>
</tr>
<tr>
<td></td>
<td>18940</td>
<td>14000</td>
</tr>
<tr>
<td></td>
<td>15600</td>
<td></td>
</tr>
</tbody>
</table>
Creating a PDF Compound Report With Drill Through Links

In this section:
Sample Drill Through PDF Compound Reports

How to:
Create a Drill Through in a PDF Compound Layout Report
Create a Drill Through in a PDF Compound Report
Specify Drill Through Hyperlinks
Specify Which Compound Layout Reports Will be Related Through Hyperlinks

Reference:
Drill Through and Drill Down Compared
Use With Other Features
Usage Notes for Drill Through

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 Composer 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.
Creating a PDF Compound Report With Drill Through Links

**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 Acrobat 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.

\[ \text{TYPE=}DATA, \ \text{COLUMN=}PRODUCT, \ \text{DRILLTHROUGH=}DOWN(CATEGORY \ \text{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.

\[ \text{DRILLMAP=}((L1 \ \text{targetreport})) \]

where:

- \( L1 \)
  
  Is the link identifier.

- \( \text{targetreport} \)
  
  Is the component name of hyperlink destination.

**Note:** The double parentheses around the DRILLMAP values are required.
Sample Component Declarations With DRILLMAP Attributes

Example:
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

- 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

Reference:

Guidelines on Links For FIRST
Rules For Drill Through Hyperlinks

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:

```plaintext
TABLE FILE GGSALES
SUM UNITS DOLLARS BY CATEGORY BY PRODUCT
ON TABLE PCHOLD FORMAT PDF
END
```

The output is:

```
<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>
```
Example:  Creating the Detail Report (Step 2)

The following syntax generates a sample detail report:

The first page of the output is:

```
PAGE 1

Category: Coffee

<table>
<thead>
<tr>
<th>Region</th>
<th>City</th>
<th>Unit Sales</th>
<th>Budget Units</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>**** Product: Capuccino</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Northeast</td>
<td>Boston</td>
<td>15358</td>
<td>15672</td>
<td>174344</td>
</tr>
<tr>
<td></td>
<td>New Haven</td>
<td>12386</td>
<td>11098</td>
<td>158995</td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td>17041</td>
<td>17662</td>
<td>208756</td>
</tr>
<tr>
<td>Southeast</td>
<td>Atlanta</td>
<td>27522</td>
<td>29171</td>
<td>352161</td>
</tr>
<tr>
<td></td>
<td>Memphis</td>
<td>21599</td>
<td>23090</td>
<td>274812</td>
</tr>
<tr>
<td></td>
<td>Orlando</td>
<td>24143</td>
<td>23092</td>
<td>317027</td>
</tr>
<tr>
<td>West</td>
<td>Los Angeles</td>
<td>23222</td>
<td>22909</td>
<td>306468</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>23311</td>
<td>23555</td>
<td>279830</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>24635</td>
<td>24121</td>
<td>309197</td>
</tr>
</tbody>
</table>
```

**** 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. 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, $
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:

```plaintext
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, $
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:

```sql
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, $
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:

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Region</th>
<th>City</th>
<th>Unit Sales</th>
<th>Budget Units</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midwest</td>
<td>Chicago</td>
<td>43300</td>
<td>43271</td>
<td>549366</td>
</tr>
<tr>
<td></td>
<td>Houston</td>
<td>46941</td>
<td>47050</td>
<td>587867</td>
</tr>
<tr>
<td></td>
<td>St. Louis</td>
<td>48941</td>
<td>49327</td>
<td>613871</td>
</tr>
<tr>
<td>Northeast</td>
<td>Boston</td>
<td>41029</td>
<td>41351</td>
<td>497234</td>
</tr>
<tr>
<td></td>
<td>New Haven</td>
<td>45847</td>
<td>46335</td>
<td>551469</td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td>50518</td>
<td>50178</td>
<td>622095</td>
</tr>
<tr>
<td>Southeast</td>
<td>Atlanta</td>
<td>53782</td>
<td>54126</td>
<td>661806</td>
</tr>
<tr>
<td></td>
<td>Memphis</td>
<td>52499</td>
<td>51585</td>
<td>638477</td>
</tr>
<tr>
<td></td>
<td>Orlando</td>
<td>50175</td>
<td>51437</td>
<td>602076</td>
</tr>
<tr>
<td>West</td>
<td>Los Angeles</td>
<td>66049</td>
<td>64432</td>
<td>800084</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>65214</td>
<td>66025</td>
<td>824457</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>65759</td>
<td>64872</td>
<td>801060</td>
</tr>
</tbody>
</table>

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 in 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.
Topics:

- Navigating Sort Groups From a Table of Contents
- Adding an HTML TOC to a Report
- Creating an Active Report
- Controlling the Display of Sorted Data With Accordion Reports
- Navigating a Multi-Page Report With the WebFOCUS Viewer
- Linking Report Pages

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 Web pages 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 657.
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 645.

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 a TOC to a Heading on page 652.

- **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 to a Report With a SET Command on page 641.
Adding an HTML TOC to a Report

You can use three different types of syntax to add an HTML TOC object to a report.

**Syntax:**  
**How to Add a TOC to a Report With 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 are required when you use a SET command to specify the number of sort fields you want to include in the TOC. Single quotation marks are not required when BYTOC is specified without a number.
How to Add a TOC to a Report With a 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.

Single quotation marks are required when you use a SET command to specify the number of sort fields you want to include in the TOC. Single quotation marks are not required when BYTOC is specified without a number or when it is specified in a PCHOLD command, with or without a number.

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

TABLE FILE CENTORD
HEADING
"SALES REPORT"
SUM LINEPRICE BY PLANT BY PRODCAT
ON TABLE SET PAGE-Num OFF
ON TABLE NOTOTAL
ON TABLE SET ONLINE-FMT 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.

```plaintext
TABLE FILE CENTORD
HEADING
"SALES REPORT"
SUM LINEPRICE BY PLANT BY PRODCAT
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET COMPOUND BYTOC
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:

```plaintext
TABLE FILE CENTORD
HEADING
"SALES REPORT"
SUM LINEPRICE BY PLANT BY PRODCAT
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE SET ONLINE-FMT HTML
ON TABLE PCHOLD FORMAT HTML BYTOC
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```
Run the report. The TOC object displays in the upper left corner.

**SALES REPORT**

<table>
<thead>
<tr>
<th>Manufacturing Plant</th>
<th>PRODCAT</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS</td>
<td>CD Players</td>
<td>$14,327,431.45</td>
</tr>
<tr>
<td></td>
<td>Camcorders</td>
<td>$119,144,483.32</td>
</tr>
<tr>
<td></td>
<td>Cameras</td>
<td>$4,142,193.13</td>
</tr>
<tr>
<td></td>
<td>DVD</td>
<td>$16,480,170.24</td>
</tr>
<tr>
<td></td>
<td>Digital Tape Recorders</td>
<td>$13,868,282.30</td>
</tr>
<tr>
<td></td>
<td>PDA Devices</td>
<td>$85,764,603.13</td>
</tr>
<tr>
<td></td>
<td>VCRs</td>
<td>$8,706,797.06</td>
</tr>
</tbody>
</table>

Double-click the icon to see the Table of Contents expand with the list of associated sort values below it: Boston, Dallas, Los Angeles, Orlando, Seattle, St. Louis.

**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 option.

**Tip:** You can also customize the look and feel of the TOC object by editing a .css file. Do not forget to make a backup copy prior to editing.

- If you are working in the Developer Studio local projects go to `\ibi\DevStudio##\ibi_html\javaassist\int1\xx` directory, where ## is the release of Developer Studio and xx is the language abbreviation for the version of Developer Studio.
If you are working in a self service or Managed Reporting environment, in Developer Studio or from a browser, go to the `\ibi\WebFOCUS##\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.

### 2002

<table>
<thead>
<tr>
<th>PLANTING</th>
<th>PRODCAT</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boston</td>
<td>CD Players</td>
<td>$29,892,829.47</td>
</tr>
<tr>
<td></td>
<td>Camcorders</td>
<td>$207,081,265.92</td>
</tr>
<tr>
<td></td>
<td>Cameras</td>
<td>$10,514,335.23</td>
</tr>
<tr>
<td></td>
<td>DVD</td>
<td>$35,609,706.25</td>
</tr>
<tr>
<td></td>
<td>Digital Tape Recorders</td>
<td>$30,242,087.41</td>
</tr>
<tr>
<td></td>
<td>PDA Devices</td>
<td>$108,214,581.22</td>
</tr>
<tr>
<td></td>
<td>VCRs</td>
<td>$22,107,937.71</td>
</tr>
</tbody>
</table>

### 2002

<table>
<thead>
<tr>
<th>PLANTING</th>
<th>PRODCAT</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dallas</td>
<td>CD Players</td>
<td>$6,887,222.95</td>
</tr>
<tr>
<td></td>
<td>Camcorders</td>
<td>$33,959,217.21</td>
</tr>
<tr>
<td></td>
<td>Cameras</td>
<td>$3,896,023.97</td>
</tr>
<tr>
<td></td>
<td>DVD</td>
<td>$16,693,051.86</td>
</tr>
<tr>
<td></td>
<td>Digital Tape Recorders</td>
<td>$11,317,892.62</td>
</tr>
<tr>
<td></td>
<td>PDA Devices</td>
<td>$67,742,440.39</td>
</tr>
</tbody>
</table>

**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.
Navigation Behavior in a Multi-Level TOC

**How to:**
Add a TOC to a Heading

**Reference:**
Navigating Between Reports
HTML Table of Contents Limits

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.

<table>
<thead>
<tr>
<th>Continent</th>
<th>Region</th>
<th>Country</th>
<th>Instrument Holder</th>
<th>Type of Instrument</th>
<th>Projected Annualized Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMERICAS</td>
<td>CENTRAL AMERICA</td>
<td>GUATEMALA</td>
<td>COMM</td>
<td>OVERNIGHT</td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.150</td>
</tr>
</tbody>
</table>

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.

![Image of a table with columns labeled 'BRAZIL', 'COMM', 'OVERNIGHT', 'ST. NOTES', 'GOVT', 'CASH', and 'OVERNIGHT'. The table contains data entries.]

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 a TOC 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:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABHEADING</td>
<td>Report heading.</td>
</tr>
<tr>
<td>TABFOOTING</td>
<td>Report footing.</td>
</tr>
<tr>
<td>HEADING</td>
<td>Page heading.</td>
</tr>
<tr>
<td>FOOTING</td>
<td>Page footing.</td>
</tr>
</tbody>
</table>

- **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:

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 an HTML TOC 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 PLANT, identified in the StyleSheet code as OBJECT=FIELD, which is the first item of that type in the declaration.

```
TABLE FILE CENTORD
HEADING
"Sales Report: PLANT"
SUM LINEPRICE BY PLANT BY PRODCAT
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, $
TYPE=HEADING, LINE=1, JUSTIFY=LEFT, $
TYPE=HEADING, LINE=1, OBJECT=FIELD, ITEM=1, TOC=1, $
ENDSTYLE
END
```
Run the report. The TOC appears as a drop-down menu in the heading, in place of the field PLANT:

<table>
<thead>
<tr>
<th>Manufacturing Plant</th>
<th>PRODCAT</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS</td>
<td>CD Players</td>
<td>$14,327,431.45</td>
</tr>
<tr>
<td></td>
<td>Camcorders</td>
<td>$119,144,483.32</td>
</tr>
<tr>
<td></td>
<td>Cameras</td>
<td>$4,142,193.13</td>
</tr>
<tr>
<td></td>
<td>DVD</td>
<td>$16,480,170.24</td>
</tr>
<tr>
<td></td>
<td>Digital Tape Recorders</td>
<td>$13,868,282.30</td>
</tr>
<tr>
<td></td>
<td>PDA Devices</td>
<td>$85,764,603.13</td>
</tr>
<tr>
<td></td>
<td>VCRs</td>
<td>$8,706,797.06</td>
</tr>
</tbody>
</table>

Click the TOC to see the list of sort values for PLANT: BOS, DAL, LA, ORL, SEA, STL.

Click each plant (city) 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 three sort (BY) fields specified in the request: CONTINENT, REGION, and COUNTRY. Each field becomes a place-holder for its TOC. (If the TOC features were not in effect, the field would display in the report.)

```plaintext
TABLE FILE SHORT
"Projected Return"
""
"For:<CONTINENT For:<REGION For:<COUNTRY"
PRINT PROJECTED_RETURN
BY CONTINENT BY REGION BY COUNTRY BY TYPE
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
  TYPE=REPORT, GRID=OFF, $
  TYPE=HEADING, LINE=1, STYLE=BOLD, $
  TYPE=HEADING, LINE=3, OBJECT=FIELD, ITEM=1, TOC=1, $
  TYPE=HEADING, LINE=3, OBJECT=FIELD, ITEM=2, TOC=2, $
  TYPE=HEADING, LINE=3, OBJECT=FIELD, ITEM=3, TOC=3, $
ENDSTYLE
END
```

The output is:

![Projected Return Table](image)

Creating Reports With WebFOCUS Language
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 and lower level drop-down lists. (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 next.

Note that if you had the selected information already in your field of view, the value would have flashed (that is, it would have appeared briefly highlighted in gray) to draw your attention to that location, but the display would not have changed.
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 city (Hong Kong) in the Far East region. The page number is now 3 since this is the third of the high-level sort groups in the TOC.

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 519.
- Define frames and populate them with reports. See *Specifying a Target Frame* on page 562.

**Reference:**  **HTML Table of Contents Limits**

The TOC feature:

- Applies to HTML output.
- Is supported with Internet Explorer Version 5.5 and later.
This feature may generate unexpected behavior when a Firefox browser is used.

Does not support the Dialogue Manager command -HTMLFORM.

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 you have installed ReportCaster, you can distribute a report with an HTML TOC by including the commands SET BASEURL=HTTP://webserver and ON TABLE PCHOLD FORMAT HTML BYTOC [n] in the report request.

Creating an Active Report

In this section:
- Controlling Active Report Formatting Options
- User Controlled Options in an Active Report
- Modifying a Chart Legend With StyleSheet Syntax
- Controlling Expiration of an Active Report
- Setting a Password for an Active Report

How to:
- Create an Active Report

Active Reports provide customizable options for creating HTML formatted reports that enable users to experience features similar to those found in Excel workbooks. An Active Report is designed for distribution to users to perform 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 e-mail and transparent to security systems.

Tip: Whenever Active Reports is mentioned, Active Flex capabilities also apply, unless otherwise noted. Use AHTML (HTML Active Reports) or FLEX (Active Report Flash/Flex) as the report output format.

Specific WebFOCUS StyleSheet commands enable developers to set the initial state of the report and control the options available to users. When developing Active Reports, the options you can control include:
Viewing reports in either tabular or expandable formats.

Applying calculations to columns and choosing the location to display results.

Controlling the display of data by hiding or freezing columns.

Limiting the number of rows displayed per page.

Adding a graphic visualization column to compare column values.

Customizing the colors for most of the report components.

**Syntax:**

**How to Create an Active Report**

To create an HTML Active report, use

```
ON TABLE {PCHOLD|HOLD|SAVE} FORMAT AHTML
```

where:

**PCHOLD**

Displays the report output to a Web browser in HTML Active format. PCHOLD is the default value.

**HOLD**

Saves the report output to a temporary file for later use in an HTML Active report. The output is saved with an associated Master File.

**SAVE**

Saves the report output to a file for later use in an HTML Active report, but the output is not saved with a Master File.

**Tip:** To create an Active Report using Adobe® Flex®, use FLEX as the format.
**Example:** Creating an Active Report

The following code generates an HTML Active Report.

All customized (non-default) values and corresponding keywords, used to specifically format an HTML Active report, are shown in bold.

```plaintext
TABLE FILE CENTORD
SUM
   ORDER_NUM
   ORDER_DATE
   QUANTITY
   LINEPRICE
   BY PLANTING AS 'Plant'
ON TABLE SET PAGE-NUM OFF
ON TABLE SET BYDISPLAY ON
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT AHTML
ON TABLE SET STYLE *
   UNITS=IN,
   SQUEEZE=ON,
   ORIENTATION=PORTRAIT,
   TYPE=REPORT,
   GRAPHCOLOR='GREEN',
   GRAPHCOLORNEG='RED',
   TYPE=REPORT,
   FONT='ARIAL',
   SIZE=9,
   COLOR='BLACK',
   BACKCOLOR='NONE',
   STYLE=NORMAL,
   FREEZE-WIDTH=AUTO,
   LINES-PER-PAGE=30,
   CALC-LOCATION =BOTTOM,
   TYPE=TITLE, STYLE=BOLD,
   TYPE=REPORT,
   OBJECT=MENU,
   COLOR=RGB(0 51 102),
   HOWER-COLOR='WHITE',
   BACKCOLOR=RGB(51 204 204),
   HOVER-BACKCOLOR='PURPLE',
   BORDER-COLOR='FUCHSIA',
   TYPE=REPORT,
   OBJECT=STATUS-AREA,
   COLOR=RGB(153 51 0),
   BACKCOLOR='AQUA',
```

660 WebFOCUS
Starting at the top of the report, notice the customized background colors, the record status and page navigation bar, the first data row (which is selected), the fourth data row (which is hovered), and the calculation row at the bottom.
Controlling Active Report Formatting Options

How to:
Control the Report View Options
Control the Window Display Option
Control the Pagination Options
Control the Row Selection Colors
Control User Menu Options
Control the Calculations Options
Control the Active Cache Option
Control the Visualization Colors Options
Control the Freeze Column Options
Control Hide Columns Options
Show or Hide Active Report Menu Options

Some of the formatting options can only be controlled or implemented from the user menus in the output of an Active Report. For more information about these options, see User Controlled Options in an Active Report on page 671.

You can set the initial state of an Active Report by customizing many report options with WebFOCUS StyleSheet syntax. You can enable or disable some options prior to distribution to users. The following syntax sections describe the options you can control.

Syntax: How to Control the Report View Options

To control the report view options, which determine whether a tabular or expandable report is created, use

ON TABLE SET EXPANDABLE=(ON|OFF)

where:

- **ON**
  - Creates an expandable report.

- **OFF**
  - Creates a tabular report. OFF is the default value.

**Note:** The pagination and freeze options are not available with the expandable report view option.
Tip: When using Active Reports, it is recommended that you set the system font display to normal to ensure that the menu icons display correctly.

To set the system font display, right-click anywhere on the desktop, select Properties from the pop-up menu, select the Settings tab in the Display Properties dialog box, click the Advanced button, set the Font Size to Small Fonts, and click OK (these steps are for Windows 2000; however, these steps may vary by operating system).

Syntax: **How to Control the Window Display Option**

To control how windows display in the report output when multiple windows are open in the Active Report Web browser, use

```
TYPE=REPORT, WINDOW-DISPLAY=coldesc, $ 
```

where:

- **coldesc**
  
  Enables the report to display multiple open windows as tabs or cascaded windows in the Active Report Web browser. Multiple windows are created in the browser when viewing data as a chart, a roll-up table, and so on.

  Valid values are CASCADE or TAB.

Syntax: **How to Control the Pagination Options**

To control the pagination options, which determine the number of rows displayed per page and the styling characteristics of the record status and page navigation bar, use the appropriate syntax section that follows.

- To control the number of rows displayed per page in the output, use

  ```
  TYPE=REPORT, LINES-PER-PAGE={n UNLIMITED}, $ 
  ```

  where:

  - **n**
    
    Specifies the number of rows to be displayed on each HTML page. The default is 20 when the LINES-PER-PAGE option is used, otherwise the server default is 57 rows.

  - **UNLIMITED**
    
    Specifies that you want to show all of the results on one HTML page.
To control the position, justification, and colors of the record status and page navigation bar, use

```
TYPE=REPORT,
OBJECT=STATUS-AREA, PAGE-LOCATION={TOP | BOTTOM}, JUSTIFY={LEFT | CENTER | RIGHT},
COLOR=textcolor, BACKCOLOR=backcolor, $
```

where:

**STATUS-AREA**
Specifies the object, which is the record status and page navigation bar.

The record status and page navigation bar shows row and page information, where your cursor is positioned, relative to the total number of rows and pages in the report (for example, 10 of 100 records (10.00%), Page 1 of 3).

**PAGE-LOCATION**
Specifies the location of the record status and page navigation bar, which is either at the top of the report above the report header, or at the bottom of the report below the report footer. TOP is the default value.

**JUSTIFY**
Specifies whether you want the information in the record status and page navigation bar centered, left-justified, or right-justified. CENTER is the default value.

**textcolor**
Specifies the text color for the record status and page navigation bar. BLACK is the default value. Valid values can be either a preset color name in single quotes, for example, 'GREEN', or the RGB (red green blue) numeric values, for example, RGB(153 153 153).

**backcolor**
Specifies the background color for the record status and page navigation bar. SILVER is the default value. Valid values can be either a preset color name in single quotes, for example, 'WHITE', or the RGB (red green blue) numeric values, for example, RGB(51 153 102).

**Note:** The pagination options are not available with the expandable report view option.
**Syntax:**  **How to Control the Row Selection Colors**

To control the colors of the row being selected or hovered over, use

```
TYPE=REPORT, OBJECT=CURRENT-ROW, HOVER-BACKCOLOR=hovercolor,
BACKCOLOR=selectcolor, $
```

where:

- **CURRENT-ROW**
  Specifies the object, which is the current row where the mouse pointer is clicked, or is hovering.

- **hovercolor**
  Specifies the background color of the row where the mouse pointer is hovering. The default value is RGB(255 252 204), which is a shade of yellow. Valid values can be either a preset color name in single quotes, for example, 'YELLOW', or the RGB (red green blue) numeric values, for example, RGB(255 255 153).

- **selectcolor**
  Specifies the background color of the row selected (click anywhere within a row). The default is RGB(51 255 204), which is a mix of green and blue. Valid values can be either a preset color name in single quotes, for example, 'BLUE', or the RGB (red green blue) numeric values, for example, RGB(153 53 0).

**Syntax:**  **How to Control User Menu Options**

To control the user menu options, which determine the location and color characteristics of the user menu in the report, use the appropriate syntax section that follows.

- To control all of the colors of the user menu, use

```
TYPE=REPORT, OBJECT=MENU, COLOR=textcolor,
HOVER-COLOR=hovertextcolor,BACKCOLOR=backcolor,
HOVER-BACKCOLOR=hoverbackcolor,BORDER-COLOR=bordercolor, $
```

where:

- **MENU**
  Specifies the object, which is the user menu.

- **textcolor**
  Specifies the color of the text (font) in the user menu. BLACK is the default value. Valid values can be either a preset color name in single quotes, for example, 'YELLOW', or the RGB (red green blue) numeric values, for example, RGB(153 153 153).
**hovertextcolor**

Specifies the color for the hover text in the user menu. BLACK is the default value. Valid values can be either a preset color name in single quotes, for example, 'RED', or the RGB (red green blue) numeric values, for example, RGB(255 255 153).

**backcolor**

Specifies the color for the background (non-text) areas in the user menu. SILVER is the default value. Valid values can be either a preset color name in single quotes, for example, 'GOLD', or the RGB (red green blue) numeric values, for example, RGB(153 51 0).

**hoverbackcolor**

Specifies the color for the background area of the user menu where your mouse pointer is hovering. WHITE is the default value. Valid values can be either a preset color name in single quotes, for example, 'ORANGE', or the RGB (red green blue) numeric values, for example, RGB(51 204 204).

You can use this, along with the hovertext color, to visually contrast the menu option where the mouse is currently hovering from the rest of the menu.

**bordercolor**

Specifies the color for the borders of the user menu. WHITE is the default value. Valid values can be either a preset color name in single quotes, for example, 'BLUE', or the RGB (red green blue) numeric values, for example, RGB(255 255 153).

---

**Syntax: How to Control the Calculations Options**

To control the calculations options, which determine the column and the calculation you want to perform, along with the styling characteristics of the calculation results row, use the appropriate syntax section that follows.

- To perform column calculations, use

  ```
  TYPE=REPORT, COLUMN=coldesc, CALCULATION=calc, $
  ```

  where:

  **coldesc**

  Specifies the description (column name or identifier) of the column where you want to perform a calculation.
calc

Specifies the calculation to be performed. The default is no calculation.

For columns containing numeric data, select from: `SUM, AVE, MAX, CNT, CNT.DST, PCT.TOT`

**Note:** PCT.TOT values appear in a separate column adjacent to the column for which it is calculated, and styling is inherited from the visualization column.

For columns containing text (alphanumeric) data, select from: `CNT, CNT.DST`

- To control the location of the calculation results row, use
  
  ```
  TYPE=REPORT, CALC-LOCATION={TOP|BOTTOM}, $
  ```
  
  where:

  **TOP**
  
  Specifies the top row of the report as the location of the calculation results. TOP is the default value.

  **BOTTOM**
  
  Specifies the bottom row of the report as the location of the calculation results.

- To control the colors of the calculation results row, use

  ```
  TYPE=REPORT, OBJECT=CALC-AREA, COLOR=color, BACKCOLOR=backcolor, $
  ```
  
  where:

  **CALC-AREA**
  
  Specifies the object, which is the calculation results row.

  **color**
  
  Specifies the color for the calculation values in the calculation results row. The default value is BLACK and is inherited from the column title font color. Valid values can be either a preset color name in single quotes, for example, 'RED', or the RGB (red green blue) numeric values, for example, RGB(255 255 153).

  **backcolor**
  
  Specifies the color for the background in the calculation results row. The default value is WHITE and is inherited from the column title background color. Valid values can be either a preset color name in single quotes, for example, 'GOLD', or the RGB (red green blue) numeric values, for example, RGB(153 51 0).
**Syntax:**   **How to Control the Active Cache Option**

Active Reports answer sets are recommended not to exceed 5,000 records, or 100 pages of output, since all of the post-retrieval processing is performed in the Web browser memory. To create answer sets in excess of 5,000 records, the Active Cache option enables you to send only the first page of Active Report output to the browser and retrieve subsequent pages from a temporary cache on the WebFOCUS Reporting Server. The server also becomes the resource for performing all calculations, sorting, and filtering when Active Cache is enabled.

To enable Active Cache for Active Reports use:

```
ON TABLE SET WEBVIEWER {ON|OFF}
ON TABLE SET CACHELINES {n|100}
```

where:

- **ON**
  - Runs the Active Report with Active Cache enabled.

- **OFF**
  - Will not enable Active Cache when the Active Report is run. OFF is the default value.

- **n|100**
  - Specifies the number of rows from cache. The default is 100.

**Tip:** It is recommended to set the number of rows retrieved five times greater then the number of lines retrieved per page (as indicated in SET LINES). The minimum number of rows retrieved is 100.

**Syntax:**   **How to Control the Visualization Colors Options**

To control the colors of the graphic values used with the visualization option, use

```
TYPE=REPORT, GRAPHCOLOR=positivecolor, GRAPHCOLORNEG=negativecolor, $
```

where:

- **positivecolor**
  - Specifies the color for the positive values displayed in the optional visualization column. BLACK is the default value. Valid values can be either a preset color name in single quotes, for example, 'BLUE', or the RGB (red green blue) numeric values, for example, RGB(255 255 153).
negativecolor

Specifies the color for the negative values displayed in the optional visualization column. BLACK is the default value. Valid values can be either a preset color name in single quotes, for example, 'VIOLET', or the RGB (red green blue) numeric values, for example, RGB(153 153 153).

**Syntax:**  
**How to Control the Freeze Column Options**

To control the freeze column options, which determine the column (and all columns to the left) that will freeze when scrolling to the right to view other columns in the report output, use:

```
TYPE=REPORT, FREEZE-COLUMN=coldesc, $
```

or

```
TYPE=REPORT, FREEZE-WIDTH=AUTO, $
```

where:

- **coldesc**
  
  Specifies the description (column name or identifier) of the column you want to freeze. You can only specify one freeze column, which must be a BY sort field. The default value is none (no freeze column).

- **AUTO**
  
  Enables the report to freeze at a particular column that will be automatically determined by WebFOCUS for optimal viewing.

If you can view the entire report output on the screen without scrolling, the freeze will not be applied.

**Note:** The freeze option is not available with the expandable report view option.

**Syntax:**  
**How to Control Hide Columns Options**

To control the hide columns options, which determine the columns that will be hidden from view in the report output, repeat the following for each column you want to hide.

```
TYPE=REPORT, COLUMN=coldesc, HIDE={ON|OFF}, $
```

where:

- **coldesc**
  
  Specifies the description (column name or identifier) of the column you want to hide. You can hide multiple columns, but you cannot hide all columns (at least one column must always be visible).
ON
- Enables the option.

OFF
- Disables the option. OFF is the default value.

**Syntax:**

**How to Show or Hide Active Report Menu Options**

To control Active Report menu options available to the users, use:

```
TYPE=REPORT, ALLOW-feature={ON|OFF}, $
```

where:

- **feature**
  - Specifies the feature. Select from:
    - **PAGINATION** to enable the use of show records options.
    - **FILTER** to enable the use of filtering or highlighting options.
    - **FREEZE** to enable the use of freeze column options.
    - **CALC** to enable the use of calculation options.
    - **HIDE** to enable the use of hide columns options.
    - **CHART** to enable the use of chart creating options.
    - **EXPORT** to enable the use of exporting options.
    - **VISUALIZE** to enable the use of graphical visualization options.
    - **SORT** to enable the use of sorting options.
    - **ROLLUP** to enable the use of roll-up table options.
    - **PIVOT** to enable the use of pivot table options.
    - **COMMENTS** to enable the use of comment options.
    - **WINDOW** to enable the use of window type options.
    - **RESTORE** to enable the use of restore options.
    - **SENDEMAIL** to enable the use of send as e-mail options.
    - **SAVECHANGE** to enable the use of save changes options.
    - **PRINT** to enable the use of print options.
    - **ACCORDION** to enable the use of accordion options.
    - **TOOLS** to enable the use Chart/Rollup Tool, Pivot Tool, and Grid Tool options.
ON

Enables the option. ON is the default value for all user-controlled options.

OFF

Disables the option.

**User Controlled Options in an Active Report**

The following options can only be implemented from the user menus in the output of an Active Report. Basic information for each of these options provides a more comprehensive understanding of this type of report.

- **Sorting Data**
  - If the ALLOW-SORT option is enabled, users have options to sort columns in either ascending and descending order. For string data types, the sorting is alphabetical. For date data types, the sorting is chronological. When the report view option is tabular, only single column sorts are supported. When the report view option is expandable, an inner sort is also supported.

- **Adding Visualization Graphics to an HTML Active Report**
  - If the ALLOW-VISUALIZE option is enabled, users have options to display visualization graphics in a new column inserted into the report to the right of the selected column. Negative values appear with colored bars that extend to the left of center, and positive values appear with colored bars that extend to the right of center.

- **Creating Charts in an Active Report**
  - If the ALLOW-CHART option is enabled, users have options to create charts using roll-up tables. All applied filters are respected, and all calculation methods are available to create aggregation charts. A selected BY sort field and a measure column are required. There is also an option to open charts in separate browser windows.

- **Exporting Active Reports**
  - If the ALLOW-EXPORT option is enabled, users have options to export all data, or filtered data only. The export method only exports data, not the JavaScript code, which makes exported reports static without the interactive options available in an Active report. The export formats supported are: Save as HTML (formatting is preserved), XML export to Excel (formatting is not preserved), and Save as CSV (formatting is not preserved).

- **Filtering or Highlighting Data**
  - If the ALLOW-FILTER option is enabled, users have options to filter or highlight data. Filtering limits the output to display only data that meets the criteria in the WHERE clause specified with the user menus. Highlighting changes the color of the data in the output based on the criteria in the WHERE clause specified with the user menus.
Applying Global Filters to Multiple Active Reports

When multiple Active reports are inserted in the same HTML page, users can apply filters on all reports containing a common BY sort field.

Modifying a Chart Legend With StyleSheet Syntax

When creating charts with Active Reports using Adobe® Flex®, you may modify the chart legend by using WebFOCUS syntax.

**Note:** This section is only applicable when using Active Reports with Adobe® Flex®, where you can click the Legend button to show/hide the chart legend. Note that the legend always appears in a Pie chart. The legend only appears in a Bar, Line, or Scatter chart if there is more than one Y-axis field.

The following StyleSheet syntax shows the chart legend for an Active Report:

```plaintext
TYPE=REPORT, OBJECT=LEGEND, LEGEND-VIEW=[DOCKING/FLAT],
POSITION-POINT=[BOTTOM-LEFT/BOTTOM-CENTER/BOTTOM-RIGHT], $
```

The examples below describe the chart legend syntax and output.

- If the chart legend is not specified, WebFOCUS sets **LEGEND-VIEW** to **DOCKING**, and **POSITION-POINT** to **BOTTOM-LEFT**, as shown in the following example.

```plaintext
TABLE FILE GGSALES
SUM UNITS DOLLARS
BY CATEGORY
BY PRODUCT
ON TABLE PCCHOLD FORMAT FLEX
END
```
The WebFOCUS syntax generates the chart legend as shown in the following image.

If `LEGEND-VIEW` is `FLAT`, the chart legend appears below the chart and there is no Legend button. In this scenario, the `POSITION-POINT` is ignored, as shown in the following example.

```
TABLE FILE GGSALES
SUM UNITS DOLLARS
BY CATEGORY
BY PRODUCT
ON TABLE PCHOLD FORMAT FLEX
ON TABLE SET STYLE *
TYPE=REPORT,OBJECT=LEGEND,LEGEND-VIEW=FLAT,$
ENDSTYLE
END
```
The WebFOCUS syntax generates the chart legend as shown in the following image.

If the legend `POSITION-POINT` is `BOTTOM-CENTER`, the chart legend can be expanded or collapsed to the bottom below the chart, as shown in the following example.

```
TABLE FILE GGSALES
SUM UNITS DOLLARS
BY CATEGORY
BY PRODUCT
ON TABLE PCHOLD FORMAT FLEX
ON TABLE SET STYLE *
TYPE=REPORT, OBJECT=LEGEND, LEGEND-VIEW=DOCKING, POSITION-POINT=BOTTOM-CENTER, $
ENDSTYLE
END
```
The WebFOCUS syntax generates the chart legend as shown in the following image.

Table sales by product

- Biscotto 11.42%
- Capuccino 5.13%
- Coffee Grinder 3.06%
- Coffee Pot 5.17%
- Croissant 17.38%
- Espresso 4.38%

If the legend POSITION-POINT is BOTTOM-RIGHT, the chart legend can be expanded and collapsed to the right below the chart, as shown in the following example.

```
TABLE FILE GGSALES
SUM UNITS DOLLARS
BY CATEGORY
BY PRODUCT
ON TABLE PCHOLD FORMAT FLEX
ON TABLE SET STYLE *
TYPE=REPORT, OBJECT=LEGEND, LEGEND-VIEW=DOCKING, POSITION-POINT=
BOTTOM-RIGHT,$
ENDSTYLE
END
```
The WebFOCUS syntax generates the chart legend as shown in the following image.

![Unit Sales BY Product](image)

- If using the procedure with the initial pie chart presentation, where `REPORT-VIEW` is `CHART`, and `CHART-TYPE` is `PIE`, the following syntax is generated.

```
TABLE FILE GGSALES
SUM DOLLARS
BY PRODUCT
ON TABLE PCHOLD FORMAT FLEX
ON TABLE SET STYLE *
    TYPE=REPORT,REPORT-VIEW=CHART,CHART-TYPE=PIE,$
    TYPE=REPORT,OBJECT=LEGEND,LEGEND-VIEW=DOCKING,POSITION-POINT=BOTTOM-CENTER,$
ENDSTYLE
END
```
The WebFOCUS syntax generates the chart legend as shown in the following image.

### Controlling Expiration of an Active Report

The AREXPIRE parameter enables you to set the date when an Active Report expires and the report output can no longer be displayed.

The syntax is:

```
SET AREXPIRE = {ymmd|xxxDAYS}
```

where:

- `ymmd`
  - Is the expiration date in the format of year, month, and day. For example, if you want the report to expire on January 1, 2009, use 090101.

- `xxx`
  - Is the number of days from the current date that the report expires. Valid values are 1 to 999.
The command can also be issued from within a request using:

```
ON TABLE SET AREXPIRE {yymmdd|xxxDAYS}
```

### Setting a Password for an Active Report

The ARPASSWORD parameter enables you to set a password that is required to view Active Report output. Prior to opening the report output, the user is prompted to enter a password to unlock the report.

The syntax is:

```
SET ARPASSWORD = password
```

where:

- **password**
  
  Is any character string up to 32 characters in length.

The command can also be issued from within a request using:

```
ON TABLE SET ARPASSWORD password
```

### 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 of data for each vertical sort field. When an Accordion Report first appears, only data values of the first (highest-level) vertical sort field are shown; all other data is hidden. You can manually expand your view to expose data values of lower-level sort fields.

When you create an Accordion Report, 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.

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 command is ignored, no message is generated, and a standard HTML report is created.
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.

PDF and Excel formats are not supported with Accordion Reports.

Requirements for Accordion Reports

**How to:**
Create Accordion Reports

**Reference:**
Distributing Accordion Reports With ReportCaster
Support 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 e-mail attachment or as an archived report in the Report Library.
**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,

```sql
SET FOCHTMLURL = http://hostname[:port]/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_html**

Is the site-customized Web server alias pointing to the WebFOCUS76/ibi_html directory (where ibi_html is the default value).

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

**Reference:** Support for Accordion Reports

The following commands are not supported when using Accordion Reports:

- **BORDER, COLUMN, FOR, IN, OVER, PAGE-NUM, ROW-TOTAL, TOTAL**

Data Visualization, 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 Reports

To enable Accordion Reports, specify the following

```sql
ON TABLE SET EXPANDABLE = {ON|OFF}
```

where:

**ON**

Enables Accordion Reports.

**OFF**

Disables Accordion Reports. OFF is the default value.
### Example: Creating an Accordion Report

This example shows how to use an EXPANDABLE command to create an Accordion Report.

```sql
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 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.

<table>
<thead>
<tr>
<th>Region</th>
<th>State</th>
<th>City</th>
<th>Category</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midwest</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Northeast</td>
<td>CT</td>
<td></td>
<td>Coffee</td>
<td>116659</td>
<td>1459160</td>
</tr>
<tr>
<td></td>
<td>MA</td>
<td></td>
<td>Food</td>
<td>125473</td>
<td>1555165</td>
</tr>
<tr>
<td></td>
<td>NY</td>
<td>New York</td>
<td>Gifts</td>
<td>70194</td>
<td>887940</td>
</tr>
<tr>
<td>Southeast</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>West</td>
<td>CA</td>
<td>Los Angeles</td>
<td>Coffee</td>
<td>109082</td>
<td>1383924</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Food</td>
<td>110417</td>
<td>1381698</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Gifts</td>
<td>78571</td>
<td>1006381</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>San Francisco</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Navigating a Multi-Page Report With the WebFOCUS Viewer

In this section:
- Using the WebFOCUS Viewer Search Option
- How to:
  - Navigate in 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.

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.

**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.
Navigating a Multi-Page Report With the WebFOCUS Viewer

- 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.

For more information about using the WebFOCUS Viewer, see the WebFOCUS Managed Reporting End User’s Manual.
**Linking Report Pages**

**How to:**
Link 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:

```html
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:

```sql
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_HTML/GGDEMO/BULLET, URL=#_next,
IMAGEALIGN=LEFT,$
TYPE=HEADING, IMAGE=/IBI_HTML/GGDEMO/GOBACK, URL=#_previous,
IMAGEALIGN=RIGHT,$
ENDSTYLE
END
```

The images display in each page heading.

PAGE 1

COFFEE GRINDER SALES BY STORE

<table>
<thead>
<tr>
<th>Store Code</th>
<th>Order Number</th>
<th>Ordered Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1019</td>
<td>9</td>
<td>265</td>
</tr>
<tr>
<td></td>
<td>139</td>
<td>148</td>
</tr>
<tr>
<td></td>
<td>369</td>
<td>382</td>
</tr>
<tr>
<td></td>
<td>549</td>
<td>325</td>
</tr>
<tr>
<td></td>
<td>729</td>
<td>381</td>
</tr>
<tr>
<td></td>
<td>909</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>1086</td>
<td>295</td>
</tr>
<tr>
<td></td>
<td>1266</td>
<td>221</td>
</tr>
</tbody>
</table>
11. Navigating Within an HTML Report

Click the image on the left of page 1 to display page 2.

PAGE 2

<table>
<thead>
<tr>
<th>Store Code</th>
<th>Order Number</th>
<th>Ordered Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1020</td>
<td>24</td>
<td>133</td>
</tr>
<tr>
<td></td>
<td>204</td>
<td>380</td>
</tr>
<tr>
<td></td>
<td>384</td>
<td>279</td>
</tr>
<tr>
<td></td>
<td>564</td>
<td>357</td>
</tr>
<tr>
<td></td>
<td>744</td>
<td>189</td>
</tr>
<tr>
<td></td>
<td>924</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>1101</td>
<td>243</td>
</tr>
<tr>
<td></td>
<td>1281</td>
<td>221</td>
</tr>
<tr>
<td></td>
<td>1461</td>
<td>262</td>
</tr>
<tr>
<td></td>
<td>1641</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>1821</td>
<td>184</td>
</tr>
</tbody>
</table>

Click the "go back" image on page 2 to redisplay page 1.

PAGE 1

**COFFEE GRINDER SALES BY STORE**

<table>
<thead>
<tr>
<th>Store Code</th>
<th>Order Number</th>
<th>Ordered Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1019</td>
<td>9</td>
<td>265</td>
</tr>
<tr>
<td></td>
<td>189</td>
<td>143</td>
</tr>
<tr>
<td></td>
<td>369</td>
<td>382</td>
</tr>
<tr>
<td></td>
<td>549</td>
<td>325</td>
</tr>
<tr>
<td></td>
<td>729</td>
<td>381</td>
</tr>
<tr>
<td></td>
<td>909</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>1086</td>
<td>295</td>
</tr>
<tr>
<td></td>
<td>1266</td>
<td>221</td>
</tr>
</tbody>
</table>
For details on including and positioning images in a report, see *Laying Out the Report Page* on page 929.

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
```
11. Navigating Within an HTML Report

The first page looks as follows:

**PAGE 1**

**COFFEE GRINDER SALES BY STORE**

<table>
<thead>
<tr>
<th>Store Code</th>
<th>Order Number</th>
<th>Ordered Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1019</td>
<td>9</td>
<td>265</td>
</tr>
<tr>
<td></td>
<td>189</td>
<td>148</td>
</tr>
<tr>
<td></td>
<td>369</td>
<td>382</td>
</tr>
<tr>
<td></td>
<td>549</td>
<td>325</td>
</tr>
<tr>
<td></td>
<td>729</td>
<td>381</td>
</tr>
<tr>
<td></td>
<td>909</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>1086</td>
<td>295</td>
</tr>
<tr>
<td></td>
<td>1266</td>
<td>221</td>
</tr>
<tr>
<td></td>
<td>1446</td>
<td>122</td>
</tr>
</tbody>
</table>

Click the page number three times to move to PAGE 4.

**PAGE 4**

<table>
<thead>
<tr>
<th>Store Code</th>
<th>Order Number</th>
<th>Ordered Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1041</td>
<td>54</td>
<td>292</td>
</tr>
<tr>
<td></td>
<td>234</td>
<td>165</td>
</tr>
<tr>
<td></td>
<td>414</td>
<td>229</td>
</tr>
<tr>
<td></td>
<td>594</td>
<td>208</td>
</tr>
<tr>
<td></td>
<td>774</td>
<td>294</td>
</tr>
<tr>
<td></td>
<td>953</td>
<td>276</td>
</tr>
<tr>
<td></td>
<td>1131</td>
<td>137</td>
</tr>
<tr>
<td></td>
<td>1311</td>
<td>273</td>
</tr>
<tr>
<td></td>
<td>1491</td>
<td>289</td>
</tr>
</tbody>
</table>

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 separates a single report into multiple HTML files based on the value of the first sort field in your report.

**Topics:**
- Bursting Reports Overview
Bursting Reports Overview

How to:
Burst Reports Into Multiple HTML Files

Reference:
Rules for Headings and Footings on Index Pages and Bursted Reports

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.

Note: When you are bursting reports in Developer Studio, you must run the request remotely against a WebFOCUS Server.

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 697.

**Reference:**  **Rules for Headings and Footings on Index Pages and Burst ed 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 1053.

**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\WebFOCUS435\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\WebFOCUS435\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:

<table>
<thead>
<tr>
<th>Region</th>
<th>Store ID</th>
<th>Category</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midwest</td>
<td>R1019</td>
<td>Coffee</td>
<td>113253</td>
<td>1393610</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Food</td>
<td>107615</td>
<td>1351523</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gifts</td>
<td>78869</td>
<td>969845</td>
</tr>
<tr>
<td></td>
<td>R1020</td>
<td>Coffee</td>
<td>109581</td>
<td>1398779</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Food</td>
<td>118068</td>
<td>1522847</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gifts</td>
<td>79932</td>
<td>1002775</td>
</tr>
<tr>
<td></td>
<td>R1250</td>
<td>Coffee</td>
<td>109943</td>
<td>1386124</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Food</td>
<td>115731</td>
<td>1463901</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gifts</td>
<td>72053</td>
<td>911261</td>
</tr>
</tbody>
</table>

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 Bursting Reports on page 693.
Bursting Reports Overview
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 698.
- A field in a segment instance does not have a data value. See Missing Field Values on page 699.
- A parent segment instance does not have child instances (missing segment instances). See Handling a Missing Segment Instance on page 713.

**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.
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 720).

**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:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>DEPARTMENT</th>
<th>MIS</th>
<th>PRODUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANNING</td>
<td>JOHN</td>
<td></td>
<td>.</td>
<td>$29,700.00</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>$21,780.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>$27,062.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
<td>$9,000.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IRVING</td>
<td>JOAN</td>
<td></td>
<td>$26,862.00</td>
<td></td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td>$18,480.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td>$18,480.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>$16,100.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>$21,120.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td>$13,200.00</td>
<td>$9,050.00</td>
<td></td>
</tr>
<tr>
<td>STEVENS</td>
<td>ALFRED</td>
<td></td>
<td></td>
<td>$11,000.00</td>
</tr>
</tbody>
</table>

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 720.

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).
**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 702.
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 702).

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
  COUNT: 22  COUNT: 20  X_RETURNS: 2  RETURNS: 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

**How to:**

Specify Missing Values 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 702.

**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.

**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:

<table>
<thead>
<tr>
<th>CITY</th>
<th>DATE</th>
<th>PROD_CODE</th>
<th>RETURNS</th>
<th>DAMAGED</th>
<th>NO_SALE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW YORK</td>
<td>10/17</td>
<td>B10</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B17</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B20</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C13</td>
<td>.</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C14</td>
<td>4</td>
<td>.</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C17</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D12</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E1</td>
<td>4</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E2</td>
<td>.</td>
<td>.</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E3</td>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>NEWARK</td>
<td>10/18</td>
<td>B10</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>10/19</td>
<td>B12</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>STAMFORD</td>
<td>12/12</td>
<td>B10</td>
<td>10</td>
<td>6</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B12</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B17</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C13</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C7</td>
<td>5</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D12</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E2</td>
<td>9</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E3</td>
<td>8</td>
<td>9</td>
<td>17</td>
</tr>
<tr>
<td>UNIONDALE</td>
<td>10/18</td>
<td>B20</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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
```
Testing for a Segment With a Missing Field Value

**How to:**
Test 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. However, you can set the ALL parameter to PASS to test for missing instances. See *Handling a Missing Segment Instance* on page 713.

**Syntax:**  
**How to Test for a Segment With a Missing Field Value**

To test for a segment with missing field values, the syntax is:

```sql
WHERE field {IS|EQ} MISSING
```

To test for the presence of field values, the syntax is:

```sql
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:

```
<table>
<thead>
<tr>
<th>CITY</th>
<th>DATE</th>
<th>PROD_CODE</th>
<th>RETURNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW YORK</td>
<td>10/17</td>
<td>B10</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B17</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B20</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C14</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C17</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D12</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E3</td>
<td>4</td>
</tr>
<tr>
<td>NEWARK</td>
<td>10/18</td>
<td>B10</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>10/19</td>
<td>B12</td>
<td>1</td>
</tr>
<tr>
<td>STAMFORD</td>
<td>12/12</td>
<td>B10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B12</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B17</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C13</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C7</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D12</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E2</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E3</td>
<td>8</td>
</tr>
<tr>
<td>UNIONDALE</td>
<td>10/18</td>
<td>B20</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C7</td>
<td>0</td>
</tr>
</tbody>
</table>
```

**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
```
**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:

<table>
<thead>
<tr>
<th>CITY</th>
<th>DATE</th>
<th>PROD_CODE</th>
<th>RETURNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW YORK</td>
<td>10/17</td>
<td>B20</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C13</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>C17</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E2</td>
<td></td>
</tr>
<tr>
<td>STAMFORD</td>
<td>12/12</td>
<td>D12</td>
<td>0</td>
</tr>
<tr>
<td>UNIONDALE</td>
<td>10/18</td>
<td>C7</td>
<td>0</td>
</tr>
</tbody>
</table>
Preserving Missing Data Values in an Output File

How to:

Distinguish Missing Data in an Extract File
Store Missing Data in HOLD Files

Reference:

Usage Notes for Holding Missing Values

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. 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 351.

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**

```plaintext
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:

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>WHOLESALEPR</th>
<th>WHOLESALEPR</th>
<th>WHOLESALEPR</th>
<th>WHOLESALEPR</th>
<th>WHOLESALEPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION</td>
<td>.</td>
<td></td>
<td>20.98</td>
<td>.</td>
<td>34.48</td>
</tr>
<tr>
<td>CHILDREN</td>
<td>54.49</td>
<td>51.38</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>CLASSIC</td>
<td>40.99</td>
<td>160.80</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>COMEDY</td>
<td>.</td>
<td></td>
<td>46.70</td>
<td>30.00</td>
<td>13.75</td>
</tr>
<tr>
<td>DRAMA</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>10.00</td>
</tr>
<tr>
<td>FOREIGN</td>
<td>13.25</td>
<td></td>
<td>62.00</td>
<td>.</td>
<td>70.99</td>
</tr>
<tr>
<td>MUSICALS</td>
<td>15.00</td>
<td></td>
<td>13.99</td>
<td>9.99</td>
<td>13.99</td>
</tr>
<tr>
<td>MYSTERY</td>
<td>.</td>
<td>9.00</td>
<td>18.00</td>
<td>9.00</td>
<td>80.97</td>
</tr>
<tr>
<td>SCI/FI</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>35.99</td>
<td>43.53</td>
</tr>
<tr>
<td>TRAIN/EX</td>
<td>.</td>
<td>60.98</td>
<td>.</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Propagating Missing Values to Reformatted Fields in a Request**

**How to:**
Control Missing Values in Reformatted Fields

**Reference:**
Usage Notes for SET COMPMISS

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:

<table>
<thead>
<tr>
<th>STORE_CODE</th>
<th>RETURNS</th>
<th>REFORMATTED RETURNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>14Z</td>
<td>2</td>
<td>2.00</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2.00</td>
</tr>
<tr>
<td>0</td>
<td>.</td>
<td>.00</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.00</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4.00</td>
</tr>
<tr>
<td>0</td>
<td>.</td>
<td>.00</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3.00</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4.00</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.00</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4.00</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>STORE_CODE</th>
<th>RETURNS</th>
<th>REFORMATTED RETURNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>14Z</td>
<td>2</td>
<td>2.00</td>
</tr>
<tr>
<td>2</td>
<td>2.00</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>.00</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4.00</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>.00</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3.00</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4.00</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4.00</td>
<td></td>
</tr>
</tbody>
</table>

**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 714 and Reporting Against Segments With Descendant Instances on page 715.

**Example:** Reporting Against Segments Without Descendant Instances

The following request displays the salary histories for each employee.

```plaintext
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:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>DAT_INC</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANNING</td>
<td>JOHN</td>
<td>82/08/01</td>
<td>$29,700.00</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>82/04/01</td>
<td>$21,780.00</td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>81/11/02</td>
<td>$25,775.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82/04/09</td>
<td>$27,062.00</td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
<td>82/04/01</td>
<td>$8,650.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82/06/11</td>
<td>$9,000.00</td>
</tr>
<tr>
<td>IRVING</td>
<td>JOAN</td>
<td>82/01/04</td>
<td>$24,420.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82/05/14</td>
<td>$26,862.00</td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td>82/05/01</td>
<td>$17,750.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82/06/01</td>
<td>$18,480.00</td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td>82/01/01</td>
<td>$18,480.00</td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>82/02/02</td>
<td>$15,000.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82/05/14</td>
<td>$16,100.00</td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>82/07/01</td>
<td>$21,120.00</td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td>82/01/01</td>
<td>$13,200.00</td>
</tr>
<tr>
<td></td>
<td>RICHARD</td>
<td>82/01/04</td>
<td>$9,050.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82/05/14</td>
<td>$9,500.00</td>
</tr>
<tr>
<td>STEVENS</td>
<td>ALFRED</td>
<td>81/01/01</td>
<td>$10,000.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82/01/01</td>
<td>$11,000.00</td>
</tr>
</tbody>
</table>

### 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.

```sql
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:

<table>
<thead>
<tr>
<th>PIN</th>
<th>LASTNAME</th>
<th>FIRSTNAME</th>
<th>COURSECODE</th>
<th>EXPENSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000010</td>
<td>VALINO</td>
<td>DANIEL</td>
<td>PDR740</td>
<td>2,300.00</td>
</tr>
<tr>
<td>000000030</td>
<td>CASSANOVA</td>
<td>LOIS</td>
<td>NAMA730</td>
<td>2,600.00</td>
</tr>
<tr>
<td></td>
<td>CASSANOVA</td>
<td>LOIS</td>
<td>EDP090</td>
<td>2,300.00</td>
</tr>
<tr>
<td>0000000350</td>
<td>FERNSTEIN</td>
<td>ERWIN</td>
<td>SSI220</td>
<td>1,850.00</td>
</tr>
<tr>
<td></td>
<td>FERNSTEIN</td>
<td>ERWIN</td>
<td>MC90</td>
<td>1,730.00</td>
</tr>
<tr>
<td></td>
<td>FERNSTEIN</td>
<td>ERWIN</td>
<td>UMI720</td>
<td>3,350.00</td>
</tr>
<tr>
<td>000000360</td>
<td>CHISOLM</td>
<td>HENRY</td>
<td>EDP090</td>
<td>.00</td>
</tr>
<tr>
<td></td>
<td>CHISOLM</td>
<td>HENRY</td>
<td>EDP690</td>
<td>3,000.00</td>
</tr>
<tr>
<td>000000370</td>
<td>WANG</td>
<td>JOHN</td>
<td>UMI710</td>
<td>2,050.00</td>
</tr>
<tr>
<td>000000380</td>
<td>ELLNER</td>
<td>DAVID</td>
<td>EDP090</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>ELLNER</td>
<td>DAVID</td>
<td>UNI780</td>
<td>3,350.00</td>
</tr>
<tr>
<td>000000410</td>
<td>CONTI</td>
<td>MARSHALL</td>
<td>EDP690</td>
<td>3,100.00</td>
</tr>
</tbody>
</table>

**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 Instances in Reports With the SET ALL Parameter* on page 717.

**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:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>DAT_INC</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANNING</td>
<td>JOHN</td>
<td>82/08/01</td>
<td>$29,700.00</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>82/04/01</td>
<td>$21,780.00</td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>81/11/02</td>
<td>$25,775.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82/04/09</td>
<td>$27,062.00</td>
</tr>
<tr>
<td>DAVIS</td>
<td>ELIZABETH</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>GARDNER</td>
<td>DAVID</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
<td>82/04/01</td>
<td>$8,650.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82/06/11</td>
<td>$9,000.00</td>
</tr>
<tr>
<td>IRVING</td>
<td>JOAN</td>
<td>82/01/04</td>
<td>$24,420.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82/05/14</td>
<td>$26,862.00</td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td>82/05/01</td>
<td>$17,750.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82/06/01</td>
<td>$18,480.00</td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td>82/01/01</td>
<td>$18,480.00</td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>82/02/02</td>
<td>$15,000.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82/05/14</td>
<td>$16,100.00</td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>82/07/01</td>
<td>$21,120.00</td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td>82/01/01</td>
<td>$13,200.00</td>
</tr>
<tr>
<td></td>
<td>RICHARD</td>
<td>82/01/04</td>
<td>$9,050.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82/05/14</td>
<td>$9,500.00</td>
</tr>
<tr>
<td>STEVENS</td>
<td>ALFRED</td>
<td>81/01/01</td>
<td>$10,000.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82/01/01</td>
<td>$11,000.00</td>
</tr>
</tbody>
</table>

Including Missing Instances in Reports With the SET ALL Parameter

**How to:**

Include a Parent Instance With Missing Descendants

You can include parent instances with missing descendants by issuing the SET ALL parameter before executing the request.

**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 in the report, use the SET ALL=PASS command.

**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.

```sql
JOIN EMPDATA.PIN IN EMPDATA TO ALL TRAINING.PIN IN TRAIN2 AS JOIN1
SET ALL = PASS
TABLE FILE EMPDATA
PRINT LASTNAME AND FIRSTNAME AND COURSECODE AND EXPENSES
BY PIN
WHERE EXPENSES GT 3000
END
```
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 TRAIN2 AS JOIN1
SET ALL = PASS
TABLE FILE EMPDATA
PRINT LASTNAME AND FIRSTNAME AND COURSECODE AND EXPENSES
BY PIN
WHERE COURSECODE EQ 'XXXX'
END
```
## Setting the NODATA Character String

### How to:

Set the NODATA 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**

```plaintext
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:

```plaintext
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.

```plaintext
SET NODATA=MISSING

TABLE FILE EMPLOYEE
PRINT CURR_SAL BY LAST_NAME BY FIRST_NAME ACROSS DEPARTMENT
END
```

The output is:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>DEPARTMENT</th>
<th>MIS</th>
<th>PRODUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANNING</td>
<td>JOHN</td>
<td></td>
<td>MISS</td>
<td>$29,700.00</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>$21,780.00</td>
<td>MISS</td>
<td></td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>$27,062.00</td>
<td>MISS</td>
<td></td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
<td>$9,000.00</td>
<td>MISS</td>
<td></td>
</tr>
<tr>
<td>IRVING</td>
<td>JOAN</td>
<td>$26,862.00</td>
<td>MISS</td>
<td></td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td>$18,480.00</td>
<td>MISS</td>
<td></td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td>$18,480.00</td>
<td>MISS</td>
<td></td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>$16,100.00</td>
<td>MISS</td>
<td></td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>MISS</td>
<td></td>
<td>$21,120.00</td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td>$13,200.00</td>
<td>MISS</td>
<td></td>
</tr>
<tr>
<td>STEVENS</td>
<td>ALFRED</td>
<td>MISS</td>
<td></td>
<td>$11,000.00</td>
</tr>
</tbody>
</table>
Setting the NODATA Character String
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 63 joins can be in effect at one time, for a total of 256 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 726.
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.

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 737.
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 753.

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 727.
**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:

![Joined Data Structure Diagram](image)

**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.
- You cannot join token-delimited 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 727.
- 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:

  \[\text{SET PASS} = \text{pswd1 IN file1, pswd2 IN file2}\]

  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

**How to:**
Correct for Lagging Values With a Unique Join

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 contains 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
```
To join these two data sources, issue the following JOIN command, using the ALL phrase:

```sql
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.

```sql
SET JOINOPT={NEW|OLD}
```

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.
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:

<table>
<thead>
<tr>
<th>TRAIN_NUM</th>
<th>OR_STATION</th>
<th>OR_CITY</th>
<th>DE_STATION</th>
<th>DE_CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>NYC</td>
<td>NEW YORK</td>
<td>ATL</td>
<td>ATLANTA</td>
</tr>
<tr>
<td>202</td>
<td>BOS</td>
<td>BOSTON</td>
<td>BLT</td>
<td>BALTIMORE</td>
</tr>
<tr>
<td>303</td>
<td>DET</td>
<td>DETROIT</td>
<td>BOS</td>
<td>BOSTON</td>
</tr>
<tr>
<td>404</td>
<td>CHI</td>
<td>CHICAGO</td>
<td>DET</td>
<td>DETROIT</td>
</tr>
<tr>
<td>505</td>
<td>BOS</td>
<td>BOSTON</td>
<td>STL</td>
<td>ST. LOUIS</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>TRAIN_NUM</th>
<th>OR_STATION</th>
<th>OR_CITY</th>
<th>DE_STATION</th>
<th>DE_CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>NYC</td>
<td>NEW YORK</td>
<td>ATL</td>
<td>ATLANTA</td>
</tr>
<tr>
<td>202</td>
<td>BOS</td>
<td>BOSTON</td>
<td>BLT</td>
<td>BALTIMORE</td>
</tr>
<tr>
<td>303</td>
<td>DET</td>
<td>DETROIT</td>
<td>BOS</td>
<td>BOSTON</td>
</tr>
<tr>
<td>404</td>
<td>CHI</td>
<td>CHICAGO</td>
<td>DET</td>
<td>DETROIT</td>
</tr>
<tr>
<td>505</td>
<td>BOS</td>
<td>BOSTON</td>
<td>STL</td>
<td>ST. LOUIS</td>
</tr>
</tbody>
</table>
Recursive Joined Structures

Reference:
Usage Notes for 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.

```
FIELD_A
NAME
```

```
FIELD_B
ADDRESS
```

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).

![Diagram of joined structure]

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 732.

**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 748) 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 a company's aircraft. 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.

![Recursive Structure Diagram]

You can then produce a report on all three levels of data with this TABLE command (the field \text{SUBDESCRIPT} describes the contents of the field \text{SUBPART}):

\begin{verbatim}
TABLE FILE AIRCRAFT
PRINT SUBPART BY PART BY SUBPART BY SUBDESCRIPT
END
\end{verbatim}

**How the JOIN Command Works**

**Reference:**

Increasing Retrieval Speed in Joined Data Sources

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 63 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 737.

- 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 748.

- A condition you specify in the JOIN command itself. See How to Create a Conditional JOIN on page 754.

**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 1427.
Creating an Equijoin

In this section:
- Joining From a Virtual Field to a Real Field Using an Equijoin
- Data Formats of Shared Fields
- Joining Fields With Different Numeric Data Types

How to:
- Join Real Fields

Reference:
- Requirements for Cross-Referenced Fields in an Equijoin
- Restrictions on Group Fields
- Usage Notes for Inner and Outer JOIN Command Syntax

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 alphanumerical 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.

IN  hostfile

Is the name of the host file.

TAG  tag1

Is a tag name of up to eight characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in the host file.

The tag name for the host file must be the same in all the JOIN commands of a joined structure.
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 727 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 eight 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 732.

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 731.

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.
Creating an Equijoin

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.

```plaintext
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 JOBFILE
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:

```
The output is:

<table>
<thead>
<tr>
<th>EMP_ID</th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>CURR_JOBCODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>071382660</td>
<td>STEVENS</td>
<td>ALFRED</td>
<td>A07</td>
</tr>
<tr>
<td>112847612</td>
<td>SMITH</td>
<td>MARY</td>
<td>B14</td>
</tr>
<tr>
<td>117593129</td>
<td>JONES</td>
<td>DIANE</td>
<td>B03</td>
</tr>
<tr>
<td>119265415</td>
<td>SMITH</td>
<td>RICHARD</td>
<td>A01</td>
</tr>
<tr>
<td>119329144</td>
<td>BANNING</td>
<td>JOHN</td>
<td>A17</td>
</tr>
<tr>
<td>123764317</td>
<td>IRVING</td>
<td>JOAN</td>
<td>A15</td>
</tr>
<tr>
<td>126724188</td>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>B04</td>
</tr>
<tr>
<td>219984371</td>
<td>MCCOY</td>
<td>JOHN</td>
<td>B02</td>
</tr>
<tr>
<td>326179357</td>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>B04</td>
</tr>
<tr>
<td>451123478</td>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>B02</td>
</tr>
<tr>
<td>543729165</td>
<td>GREENSPAN</td>
<td>MARY</td>
<td>A07</td>
</tr>
<tr>
<td>818692173</td>
<td>CROSS</td>
<td>BARBARA</td>
<td>A17</td>
</tr>
<tr>
<td>111111111</td>
<td>NEWMAN</td>
<td>FRED</td>
<td>C07</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>JOB_DESC</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEVENS</td>
<td>ALFRED</td>
<td>SECRETARY</td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td>FILE QUALITY</td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td>PROGRAMMER ANALYST</td>
</tr>
<tr>
<td>SMITH</td>
<td>RICHARD</td>
<td>PRODUCTION CLERK</td>
</tr>
<tr>
<td>BANNING</td>
<td>JOHN</td>
<td>DEPARTMENT MANAGER</td>
</tr>
<tr>
<td>IRVING</td>
<td>JOAN</td>
<td>ASSIST.MANAGER</td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>SYSTEMS ANALYST</td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td>PROGRAMMER</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>SYSTEMS ANALYST</td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>PROGRAMMER</td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
<td>SECRETARY</td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>DEPARTMENT MANAGER</td>
</tr>
</tbody>
</table>
**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.

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>COURSE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEVENS</td>
<td>ALFRED</td>
<td>FILE DESCRPT &amp; MAINT</td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td>BASIC REPORT PREP FOR PROG</td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td>FOCUS INTERNALS</td>
</tr>
<tr>
<td>SMITH</td>
<td>RICHARD</td>
<td>BASIC RPT NON-DP MGRS</td>
</tr>
<tr>
<td>BANNING</td>
<td>JOHN</td>
<td></td>
</tr>
<tr>
<td>IRVING</td>
<td>JOAN</td>
<td></td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td></td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td></td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>DECISION SUPPORT WORKSHOP</td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>FILE DESCRPT &amp; MAINT</td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
<td></td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>HOST LANGUAGE INTERFACE</td>
</tr>
<tr>
<td>NEWMAN</td>
<td>FRED</td>
<td></td>
</tr>
</tbody>
</table>

**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
```

Creating Reports With WebFOCUS Language
The structure created by the two joins has two independent paths:

```
  SEG01
01    S1
**************
*EMP_ID      **I
*CURR_JOBCODE**I
*LAST_NAME   **
*FIRST_NAME  **
**************

I
+-----------------+
I                 I
I SEG01           I SEG01
02    I KM        03    I KM
**************    **************
:EMP_ID      ::K :JOBCODE     ::K
:COURSE_CODE :: :JOB_DESC    ::
:COURSE_NAME :: :
:            ::
:............::   :............::
:           ::   :
:JOINED  EDINFO JOINED  JOBINFO
```

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 DESCRIPT & 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 DESCRIPT & 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:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>COURSE_NAME</th>
<th>JOB_DESC</th>
</tr>
</thead>
<tbody>
<tr>
<td>STERVENS</td>
<td>ALFRED</td>
<td>FILE DESCRPT &amp; MAINT</td>
<td>SECRETARY</td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td>BASIC REPORT PREP FOR PROG</td>
<td>FILE QUALITY</td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td>FOCUS internals</td>
<td>PROGRAMMER ANALYST</td>
</tr>
<tr>
<td>SMITH</td>
<td>RICHARD</td>
<td>BASIC RPT NON-DP MGRS</td>
<td>PRODUCTION CLERK</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>DECISION SUPPORT WORKSHOP</td>
<td>SYSTEMS ANALYST</td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>FILE DESCRPT &amp; MAINT</td>
<td>PROGRAMMER</td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>HOST LANGUAGE INTERFACE</td>
<td>DEPARTMENT MANAGER</td>
</tr>
</tbody>
</table>

**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 Fusion 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 comma-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 data is not in the same sort order, errors are displayed. A many-to-many join is not supported. A comma-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.

**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

**How to:**
Join From a Virtual Field to a Real Field

**Reference:**
Notes on Using Virtual Fields With Joined Data Sources

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 757.
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 749.

- **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. 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 eight 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 727 for more information.

IN crfile

Is the name of the cross-referenced file.

TAG tag2

Is a tag name of up to eight characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in 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 *Usage Notes for Recursive Joined Structures* on page 732.

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 731.

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; terminates the command.

**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 757.

**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
```
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 752.

  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 746.

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

**How to:**

Enable Joins With Data Type Conversion

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.

**Syntax:**

**How to Enable Joins With Data Type Conversion**

To enable joins with data type conversion, issue the command

```plaintext
SET JOINOPT = [NEW | OLD]
```

where:

- **NEW**
  
  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:

```plaintext
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:

```plaintext
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**

**How to:**

Create 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, 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 the field name in the host Master File whose segment will be joined to the cross-referenced data source. The field name must be at the lowest level segment in its data source that is referenced.

**tag1**

Is the optional tag name that is used as a unique qualifier for fields and aliases in the host data source.

**hfld2**

Is a data source field with which to associate a DEFINE-based conditional JOIN. For a DEFINE-based conditional join, the KEEPDEFINES setting must be ON, and you must create the virtual fields before issuing the JOIN command.

**MULTIPLE**

Specifies a one-to-many relationship between `from_file` and `to_file`. Note that ALL is a synonym for MULTIPLE.
UNIQUE
Specifies a one-to-one relationship between from_file and to_file. Note that ONE is a synonym for UNIQUE.

**Note:** Unique returns only one instance and, if there is no matching instance in the cross-referenced file, it supplies default values (blank for alphanumeric fields and zero for numeric fields).

crfile
Is the cross-referenced Master File.

crfld
Is the join field name in the cross-referenced Master File. It can be any field in the segment.

tag2
Is the optional tag name that is used as a unique qualifier for fields and aliases in the cross-referenced data source.

joinname
Is the name associated with the joined structure.

expression1, expression2
Are any expressions that are acceptable in a DEFINE FILE command. All fields used in the expressions must lie on a single path.

**Note:** Single line JOIN syntax is not supported. The END command is required.

**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:

<table>
<thead>
<tr>
<th>Title</th>
<th>Last Transaction</th>
<th>Release Date</th>
<th>Years Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALICE IN WONDERLAND</td>
<td>91/06/22</td>
<td>51/07/21</td>
<td>39</td>
</tr>
<tr>
<td>ALIEN</td>
<td>91/06/18</td>
<td>80/04/04</td>
<td>11</td>
</tr>
<tr>
<td>ALL THAT JAZZ</td>
<td>91/06/25</td>
<td>80/05/11</td>
<td>11</td>
</tr>
<tr>
<td>ANNIE HALL</td>
<td>91/06/24</td>
<td>78/04/16</td>
<td>13</td>
</tr>
<tr>
<td>BAMBI</td>
<td>91/06/22</td>
<td>42/07/03</td>
<td>49</td>
</tr>
<tr>
<td>BIRDS, THE</td>
<td>91/06/23</td>
<td>63/09/27</td>
<td>27</td>
</tr>
<tr>
<td>CABARET</td>
<td>91/06/25</td>
<td>73/07/14</td>
<td>17</td>
</tr>
<tr>
<td>CASABLANCA</td>
<td>91/06/27</td>
<td>42/03/28</td>
<td>49</td>
</tr>
<tr>
<td>CITIZEN KANE</td>
<td>91/06/22</td>
<td>41/08/11</td>
<td>49</td>
</tr>
<tr>
<td>CYRANO DE BERGERAC</td>
<td>91/06/20</td>
<td>50/11/09</td>
<td>40</td>
</tr>
<tr>
<td>DEATH IN VENICE</td>
<td>91/06/26</td>
<td>73/07/27</td>
<td>17</td>
</tr>
<tr>
<td>DOG DAY AFTERNOON</td>
<td>91/06/23</td>
<td>76/04/04</td>
<td>15</td>
</tr>
<tr>
<td>EAST OF EDEN</td>
<td>91/06/20</td>
<td>55/01/12</td>
<td>36</td>
</tr>
<tr>
<td>GONE WITH THE WIND</td>
<td>91/06/24</td>
<td>39/06/04</td>
<td>52</td>
</tr>
<tr>
<td>JAWS</td>
<td>91/06/27</td>
<td>78/05/13</td>
<td>13</td>
</tr>
<tr>
<td>MALTESE FALCON, THE</td>
<td>91/06/19</td>
<td>41/11/14</td>
<td>49</td>
</tr>
<tr>
<td>MARTY</td>
<td>91/06/19</td>
<td>55/10/26</td>
<td>35</td>
</tr>
<tr>
<td>NORTH BY NORTHWEST</td>
<td>91/06/21</td>
<td>59/02/09</td>
<td>32</td>
</tr>
<tr>
<td>ON THE WATERFRONT</td>
<td>91/06/24</td>
<td>54/07/06</td>
<td>36</td>
</tr>
<tr>
<td>PHILADELPHIA STORY, THE</td>
<td>91/06/21</td>
<td>40/05/06</td>
<td>51</td>
</tr>
<tr>
<td>PSYCHO</td>
<td>91/06/17</td>
<td>60/05/16</td>
<td>31</td>
</tr>
<tr>
<td>REAR WINDOW</td>
<td>91/06/17</td>
<td>54/12/15</td>
<td>36</td>
</tr>
<tr>
<td>SHAGGY DOG, THE</td>
<td>91/06/25</td>
<td>59/01/09</td>
<td>32</td>
</tr>
<tr>
<td>SLEEPING BEAUTY</td>
<td>91/06/24</td>
<td>75/08/30</td>
<td>15</td>
</tr>
<tr>
<td>TIN DRUM, THE</td>
<td>91/06/17</td>
<td>80/03/01</td>
<td>11</td>
</tr>
<tr>
<td>VERTIGO</td>
<td>91/06/27</td>
<td>58/11/25</td>
<td>32</td>
</tr>
</tbody>
</table>
Preserving Virtual Fields During Join Parsing

In this section:
- Preserving Virtual Fields Using KEEPDEFINES
- Preserving Virtual Fields Using DEFINE FILE SAVE and RETURN
- Screening Segments With Conditional JOIN Expressions
- Parsing WHERE Criteria in a Join

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

How to:
- Use KEEPDEFINES

Reference:
- Usage Notes for 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

```plaintext
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:

```plaintext
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:

```plaintext
? 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:

```plaintext
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:

<table>
<thead>
<tr>
<th>MOVIECODE</th>
<th>TRANSDATE</th>
<th>RETURNDATE</th>
<th>DAYSKEPT</th>
<th>ACTUAL_DAYS</th>
</tr>
</thead>
<tbody>
<tr>
<td>001MCA</td>
<td>91/06/27</td>
<td>91/06/29</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>692PAR</td>
<td>91/06/27</td>
<td>91/06/29</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>259MGM</td>
<td>91/06/19</td>
<td>91/06/21</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>MOVIECODE</th>
<th>TRANSDATE</th>
<th>RETURNDATE</th>
<th>DAYSKEPT</th>
<th>ACTUAL_DAYS</th>
</tr>
</thead>
<tbody>
<tr>
<td>040ORI</td>
<td>91/06/20</td>
<td>91/06/21</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>505MGM</td>
<td>91/06/21</td>
<td>91/06/22</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>710VES</td>
<td>91/06/26</td>
<td>91/06/27</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>MOVIECODE</th>
<th>TRANSDATE</th>
<th>RETURNDATE</th>
<th>DAYSKEPT</th>
<th>ACTUAL_DAYS</th>
</tr>
</thead>
<tbody>
<tr>
<td>001MCA</td>
<td>91/06/27</td>
<td>91/06/29</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>692PAR</td>
<td>91/06/27</td>
<td>91/06/29</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>259MGM</td>
<td>91/06/19</td>
<td>91/06/21</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

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 193.

**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:

<table>
<thead>
<tr>
<th>FILE</th>
<th>FIELD NAME</th>
<th>FORMAT</th>
<th>SEGMENT</th>
<th>VIEW</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIDEOTRK A</td>
<td>A5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIDEOTRK B</td>
<td>A5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
```

The output of the `DEFINE` query shows that C is the only virtual field defined:

<table>
<thead>
<tr>
<th>FILE</th>
<th>FIELD NAME</th>
<th>FORMAT</th>
<th>SEGMENT</th>
<th>VIEW</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIDEOTRK C</td>
<td>A10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
```

The output of the `DEFINE` query shows that field C is still defined:

<table>
<thead>
<tr>
<th>FILE</th>
<th>FIELD NAME</th>
<th>FORMAT</th>
<th>SEGMENT</th>
<th>VIEW</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIDEOTRK C</td>
<td>A10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
```

The output of the `DEFINE` query shows that virtual fields C and D are defined:

<table>
<thead>
<tr>
<th>FILE</th>
<th>FIELD NAME</th>
<th>FORMAT</th>
<th>SEGMENT</th>
<th>VIEW</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIDEOTRK C</td>
<td>A10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIDEOTRK D</td>
<td>A10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The `DEFINE FILE RETURN` command clears virtual field D created in the current context (after the `JOIN`):

```
DEFINE FILE VIDEOTRK RETURN
END
```

The output of the `DEFINE` query shows that virtual field D was cleared, but C is still defined:

<table>
<thead>
<tr>
<th>FILE</th>
<th>FIELD NAME</th>
<th>FORMAT</th>
<th>SEGMENT</th>
<th>VIEW</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIDEOTRK C</td>
<td>A10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The following DEFINE FILE RETURN command does not clear virtual field C because field C was not created using a DEFINE FILE SAVE command:

```plaintext
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

<table>
<thead>
<tr>
<th>How to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display a Joined Structure</td>
</tr>
<tr>
<td>List Joined Structures</td>
</tr>
</tbody>
</table>

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. For an illustration, see *Displaying Joined Structures* on page 763.
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   SI
      *************
       *PIN      **I
       *LASTNAME  **
       *FIRSTNAME **
       *MIDINITIAL **
        *       **
      *************
       *************
        I
        I
         I
       I SLHISTORY
      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:

```sql
? JOIN
```

This displays every JOIN command currently in effect. For example:

```
JOINS CURRENTLY ACTIVE

HOST FIELD | CROSSREFERENCE FIELD FILE TAG AS ALL WH
------- ---- | --------------------- ---- --- -- --- --
JOBCODE JOBCODE | 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

In this section:
Clearing a Conditional Join
How to:
Clear a Join

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 765.

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 FIELD    FILE     TAG    CROSSREFERENCE 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
```

<table>
<thead>
<tr>
<th>HOST FIELD</th>
<th>FILE</th>
<th>TAG</th>
<th>CROSSREFERENCE FIELD</th>
<th>FILE</th>
<th>TAG</th>
<th>AS</th>
<th>ALL</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODCODE</td>
<td>VIDEOTRK</td>
<td>PCD</td>
<td>GGSALES</td>
<td>JW1</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOVIECODE</td>
<td>MOVIES</td>
<td>TRANSDATE</td>
<td>VIDEOTRK</td>
<td>JW3</td>
<td>N</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

**Topics:**
- Merging Data
- MATCH Processing
- MATCH Processing With Common High-Order Sort Fields
- Fine-Tuning MATCH Processing
- Universal Concatenation
- Merging Concatenated Data Sources
- Cartesian Product
Merging Data

How to:
Merge Data Sources

Reference:
Usage Notes for Match Requests

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. The merge process does not change the original data sources. For more information on HOLD files, see Saving and Reusing Your Report Output on page 351.

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 32 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 774.

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 single-segment HOLD file. You can print the contents of the HOLD file using the PRINT command with the wildcard character (*). For related information, see Merging Data Sources on page 769.

Reference: Usage Notes for Match Requests

- The ACROSS and WHERE TOTAL phrases, and the COMPUTE command, are not permitted in a MATCH request. You can, however, use the DEFINE command.
A total of 32 BY phrases and the maximum number of display fields can be used in each MATCH request. The maximum number of display fields is determined by a combination of factors.

For details, see Displaying Report Data on page 43.

Up to 32 sort sets are supported, including the number of common sort fields.

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.

You cannot use BY HIGHEST in a MATCH request.

The following prefix operators are not supported in MATCH requests: DST., DST.CNT., RNK., ST., and CT.

Example:  Merging Data Sources

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.

Note that if you are working at the command line in Developer Studio, after you enter the command RUN, a message indicates how many records were retrieved, and if you are entering the MATCH request at the command line, prompts you for the name of the next data source to be merged.
The output is:

<table>
<thead>
<tr>
<th>EMP_ID</th>
<th>COURSE_CODE</th>
<th>CURR_SAL</th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>071382660</td>
<td>101</td>
<td>$11,000.00</td>
<td>STEVENS</td>
<td>ALFRED</td>
</tr>
<tr>
<td>112847612</td>
<td>103</td>
<td>$13,200.00</td>
<td>SMITH</td>
<td>MARY</td>
</tr>
<tr>
<td>117593129</td>
<td>203</td>
<td>$18,480.00</td>
<td>JONES</td>
<td>DIANE</td>
</tr>
<tr>
<td>119265415</td>
<td>108</td>
<td>$9,500.00</td>
<td>SMITH</td>
<td>RICHARD</td>
</tr>
<tr>
<td>119329144</td>
<td></td>
<td>$29,700.00</td>
<td>BANNING</td>
<td>JOHN</td>
</tr>
<tr>
<td>123764317</td>
<td></td>
<td>$26,862.00</td>
<td>IRVING</td>
<td>JOAN</td>
</tr>
<tr>
<td>126724188</td>
<td></td>
<td>$21,120.00</td>
<td>ROMANS</td>
<td>ANTHONY</td>
</tr>
<tr>
<td>212289111</td>
<td>103</td>
<td>.00</td>
<td>MCCOY</td>
<td>JOHN</td>
</tr>
<tr>
<td>219984371</td>
<td></td>
<td>$18,480.00</td>
<td>MCCOY</td>
<td>JOHN</td>
</tr>
<tr>
<td>315548712</td>
<td>108</td>
<td>.00</td>
<td>MCCOY</td>
<td>JOHN</td>
</tr>
<tr>
<td>326179357</td>
<td>301</td>
<td>$21,780.00</td>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
</tr>
<tr>
<td>451123478</td>
<td>101</td>
<td>$16,100.00</td>
<td>MCKNIGHT</td>
<td>ROGER</td>
</tr>
<tr>
<td>543729165</td>
<td></td>
<td>$9,000.00</td>
<td>GREENSPAN</td>
<td>MARY</td>
</tr>
<tr>
<td>818692173</td>
<td>302</td>
<td>$27,062.00</td>
<td>CROSS</td>
<td>BARBARA</td>
</tr>
</tbody>
</table>

**MATCH Processing**

**How to:**

Specify Merge Phrases

**Reference:**

Merge Phrases

The way MATCH merges data depends on the order in which you name data sources in the request, the BY fields, display commands, 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 retrieved records common high-order sort fields as specified in the merge phrase (for example, OLD-OR-NEW). For more information, see Merge Phrases on page 774.

4. 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.

5. It writes final records to the HOLD file.
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](image1)

**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 and New](image2)

**OLD-NOT-NEW** specifies that records that appear only in the old data source appear in the HOLD file.

![Old and New](image3)
**NEW-NOT-OLD** specifies that records that appear only in the new data source appear in the HOLD file.

![Venn diagram showing OLD and New sets]

**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).

![Venn diagram showing OLD and New sets]

**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.

![Venn diagram showing OLD and New sets]

**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.

![Venn diagram showing OLD and New sets]

**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:

<table>
<thead>
<tr>
<th>EMPLOYEE Data Source</th>
<th>EDUCFILE Data Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP_ID</td>
<td>EMP_ID</td>
</tr>
<tr>
<td>071382660</td>
<td>071382660</td>
</tr>
<tr>
<td>119329144</td>
<td>212289111</td>
</tr>
<tr>
<td>112847612</td>
<td>112847612</td>
</tr>
<tr>
<td>LAST_NAME</td>
<td>COURSE_CODE</td>
</tr>
<tr>
<td>STEVENS</td>
<td>101</td>
</tr>
<tr>
<td>BANNING</td>
<td>103</td>
</tr>
<tr>
<td>SMITH</td>
<td>103</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>EMP_ID</th>
<th>LAST_NAME</th>
<th>COURSE_CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>071382660</td>
<td>STEVENS</td>
<td>101</td>
</tr>
<tr>
<td>112847612</td>
<td>SMITH</td>
<td>103</td>
</tr>
<tr>
<td>117593129</td>
<td>JONES</td>
<td>203</td>
</tr>
<tr>
<td>119265415</td>
<td>SMITH</td>
<td>108</td>
</tr>
<tr>
<td>119329144</td>
<td>BANNING</td>
<td></td>
</tr>
<tr>
<td>123764317</td>
<td>IRVING</td>
<td></td>
</tr>
<tr>
<td>126724188</td>
<td>ROMANS</td>
<td></td>
</tr>
<tr>
<td>212289111</td>
<td></td>
<td>103</td>
</tr>
<tr>
<td>219984371</td>
<td>MCCOY</td>
<td></td>
</tr>
<tr>
<td>315548712</td>
<td></td>
<td>108</td>
</tr>
<tr>
<td>326179357</td>
<td>BLACKWOOD</td>
<td>301</td>
</tr>
<tr>
<td>451123478</td>
<td>MCKNIGHT</td>
<td>101</td>
</tr>
<tr>
<td>543729165</td>
<td>GREENSPAN</td>
<td></td>
</tr>
<tr>
<td>818692173</td>
<td>CROSS</td>
<td>302</td>
</tr>
</tbody>
</table>
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 LAST_NAME
BY FIRST_NAME
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:

<table>
<thead>
<tr>
<th>EMP_ID</th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>LASTNAME</th>
<th>FIRSTNAME</th>
<th>PIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>071382660</td>
<td>STEVENS</td>
<td>ALFRED</td>
<td>ADAMS</td>
<td>RUTH</td>
<td>000000040</td>
</tr>
<tr>
<td>112847612</td>
<td>SMITH</td>
<td>MARY</td>
<td>ADDAMS</td>
<td>PETER</td>
<td>000000050</td>
</tr>
<tr>
<td>117593129</td>
<td>JONES</td>
<td>DIANE</td>
<td>ANDERSON</td>
<td>TIM</td>
<td>000000100</td>
</tr>
<tr>
<td>119265415</td>
<td>SMITH</td>
<td>RICHARD</td>
<td>BELLA</td>
<td>MICHAEL</td>
<td>000000020</td>
</tr>
<tr>
<td>119329144</td>
<td>BANNING</td>
<td>JOHN</td>
<td>CASSANOVA</td>
<td>LOIS</td>
<td>000000030</td>
</tr>
<tr>
<td>123764317</td>
<td>IRVING</td>
<td>JOAN</td>
<td>CASTALANETTA</td>
<td>MARIE</td>
<td>000000270</td>
</tr>
<tr>
<td>126724188</td>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>CHISOLM</td>
<td>HENRY</td>
<td>000000360</td>
</tr>
<tr>
<td>219984371</td>
<td>MCCOY</td>
<td>JOHN</td>
<td>CONRAD</td>
<td>ADAM</td>
<td>000000250</td>
</tr>
<tr>
<td>326179357</td>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>CONTI</td>
<td>MARSHALL</td>
<td>000000410</td>
</tr>
<tr>
<td>451123478</td>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>CVEK</td>
<td>MARCUS</td>
<td>000000130</td>
</tr>
<tr>
<td>543729165</td>
<td>GREENSPAN</td>
<td>MARY</td>
<td>DONATELLO</td>
<td>ERICA</td>
<td>000000320</td>
</tr>
<tr>
<td>818692173</td>
<td>CROSS</td>
<td>BARBARA</td>
<td>DUBOIS</td>
<td>ERIC</td>
<td>000000210</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ELLNER</td>
<td>DAVID</td>
<td>000000380</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FERNESTIN</td>
<td>ERWIN</td>
<td>000000350</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GORDON</td>
<td>LAURA</td>
<td>000000180</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GOTLIBE</td>
<td>CHRIS</td>
<td>000000340</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GRAFF</td>
<td>ELAINE</td>
<td>000000390</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HIRSCHMAN</td>
<td>ROSE</td>
<td>000000160</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>KASHMAN</td>
<td>YOLANDA</td>
<td>000000240</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LASTRA</td>
<td>KAREN</td>
<td>000000200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LEWIS</td>
<td>CASSANDRA</td>
<td>000000220</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>100</td>
<td>b</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>y</td>
<td>200</td>
<td>d</td>
<td>70</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>100</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>100</td>
<td>b</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>y</td>
<td>200</td>
<td>c</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>y</td>
<td>200</td>
<td>d</td>
<td>40</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>100</td>
<td>b</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>y</td>
<td>200</td>
<td>d</td>
<td>70</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>100</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>b</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>y</td>
<td>200</td>
<td>c</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>d</td>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>100</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>100</td>
<td>b</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>y</td>
<td>200</td>
<td>c</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>y</td>
<td>200</td>
<td>d</td>
<td>40</td>
</tr>
</tbody>
</table>

Note that the records for file A are printed for every occurrence of the record in file B.
Universal Concatenation

In this section:
- Field Name and Format Matching

How to:
- Concatenate Data Sources

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 143.

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 file1
    main 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 EXPERSO 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.
- EXPERSO contains the field values SSN=987654321 and WAGE=100.00.

The following annotated request concatenates the two data sources:

```sql
DEFINE FILE EXPERSO
1. EMP_ID/A9  = SSN;
   CURR_SAL/D12.2  = WAGE;
END
2. TABLE FILE EMPLOYEE
   PRINT CURR_SAL
   BY EMP_ID
3. MORE
   FILE EXPERSO
END
```

1. The request must re-map the field names and formats in the EXPERSO 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, EXPERSO. 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 re-mapped 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:

<table>
<thead>
<tr>
<th>Usage Format Type</th>
<th>Correspondence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Format type and length must be equal.</td>
</tr>
<tr>
<td>I, F, D</td>
<td>Format type must be the same.</td>
</tr>
<tr>
<td>P</td>
<td>Format type and scale must be equal.</td>
</tr>
<tr>
<td>DATE (new)</td>
<td>Format information (type, length, components, and order) must always correspond.</td>
</tr>
<tr>
<td>DATE (old)</td>
<td>Edit options must be the same.</td>
</tr>
<tr>
<td>DATE -TIME</td>
<td>Format information (type, length, components, and order) must always correspond.</td>
</tr>
</tbody>
</table>

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. *Master Files and Diagrams* on page 1441, contains the Master Files referenced in the request.

```
DEFINE FILE EMPDATA
  1. NEWID/A11=EDIT (ID,'999-99-9999');
  END

DEFINE FILE SALHIST
  2. NEWID/A11=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
  6. FILE SALHIST
     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.
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 file1 main request
   MORE
   2. FILE file2 subrequest
      MORE
      3. FILE file3 subrequest
         RUN
      4. FILE file4 main request
      5. [AFTER MATCH merge_phrase]
         MORE
      6. FILE file5 subrequest
         MORE
      7. FILE file6 subrequest
         RUN
      8. FILE file7 main request
      9. [AFTER MATCH merge_phrase]
         MORE
     10. FILE file8 subrequest
         MORE
     11. FILE file9 subrequest
         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:

<table>
<thead>
<tr>
<th>SSN</th>
<th>CURRENT</th>
<th>FIRST</th>
<th>EXPENSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000010</td>
<td>$55,500.00</td>
<td>DANIEL</td>
<td>2,300.00</td>
</tr>
<tr>
<td>000000020</td>
<td>$62,500.00</td>
<td>MICHAEL</td>
<td>.</td>
</tr>
<tr>
<td>000000030</td>
<td>$70,000.00</td>
<td>LOIS</td>
<td>2,600.00</td>
</tr>
<tr>
<td>000000030</td>
<td>$70,000.00</td>
<td>LOIS</td>
<td>2,300.00</td>
</tr>
<tr>
<td>000000040</td>
<td>$62,500.00</td>
<td>RUTH</td>
<td>3,400.00</td>
</tr>
<tr>
<td>000000050</td>
<td>$54,100.00</td>
<td>PETER</td>
<td>3,300.00</td>
</tr>
<tr>
<td>000000060</td>
<td>$55,500.00</td>
<td>DORINA</td>
<td>.</td>
</tr>
<tr>
<td>000000070</td>
<td>$83,000.00</td>
<td>EVELYN</td>
<td>.</td>
</tr>
<tr>
<td>000000080</td>
<td>$43,400.00</td>
<td>PAMELA</td>
<td>3,200.00</td>
</tr>
<tr>
<td>000000080</td>
<td>$43,400.00</td>
<td>PAMELA</td>
<td>3,350.00</td>
</tr>
<tr>
<td>000000090</td>
<td>$33,000.00</td>
<td>MARIANNE</td>
<td>.</td>
</tr>
<tr>
<td>000000100</td>
<td>$32,400.00</td>
<td>TIM</td>
<td>3,100.00</td>
</tr>
<tr>
<td>000000110</td>
<td>$19,300.00</td>
<td>ANTHONY</td>
<td>1,800.00</td>
</tr>
<tr>
<td>000000110</td>
<td>$19,300.00</td>
<td>ANTHONY</td>
<td>2,500.00</td>
</tr>
<tr>
<td>000000110</td>
<td>$19,300.00</td>
<td>ANTHONY</td>
<td>2,400.00</td>
</tr>
<tr>
<td>000000120</td>
<td>$49,500.00</td>
<td>KATE</td>
<td>2,200.00</td>
</tr>
<tr>
<td>000000130</td>
<td>$62,500.00</td>
<td>MARCUS</td>
<td>.</td>
</tr>
</tbody>
</table>
**Example:** Merging Concatenated Data Sources Without a Common Sort Field

In this example, the merged data sources do not share a sort field:

```hll
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:

<table>
<thead>
<tr>
<th>SSN</th>
<th>CURRENT</th>
<th>FIRST</th>
<th>EID</th>
<th>EXPENSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000010</td>
<td>$55,500.00</td>
<td>DANIEL</td>
<td>000000010</td>
<td>2,300.00</td>
</tr>
<tr>
<td>000000020</td>
<td>$62,500.00</td>
<td>MICHAEL</td>
<td>000000030</td>
<td>2,600.00</td>
</tr>
<tr>
<td>000000030</td>
<td>$70,000.00</td>
<td>LOIS</td>
<td>000000030</td>
<td>2,300.00</td>
</tr>
<tr>
<td>000000040</td>
<td>$62,500.00</td>
<td>RUTH</td>
<td>000000040</td>
<td>3,400.00</td>
</tr>
<tr>
<td>000000050</td>
<td>$54,100.00</td>
<td>PETER</td>
<td>000000050</td>
<td>3,300.00</td>
</tr>
<tr>
<td>000000060</td>
<td>$55,500.00</td>
<td>DORINA</td>
<td>000000080</td>
<td>3,200.00</td>
</tr>
<tr>
<td>000000070</td>
<td>$83,000.00</td>
<td>EVELYN</td>
<td>000000080</td>
<td>3,350.00</td>
</tr>
<tr>
<td>000000080</td>
<td>$43,400.00</td>
<td>PAMELA</td>
<td>000000100</td>
<td>3,100.00</td>
</tr>
<tr>
<td>000000090</td>
<td>$33,000.00</td>
<td>MARIANNE</td>
<td>000000110</td>
<td>1,800.00</td>
</tr>
<tr>
<td>000000100</td>
<td>$32,400.00</td>
<td>TIM</td>
<td>000000110</td>
<td>2,500.00</td>
</tr>
<tr>
<td>000000110</td>
<td>$19,300.00</td>
<td>ANTHONY</td>
<td>000000110</td>
<td>2,400.00</td>
</tr>
<tr>
<td>000000120</td>
<td>$49,500.00</td>
<td>KATE</td>
<td>000000120</td>
<td>2,200.00</td>
</tr>
<tr>
<td>000000130</td>
<td>$62,500.00</td>
<td>MARCUS</td>
<td>000000140</td>
<td>3,600.00</td>
</tr>
<tr>
<td>000000140</td>
<td>$62,500.00</td>
<td>VERONICA</td>
<td>000000150</td>
<td>3,400.00</td>
</tr>
<tr>
<td>000000150</td>
<td>$40,900.00</td>
<td>KARL</td>
<td>000000160</td>
<td>1,000.00</td>
</tr>
<tr>
<td>000000160</td>
<td>$62,500.00</td>
<td>ROSE</td>
<td>000000180</td>
<td>1,250.00</td>
</tr>
<tr>
<td>000000170</td>
<td>$30,800.00</td>
<td>WILLIAM</td>
<td>000000190</td>
<td>3,150.00</td>
</tr>
</tbody>
</table>

**Cartesian Product**

How to:

Enable/Disable Cartesian Product

Reference:

Usage Notes for 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 143.
**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:

<table>
<thead>
<tr>
<th>CAR</th>
<th>MODEL</th>
<th>STANDARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAGUAR</td>
<td>V12XKE AUTO</td>
<td>4 WHEEL DISC BRAKES</td>
</tr>
<tr>
<td></td>
<td>V12XKE AUTO</td>
<td>POWER STEERING</td>
</tr>
<tr>
<td></td>
<td>V12XKE AUTO</td>
<td>RECLINING BUCKET SEATS</td>
</tr>
<tr>
<td></td>
<td>V12XKE AUTO</td>
<td>WHITETWALL RADIAL PLY TIRES</td>
</tr>
<tr>
<td></td>
<td>V12XKE AUTO</td>
<td>WRAP AROUND BUMPERS</td>
</tr>
<tr>
<td></td>
<td>XJ12L AUTO</td>
<td>4 WHEEL DISC BRAKES</td>
</tr>
<tr>
<td></td>
<td>XJ12L AUTO</td>
<td>POWER STEERING</td>
</tr>
<tr>
<td></td>
<td>XJ12L AUTO</td>
<td>RECLINING BUCKET SEATS</td>
</tr>
<tr>
<td></td>
<td>XJ12L AUTO</td>
<td>WHITETWALL RADIAL PLY TIRES</td>
</tr>
<tr>
<td></td>
<td>XJ12L AUTO</td>
<td>WRAP AROUND BUMPERS</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>CAR</th>
<th>MODEL</th>
<th>STANDARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAGUAR</td>
<td>V12XKE AUTO</td>
<td>4 WHEEL DISC BRAKES</td>
</tr>
<tr>
<td></td>
<td>XJ12L AUTO</td>
<td>POWER STEERING</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>RECLINING BUCKET SEATS</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>WHITETWALL RADIAL PLY TIRES</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>WRAP AROUND BUMPERS</td>
</tr>
</tbody>
</table>
16 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 1245.
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 1205.

- **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 1053.

- **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 929.

- **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 825.

- **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 429.
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 show 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):

<table>
<thead>
<tr>
<th>Date Of Order</th>
<th>Order Number</th>
<th>Line Of Order</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct 1, 00</td>
<td>79613</td>
<td>2000/10/01</td>
<td>$1,417.29</td>
</tr>
<tr>
<td>Oct 2, 00</td>
<td>78360</td>
<td>2000/10/02</td>
<td>$5,485.05</td>
</tr>
<tr>
<td>Oct 3, 00</td>
<td>79327</td>
<td>2000/10/03</td>
<td>$628,241.37</td>
</tr>
<tr>
<td>Oct 4, 00</td>
<td>78373</td>
<td>2000/10/04</td>
<td>$2,659.47</td>
</tr>
<tr>
<td>Oct 5, 00</td>
<td>74577</td>
<td>2000/10/05</td>
<td>$209,286.00</td>
</tr>
<tr>
<td>Oct 6, 00</td>
<td>88950</td>
<td>2000/10/06</td>
<td>$3,670.68</td>
</tr>
<tr>
<td>Oct 7, 00</td>
<td>88840</td>
<td>2000/10/07</td>
<td>$4,251.87</td>
</tr>
<tr>
<td>Oct 8, 00</td>
<td>66977</td>
<td>2000/10/08</td>
<td>$669,979.35</td>
</tr>
<tr>
<td>Oct 9, 00</td>
<td>66974</td>
<td>2000/10/09</td>
<td>$122,298.40</td>
</tr>
<tr>
<td>Oct 10, 00</td>
<td>68922</td>
<td>2000/10/10</td>
<td>$867,979.35</td>
</tr>
<tr>
<td>Oct 11, 00</td>
<td>68974</td>
<td>2000/10/11</td>
<td>$122,298.40</td>
</tr>
</tbody>
</table>

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

In this section:

How to Choose a Type of Style Sheet

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 803.
**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:

<table>
<thead>
<tr>
<th>Date Of Order</th>
<th>Order Number</th>
<th>Order Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct 1, 00</td>
<td>79613</td>
<td>$1,417.29</td>
</tr>
<tr>
<td>Oct 2, 00</td>
<td>78360</td>
<td>$5,485.05</td>
</tr>
<tr>
<td>Oct 3, 00</td>
<td>79327</td>
<td><strong>$628,241.37</strong></td>
</tr>
<tr>
<td>Oct 4, 00</td>
<td>78373</td>
<td>$2,650.47</td>
</tr>
<tr>
<td>Oct 6, 00</td>
<td>74577</td>
<td>$209,286.00</td>
</tr>
<tr>
<td>Oct 7, 00</td>
<td>88950</td>
<td>$3,670.68</td>
</tr>
<tr>
<td>Oct 9, 00</td>
<td>68840</td>
<td>$4,251.87</td>
</tr>
<tr>
<td><strong>Oct 10, 00</strong></td>
<td><strong>68922</strong></td>
<td><strong>$867,979.35</strong></td>
</tr>
<tr>
<td>Oct 11, 00</td>
<td>68974</td>
<td>$122,298.40</td>
</tr>
</tbody>
</table>
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. "C e n t u r y C o r p o r a t i o n"
    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,
       WHEN=LinePrice GT 500000, $
    7. TYPE=Data, COLUMN=LINEPRICE, BACKCOLOR=Aqua, COLOR=Navy, STYLE=Bold,
       WHEN=LinePrice GT 500000, $
    8. TYPE=Title, FONT='Helvetica', $
    9. TYPE=Heading, FONT='Helvetica', STYLE=Bold, SIZE=14, JUSTIFY=Center,
       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 826.

- **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 819.

- **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 816.

- **Tools.** You can use Report Painter to generate most formatting code in a procedure. In some cases, you can generate formatting code with Report Painter and then edit it manually to take advantage of special code-based formatting techniques. (Note that in a few cases, after you manually edit generated code, you may not be able to reopen it in the tool that generated it, though you will always be able to edit it manually.) For more information about formatting reports using Report Painter, see the Creating Reports With Report Painter manual.

- **Samples.** Developer Studio comes with several sample StyleSheets. You can apply one of these samples to a report. You can also use a sample as a template, first copying it and then customizing the copy. You can find sample StyleSheets in the IBI\desktop52\home\bin directory. They are also listed in Report Painter (in the Style tab of the Report Options dialog box).
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 520.
- **Link to a Web page**, compose and send an e-mail message, and connect to other kinds of Internet resources. For more information, see *Linking to a URL* on page 525.
- **Execute JavaScript functions** to perform additional analysis of report data. For more information, see *Linking to a JavaScript Function* on page 528.
- **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 640.
- **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 685.
Creating and Managing a WebFOCUS StyleSheet

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 891 and Controlling Report Formatting on page 825.

Unless otherwise noted, all StyleSheet references in this chapter refer to WebFOCUS StyleSheets.

Topics:
- Creating a WebFOCUS StyleSheet
- General WebFOCUS StyleSheet Syntax
- Reusing WebFOCUS StyleSheet Declarations With Macros
- WebFOCUS StyleSheet Attribute Inheritance
Creating a WebFOCUS StyleSheet

In this section:
Creating a WebFOCUS StyleSheet Within a Report Request
Creating and Applying a WebFOCUS StyleSheet File

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 808.

- **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 810.

Creating a WebFOCUS StyleSheet Within a Report Request

**How to:**
Create 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.

dehlaration

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 812.

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
Creating a WebFOCUS StyleSheet

The output is:

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>

**End of Report**

Creating and Applying a WebFOCUS StyleSheet File

How to:
Apply a WebFOCUS StyleSheet File to a Report

Reference:
Naming and Storing a WebFOCUS StyleSheet File
Sample WebFOCUS StyleSheet Files

You can create a StyleSheet as a separate file and apply it to as many reports as you wish. A StyleSheet file contains declarations and optional comments and nothing else (unlike an inline StyleSheet, it does not contain the ON TABLE SET STYLE and ENDSTYLE commands). You can apply a StyleSheet file to a report using the SET STYLE and ENDSTYLE command, as described in How to Apply a WebFOCUS StyleSheet File to a Report on page 811. For information about StyleSheet declarations, see General WebFOCUS StyleSheet Syntax on page 812.
As an alternative to creating a StyleSheet file from scratch, you can copy one of the sample StyleSheet files provided with WebFOCUS and Developer Studio to use as a template. The sample StyleSheet files are described in Sample WebFOCUS StyleSheet Files on page 812.

Whether you create a StyleSheet file from scratch 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 811.

After you have created a StyleSheet file, you can check it for syntax errors using the CHECK STYLE command, as described in Validating WebFOCUS StyleSheet Syntax on page 815.

Reference:  Naming and Storing a WebFOCUS StyleSheet File

When you create a StyleSheet file to be used in:

- **WebFOCUS**, save the file in the WebFOCUS#asedir\appname\app directory (where # is the release number). For example, use WebFOCUS76\basedir\appname\app for Release 7.6.1.

  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.

- **Developer Studio**, for development or running a report locally, it should reside in the application folder or in the ibi\devstudio#\bin directory (where # is the release number). For example, use ibi\devstudio76\bin for Release 7.6.1.

  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.
Reference: Sample WebFOCUS StyleSheet Files

Many sample StyleSheet files are available in Developer Studio in the ibi\devstudio#\bin directory (where # is the release number). For example, use ibi\devstudio76\bin for Release 7.6.1.

You can apply any of these sample StyleSheets directly to a report, or you can use them as templates to create your own customized StyleSheets. If you use them as templates, be sure to make a copy of the original file and edit the copy, leaving the original unchanged.

The following is a list of the file names for the sample StyleSheets available in the Report Painter and Report Assistant:

altbycol.sty  deflt4.sty  proslate.sty
basicycel.sty  defltblu.sty  sales.sty
blackred.sty  elephant.sty  slate.sty
boyscout.sty  eursteel.sty  soupcan.sty
citrus.sty  fclabels.sty  sports.sty
classic.sty  fuschia.sty  teal.sty
corporat.sty  greenbar.sty  usa.sty
deflt.sty  marble.sty  usasty.sty
deflt1.sty  olive.sty  wood.sty
deflt2.sty  paper.sty  yankee.sty
deflt3.sty  problue.sty

General WebFOCUS StyleSheet Syntax

In this section:

Improving WebFOCUS StyleSheet Readability
Adding a Comment to a WebFOCUS StyleSheet
Validating WebFOCUS StyleSheet Syntax

How to:

Specify a WebFOCUS StyleSheet Declaration

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 815.
For information about identifying a report component, see Identifying a Report Component in a WebFOCUS StyleSheet on page 849.

**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.

```plaintext
TABLE FILE CENTORD
HEADING " "
"Century Corporation" " "
"Order Revenue - 2000 Q3" " "
"page <TABPAGENO" " "
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 STYLESET *
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.

Validating WebFOCUS StyleSheet Syntax

You can check a StyleSheet file to see if there are any syntax errors in it using the CHECK STYLE command. CHECK STYLE does not check the spelling of font names, nor does it verify the availability of specified fonts on your system.

You can issue the CHECK STYLE command in the Command Console in Developer Studio.
**Syntax:**  How to Validate WebFOCUS StyleSheet Syntax

To validate the syntax of a StyleSheet file, issue this command from the Command Console in Developer Studio:

```
CHECK STYLE file
```

where:

- **file**
  
  Is the name of the StyleSheet file. Do not include the file extension.

Reusing WebFOCUS StyleSheet Declarations With Macros

**In this section:**

- Defining a WebFOCUS StyleSheet Macro
- Applying a WebFOCUS StyleSheet Macro

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**

**How to:**

Define 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

How to:

Apply 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. For an example, see Defining, Applying, and Overriding a WebFOCUS StyleSheet Macro on page 818.

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 849.

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 825.
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 *
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).
Sales Report

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>

Sales Report - Page 1

WebFOCUS StyleSheet Attribute Inheritance

Reference:
WebFOCUS StyleSheet Inheritance Hierarchy

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:

```text
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 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 attributes 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.

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 913.

**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

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>
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
SUM UNITS DOLLARS
BY CATEGORY BY PRODUCT
HEADING
"Sales Report"
ON TABLE SET PAGE-NUM OFF
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**

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>

**Page 1 of 5**
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.

**Topics:**

- Generating an Internal Cascading Style Sheet for HTML Reports
- Selecting a Unit of Measurement
- Conditionally Formatting, Displaying, and Linking in a StyleSheet
- 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

How to:
Generate an Internal Cascading Style Sheet

Reference:
Requirements for Internal Cascading Style Sheets

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 827. For more information on all other attributes, see *Laying Out the Report Page* on page 929.

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 891.

**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. This is the default for reports created in the Report Painter.

**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. OFF is the default value.

**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**

**How to:**
Set the 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**

**In this section:**

Applying Sequential Conditional Formatting

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.
Applying Sequential Conditional Formatting

How to:
Conditionally Format, Display, or Link in a StyleSheet

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).

For an example, see Using Sequential Conditional Formatting on page 831.

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 849.
**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 sort field in a graph or tabular report. You cannot specify a horizontal sort field (ACROSS).
- 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, date, or date-time. 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**

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Date Of Order</th>
<th>Order Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>94710</td>
<td>2001/01/02</td>
<td>$406,964.24</td>
</tr>
<tr>
<td>94680</td>
<td>2001/01/02</td>
<td>$421,916.60</td>
</tr>
<tr>
<td>94670</td>
<td>2001/01/02</td>
<td>$513,868.76</td>
</tr>
<tr>
<td>94550</td>
<td>2001/01/02</td>
<td>$496,323.64</td>
</tr>
<tr>
<td>94530</td>
<td>2001/01/02</td>
<td>$3,472.41</td>
</tr>
<tr>
<td>94520</td>
<td>2001/01/02</td>
<td>$261,808.72</td>
</tr>
<tr>
<td>94490</td>
<td>2000/12/31</td>
<td>$633,723.06</td>
</tr>
<tr>
<td>94460</td>
<td>2001/01/02</td>
<td>$3,872.39</td>
</tr>
<tr>
<td>94430</td>
<td>2001/01/02</td>
<td>$3,033.38</td>
</tr>
<tr>
<td>94410</td>
<td>2001/01/02</td>
<td>$2,337.28</td>
</tr>
</tbody>
</table>
**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**

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Date Of Order</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>94710</td>
<td>2001/01/02</td>
<td>$406,964.24</td>
</tr>
<tr>
<td>94680</td>
<td>2001/01/02</td>
<td>$421,916.60</td>
</tr>
<tr>
<td>94670</td>
<td>2001/01/02</td>
<td>$513,868.76</td>
</tr>
<tr>
<td>94550</td>
<td>2001/01/02</td>
<td>$496,323.64</td>
</tr>
<tr>
<td>94530</td>
<td>2001/01/02</td>
<td>$3,472.41</td>
</tr>
<tr>
<td>94520</td>
<td>2001/01/02</td>
<td>$261,808.72</td>
</tr>
<tr>
<td>94490</td>
<td>2000/12/31</td>
<td>$633,723.06</td>
</tr>
<tr>
<td>94460</td>
<td>2001/01/02</td>
<td>$3,872.39</td>
</tr>
<tr>
<td>94430</td>
<td>2001/01/02</td>
<td>$3,033.38</td>
</tr>
<tr>
<td>94410</td>
<td>2001/01/02</td>
<td>$2,337.28</td>
</tr>
</tbody>
</table>
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 STYLESHEET *
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

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Date Of Order</th>
<th>Order Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>94710</td>
<td>2001/01/02</td>
<td>$406,964.24</td>
</tr>
<tr>
<td>94680</td>
<td>2001/01/02</td>
<td>$421,916.60</td>
</tr>
<tr>
<td>94670</td>
<td>2001/01/02</td>
<td>$513,868.76</td>
</tr>
<tr>
<td>94550</td>
<td>2001/01/02</td>
<td>$496,323.64</td>
</tr>
<tr>
<td>94530</td>
<td>2001/01/02</td>
<td>$3,472.41</td>
</tr>
<tr>
<td>94520</td>
<td>2001/01/02</td>
<td>$261,808.72</td>
</tr>
<tr>
<td>94490</td>
<td>2000/12/31</td>
<td>$633,723.06</td>
</tr>
<tr>
<td>94460</td>
<td>2001/01/02</td>
<td>$3,872.39</td>
</tr>
<tr>
<td>94430</td>
<td>2001/01/02</td>
<td>$3,033.38</td>
</tr>
<tr>
<td>94410</td>
<td>2001/01/02</td>
<td>$2,337.28</td>
</tr>
</tbody>
</table>
**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.

```sql
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, $
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**

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Date Of Order</th>
<th>Order Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>94710</td>
<td>2001/01/02</td>
<td>$406,964.24</td>
</tr>
<tr>
<td>94680</td>
<td>2001/01/02</td>
<td>$421,916.60</td>
</tr>
<tr>
<td>94670</td>
<td>2001/01/02</td>
<td>$513,868.76</td>
</tr>
<tr>
<td>94550</td>
<td>2001/01/02</td>
<td>$496,323.64</td>
</tr>
<tr>
<td>94530</td>
<td>2001/01/02</td>
<td>$3,472.41</td>
</tr>
<tr>
<td>94520</td>
<td>2001/01/02</td>
<td>$261,808.72</td>
</tr>
<tr>
<td>94490</td>
<td>2000/12/31</td>
<td>$633,723.06</td>
</tr>
<tr>
<td>94460</td>
<td>2001/01/02</td>
<td>$3,372.39</td>
</tr>
<tr>
<td>94430</td>
<td>2001/01/02</td>
<td>$3,033.38</td>
</tr>
<tr>
<td>94410</td>
<td>2001/01/02</td>
<td>$2,337.28</td>
</tr>
</tbody>
</table>
**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.

```plaintext
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
```
# 18. Controlling Report Formatting

## Employee List for Boston

### For Pay Levels 5+

**Resigned Employees Shown in Red Bold**

<table>
<thead>
<tr>
<th>Employee ID#</th>
<th>Last Name</th>
<th>First Name</th>
<th>Pay Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>GLIOZZO</td>
<td>ANTHONY</td>
<td>5</td>
</tr>
<tr>
<td>46</td>
<td>HEBERT</td>
<td>BRYAN</td>
<td>5</td>
</tr>
<tr>
<td>70</td>
<td>MIKITKA</td>
<td>MARK</td>
<td>5</td>
</tr>
<tr>
<td>79</td>
<td>PALMER</td>
<td>TED</td>
<td>5</td>
</tr>
<tr>
<td>85</td>
<td>PHILLIPS</td>
<td>CLAIRE</td>
<td>5</td>
</tr>
<tr>
<td>91</td>
<td>ROUSSEAU</td>
<td>DIANE</td>
<td>5</td>
</tr>
<tr>
<td>104</td>
<td>WHEELAN</td>
<td>JAMES</td>
<td>5</td>
</tr>
<tr>
<td>129</td>
<td>HAZARD</td>
<td>DAVID</td>
<td>6</td>
</tr>
<tr>
<td>142</td>
<td>FLYNN</td>
<td>PAUL</td>
<td>6</td>
</tr>
</tbody>
</table>
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:
```

<table>
<thead>
<tr>
<th>Status</th>
<th>Last</th>
<th>First</th>
<th>Pay Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECLINED</td>
<td>ROUSSEAU DIANE</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>EMPLOYED</td>
<td>MIKITKA MARK</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WHELEHAN JAMES</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FLynn PAUL</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>RESIGNED</td>
<td>HEBERT BRYAN</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PHILLIPS CLAIRE</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HAZARD DAVID</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>TERMINAT</td>
<td>GIOZZO ANTHONY</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PALMER TED</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
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):

```plaintext
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 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, $
ENDSTYLE

END
```

The output is:

<table>
<thead>
<tr>
<th>Employee List for Boston</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>For Pay Levels 5+</strong></td>
</tr>
<tr>
<td>Current Status</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>DECLINED</td>
</tr>
<tr>
<td>EMPLOYED</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>RESIGNED</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TERMINAT</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
**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, $
END
```

The output is:

<table>
<thead>
<tr>
<th>Category</th>
<th>PERIOD</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
<th>MOVAVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1</td>
<td>61666</td>
<td>801123</td>
<td>801,123</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>54870</td>
<td>682340</td>
<td>741,731.5</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>61608</td>
<td>765078</td>
<td>749,513.7</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>57050</td>
<td>691274</td>
<td>712,897.3</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>59229</td>
<td>720444</td>
<td>725,598.7</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>58466</td>
<td>742457</td>
<td>718,058.3</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>60771</td>
<td>747253</td>
<td>736,718.0</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>54633</td>
<td>655896</td>
<td>715,202.0</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>57829</td>
<td>730327</td>
<td>711,158.7</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>57012</td>
<td>724412</td>
<td>703,545.0</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>51110</td>
<td>620264</td>
<td>691,667.7</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>58981</td>
<td>762328</td>
<td>702,334.7</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>694,975.6</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>719,879.4</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>705,729.9</td>
</tr>
</tbody>
</table>
```
Conditionally Including Summary Lines, Underlines, Skipped Lines, and Page Breaks

How to:
Conditionally Display 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 315.
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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>94710</td>
<td>2001/01/02</td>
<td>$406,964.24</td>
</tr>
<tr>
<td>94680</td>
<td>2001/01/02</td>
<td>$421,916.60</td>
</tr>
<tr>
<td>94670</td>
<td>2001/01/02</td>
<td>$513,868.76</td>
</tr>
<tr>
<td>94550</td>
<td>2001/01/02</td>
<td>$496,323.64</td>
</tr>
<tr>
<td>94530</td>
<td>2001/01/02</td>
<td>$3,472.41</td>
</tr>
</tbody>
</table>

--- 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.

```plaintext
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**

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Date Of Order</th>
<th>Order Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>94710</td>
<td>2001/01/02</td>
<td>$406,964.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+++ Order total is greater than or equal to $200,000 +++</td>
</tr>
<tr>
<td>94680</td>
<td>2001/01/02</td>
<td>$421,916.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+++ Order total is greater than or equal to $200,000 +++</td>
</tr>
<tr>
<td>94670</td>
<td>2001/01/02</td>
<td>$513,868.76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+++ Order total is greater than or equal to $200,000 +++</td>
</tr>
<tr>
<td>94550</td>
<td>2001/01/02</td>
<td>$496,323.64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+++ Order total is greater than or equal to $200,000 +++</td>
</tr>
<tr>
<td>94530</td>
<td>2001/01/02</td>
<td>$3,472.41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--- Order total is less than $200,000 ---</td>
</tr>
</tbody>
</table>
Controlling the Display of Empty Reports

**How to:**

Control the Display of an Empty Report

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 WP or DOC formats.

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
Specifications 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
Specifications that a message will be displayed indicating that there is no report output. OFF is the default value.

Formatting a Report Using Only StyleSheet Defaults

How to:
Format 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.
Identifying a Report Component in a WebFOCUS StyleSheet

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 807.

For information about identifying components in a graph, see Creating a Graph on page 1245.

Unless otherwise noted, all StyleSheet references in this chapter refer to WebFOCUS StyleSheets.
Identifying an Entire Report, Column, or Row

How to:
Identify an Entire Report
Identify an Entire Column
Identify an Entire Financial Modeling Language (FML) Row
Identify an Entire Total or Subtotal Row

You can apply formatting to an:

- **Entire report.** For more information, see *How to Identify an Entire Report* on page 851.

- **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 853.

- **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 856, and *How to Identify an Entire Total or Subtotal Row* on page 857.

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 875.
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
```

```
<table>
<thead>
<tr>
<th>Plant</th>
<th>Line Total</th>
<th>Line Cost of Goods Sold</th>
<th>Line Total</th>
<th>Line Cost of Goods Sold</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS</td>
<td>$1,144,222.47</td>
<td>955,189.00</td>
<td>$125,470,864.68</td>
<td>109,839,382.00</td>
</tr>
<tr>
<td>DAL</td>
<td>$291,718.75</td>
<td>230,249.00</td>
<td>$46,868,840.52</td>
<td>41,129,581.00</td>
</tr>
<tr>
<td>LA</td>
<td>$1,433.30</td>
<td>986.00</td>
<td>$27,668,201.82</td>
<td>24,027,600.00</td>
</tr>
<tr>
<td>ORL</td>
<td>.</td>
<td>.</td>
<td>$46,955,479.15</td>
<td>41,143,292.00</td>
</tr>
<tr>
<td>SEA</td>
<td>.</td>
<td>.</td>
<td>$10,906,551.21</td>
<td>9,578,621.00</td>
</tr>
<tr>
<td>STL</td>
<td>$560,831.70</td>
<td>458,317.00</td>
<td>$93,875,795.10</td>
<td>81,807,263.00</td>
</tr>
</tbody>
</table>
```

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
```

Creating Reports With WebFOCUS Language 851
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.

```plaintext
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**

<table>
<thead>
<tr>
<th>Product Name:</th>
<th>Quantity In Stock:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Hd VCR LCD Menu</td>
<td>43068</td>
</tr>
<tr>
<td>250 8MM Camcorder 40 X</td>
<td>60073</td>
</tr>
<tr>
<td>330DX Digital Camera 1024K P</td>
<td>12707</td>
</tr>
<tr>
<td>750SL Digital Camcorder 300 X</td>
<td>10758</td>
</tr>
<tr>
<td>AR2 35MM Camera 8 X</td>
<td>11499</td>
</tr>
<tr>
<td>AR3 35MM Camera 10 X</td>
<td>12444</td>
</tr>
<tr>
<td>Combo Player - 4 Hd VCR + DVD</td>
<td>13527</td>
</tr>
<tr>
<td>QX Portable CD Player</td>
<td>22000</td>
</tr>
<tr>
<td>ZC Digital PDA - Standard</td>
<td>33000</td>
</tr>
<tr>
<td>ZT Digital PDA - Commercial</td>
<td>21000</td>
</tr>
</tbody>
</table>

**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** 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** 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.

Options for identifying columns in a StyleSheet are:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nn</td>
<td>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 an example, see <em>Identifying a Column of Data</em> on page 863.</td>
</tr>
<tr>
<td>Pn</td>
<td>Identifies a column by its position in the report. To determine the value of n, count vertical sort (BY) fields, display fields, and ROW-TOTAL fields from left to right. Do not count NOPRINT fields.</td>
</tr>
<tr>
<td>Cn</td>
<td>Identifies a display column by its position in the report. To determine the value of n, count only display fields from left to right, including NOPRINT fields. Do not count vertical sort (BY) fields or ROW-TOTAL fields.</td>
</tr>
<tr>
<td>Bn</td>
<td>Identifies a vertical sort (BY) column by its position in the report. To determine the value of n, count only vertical sort (BY) fields, including NOPRINTs, from left to right.</td>
</tr>
</tbody>
</table>

To select all display fields use C*.

To select all BY fields use B*.
### Identifying an Entire Column

#### Description

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>field</em></td>
<td>Identifies a column by its field name. When a field occurs more than once, use field(n) to select a particular occurrence or field(*) to select all occurrences of the field.</td>
</tr>
<tr>
<td><strong>ROWTOTAL</strong></td>
<td>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(n) to select a particular total column. Use ROWTOTAL(field) to select the row total column for a particular field. Use ROWTOTAL(*) to select all row total columns in the report.</td>
</tr>
</tbody>
</table>

#### 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.

```plaintext
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
```
Identifying an Entire Horizontal (ACROSS) Column

Example:

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
```
Identifying an Entire Report, Column, or Row

The output is:

```
<table>
<thead>
<tr>
<th>Cost Analysis</th>
<th>Line cost of</th>
<th>Line cost of</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Plant</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TABLE FILE</td>
<td>LEDGER</td>
</tr>
<tr>
<td></td>
<td>1010 AS 'CASH</td>
<td>1010 AS 'CASH</td>
</tr>
<tr>
<td></td>
<td>On TABLE SET</td>
<td>Style OFF</td>
</tr>
<tr>
<td></td>
<td>LABEL=COH,</td>
<td>$TYPE=REPORT, LABEL=COH, STYLE=ITALIC, $</td>
</tr>
<tr>
<td></td>
<td>END</td>
<td>END</td>
</tr>
</tbody>
</table>
```

**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:

<table>
<thead>
<tr>
<th>AMOUNT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH ON HAND</td>
<td>8,784</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
<td>4,494</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
<td>7,961</td>
</tr>
<tr>
<td>TOTAL CASH</td>
<td>21,239</td>
</tr>
</tbody>
</table>

**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.

```plaintext
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:

<table>
<thead>
<tr>
<th>PROD CODE</th>
<th>RETURNS</th>
<th>DAMAGED</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>B10</td>
<td>13</td>
<td>10</td>
<td>23</td>
</tr>
<tr>
<td>B12</td>
<td>4</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>B17</td>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>B20</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>C13</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>C17</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C7</td>
<td>5</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>D12</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>E1</td>
<td>4</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>E2</td>
<td>9</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>E3</td>
<td>12</td>
<td>11</td>
<td>23</td>
</tr>
</tbody>
</table>

**TOTAL** 58 45 103
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 825.

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:

<table>
<thead>
<tr>
<th>PRODUCT CODE</th>
<th>RETURNS</th>
<th>DAMAGED</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>B10</td>
<td>13</td>
<td>10</td>
<td>23</td>
</tr>
<tr>
<td>B12</td>
<td>4</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>B17</td>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>B20</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>C13</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>C17</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C7</td>
<td>5</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>D12</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>E1</td>
<td>4</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>E2</td>
<td>9</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>E3</td>
<td>12</td>
<td>11</td>
<td>23</td>
</tr>
</tbody>
</table>
Identifying Data

In this section:
Identifying Totals and Subtotals

How to:
Identify All Data
Identify a Column of Data
Identify a Row of Horizontal Sort (ACROSS) 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 861.
- **Columns of data,** including sort columns and display columns. For more information, see *How to Identify a Column of Data* on page 863.
- **Sort rows** (that is, ACROSS field values). For more information, see *How to Identify a Row of Horizontal Sort (ACROSS) Data* on page 864.
- **Totals and subtotals.** For more information, see *Identifying Totals and Subtotals* on page 866.
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 PLANTING 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

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
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 PLANT LNG 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:

```
<table>
<thead>
<tr>
<th>PLANT</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>2002 TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boston</td>
<td>165,938</td>
<td>202,100</td>
<td>84,571</td>
<td>86,227</td>
<td>538,836</td>
</tr>
<tr>
<td>Dallas</td>
<td>62,693</td>
<td>63,403</td>
<td>35,747</td>
<td>42,094</td>
<td>203,937</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>55,542</td>
<td>30,326</td>
<td>17,188</td>
<td>16,869</td>
<td>119,925</td>
</tr>
<tr>
<td>Orlando</td>
<td>76,261</td>
<td>57,639</td>
<td>32,595</td>
<td>35,895</td>
<td>202,390</td>
</tr>
<tr>
<td>Seattle</td>
<td>16,887</td>
<td>17,496</td>
<td>5,454</td>
<td>5,512</td>
<td>45,349</td>
</tr>
<tr>
<td>St Louis</td>
<td>150,293</td>
<td>109,284</td>
<td>65,161</td>
<td>80,530</td>
<td>405,268</td>
</tr>
</tbody>
</table>
```

TOTAL UNITS 527,614 480,248 240,716 267,127 1,515,705
**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 853.

**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:

<table>
<thead>
<tr>
<th>Product Name</th>
<th>Quantity</th>
<th>Line Total</th>
<th>Line Cost of Goods Sold</th>
</tr>
</thead>
<tbody>
<tr>
<td>110 VHS-C Camcorder 20X</td>
<td>266</td>
<td>$70,901.90</td>
<td>66,234.00</td>
</tr>
<tr>
<td>120 VHS-C Camcorder 40X</td>
<td>266</td>
<td>$76,511.09</td>
<td>68,894.00</td>
</tr>
<tr>
<td>150 8MM Camcorder 20X</td>
<td>121</td>
<td>$28,473.78</td>
<td>29,040.00</td>
</tr>
<tr>
<td>DVD Upgrade Unit for VCR</td>
<td>121</td>
<td>$18,177.77</td>
<td>16,819.00</td>
</tr>
<tr>
<td>QX Portable CD Player</td>
<td>266</td>
<td>$33,847.80</td>
<td>26,334.00</td>
</tr>
</tbody>
</table>
**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.

```plaintext
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
```
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.

<table>
<thead>
<tr>
<th>Product Name:</th>
<th>Manufacturing Plant</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Plant Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>330DX Digital Camera 1024K P</td>
<td>3,949</td>
<td>2,767</td>
<td>562</td>
<td>561</td>
<td>2</td>
<td>1,132</td>
</tr>
<tr>
<td>650DL Digital Camcorder 150 X</td>
<td>44,930</td>
<td>14,548</td>
<td>7,995</td>
<td>17,972</td>
<td>4,967</td>
<td>31,436</td>
</tr>
<tr>
<td>750SL Digital Camcorder 300 X</td>
<td>4,098</td>
<td>2</td>
<td>6</td>
<td>283</td>
<td>642</td>
<td>1,644</td>
</tr>
<tr>
<td>Combo Player - 4 Hd VCR + DVD</td>
<td>27,687</td>
<td>15,985</td>
<td>9,634</td>
<td>13,816</td>
<td>2,359</td>
<td>22,097</td>
</tr>
<tr>
<td>DVD Upgrade Unit for Cent. VCR</td>
<td>47,112</td>
<td>25,602</td>
<td>16,152</td>
<td>23,429</td>
<td>4,539</td>
<td>32,875</td>
</tr>
<tr>
<td>QX Portable CD Player</td>
<td>108,696</td>
<td>35,355</td>
<td>26,215</td>
<td>46,388</td>
<td>8,056</td>
<td>83,564</td>
</tr>
<tr>
<td>R5 Micro Digital Tape Recorder</td>
<td>188,384</td>
<td>73,255</td>
<td>41,169</td>
<td>63,120</td>
<td>15,385</td>
<td>142,714</td>
</tr>
<tr>
<td>ZC Digital PDA - Standard</td>
<td>45,890</td>
<td>13,154</td>
<td>4,664</td>
<td>10,249</td>
<td>6,439</td>
<td>30,213</td>
</tr>
<tr>
<td>ZT Digital PDA - Commercial</td>
<td>181,750</td>
<td>70,111</td>
<td>41,149</td>
<td>61,244</td>
<td>15,039</td>
<td>130,942</td>
</tr>
</tbody>
</table>

Identifying Totals and Subtotals

**How to:**

Identify a Grand Total, Subtotal, or Subtotal Calculation

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 850.

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:

<table>
<thead>
<tr>
<th></th>
<th>DEPARTMENT</th>
<th>PAY_DATE</th>
<th>DED_AMT</th>
<th>GROSS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>8/1/30</td>
<td>$1,406.79</td>
<td>$2,197.75</td>
</tr>
<tr>
<td>MIS</td>
<td></td>
<td>8/1/31</td>
<td>$1,406.79</td>
<td>$2,197.75</td>
</tr>
<tr>
<td>*TOTAL MIS</td>
<td></td>
<td></td>
<td>$2,813.58</td>
<td>$4,295.50</td>
</tr>
<tr>
<td></td>
<td>DEPT_NET</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>**</td>
<td></td>
<td>$2,311.98</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRODUCTION</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>8/1/30</td>
<td>$141.66</td>
<td>$833.33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8/2/30</td>
<td>$1,560.09</td>
<td>$3,175.84</td>
<td></td>
</tr>
<tr>
<td>*TOTAL PRODUCTION</td>
<td></td>
<td>$1,701.75</td>
<td>$4,539.17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DEPT_NET</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>**</td>
<td></td>
<td>$2,531.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TOTAL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$5,578.35</td>
<td>$10,421.47</td>
</tr>
</tbody>
</table>

**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 869 for an example.
    - **SUBTOTAL** which is a subtotal (generated by SUBTOTAL, SUB-TOTAL, RECOMPUTE, or SUMMARIZE). See *Identifying Subtotals* on page 870 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 872 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 853.
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:

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Product Name:</th>
<th>Quantity</th>
<th>Line Total</th>
<th>Cost Of Goods Sold</th>
</tr>
</thead>
<tbody>
<tr>
<td>48053</td>
<td>150 SMM Camcorder 20 X</td>
<td>1</td>
<td>$338.28</td>
<td>240.00</td>
</tr>
<tr>
<td></td>
<td>Combo Player - 4 Hd VCR + DVD</td>
<td>1</td>
<td>$415.73</td>
<td>289.00</td>
</tr>
<tr>
<td></td>
<td>DVD Upgrade Unit for Cent. VCR</td>
<td>1</td>
<td>$210.42</td>
<td>139.00</td>
</tr>
<tr>
<td></td>
<td>R5 Micro Digital Tape Recorder</td>
<td>1</td>
<td>$88.52</td>
<td>69.00</td>
</tr>
<tr>
<td></td>
<td>ZT Digital PDA - Commercial</td>
<td>1</td>
<td>$523.75</td>
<td>349.00</td>
</tr>
<tr>
<td>48798</td>
<td>150 SMM Camcorder 20 X</td>
<td>1</td>
<td>$319.89</td>
<td>240.00</td>
</tr>
<tr>
<td></td>
<td>Combo Player - 4 Hd VCR + DVD</td>
<td>1</td>
<td>$418.26</td>
<td>289.00</td>
</tr>
<tr>
<td></td>
<td>DVD Upgrade Unit for Cent. VCR</td>
<td>1</td>
<td>$200.53</td>
<td>139.00</td>
</tr>
<tr>
<td></td>
<td>R5 Micro Digital Tape Recorder</td>
<td>1</td>
<td>$92.09</td>
<td>69.00</td>
</tr>
<tr>
<td></td>
<td>ZT Digital PDA - Commercial</td>
<td>1</td>
<td>$483.52</td>
<td>349.00</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>10</td>
<td>$3,091.09</td>
<td>2,172.00</td>
</tr>
</tbody>
</table>
**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.

```plaintext
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
ON PRODNAME 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
```
### Identifying a Report Component in a WebFOCUS StyleSheet

<table>
<thead>
<tr>
<th>Manufacturing Plant</th>
<th>Order Number</th>
<th>Product Name</th>
<th>Quantity</th>
<th>Line Total</th>
<th>Goods Sold</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS</td>
<td>35774</td>
<td>110 VHS-C Camcorder 20 X</td>
<td>146</td>
<td>$38,817.56</td>
<td>36,354.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>120 VHS-C Camcorder 40 X</td>
<td>146</td>
<td>$40,426.90</td>
<td>37,814.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.5d VCR LCD Menu</td>
<td>279</td>
<td>$39,693.32</td>
<td>35,991.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>650DL Digital Camcorder 150 X</td>
<td>146</td>
<td>$116,907.15</td>
<td>103,660.00</td>
</tr>
</tbody>
</table>

**TOTAL ORDER_NUM 35774**

| Line Cost of | 717 | $235,844.93 | 213,819.00 |

| 48041         | 159 SMM Camcorder 20 X            | 1 | $307.43 | 240.00 |
|              | Combo Player - 4 Hdd 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 FPA - Commercial       | 1 | $528.38 | 349.00 |

**TOTAL ORDER_NUM 48041**

| 5 | $1,523.07 | 1,086.00 |

**TOTAL PLANT BOS**

| 722 | $237,368.00 | **214,905.00** |

**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:

```
<table>
<thead>
<tr>
<th>Manufacturing Plant</th>
<th>Order Number</th>
<th>Quantity</th>
<th>Line Total</th>
<th>Line Cost of Goods Sold</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS</td>
<td>28004</td>
<td>6</td>
<td>$2,023.62</td>
<td>1,395.00</td>
</tr>
<tr>
<td></td>
<td>28011</td>
<td>597</td>
<td>$25,556.08</td>
<td>125,306.00</td>
</tr>
<tr>
<td></td>
<td>28024</td>
<td>597</td>
<td>$44,012.51</td>
<td>125,306.00</td>
</tr>
<tr>
<td></td>
<td>28035</td>
<td>5</td>
<td>$1,541.26</td>
<td>986.00</td>
</tr>
<tr>
<td></td>
<td>28036</td>
<td>542</td>
<td>$29,859.27</td>
<td>109,331.00</td>
</tr>
<tr>
<td></td>
<td>28043/</td>
<td>675</td>
<td>$66,9129.48</td>
<td>135,574.00</td>
</tr>
<tr>
<td></td>
<td>28040</td>
<td>5</td>
<td>$1,659.77</td>
<td>1,086.00</td>
</tr>
<tr>
<td></td>
<td>28041</td>
<td>5</td>
<td>$1,642.86</td>
<td>1,086.00</td>
</tr>
<tr>
<td></td>
<td>28042</td>
<td>5</td>
<td>$1,724.17</td>
<td>1,086.00</td>
</tr>
<tr>
<td></td>
<td>28043</td>
<td>4</td>
<td>$2,020.11</td>
<td>1,347.00</td>
</tr>
<tr>
<td></td>
<td>28045</td>
<td>1,142</td>
<td>$260,105.24</td>
<td>227,651.00</td>
</tr>
</tbody>
</table>

**QTY 655661**
Identifying a Heading, Footing, Title, or FML Free Text

In this section:
- Identifying a Column or Row Title
- Identifying a Heading or Footing

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

How to:

Identify a Column Title
Identify a Horizontal Sort Title or Value
Identify Free Text in an FML Report

Within aStyleSheet 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 |

<table>
<thead>
<tr>
<th>PAY_DATE</th>
<th>GROSS</th>
<th>DED_AMT</th>
<th>GROSS</th>
<th>DED_AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>51/11/30</td>
<td>$2,147.75</td>
<td>$1,408.79</td>
<td>$633.33</td>
<td>$141.66</td>
</tr>
<tr>
<td>51/12/31</td>
<td>$2,147.75</td>
<td>$1,408.79</td>
<td>$633.33</td>
<td>$141.66</td>
</tr>
<tr>
<td>52/01/29</td>
<td>$3,247.75</td>
<td>$1,740.89</td>
<td>$3,708.84</td>
<td>$1,560.99</td>
</tr>
<tr>
<td>52/02/26</td>
<td>$3,247.75</td>
<td>$1,740.89</td>
<td>$4,959.84</td>
<td>$2,081.69</td>
</tr>
<tr>
<td>52/03/31</td>
<td>$3,247.75</td>
<td>$1,740.89</td>
<td>$4,959.84</td>
<td>$2,081.69</td>
</tr>
<tr>
<td>52/04/30</td>
<td>$5,890.84</td>
<td>$3,855.33</td>
<td>$4,959.84</td>
<td>$2,081.69</td>
</tr>
<tr>
<td>52/05/28</td>
<td>$6,649.50</td>
<td>$3,954.35</td>
<td>$7,045.84</td>
<td>$3,483.68</td>
</tr>
<tr>
<td>52/06/30</td>
<td>$7,460.00</td>
<td>$4,117.03</td>
<td>$7,045.84</td>
<td>$3,483.68</td>
</tr>
<tr>
<td>52/07/30</td>
<td>$7,460.00</td>
<td>$4,117.03</td>
<td>$7,045.84</td>
<td>$3,483.68</td>
</tr>
<tr>
<td>52/08/31</td>
<td>$8,000.00</td>
<td>$4,575.72</td>
<td>$9,523.84</td>
<td>$4,511.12</td>
</tr>
</tbody>
</table>

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 853.

**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 *How to Identify a Row of Horizontal Sort (ACROSS) Data* on page 864.
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**

<table>
<thead>
<tr>
<th>Plant</th>
<th>2001 Quantity Sold</th>
<th>2001 Product Cost</th>
<th>2002 Quantity Sold</th>
<th>2002 Product Cost</th>
<th>TOTAL Quantity Sold</th>
<th>TOTAL Product Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS</td>
<td>491,080</td>
<td>109839382</td>
<td>538,836</td>
<td>120518527</td>
<td>1,029,916</td>
<td>230357909</td>
</tr>
<tr>
<td>DAL</td>
<td>185,785</td>
<td>41129581</td>
<td>203,937</td>
<td>45147907</td>
<td>389,722</td>
<td>86277488</td>
</tr>
<tr>
<td>LA</td>
<td>109,326</td>
<td>24027600</td>
<td>119,925</td>
<td>26356066</td>
<td>229,251</td>
<td>50383666</td>
</tr>
<tr>
<td>ORL</td>
<td>184,519</td>
<td>41143292</td>
<td>202,390</td>
<td>45127226</td>
<td>386,909</td>
<td>86270518</td>
</tr>
<tr>
<td>SEA</td>
<td>41,331</td>
<td>9578621</td>
<td>45,349</td>
<td>10510177</td>
<td>86,680</td>
<td>20088798</td>
</tr>
<tr>
<td>STL</td>
<td>369,456</td>
<td>81807263</td>
<td>405,268</td>
<td>89734521</td>
<td>774,724</td>
<td>171541784</td>
</tr>
</tbody>
</table>

**Syntax:**  

*How to Identify Free Text in an FML Report*

`TYPE=FREETEXT, 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.

```FML
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 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:

```
<table>
<thead>
<tr>
<th></th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>--- CASH ACCOUNTS ---</td>
<td></td>
</tr>
<tr>
<td>CASH ON HAND</td>
<td>8,784</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
<td>4,494</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
<td>7,961</td>
</tr>
<tr>
<td>--- OTHER CURRENT ASSETS</td>
<td></td>
</tr>
<tr>
<td>ACCOUNTS RECEIVABLE</td>
<td>18,829</td>
</tr>
<tr>
<td>INVENTORY</td>
<td>27,307</td>
</tr>
</tbody>
</table>
```
Identifying a Heading or Footing

How to:
Identify a Heading or Footing
Identify an Individual Line in a Heading or Footing
Identify a Text String in a Heading or Footing
Identify an Embedded Field in 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.

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
```
Identifying a Heading, Footing, Title, or FML Free Text

The following output goes with the previous code example:

```plaintext
PAGE    1

CONFIDENTIAL INFORMATION
SWIFTY INFORMATION GROUP - EMPLOYEE LIST BY DEPARTMENT

                      TYPE = TABHEADING

EMPLOYEE LIST FOR DEPARTMENT: PRODUCTION

                      TYPE = HEADING

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>CURR_SAL</th>
<th>HIRE_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID: 119329144</td>
<td>JOHN</td>
<td>$29,700.00</td>
<td>08/06/01</td>
</tr>
<tr>
<td>** REVIEW SALARY FOR JOHN BANNING</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID: 326179357</td>
<td>ROSEMARIE</td>
<td>$21,780.00</td>
<td>08/04/01</td>
</tr>
<tr>
<td>** REVIEW SALARY FOR ROSEMARIE BLACKWOOD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CONFIDENTIAL INFORMATION

                      TYPE = FOOTING

***END OF REPORT***

                      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

```plaintext
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

<table>
<thead>
<tr>
<th>Manufacturing Plant</th>
<th>Quantity</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS</td>
<td>1,033,818</td>
<td>$262,433,960.63</td>
</tr>
<tr>
<td>DAL</td>
<td>390,844</td>
<td>$97,751,846.65</td>
</tr>
<tr>
<td>LA</td>
<td>229,256</td>
<td>$57,507,080.82</td>
</tr>
<tr>
<td>ORL</td>
<td>386,909</td>
<td>$97,616,855.94</td>
</tr>
<tr>
<td>SEA</td>
<td>86,680</td>
<td>$22,742,743.88</td>
</tr>
<tr>
<td>STL</td>
<td>776,743</td>
<td>$195,970,536.09</td>
</tr>
</tbody>
</table>

***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 *How to Identify a Heading or Footing* on page 880.

*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**

<table>
<thead>
<tr>
<th>YEAR</th>
<th>2000</th>
<th>2001</th>
<th>2002</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturing Plant</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOS</td>
<td>3,902,491,080</td>
<td>538,836</td>
<td></td>
</tr>
<tr>
<td>DAL</td>
<td>1,122</td>
<td>185,785</td>
<td>203,937</td>
</tr>
<tr>
<td>LA</td>
<td>5109,326</td>
<td>119,925</td>
<td></td>
</tr>
<tr>
<td>ORL</td>
<td>184,519</td>
<td>202,390</td>
<td></td>
</tr>
<tr>
<td>SEA</td>
<td>41,331</td>
<td>45,349</td>
<td></td>
</tr>
<tr>
<td>STL</td>
<td>2,019</td>
<td>369,456</td>
<td>405,268</td>
</tr>
</tbody>
</table>
How to Identify a Text String in a Heading or Footing

Syntax: 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 879.

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, ITEM=3, STYLE=BOLD, $
ENDSTYLE
END
```

For an example, see Identifying a Text String in a Heading Using Spot Markers on page 885.
**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.

```webfocus
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:

**Third Quarter, 2002: Sales Quantity Analysis**

<table>
<thead>
<tr>
<th>Manufacturing Plant</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS</td>
<td>1,033,818</td>
</tr>
<tr>
<td>DAL</td>
<td>390,844</td>
</tr>
<tr>
<td>LA</td>
<td>229,256</td>
</tr>
<tr>
<td>ORL</td>
<td>386,909</td>
</tr>
<tr>
<td>SEA</td>
<td>86,680</td>
</tr>
<tr>
<td>STL</td>
<td>776,743</td>
</tr>
</tbody>
</table>
How to Identify an Embedded Field in a Heading or Footing

Syntax:

\[ \text{TYPE}=\text{type}, \ [\text{LINE}=\text{line}_#, \ \text{OBJECT}=\text{FIELD}, \ [\text{ITEM}=\text{item}_#] \]

where:

\text{type}

Identifies a type of heading or footing. Select from \text{HEADING}, \text{FOOTING}, \text{TABHEADING}, \text{TABFOOTING}, \text{SUBHEAD}, or \text{SUBFOOT}. For details, see \textit{Identifying a Heading or Footing} on page 879.

\text{line}_#

Identifies a line by its position in the heading or footing. You only need to include the \text{LINE} attribute if you have a multi-line heading or footing.

\text{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.

\textbf{Note:} BORDER options are not supported on specific ITEMS in a \text{HEADING}, \text{FOOTING}, \text{SUBHEAD}, \text{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.

\begin{verbatim}
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=FIELD, COLOR=RED, STYLE=BOLD, $
ENDSTYLE
END
\end{verbatim}
Identifying a Page Number, Underline, or Skipped Line

How to:

Identify 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 For 2000 By Plant"
MANUFACTURING PLANT QUANTITY:
BOS   3,902
DAL   1,122
LA    5
STL   2,019
```
The following output goes with the previous code example.

```
TYPE=PAGENUM  PAGE  1
```

<table>
<thead>
<tr>
<th>Sales By Plant</th>
<th>YEAR</th>
<th>Quantity:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturing Plant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOS</td>
<td>2000</td>
<td>3,902</td>
</tr>
<tr>
<td></td>
<td>2001</td>
<td>491,080</td>
</tr>
<tr>
<td></td>
<td>2002</td>
<td>538,836</td>
</tr>
<tr>
<td>DAL</td>
<td>2000</td>
<td>1,122</td>
</tr>
<tr>
<td></td>
<td>2001</td>
<td>185,785</td>
</tr>
<tr>
<td></td>
<td>2002</td>
<td>203,937</td>
</tr>
</tbody>
</table>

**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.

```plaintext
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
Manufacturing Plant      YEAR    Quantity:

BOS  2000    3,902
     2001    491,080
     2002    538,836

DAL  2000    1,122
     2001    185,785
     2002    203,937

LA   2000    5
     2001   109,326
     2002   119,925
```

Creating Reports With WebFOCUS Language
**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

<table>
<thead>
<tr>
<th>Product Name:</th>
<th>Quantity In Stock:</th>
</tr>
</thead>
<tbody>
<tr>
<td>110 VHS-C Camcorder 20 X</td>
<td>4000</td>
</tr>
<tr>
<td>120 VHS-C Camcorder 40 X</td>
<td>2300</td>
</tr>
<tr>
<td>340SX Digital Camera 65K P</td>
<td>990</td>
</tr>
<tr>
<td>650DL Digital Camcorder 150 X</td>
<td>2972</td>
</tr>
<tr>
<td>DVD Upgrade Unit for Cent. VCR</td>
<td>199</td>
</tr>
<tr>
<td>R5 Micro Digital Tape Recorder</td>
<td>1990</td>
</tr>
</tbody>
</table>
```
Cascading Style Sheets (CSS) provide a standard way of formatting HTML documents. To format a WebFOCUS report HTML 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).
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 826.

- **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 893.

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 include:

- **Increased formatting options.** You can take advantage of all Cascading Style Sheet features, including features that are unavailable in WebFOCUS StyleSheets.

- **Improved performance.** You can reduce transmission bandwidth and download time.

- **Reduced effort** by applying the same Cascading Style Sheet to reports and other documents.

- **Easier standards conformance** for enterprises that currently implement their presentation standards using Cascading Style Sheets.
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 826

**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:

```
.pageFooting {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 895.
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

**How to:**
Format a Report Using an External Cascading Style Sheet

**Reference:**
A CSS Rule for the ColumnTitle Class
A CSS Rule for the TD Element
Applying a CSS Class to ACROSS Values in a Report
Linking to the ReportStyles 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 901.

- **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 905.

  (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 907.)

- **A link to the external Cascading Style Sheet** from the report. For more information, see *Linking to an External Cascading Style Sheet* on page 909.

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 897.
For an example that demonstrates how these items work together, see *Formatting a Report Using an External CSS* on page 899, and see 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 898.

     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 898.

     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 901.

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 898. For more information, see Applying External Cascading Style Sheet Formatting on page 905.

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.

   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 898. For more information, see Linking to an External Cascading Style Sheet on page 909.
Problems? If you encounter problems, see Troubleshooting External Cascading Style Sheets on page 925.

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;}
```

For a more comprehensive example that shows a complete external Cascading Style Sheet, a report procedure with an inline WebFOCUS StyleSheet, and generated report output, see Formatting a Report Using an External CSS on page 899.

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 Formatting a Report Using an External CSS on page 899.

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, $
```

For a more comprehensive example that shows a complete external Cascading Style Sheet, a report procedure with an inline WebFOCUS StyleSheet, and generated report output, see Formatting a Report Using an External CSS on page 899.

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:

```html
<HEAD>
<TITLE>Accounts Receivable Report</TITLE>
<LINK REL="STYLESHEET" HREF="http://srv3/css/reports.css" TYPE="text/css">
</HEAD>
```

For a more comprehensive example that shows a complete external Cascading Style Sheet, a report procedure with an inline WebFOCUS StyleSheet, and generated report output, see Formatting a Report Using an External CSS on page 899.

**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**

```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
```
Formatting a Report With an External Cascading Style Sheet

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

<table>
<thead>
<tr>
<th>Product Code</th>
<th>Product</th>
<th>Unit Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>B141</td>
<td>Hazelnut</td>
<td>58.00</td>
</tr>
<tr>
<td>B142</td>
<td>French Roast</td>
<td>81.00</td>
</tr>
<tr>
<td>B144</td>
<td>Kona</td>
<td>76.00</td>
</tr>
<tr>
<td>F101</td>
<td>Scone</td>
<td>13.00</td>
</tr>
<tr>
<td>F102</td>
<td>Biscotti</td>
<td>17.00</td>
</tr>
<tr>
<td>F103</td>
<td>Croissant</td>
<td>28.00</td>
</tr>
<tr>
<td>G100</td>
<td>Mug</td>
<td>26.00</td>
</tr>
<tr>
<td>G104</td>
<td>Thermos</td>
<td>96.00</td>
</tr>
<tr>
<td>G110</td>
<td>Coffee Grinder</td>
<td>125.00</td>
</tr>
<tr>
<td>G121</td>
<td>Coffee Pot</td>
<td>140.00</td>
</tr>
</tbody>
</table>

### Working With an External Cascading Style Sheet

#### In this section:

- Choosing an External Cascading Style Sheet
- External Cascading Style Sheet Location
- Using Several External Cascading Style Sheets
- Editing an External Cascading Style Sheet
- Choosing a Cascading Style Sheet Rule
- Naming a Cascading Style Sheet Class

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 902.

- Where an external CSS can reside. For more information, see External Cascading Style Sheet Location on page 902.
How you can apply multiple Cascading Style Sheets to one report. For more information, see Using Several External Cascading Style Sheets on page 902.

Editing an external CSS. For more information, see Editing an External Cascading Style Sheet on page 903.

Using CSS rules and classes to specify report formatting. For more information, see Choosing a Cascading Style Sheet Rule on page 903.

Suggestions for naming Cascading Style Sheet classes. For more information, see Naming a Cascading Style Sheet Class on page 904.

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 907.

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 Web pages used by your WebFOCUS applications, as well as rules for other kinds of Web pages. 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 the previous bullet 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 a report formatting 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.

<table>
<thead>
<tr>
<th>To format:</th>
<th>Use a rule for:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The entire report</td>
<td>BODY or TD</td>
</tr>
<tr>
<td>Any report component</td>
<td>A generic class (that is, one declared without an element)</td>
</tr>
</tbody>
</table>

To choose between using a rule for BODY or for TD, note that a rule for:

- **BODY** will specify default formatting for the entire Web page 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 906.
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 898.
- The TD element to format an entire report, see A CSS Rule for the TD Element on page 898.

**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

**How to:**

Use the CLASS Attribute to Apply CSS 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 906. For information about specifying different types of report components, see *Identifying a Report Component in a WebFOCUS StyleSheet* on page 849.

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, not the Report Assistant, Report Painter, or Graph Assistant.

- **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 898.

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 907.
**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 849. 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 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 797.
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 519.

For an example, see Applying a CSS Class to ACROSS Values in a Report on page 898.

Combining an External CSS With Other Formatting Methods

In this section:

Combining an External CSS With WebFOCUS StyleSheets

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 StyleSheet.** 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 WebFOCUS StyleSheets on page 908.

- **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 WebFOCUS StyleSheets

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 903. For an example, see A Report Column Inheriting Formatting From the TD Element on page 914.

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 826.

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:

\[\text{TYPE=Data, COLUMN=Country, COLOR=Orange, }\$
\text{TYPE=Data, CLASS=TextColor, }\$

because they would both try to assign a color to the Country column.
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, $
2. TYPE=Heading, CLASS=HeadingFontSize, $
3. TYPE=Data, Column=Country, BACKCOLOR=Yellow, $
4. TYPE=Data, Column=Car, CLASS=DataBackgroundColor, $
5. 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

In this section:

Using the CSSURL Attribute and Parameter

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 910.

- **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 898.
Using the CSSURL Attribute and Parameter

**How to:**

Use the CSSURL Attribute to Link to an External CSS
Use the CSSURL Parameter to Link to an External CSS

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 911. For more information about the CSS parameter, see *How to Use the CSSURL Parameter to Link to an External CSS* on page 913.
Syntax: How to Use the CSSURL Attribute to Link to an External CSS

To link an external Cascading Style Sheet (CSS) to a 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.

For an example, see Linking to an External Cascading Style Sheet Using the CSSURL Attribute on page 911.

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

<table>
<thead>
<tr>
<th>Product Code</th>
<th>Product</th>
<th>Unit Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>B141</td>
<td>Hazelnut</td>
<td>58.00</td>
</tr>
<tr>
<td>B142</td>
<td>French Roast</td>
<td>81.00</td>
</tr>
<tr>
<td>B144</td>
<td>Kona</td>
<td>76.00</td>
</tr>
<tr>
<td>F101</td>
<td>Scone</td>
<td>13.00</td>
</tr>
<tr>
<td>F102</td>
<td>Biscotti</td>
<td>17.00</td>
</tr>
<tr>
<td>F103</td>
<td>Croissant</td>
<td>28.00</td>
</tr>
<tr>
<td>G100</td>
<td>Mug</td>
<td>26.00</td>
</tr>
<tr>
<td>G104</td>
<td>Thermos</td>
<td>96.00</td>
</tr>
<tr>
<td>G110</td>
<td>Coffee Grinder</td>
<td>125.00</td>
</tr>
<tr>
<td>G121</td>
<td>Coffee Pot</td>
<td>140.00</td>
</tr>
</tbody>
</table>
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 898.

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:

<table>
<thead>
<tr>
<th>Product Code</th>
<th>Product</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>B141</td>
<td>Hazelnut</td>
<td>Coffee Connection</td>
</tr>
<tr>
<td>B142</td>
<td>French Roast</td>
<td>European Specialities,</td>
</tr>
<tr>
<td>B144</td>
<td>Kona</td>
<td>Evelina Imports, Ltd</td>
</tr>
<tr>
<td>F101</td>
<td>Scone</td>
<td>Ridgewood Bakeries</td>
</tr>
<tr>
<td>F102</td>
<td>Biscotti</td>
<td>Delancey Bakeries</td>
</tr>
<tr>
<td>F103</td>
<td>Croissant</td>
<td>West Side Bakers</td>
</tr>
<tr>
<td>G100</td>
<td>Mug</td>
<td>NY Ceramic Supply</td>
</tr>
<tr>
<td>G104</td>
<td>Thermos</td>
<td>ThermoTech, Inc</td>
</tr>
<tr>
<td>G110</td>
<td>Coffee Grinder</td>
<td>Appliance Craft</td>
</tr>
<tr>
<td>G121</td>
<td>Coffee Pot</td>
<td>Appliance Craft</td>
</tr>
</tbody>
</table>
Using External Cascading Style Sheets With Non-HTML Reports

How to:
Use an External CSS With Multiple Output Types

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 917, and an example in *Using an External CSS With PDF and HTML Output* on page 919.
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.
Using an External CSS With PDF and HTML Output

This report procedure (videorpt.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.

**videorpt.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, $
    ENDSSTYLE
    END

**reports.css**

11. .bold {font-weight: bolder}
12. .blueBack {background: lightblue}
13. TABLE {border:0}
13. 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:

```
<table>
<thead>
<tr>
<th>Cust ID</th>
<th>Last Name</th>
<th>First Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0925</td>
<td>CRUZ</td>
<td>IVY</td>
</tr>
<tr>
<td>0944</td>
<td>HANDLER</td>
<td>EVAN</td>
</tr>
<tr>
<td>1118</td>
<td>WILSON</td>
<td>KELLY</td>
</tr>
<tr>
<td>1133</td>
<td>KRAMER</td>
<td>CHERYL</td>
</tr>
<tr>
<td>1237</td>
<td>GOODMAN</td>
<td>JOHN</td>
</tr>
</tbody>
</table>
```
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 916.

- 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 907.

- **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 907.

- **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.
FAQ About Using External Cascading Style Sheets

- **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 4.3.5 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.

**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.)

You can:

- **E-mail them** to books_info@ibi.com. Please include your name and phone number, and put Cascading Style Sheet FAQ in the subject line.

- **Send them** to:

  Documentation Services  
  Attn: Core Reporting Group  
  Information Builders  
  Two Penn Plaza  
  New York, NY 10121-2898

  Please include your name, phone number, e-mail address, and postal address.
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 a Cascading Style Sheet Rule on page 903.

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:

```html
<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 905.

**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 907. For information about internal Cascading Style Sheets, see *Generating an Internal Cascading Style Sheet for HTML Reports* on page 826.

**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 921, and *Troubleshooting External Cascading Style Sheets* on page 925.
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. You can:

- **E-mail them** to books_info@ibi.com. Please include your name and phone number, and put *Cascading Style Sheet troubleshooting* in the subject line.

- **Send them** to:
  
  Documentation Services  
  Attn: Core Reporting Group  
  Information Builders  
  Two Penn Plaza  
  New York, NY 10121-2898  

  Please include your name, phone number, e-mail address, and postal address.
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 4.3.5 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.
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 907.
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 429 and Saving and Reusing Your Report Output on page 351.

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.
Selecting Page Size, Orientation, and Color

**How to:**
Set Page Size
Set Page Orientation
Set Page Color

**Reference:**
Page Size, Orientation, and Color Attributes

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGESIZE</td>
<td>Sets page size.</td>
<td>In the Development environment:</td>
</tr>
<tr>
<td>ORIENTATION</td>
<td>Sets page orientation.</td>
<td>In the Development environment:</td>
</tr>
<tr>
<td>PAGECOLOR</td>
<td>Sets page color.</td>
<td>HTML report with internal Cascading Style Sheet</td>
</tr>
</tbody>
</table>
**Syntax:**  

**How to Set Page Size**

This syntax applies to a PDF or PS 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCREEN</td>
<td>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. <strong>Developer Studio.</strong> This value is most useful in conjunction with the Report Viewer when maximized, using the proprietary WebFOCUS report format.</td>
</tr>
<tr>
<td>LETTER</td>
<td>8.5 x 11 inches. LETTER is the default value.</td>
</tr>
<tr>
<td>LEGAL</td>
<td>8.5 x 14 inches.</td>
</tr>
<tr>
<td>TABLOID</td>
<td>11 x 17 inches.</td>
</tr>
<tr>
<td>LEDGER</td>
<td>17 x 11 inches.</td>
</tr>
<tr>
<td>C</td>
<td>17 x 22 inches.</td>
</tr>
<tr>
<td>D</td>
<td>22 x 34 inches.</td>
</tr>
<tr>
<td>E</td>
<td>34 x 44 inches.</td>
</tr>
<tr>
<td>STATEMENT</td>
<td>5.5 x 8.5 inches.</td>
</tr>
</tbody>
</table>
### Selecting Page Size, Orientation, and Color

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECUTIVE</td>
<td>7.5 x 10.5 inches.</td>
</tr>
<tr>
<td>FOLIO</td>
<td>8.5 x 13 inches.</td>
</tr>
<tr>
<td>10x14</td>
<td>10 x 14 inches.</td>
</tr>
<tr>
<td>A3</td>
<td>297 x 420 millimeters.</td>
</tr>
<tr>
<td>A4</td>
<td>210 x 297 millimeters.</td>
</tr>
<tr>
<td>A5</td>
<td>148 x 210 millimeters.</td>
</tr>
<tr>
<td>B4</td>
<td>250 x 354 millimeters.</td>
</tr>
<tr>
<td>B5</td>
<td>182 x 257 millimeters.</td>
</tr>
<tr>
<td>QUARTO</td>
<td>215 x 275 millimeters.</td>
</tr>
<tr>
<td>ENVELOPE-9</td>
<td>3.875 x 8.875 inches.</td>
</tr>
<tr>
<td>ENVELOPE-10</td>
<td>4.125 x 9.5 inches.</td>
</tr>
<tr>
<td>ENVELOPE-11</td>
<td>4.5 x 10.375 inches.</td>
</tr>
<tr>
<td>ENVELOPE-12</td>
<td>4.5 x 11 inches.</td>
</tr>
<tr>
<td>ENVELOPE-14</td>
<td>5 x 11.5 inches.</td>
</tr>
<tr>
<td>ENVELOPE-MONARCH</td>
<td>3.875 x 7.5 inches.</td>
</tr>
<tr>
<td>ENVELOPE-PERSONAL</td>
<td>3.625 x 6.5 inches.</td>
</tr>
<tr>
<td>ENVELOPE-DL</td>
<td>110 x 220 millimeters.</td>
</tr>
<tr>
<td>ENVELOPE-C3</td>
<td>324 x 458 millimeters.</td>
</tr>
<tr>
<td>ENVELOPE-C4</td>
<td>229 x 324 millimeters.</td>
</tr>
<tr>
<td>ENVELOPE-C5</td>
<td>162 x 229 millimeters.</td>
</tr>
<tr>
<td>ENVELOPE-C6</td>
<td>114 x 162 millimeters.</td>
</tr>
<tr>
<td>ENVELOPE-C65</td>
<td>114 x 229 millimeters.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>ENVELOPE-B4</td>
<td>250 x 353 millimeters.</td>
</tr>
<tr>
<td>ENVELOPE-B5</td>
<td>176 x 250 millimeters.</td>
</tr>
<tr>
<td>ENVELOPE-B6</td>
<td>176 x 125 millimeters.</td>
</tr>
<tr>
<td>ENVELOPE-ITALY</td>
<td>110 x 230 millimeters.</td>
</tr>
<tr>
<td>GERMAN-STANDARD-FANFOLD</td>
<td>8.5 x 12 inches.</td>
</tr>
<tr>
<td>GERMAN-LEGAL-FANFOLD</td>
<td>8.5 x 13 inches.</td>
</tr>
</tbody>
</table>

**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 1205.
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
Product Number: | Store Name: | State: | Quantity In Stock: |
-----------------|-----------|-------|---------------------|
1004             | eMart     | CA    | 689,088             |
```

Setting Page Margins

How to:

- Set the Unit of Measurement
- Set Margin Size

Reference:

Page Margin Attributes

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNITS</td>
<td>Sets unit of measurement.</td>
<td>PDF, PS, HTML report with internal Cascading Style Sheet</td>
</tr>
<tr>
<td></td>
<td>Used when specifying margin size or other page characteristics. If you</td>
<td></td>
</tr>
<tr>
<td></td>
<td>change the current unit of measurement, the new value is applied to all</td>
<td></td>
</tr>
<tr>
<td></td>
<td>instances in which unit of measurement is used.</td>
<td></td>
</tr>
<tr>
<td>TOPMARGIN, BOTTOMMARGIN, LEFTMARGIN, RIGHTMARGIN</td>
<td>Sets size of top, bottom, left, and right margin.</td>
<td>PDF, PS, HTML report with internal Cascading Style Sheet</td>
</tr>
</tbody>
</table>

### 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

\[
\text{UNITS} = \text{units}
\]

Outside of a report request, use

\[
\text{SET UNITS} = \text{units}
\]
Within a report request, use

\texttt{ON \ TABLE \ SET \ UNITS \ units}

where:

\textit{units}

Is the unit of measure. Values can be:

- \texttt{INCHES} which specifies the unit of measure as inches. This is the default value.
- \texttt{CM} which specifies the unit of measure as centimeters.
- \texttt{PTS} which specifies the unit of measure as points. Points is a common measurement scale for typefaces.

\textbf{Syntax: How to Set Margin Size}

This syntax applies to a PDF, PS, or HTML report with internal Cascading Style Sheet.

\begin{verbatim}
[TYPE=REPORT,] \ [TOPMARGIN={value|.25},] \ [BOTTOMMARGIN={value=.25},] \ [LEFTMARGIN={value|.25},] \ [RIGHTMARGIN={value|.25},] $
\end{verbatim}

where:

- \texttt{TYPE=REPORT}
  Applies the margin size to the entire report. Not required, as it is the default.

- \texttt{TOPMARGIN}
  Sets the top boundary of the report content.

- \texttt{BOTTOMMARGIN}
  Sets the bottom boundary of the report content.

- \texttt{LEFTMARGIN}
  Sets the left boundary of the report content.

- \texttt{RIGHTMARGIN}
  Sets the right boundary of the report content.

- \texttt{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, $
END
```

The output is:

<table>
<thead>
<tr>
<th>Region</th>
<th>State</th>
<th>City</th>
<th>Category</th>
<th>Product</th>
<th>Dollar Sales</th>
<th>Budget Dollars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midwest</td>
<td>IL</td>
<td>Chicago</td>
<td>Gifts</td>
<td>Thermos</td>
<td>2256982</td>
<td>1833504</td>
</tr>
<tr>
<td></td>
<td>MO</td>
<td>St. Louis</td>
<td>Gifts</td>
<td>Thermos</td>
<td>2359631</td>
<td>1896623</td>
</tr>
<tr>
<td></td>
<td>TX</td>
<td>Houston</td>
<td>Gifts</td>
<td>Thermos</td>
<td>2051560</td>
<td>1646429</td>
</tr>
<tr>
<td>Northeast</td>
<td>CT</td>
<td>New Haven</td>
<td>Gifts</td>
<td>Thermos</td>
<td>1955235</td>
<td>1592177</td>
</tr>
<tr>
<td></td>
<td>MA</td>
<td>Boston</td>
<td>Gifts</td>
<td>Thermos</td>
<td>1740522</td>
<td>1403566</td>
</tr>
<tr>
<td></td>
<td>NY</td>
<td>New York</td>
<td>Gifts</td>
<td>Thermos</td>
<td>2025193</td>
<td>1636563</td>
</tr>
<tr>
<td>Southeast</td>
<td>FL</td>
<td>Orlando</td>
<td>Gifts</td>
<td>Thermos</td>
<td>2261866</td>
<td>1814452</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>Atlanta</td>
<td>Gifts</td>
<td>Thermos</td>
<td>1937991</td>
<td>1565867</td>
</tr>
<tr>
<td></td>
<td>TN</td>
<td>Memphis</td>
<td>Gifts</td>
<td>Thermos</td>
<td>1983511</td>
<td>1582051</td>
</tr>
<tr>
<td>West</td>
<td>CA</td>
<td>Los Angeles</td>
<td>Gifts</td>
<td>Thermos</td>
<td>2263020</td>
<td>1827565</td>
</tr>
<tr>
<td></td>
<td></td>
<td>San Francisco</td>
<td>Gifts</td>
<td>Thermos</td>
<td>2108905</td>
<td>1752967</td>
</tr>
<tr>
<td></td>
<td>WA</td>
<td>Seattle</td>
<td>Gifts</td>
<td>Thermos</td>
<td>2020581</td>
<td>1660214</td>
</tr>
</tbody>
</table>
Positioning a Report Component

**How to:**
- Specify the Starting Position of a Column
- Add Blank Space Around a Report Component

**Reference:**
- Positioning Attributes

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 957.

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 1053.

### Reference: Positioning Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSITION</td>
<td>Sets absolute or relative starting position of a column.</td>
<td>PDF, PS</td>
</tr>
<tr>
<td></td>
<td>An absolute position is the distance from the left margin of the printed paper.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A relative position is the distance from the default position. After the first column, the default position is the end of the preceding column.</td>
<td></td>
</tr>
<tr>
<td>TOPGAP</td>
<td>Adds blank space to the top or bottom of a report.</td>
<td>PDF, PS</td>
</tr>
<tr>
<td>BOTTOMGAP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEFTGAP</td>
<td>Adds blank space to the left or right of a report.</td>
<td>PDF, PS</td>
</tr>
<tr>
<td>RIGHTGAP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**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 849.

`+`
- 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:

<table>
<thead>
<tr>
<th>Product</th>
<th>Ordered Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscotti</td>
<td>6140</td>
</tr>
<tr>
<td>Coffee Grinder</td>
<td>2493</td>
</tr>
<tr>
<td>Coffee Pot</td>
<td>3100</td>
</tr>
<tr>
<td>Croissant</td>
<td>7465</td>
</tr>
<tr>
<td>French Roast</td>
<td>12965</td>
</tr>
<tr>
<td>Hazelnut</td>
<td>4186</td>
</tr>
<tr>
<td>Kona</td>
<td>2591</td>
</tr>
<tr>
<td>Mug</td>
<td>5332</td>
</tr>
<tr>
<td>Scone</td>
<td>5949</td>
</tr>
<tr>
<td>Thermos</td>
<td>2362</td>
</tr>
</tbody>
</table>
**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.

```plaintext
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

<table>
<thead>
<tr>
<th>Product</th>
<th>Ordered Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscotti</td>
<td>6140</td>
</tr>
<tr>
<td>Coffee Grinder</td>
<td>2493</td>
</tr>
<tr>
<td>Coffee Pot</td>
<td>3100</td>
</tr>
<tr>
<td>Croissant</td>
<td>7465</td>
</tr>
<tr>
<td>French Roast</td>
<td>12965</td>
</tr>
<tr>
<td>Hazelnut</td>
<td>4186</td>
</tr>
<tr>
<td>Kona</td>
<td>2591</td>
</tr>
<tr>
<td>Mug</td>
<td>5332</td>
</tr>
<tr>
<td>Scone</td>
<td>5949</td>
</tr>
<tr>
<td>Thermos</td>
<td>2362</td>
</tr>
</tbody>
</table>
```

**Syntax:**  **How to Add Blank Space Around a Report Component**

This syntax applies to a PDF or PS report.

```plaintext
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.
**BOTTOMGAP**

Indicates how much space to add below the report.

**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 849.

**identifier**

Selects one or more columns using the COLUMN attribute described in *Identifying a Report Component in a WebFOCUS StyleSheet* on page 849.

**acrosscolumn**

Selects the same column under every occurrence of an ACROSS sort field using the ACROSCOLUMN attribute described in *Identifying a Report Component in a WebFOCUS StyleSheet* on page 849.

**LEFTGAP**

Indicates how much space to add to the left of a report component.

**RIGHTGAP**

Indicates how much space to add to the right of a report component.

**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, FOCUS 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.

```plaintext
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

<table>
<thead>
<tr>
<th>Product</th>
<th>Ordered Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscotti</td>
<td>6140</td>
</tr>
<tr>
<td>Coffee Grinder</td>
<td>2493</td>
</tr>
<tr>
<td>Coffee Pot</td>
<td>3100</td>
</tr>
<tr>
<td>Croissant</td>
<td>7465</td>
</tr>
<tr>
<td>French Roast</td>
<td>12965</td>
</tr>
<tr>
<td>Hazelnut</td>
<td>4186</td>
</tr>
<tr>
<td>Kona</td>
<td>2591</td>
</tr>
<tr>
<td>Mug</td>
<td>5332</td>
</tr>
<tr>
<td>Scone</td>
<td>5949</td>
</tr>
<tr>
<td>Thermos</td>
<td>2362</td>
</tr>
</tbody>
</table>
```
**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.

```sql
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

<table>
<thead>
<tr>
<th>Product Type</th>
<th>Product Category</th>
<th>Cost of Goods Sold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital</td>
<td>CD Players</td>
<td>30,519,126</td>
</tr>
<tr>
<td></td>
<td>Camcorders</td>
<td>91,518,330</td>
</tr>
<tr>
<td></td>
<td>Cameras</td>
<td>1,785,627</td>
</tr>
<tr>
<td></td>
<td>DVD</td>
<td>47,275,593</td>
</tr>
<tr>
<td></td>
<td>Digital Tape Recorders</td>
<td>36,157,863</td>
</tr>
<tr>
<td></td>
<td>PDA Devices</td>
<td>202,121,656</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>409,380,195</td>
</tr>
</tbody>
</table>
```
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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SQUEEZE</strong></td>
<td>Sets column width.</td>
<td>HTML (requires FONT=DEFAULT-FIXED)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PDF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PS</td>
</tr>
<tr>
<td><strong>SET SPACES</strong></td>
<td>Sets number of spaces between columns.</td>
<td>HTML (requires SET STYLEMODE=FIXED)</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Applies to</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>Sets column order.</td>
<td>PDF, PS, HTML</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXL2K (Note: does not work with EXL2K FORMULA)</td>
</tr>
<tr>
<td>FOLD–LINE</td>
<td>Reduces report width by stacking columns.</td>
<td>PDF, PS</td>
</tr>
<tr>
<td>OVER</td>
<td>Stacks columns by placing them over one another.</td>
<td>HTML, PDF, PS</td>
</tr>
<tr>
<td>IN {n</td>
<td>+n}</td>
<td>Sets absolute or relative starting position of a column.</td>
</tr>
</tbody>
</table>

**Determining Column Width**

**How to:**

- Determine Column Width (HTML)
- Determine Column Width (PDF or PS)

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 1053.
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 950.

\[
\text{[TYPE=REPORT,] \ SQUEEZE=\{ON\|OFF\}, \ $}
\]

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 widest 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 field format in the Master File. Blank spaces pad the column width up to the length of the column title or field format, whichever is greater.

Note: 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.

\[
\text{SET \ PAGE-\text{NUM} = \text{OFF}} \text{\linebreak TABLE \ FILE \ GGSALES} \text{\linebreak SUM \ UNITS \ DOLLARS} \text{\linebreak BY \ CATEGORY \ BY \ PRODUCT} \text{\linebreak ON \ TABLE \ SET \ STYLE *} \text{\linebreak TYPE=REPORT, \ GRID=OFF, \ FONT=COURIER, \ $} \text{\linebreak ENDSTYLE} \text{\linebreak END}
\]
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:

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>

**Example:** Using Column Width Based on Field Format in Master File (HTML)

This request sets SQUEEZE to OFF for an HTML report. Column width is based on the field format in the Master File.

```
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:

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>

**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 948.

```
[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 849. 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. ON is the default value.

- **OFF**
  - Determines column width based on the field format in the Master File. Blank spaces pad the column width up to the length of the column title or field format, whichever is greater.
width

Is a measurement for the column width, specified with the UNITS attribute.

If the widest data value exceeds the specified measurement:

<table>
<thead>
<tr>
<th>And the field is...</th>
<th>The following displays...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphanumeric</td>
<td>As much of the value as will fit in the specified width, followed by an exclamation mark (!) to indicate truncation.</td>
</tr>
<tr>
<td>Numeric</td>
<td>Asterisks (*) in place of the field value.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Product</th>
<th>Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscotti</td>
<td>421377</td>
</tr>
<tr>
<td>Capuccino</td>
<td>189217</td>
</tr>
<tr>
<td>Coffee Grinder</td>
<td>186534</td>
</tr>
<tr>
<td>Coffee Pot</td>
<td>190695</td>
</tr>
<tr>
<td>Croissant</td>
<td>630054</td>
</tr>
<tr>
<td>Espresso</td>
<td>308986</td>
</tr>
<tr>
<td>Latte</td>
<td>878063</td>
</tr>
<tr>
<td>Mug</td>
<td>360570</td>
</tr>
<tr>
<td>Scone</td>
<td>333414</td>
</tr>
<tr>
<td>Thermos</td>
<td>190081</td>
</tr>
</tbody>
</table>
Controlling Column Spacing

How to:
Control 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:

<table>
<thead>
<tr>
<th>Order Number:</th>
<th>28003</th>
<th>28004</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plant Quantity:</td>
<td>Line Total</td>
<td>Quantity:</td>
</tr>
<tr>
<td>BOS</td>
<td>.</td>
<td>6</td>
</tr>
<tr>
<td>LA</td>
<td>5</td>
<td>$1,688.47</td>
</tr>
</tbody>
</table>

Changing Column Order

**How to:**
Change 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, and EXL2K reports.

```
[TYP=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 849.

Creating Reports With WebFOCUS Language 953
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:

<table>
<thead>
<tr>
<th>Sales</th>
<th>Product</th>
<th>Store Name:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$469,771.73</td>
<td>CD Players</td>
<td>City Video</td>
</tr>
<tr>
<td>$5,488,218.35</td>
<td>Camcorders</td>
<td></td>
</tr>
<tr>
<td>$359,190.46</td>
<td>Cameras</td>
<td></td>
</tr>
<tr>
<td>$1,023,484.18</td>
<td>DVD</td>
<td></td>
</tr>
<tr>
<td>$1,151,410.41</td>
<td>Digital Tape Recorders</td>
<td></td>
</tr>
<tr>
<td>$6,444,433.57</td>
<td>FDA Devices</td>
<td></td>
</tr>
<tr>
<td>$362,542.59</td>
<td>VCRs</td>
<td></td>
</tr>
<tr>
<td>$20,371,041.94</td>
<td>CD Players</td>
<td>eMart</td>
</tr>
<tr>
<td>$142,957,872.94</td>
<td>Camcorders</td>
<td></td>
</tr>
<tr>
<td>$2,428,149.11</td>
<td>Cameras</td>
<td></td>
</tr>
<tr>
<td>$18,215,609.08</td>
<td>DVD</td>
<td></td>
</tr>
<tr>
<td>$14,123,119.89</td>
<td>Digital Tape Recorders</td>
<td></td>
</tr>
<tr>
<td>$82,376,713.78</td>
<td>FDA Devices</td>
<td></td>
</tr>
<tr>
<td>$10,015,249.62</td>
<td>VCRs</td>
<td></td>
</tr>
</tbody>
</table>
Stacking Columns

**How to:**
- Stack Columns With FOLD-LINE
- Stack Columns With OVER

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** to reduce 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. FOLD-LINE is available for PDF and PS reports, but not cell-based formats such as HTML and EXL2K.

- **OVER** to stack 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 1307.

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

```plaintext
display_command fieldname ... FOLD-LINE fieldname ...
```

or

```plaintext
{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.

```sql
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:

<table>
<thead>
<tr>
<th>Category</th>
<th>Unit Sales</th>
<th>Budget Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1376266</td>
<td>1385923</td>
</tr>
<tr>
<td>Food</td>
<td>1384845</td>
<td>1377564</td>
</tr>
<tr>
<td>Gifts</td>
<td>927880</td>
<td>931007</td>
</tr>
</tbody>
</table>

Without FOLD-LINE, the report looks like this:

<table>
<thead>
<tr>
<th>Category</th>
<th>Unit Sales</th>
<th>Budget Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1376266</td>
<td>1385923</td>
</tr>
<tr>
<td>Food</td>
<td>1384845</td>
<td>1377564</td>
</tr>
<tr>
<td>Gifts</td>
<td>927880</td>
<td>931007</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>MIS</th>
<th>PRODUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROSS</td>
<td>$50,499.09</td>
<td>$50,922.38</td>
</tr>
<tr>
<td>DED_AMT</td>
<td>$28,187.11</td>
<td>$23,391.24</td>
</tr>
<tr>
<td>NET</td>
<td>$22,311.98</td>
<td>$27,531.14</td>
</tr>
</tbody>
</table>

Without the use of OVER, the HTML report looks like this:

```
DEPARTMENT
MIS                  PRODUCTION
<table>
<thead>
<tr>
<th>GROSS</th>
<th>DED_AMT</th>
<th>NET</th>
<th>GROSS</th>
<th>DED_AMT</th>
<th>NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>$50,499.09</td>
<td>$28,187.11</td>
<td>$22,311.98</td>
<td>$50,922.38</td>
<td>$23,391.24</td>
<td>$27,531.14</td>
</tr>
</tbody>
</table>
```

**Positioning a Column**

**How to:**
Position 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:

<table>
<thead>
<tr>
<th>Total # Problems</th>
<th>Product Name:</th>
<th># Problems by Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>95</td>
<td>120 VHS-C Camcorder 40 X</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2 Hd VCR LCD Menu</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>250 8MM Camcorder 40 X</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>650DL Digital Camcorder 150 X</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>AR2 35MM Camera 8 X</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>AR3 35MM Camera 10 X</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Combo Player - 4 Hd VCR + DVD</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>DVD Upgrade Unit for Cent. VCR</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>QX Portable CD Player</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>R5 Micro Digital Tape Recorder</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>ZC Digital PDA - Standard</td>
<td>7</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Problem</th>
<th>BOS</th>
<th>ORL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect Labeling</td>
<td>39921</td>
<td>19877</td>
</tr>
<tr>
<td>Mechanical Failure</td>
<td>19564</td>
<td>13792</td>
</tr>
<tr>
<td>Missing Component</td>
<td>38285</td>
<td>18709</td>
</tr>
<tr>
<td>Physical Damage</td>
<td>65497</td>
<td>16175</td>
</tr>
<tr>
<td>Power Failure</td>
<td>41472</td>
<td>14277</td>
</tr>
<tr>
<td>Remote Failure</td>
<td>59655</td>
<td>8635</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>LAST_NAME</th>
<th>GROSS</th>
<th>DED_AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>BLACKWOOD</td>
<td>$9,075.00</td>
<td>$6,307.00</td>
</tr>
<tr>
<td></td>
<td>CROSS</td>
<td>$22,013.75</td>
<td>$15,377.40</td>
</tr>
<tr>
<td></td>
<td>GREENSPAN</td>
<td>$2,970.84</td>
<td>$505.04</td>
</tr>
<tr>
<td></td>
<td>JONES</td>
<td>$6,099.50</td>
<td>$2,866.18</td>
</tr>
<tr>
<td></td>
<td>MCCOY</td>
<td>$1,540.00</td>
<td>$458.69</td>
</tr>
<tr>
<td></td>
<td>SMITH</td>
<td>$8,800.00</td>
<td>$2,672.80</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>BANNING</td>
<td>$2,475.00</td>
<td>$1,427.24</td>
</tr>
<tr>
<td></td>
<td>IRVING</td>
<td>$17,094.00</td>
<td>$11,949.44</td>
</tr>
<tr>
<td></td>
<td>MCKNIGHT</td>
<td>$9,130.00</td>
<td>$3,593.92</td>
</tr>
<tr>
<td></td>
<td>ROMANS</td>
<td>$7,040.00</td>
<td>$3,507.88</td>
</tr>
<tr>
<td></td>
<td>SMITH</td>
<td>$6,183.36</td>
<td>$1,104.96</td>
</tr>
<tr>
<td></td>
<td>STEVENS</td>
<td>$9,000.02</td>
<td>$1,807.80</td>
</tr>
</tbody>
</table>

**Suppressing Column Display**

**How to:**
Suppress Column Display

**Reference:**
Column Suppression Commands

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.
Column Suppression Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOPRINT or</td>
<td>Suppresses column display.</td>
<td>HTML, PDF,</td>
</tr>
<tr>
<td>SUP-PRINT</td>
<td></td>
<td>PS</td>
</tr>
</tbody>
</table>

How to Suppress Column Display

display_commandfieldname {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

<table>
<thead>
<tr>
<th>PROD_CODE</th>
<th>UNIT SOLD</th>
<th>DELIVER AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>B10</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>B17</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>B20</td>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>C17</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>D12</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>E1</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>E3</td>
<td>35</td>
<td>25</td>
</tr>
</tbody>
</table>

**Page 2**
SALES REPORT FOR NEWARK

<table>
<thead>
<tr>
<th>PROD_CODE</th>
<th>UNIT SOLD</th>
<th>DELIVER AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>B10</td>
<td>13</td>
<td>30</td>
</tr>
<tr>
<td>B12</td>
<td>29</td>
<td>30</td>
</tr>
</tbody>
</table>
Without NOPRINT, the report would unnecessarily repeat the city:

**Page 1**

**SALES REPORT FOR NEW YORK**

<table>
<thead>
<tr>
<th>CITY</th>
<th>PROD CODE</th>
<th>UNIT</th>
<th>SOLD</th>
<th>DELIVER</th>
<th>AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW YORK</td>
<td>B10</td>
<td>30</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B17</td>
<td>20</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B20</td>
<td>15</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C17</td>
<td>12</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>D12</td>
<td>20</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E1</td>
<td>30</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E3</td>
<td>35</td>
<td>25</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Page 2**

**SALES REPORT FOR NEWARK**

<table>
<thead>
<tr>
<th>CITY</th>
<th>PROD CODE</th>
<th>UNIT</th>
<th>SOLD</th>
<th>DELIVER</th>
<th>AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEWARK</td>
<td>B10</td>
<td>13</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B12</td>
<td>29</td>
<td>30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**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):

<table>
<thead>
<tr>
<th>Product</th>
<th>Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capuccino</td>
<td>189217</td>
</tr>
<tr>
<td>Espresso</td>
<td>308986</td>
</tr>
<tr>
<td>Latte</td>
<td>878063</td>
</tr>
<tr>
<td><strong>TOTAL Coffee</strong></td>
<td><strong>1376266</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product</th>
<th>Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscotti</td>
<td>421377</td>
</tr>
<tr>
<td>Croissant</td>
<td>630054</td>
</tr>
<tr>
<td>Scone</td>
<td>333414</td>
</tr>
<tr>
<td><strong>TOTAL Food</strong></td>
<td><strong>1384845</strong></td>
</tr>
</tbody>
</table>

Creating Reports With WebFOCUS Language 965
Without SUP-PRINT, the report would unnecessarily repeat the category:

<table>
<thead>
<tr>
<th>Category Product</th>
<th>Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td></td>
</tr>
<tr>
<td>Capuccino</td>
<td>189217</td>
</tr>
<tr>
<td>Espresso</td>
<td>308986</td>
</tr>
<tr>
<td>Latte</td>
<td>878063</td>
</tr>
</tbody>
</table>

*TOTAL Coffee 1376266

<table>
<thead>
<tr>
<th>Category Product</th>
<th>Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Food</td>
<td></td>
</tr>
<tr>
<td>Biscotti</td>
<td>421377</td>
</tr>
<tr>
<td>Croissant</td>
<td>630054</td>
</tr>
<tr>
<td>Scone</td>
<td>333414</td>
</tr>
</tbody>
</table>

*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
SMITH
SMITH
STEVENS
```
Inserting a Page Break

In this section:
Preventing an Undesirable Split

How to:
Insert a Page Break

Reference:
Page Break Commands
Working With Multi-Table HTML Reports

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 Web page. It does not, by itself, create a new Web page. To create multiple Web pages 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 691.

- Use SET STYLEMODE=PAGED in conjunction with PAGE-BREAK. This is useful when you are distributing a report via ReportCaster. For an example, see Displaying a Multiple-Table HTML Report on page 971.

- 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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE-BREAK</td>
<td>Generates new page.</td>
<td>HTML, PDF, PS</td>
</tr>
<tr>
<td>NOSPLIT</td>
<td>Prevents undesirable page break.</td>
<td>PDF, PS</td>
</tr>
<tr>
<td>SET LINES</td>
<td>Synchronizes report page with browser page.</td>
<td>HTML</td>
</tr>
</tbody>
</table>

Reference: Working With Multi-Table HTML Reports

- You can control where a report breaks using SET LINES or PAGE-BREAK in the request.
- If SET LINES=999999, the entire report is matched to a single displayed Web page, even if SET STYLEMODE=PAGED.
- 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.
Inserting a Page Break

**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 825 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
```

The first two pages of the report are displayed to illustrate where the page breaks occur:
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.

\[
\text{SET STYLEMODE = PAGED} \\
\text{SET LINES = 12} \\
\text{TABLE FILE CENTORD} \\
\text{HEADING} \\
"\text{SALES OVER $200,000}" \\
\text{PRINT LINEPRICE AS 'Sales'} \\
\text{BY SNAME BY ORDER_NUM} \\
\text{WHERE LINEPRICE GT 200000} \\
\text{ON TABLE SET STYLE *} \\
\text{TYPE=REPORT, GRID=OFF, SQUEEZE=OFF, $} \\
\text{ENDSTYLE} \\
\text{END}
\]
Two pages of the report follow, showing consistent alignment:

**PAGE 10**

**SALES OVER $200,000**

<table>
<thead>
<tr>
<th>Store Name</th>
<th>Order Number</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio Expert</td>
<td>59079</td>
<td>$241,754.82</td>
</tr>
<tr>
<td></td>
<td>59487</td>
<td>$299,272.98</td>
</tr>
<tr>
<td></td>
<td>59515</td>
<td>$241,570.90</td>
</tr>
<tr>
<td></td>
<td>74080</td>
<td>$200,724.66</td>
</tr>
<tr>
<td></td>
<td>74549</td>
<td>$326,165.31</td>
</tr>
<tr>
<td></td>
<td>74557</td>
<td>$267,960.68</td>
</tr>
</tbody>
</table>

**PAGE 11**

**SALES OVER $200,000**

<table>
<thead>
<tr>
<th>Store Name</th>
<th>Order Number</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio Expert</td>
<td>74610</td>
<td>$222,234.35</td>
</tr>
<tr>
<td></td>
<td>76810</td>
<td>$286,025.60</td>
</tr>
<tr>
<td></td>
<td>77790</td>
<td>$504,891.18</td>
</tr>
<tr>
<td>City Video</td>
<td>54691</td>
<td>$256,552.45</td>
</tr>
<tr>
<td></td>
<td>74691</td>
<td>$250,675.75</td>
</tr>
<tr>
<td>Consumer Merchandise</td>
<td>37395</td>
<td>$283,885.51</td>
</tr>
</tbody>
</table>
The same two pages illustrate inconsistent alignment with SQUEEZE set to ON:

PAGE 10

SALES OVER $200,000

<table>
<thead>
<tr>
<th>Store Name:</th>
<th>Order Number</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio Expert</td>
<td>59079</td>
<td>$241,754.82</td>
</tr>
<tr>
<td></td>
<td>59487</td>
<td>$299,272.98</td>
</tr>
<tr>
<td></td>
<td>59515</td>
<td>$241,570.90</td>
</tr>
<tr>
<td></td>
<td>74080</td>
<td>$200,724.66</td>
</tr>
<tr>
<td></td>
<td>74549</td>
<td>$326,165.31</td>
</tr>
<tr>
<td></td>
<td>74557</td>
<td>$267,960.68</td>
</tr>
</tbody>
</table>

PAGE 11

SALES OVER $200,000

<table>
<thead>
<tr>
<th>Store Name:</th>
<th>Order Number</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio Expert</td>
<td>74610</td>
<td>$222,234.35</td>
</tr>
<tr>
<td></td>
<td>76810</td>
<td>$286,025.60</td>
</tr>
<tr>
<td></td>
<td>77790</td>
<td>$504,891.18</td>
</tr>
<tr>
<td>City Video</td>
<td>54691</td>
<td>$256,552.45</td>
</tr>
<tr>
<td></td>
<td>74691</td>
<td>$250,675.75</td>
</tr>
<tr>
<td>Consumer Merchandise</td>
<td>37395</td>
<td>$283,885.51</td>
</tr>
</tbody>
</table>

Preventing an Undesirable Split

How to:
Prevent 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:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SMITH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HLTH</td>
<td>$50.87</td>
<td>LIFE</td>
<td>$30.52</td>
</tr>
<tr>
<td></td>
<td>SAVE</td>
<td>$122.10</td>
<td>STAT</td>
<td>$85.47</td>
</tr>
<tr>
<td></td>
<td>CITY</td>
<td>$1.43</td>
<td>FED</td>
<td>$121.55</td>
</tr>
<tr>
<td></td>
<td>FICA</td>
<td>$100.10</td>
<td>HLTH</td>
<td>$22.75</td>
</tr>
<tr>
<td></td>
<td>LIFE</td>
<td>$13.65</td>
<td>SAVE</td>
<td>$54.60</td>
</tr>
<tr>
<td></td>
<td>STAT</td>
<td>$20.02</td>
<td>CITY</td>
<td>$.75</td>
</tr>
<tr>
<td></td>
<td>FED</td>
<td>$64.10</td>
<td>FICA</td>
<td>$52.79</td>
</tr>
<tr>
<td></td>
<td>STAT</td>
<td>$10.56</td>
<td>CITY</td>
<td>$.92</td>
</tr>
<tr>
<td>STEVENS</td>
<td>FED</td>
<td>$77.92</td>
<td>FICA</td>
<td>$64.17</td>
</tr>
<tr>
<td></td>
<td>HLTH</td>
<td>$8.68</td>
<td>LIFE</td>
<td>$5.21</td>
</tr>
<tr>
<td></td>
<td>SAVE</td>
<td>$20.83</td>
<td>STAT</td>
<td>$12.83</td>
</tr>
<tr>
<td>PAY_DATE</td>
<td>LAST_NAME</td>
<td>DED_CODE</td>
<td>DED_AMT</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>----------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>82/02/26</td>
<td>CROSS</td>
<td>CITY</td>
<td>$7.52</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FED</td>
<td>$638.96</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FICA</td>
<td>$526.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>HLTH</td>
<td>$32.22</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LIFE</td>
<td>$19.33</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAVE</td>
<td>$77.32</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>STAT</td>
<td>$105.24</td>
<td></td>
</tr>
<tr>
<td>IRVING</td>
<td>CITY</td>
<td>$6.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FED</td>
<td>$518.92</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FICA</td>
<td>$427.35</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HLTH</td>
<td>$50.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LIFE</td>
<td>$30.52</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SAVE</td>
<td>$122.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>STAT</td>
<td>$85.47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>CITY</td>
<td>$2.51</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FED</td>
<td>$213.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FICA</td>
<td>$175.56</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HLTH</td>
<td>$18.81</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LIFE</td>
<td>$11.29</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SAVE</td>
<td>$45.14</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Without NOSPLIT, the information for CROSS falls on the first and second pages:

<table>
<thead>
<tr>
<th></th>
<th>LIFE</th>
<th>SAVE</th>
<th>STAT</th>
<th>CITY</th>
<th>PED</th>
<th>FICA</th>
<th>HLTH</th>
<th>LIFE</th>
<th>SAVE</th>
<th>STAT</th>
<th>CITY</th>
<th>FED</th>
<th>PICA</th>
<th>STAT</th>
<th>82/02/26 CROSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMITH</td>
<td></td>
<td></td>
<td></td>
<td>$1.43</td>
<td>$121.55</td>
<td>$100.10</td>
<td>$22.75</td>
<td>$13.65</td>
<td>$54.60</td>
<td>$20.02</td>
<td>$ .75</td>
<td>$64.10</td>
<td>$52.79</td>
<td>$10.56</td>
<td></td>
</tr>
<tr>
<td>STEVENS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LIFE</td>
<td>SAVE</td>
<td>STAT</td>
<td>CITY</td>
<td>PED</td>
<td>FICA</td>
<td>HLTH</td>
<td>LIFE</td>
<td>SAVE</td>
<td>STAT</td>
<td>CITY</td>
<td>FED</td>
<td>PICA</td>
<td>STAT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$30.52</td>
<td>$122.10</td>
<td>$85.47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CROSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

82/02/26 CROSS
<table>
<thead>
<tr>
<th>PAY_DATE</th>
<th>LAST_NAME</th>
<th>DED_CODE</th>
<th>DED_AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>82/02/26</td>
<td>CROSS</td>
<td>HLTH</td>
<td>$32.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LIFE</td>
<td>$19.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAVE</td>
<td>$77.32</td>
</tr>
<tr>
<td></td>
<td>IRVING</td>
<td>STAT</td>
<td>$105.24</td>
</tr>
<tr>
<td></td>
<td>CITY</td>
<td>CICA</td>
<td>$427.35</td>
</tr>
<tr>
<td></td>
<td>PECI</td>
<td>HLTH</td>
<td>$50.87</td>
</tr>
<tr>
<td></td>
<td>LIFE</td>
<td>LIFE</td>
<td>$30.52</td>
</tr>
<tr>
<td></td>
<td>SAVE</td>
<td>STAT</td>
<td>$122.10</td>
</tr>
<tr>
<td></td>
<td>MCKNIGHT</td>
<td>CITY</td>
<td>$2.51</td>
</tr>
<tr>
<td></td>
<td>PECI</td>
<td>PECI</td>
<td>$213.18</td>
</tr>
<tr>
<td></td>
<td>HLTH</td>
<td>HLTH</td>
<td>$18.81</td>
</tr>
<tr>
<td></td>
<td>LIFE</td>
<td>LIFE</td>
<td>$11.29</td>
</tr>
<tr>
<td></td>
<td>SAVE</td>
<td>STAT</td>
<td>$45.14</td>
</tr>
<tr>
<td></td>
<td>SMITH</td>
<td>CITY</td>
<td>$1.43</td>
</tr>
<tr>
<td></td>
<td>PECI</td>
<td>PECI</td>
<td>$121.55</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PICA</td>
<td>$100.10</td>
</tr>
</tbody>
</table>
Inserting Page Numbers

In this section:
- Inserting the Total Page Count
- Displaying the Total Page Count Within a Sort Group
- Assigning Any Page Number to the First Page
- Controlling the Display of Page Numbers

How to:
- Insert the Current Page Number

Reference:
- Page Number Commands

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.

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.

If you enable Section 508 accessibility, a default page number is not included in the HTML table.

Reference: Page Number Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;BYLASTPAGE</td>
<td>Used with REPAGE. Inserts the total page count within the sort group that has the REPAGE option.</td>
<td>HTML, PDF, PS</td>
</tr>
</tbody>
</table>
## Inserting Page Numbers

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPAGE</td>
<td>Resets page number to one.</td>
<td>HTML, PDF, PS</td>
</tr>
<tr>
<td>&lt;TABPAGENO</td>
<td>Inserts the current page number. TABPAGENO suppresses the default page number, and the top two lines of a page are blank.</td>
<td>HTML, PDF, PS</td>
</tr>
<tr>
<td>&lt;TABLASTPAGE</td>
<td>Inserts the total page count in the report.</td>
<td>HTML, PDF, PS</td>
</tr>
<tr>
<td>SET FOCFIRSTPAGE</td>
<td>Assigns the designated page number to the first page.</td>
<td>HTML, PDF, PS</td>
</tr>
<tr>
<td>SET PAGE-NUM</td>
<td>Controls page number display.</td>
<td>HTML, PDF, PS</td>
</tr>
</tbody>
</table>

### 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.

```plaintext
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:

<table>
<thead>
<tr>
<th>Region</th>
<th>State</th>
<th>City</th>
<th>Budget Dollars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midwest</td>
<td>IL</td>
<td>Chicago</td>
<td>3966856</td>
</tr>
<tr>
<td></td>
<td>MO</td>
<td>St. Louis</td>
<td>3646838</td>
</tr>
<tr>
<td></td>
<td>TX</td>
<td>Houston</td>
<td>3680679</td>
</tr>
</tbody>
</table>

Sales Quota for Midwest Cities
Page 1

**Inserting the Total Page Count**

**How to:**
Insert the Total Page Count

**Reference:**
Usage Notes for TABLASTPAGE

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:

```plaintext
<TABLASTPAGE
```
Usage Notes for TABLASTPAGE

- TABLASTPAGE, like TABPAGENO, is a legitimate field that can be used in a COMPUTE. However, TABLASTPAGE cannot be used in a DEFINE. DEFINEs are evaluated at retrieval time, which is before the number of pages of output is known.

- 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 pooled or 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.

```sql
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
```
Displaying the Total Page Count Within a Sort Group

How to:
Display the Total Number of Pages Within Each 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
Inserting Page Numbers

The heading or footing can use the following syntax to display “Page x of y”

\{HEADING|FOOTING\}
"Page <TABPAGENO of <BYLASTPAGE>"

where:

\textit{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.

\textbf{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:

\begin{verbatim}
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
\end{verbatim}
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:

<table>
<thead>
<tr>
<th>Region</th>
<th>Category</th>
<th>City</th>
<th>Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midwest</td>
<td>Food</td>
<td>Chicago</td>
<td>29413</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Houston</td>
<td>27504</td>
</tr>
<tr>
<td></td>
<td></td>
<td>St. Louis</td>
<td>29188</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Region</th>
<th>Category</th>
<th>City</th>
<th>Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Food</td>
<td>Boston</td>
<td>47064</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Haven</td>
<td>46214</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New York</td>
<td>51964</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Region</th>
<th>Category</th>
<th>City</th>
<th>Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Southeast</td>
<td>Food</td>
<td>Atlanta</td>
<td>43639</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Memphis</td>
<td>35349</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Orlando</td>
<td>40606</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Region</th>
<th>Category</th>
<th>City</th>
<th>Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>West</td>
<td>Food</td>
<td>Los Angeles</td>
<td>20773</td>
</tr>
<tr>
<td></td>
<td></td>
<td>San Francisco</td>
<td>22987</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Seattle</td>
<td>26676</td>
</tr>
</tbody>
</table>
Assigning Any Page Number to the First Page

How to: Assign a 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

<table>
<thead>
<tr>
<th>Store Name</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>AV VideoTown</td>
<td>$63,993,429.34</td>
</tr>
<tr>
<td>Audio Expert</td>
<td>$247,093,518.58</td>
</tr>
<tr>
<td>City Video</td>
<td>$15,299,051.29</td>
</tr>
<tr>
<td>Consumer Merchandise</td>
<td>$30,080,503.15</td>
</tr>
<tr>
<td>TV City</td>
<td>$83,041,182.81</td>
</tr>
<tr>
<td>Web Sales</td>
<td>$4,027,582.48</td>
</tr>
<tr>
<td>eMart</td>
<td>$290,487,756.36</td>
</tr>
</tbody>
</table>
```
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
Store Name: Sales
 eMart $290,487,756.36
PAGE 4
Sales By Product
Product Sales
 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.

   This setting issues page ejects.

   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.

   This setting does not issue page ejects.

   TOP 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
```

TABPAGENO inserts the page number in the sort footing. The first page of the report is:
Adding Grids and Borders

How to:
Control Grid Display in HTML Reports
Add and Format Borders
Add and Adjust Grid Lines (PDF or PS)

Reference:
Grid Display Attributes

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 an HTML, PDF, EXL2K, 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. BORDERS only work with the following syntax because they can only be implemented with internal Cascading Style Sheets:

ON TABLE SET HTMLCSS ON

For an example, see Inserting and Formatting a Border on page 995.

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRID</td>
<td>Controls grid display.</td>
<td>HTML</td>
</tr>
<tr>
<td>HGRID</td>
<td>Controls horizontal grid display and grid line density.</td>
<td>PDF, PS</td>
</tr>
<tr>
<td>VGRID</td>
<td>Control vertical grid display and grid line density.</td>
<td>PDF, PS</td>
</tr>
</tbody>
</table>

**Note:** When viewing PDF reports with the Acrobat 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:

\[
\text{TYPE} = \text{type}, \quad \text{BORDER-} \text{position} = \text{option}, \\
\hspace{1cm} \left\{ \begin{array}{l}
\text{BORDER[-position]}-\text{STYLE}=\text{line_style}, \\
\text{BORDER[-position]}-\text{COLOR} = \{ \text{color} | \text{RGB}(r \ g \ b) \}, \\
\end{array} \right. \\
\]

where:

**type**
Identifies the report component to which borders are applied. See *Identifying a Report Component in a WebFOCUS StyleSheet* on page 849, 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:

<table>
<thead>
<tr>
<th>Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>No border is drawn.</td>
</tr>
<tr>
<td>SOLID</td>
<td>Solid line.</td>
</tr>
<tr>
<td>DOTTED</td>
<td>Dotted line.</td>
</tr>
<tr>
<td>DASHED</td>
<td>Dashed line.</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>Double line.</td>
</tr>
<tr>
<td>GROOVE*</td>
<td>3D groove. (Not supported with Excel 2003, which has no option for specifying this type of border.)</td>
</tr>
<tr>
<td>RIDGE*</td>
<td>3D ridge. (Not supported with Excel 2003, which has no option for specifying this type of border.)</td>
</tr>
<tr>
<td>INSET*</td>
<td>3D inset.</td>
</tr>
<tr>
<td>OUTSET*</td>
<td>3D outset.</td>
</tr>
</tbody>
</table>

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 1205.

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=TABHEADING, STYLE=BOLD, JUSTIFY= CENTER, BORDER=HEAVY,
   BORDER-COLOR=RED, BORDER-STYLE=DOTTED, $
ENDSTYLE
END
```

The output is:

```
Sales Report
**CONFIDENTIAL**
December 2002

<table>
<thead>
<tr>
<th>Category</th>
<th>Budget Units</th>
<th>Unit Sales</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1385923</td>
<td>1376266</td>
<td>17293886</td>
<td>17231455</td>
</tr>
<tr>
<td>Food</td>
<td>1377564</td>
<td>1384845</td>
<td>17267160</td>
<td>17229333</td>
</tr>
<tr>
<td>Gifts</td>
<td>931007</td>
<td>927880</td>
<td>11659732</td>
<td>11695502</td>
</tr>
</tbody>
</table>
```

**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):

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>

**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
```
### Syntax: How to Add and Adjust Grid Lines (PDF or PS)

This syntax applies to a PDF or PS report.

\[ \text{TYPE} = \text{type}, \ (\text{HGRID} | \text{VGRID}) = \{ \text{ON} | \text{OFF} | \text{HEAVY} \}, \ $ \]

where:

- **type**
  - Identifies the report component to which grid lines are applied. See *Identifying a Report Component in a WebFOCUS StyleSheet* on page 849, 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.

---

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>
**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, $
END
```

In the PDF report, the lines make it easier to distinguish the data by state:

<table>
<thead>
<tr>
<th>State</th>
<th>Number of Households</th>
<th>Avg. Size</th>
<th>Avg. Income</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>11478067</td>
<td>3</td>
<td>44925</td>
</tr>
<tr>
<td>FL</td>
<td>5968392</td>
<td>2</td>
<td>34264</td>
</tr>
<tr>
<td>NY</td>
<td>6799434</td>
<td>3</td>
<td>42023</td>
</tr>
</tbody>
</table>
Adding Underlines and Skipped Lines

How to:

Add a Blank Line
Format a Blank Line
Underline a Sort Group
Format an Underline
Add or Remove a Report Component Underline
Remove an Underline From a Column Title
Extend an Underline to the Entire Report Column

Reference:

Section Separation Features
Usage Notes for the EXTUNDERLINE Attribute

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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKIP-LINE</td>
<td>Adds a blank line.</td>
<td>HTML (requires GRID=OFF) PDF PS</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Applies to</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>TYPE=SKIPLINE</td>
<td>Formats a blank line.</td>
<td>PDF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PS</td>
</tr>
<tr>
<td>UNDER-LINE</td>
<td>Underlines a sort group.</td>
<td>HTML (requires GRID=OFF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PDF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PS</td>
</tr>
<tr>
<td>TYPE=UNDERLINE</td>
<td>Formats an underline.</td>
<td>PDF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PS</td>
</tr>
<tr>
<td>STYLE={+</td>
<td>-}UNDERLINE</td>
<td>Adds an underline to a report component, or removes an underline from a</td>
</tr>
<tr>
<td></td>
<td>report component other than a column title.</td>
<td>PDF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PS</td>
</tr>
<tr>
<td>BAR AS '{-</td>
<td>=}'</td>
<td>Selects a light or heavy underline in an FML report.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PDF (displays single or double underline)</td>
</tr>
</tbody>
</table>

**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 315, 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.

```sql
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:

<table>
<thead>
<tr>
<th>EMP_ID</th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>CURR_SAL</th>
<th>INCREASE</th>
<th>NEWSAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>071382660</td>
<td>STEVENS</td>
<td>ALFRED</td>
<td>$11,000.00</td>
<td>$550.00</td>
<td>$11,550.00</td>
</tr>
<tr>
<td>112847612</td>
<td>SMITH</td>
<td>MARY</td>
<td>$13,200.00</td>
<td>$660.00</td>
<td>$13,860.00</td>
</tr>
<tr>
<td>117593129</td>
<td>JONES</td>
<td>DIANE</td>
<td>$18,480.00</td>
<td>$924.00</td>
<td>$19,404.00</td>
</tr>
<tr>
<td>119265415</td>
<td>SMITH</td>
<td>RICHARD</td>
<td>$9,500.00</td>
<td>$475.00</td>
<td>$9,975.00</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Product</th>
<th>Quantity In Stock</th>
</tr>
</thead>
<tbody>
<tr>
<td>110 VHS-C Camcorder 20 X</td>
<td>4000</td>
</tr>
<tr>
<td>120 VHS-C Camcorder 40 X</td>
<td>2300</td>
</tr>
<tr>
<td>340SX Digital Camera 65K P</td>
<td>990</td>
</tr>
<tr>
<td>650DL Digital Camcorder 150 X</td>
<td>2972</td>
</tr>
<tr>
<td>DVD Upgrade Unit for Cent. VCR</td>
<td>199</td>
</tr>
<tr>
<td>R5 Micro Digital Tape Recorder</td>
<td>1990</td>
</tr>
</tbody>
</table>
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 315, 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:

<table>
<thead>
<tr>
<th>BANK NAME</th>
<th>EMP ID</th>
<th>BANK ACCT</th>
<th>LAST NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>071382660</td>
<td>STEVENS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>112847612</td>
<td>SMITH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>119265415</td>
<td>SMITH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>126724188</td>
<td>ROMANS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>219984371</td>
<td>MCCOY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>543729165</td>
<td>GREENSPAN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ASSOCIATED | 123764317 | 819000702 IRVING |
| 326179357 | BLACKWOOD |
| 451123478 | MCKNIGHT |

BANK ASSOCIATION | 818692173 | 163800144 CROSS |

BEST BANK | 119329144 | 160633 BANNING |

STATE | 117593129 | 40950036 JONES |

**Syntax:**

**How to Format an Underline**

```plaintext
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 *Formatting Report Data* on page 1205.

**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:

[Table showing sales data with a red underline]
**Syntax:** **How to Add or Remove a Report Component Underline**

```plaintext
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 849.

- **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 849.

- **+**
  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.

```plaintext
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 849.
**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.

```plaintext
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:

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>TITLE</th>
<th>DIRECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JAWS</td>
<td>SPIELBERG S.</td>
</tr>
<tr>
<td></td>
<td>ROBOCOP</td>
<td>VERHOVEN P.</td>
</tr>
<tr>
<td></td>
<td>TOTAL RECALL</td>
<td>VERHOVEN P.</td>
</tr>
<tr>
<td></td>
<td>TOP GUN</td>
<td>SCOTT T.</td>
</tr>
<tr>
<td></td>
<td>RAMBO III</td>
<td>MCDONALD P.</td>
</tr>
<tr>
<td>CHILDREN</td>
<td>SMURFS, THE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHAGGY DOG, THE</td>
<td>BARTON C.</td>
</tr>
<tr>
<td></td>
<td>SCOOBY-DOO-A DOG IN THE RUFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ALICE IN WONDERLAND</td>
<td>GEROMINI</td>
</tr>
<tr>
<td></td>
<td>SESAME STREET-BEDTIME STORIES AND SONGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ROMPER ROOM-ASK MISS MOLLY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SLEEPING BEAUTY</td>
<td>DISNEY W.</td>
</tr>
<tr>
<td></td>
<td>BAMBI</td>
<td>DISNEY W.</td>
</tr>
<tr>
<td>CLASSIC</td>
<td>EAST OF EDEN</td>
<td>KAZAN E.</td>
</tr>
<tr>
<td></td>
<td>CITIZEN KANE</td>
<td>WELLES O.</td>
</tr>
<tr>
<td></td>
<td>CYRANO DE BERGERAC</td>
<td>GORDON M.</td>
</tr>
<tr>
<td></td>
<td>MARTY</td>
<td>MANN D.</td>
</tr>
<tr>
<td></td>
<td>MALTESE FALCON, THE</td>
<td>HUSTON J.</td>
</tr>
<tr>
<td></td>
<td>GONE WITH THE WIND</td>
<td>FLEMING V.</td>
</tr>
</tbody>
</table>

---

Creating Reports With WebFOCUS Language 1009
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, and PPT.

\[
\text{TYPE = TITLE, [COLUMN = colspec,] STYLE = [+|-]EXTUNDERLINE ,}$
\]

where:

\[\text{colspec}\]

Is any valid column specification.

\[\text{+EXTUNDERLINE}\]

Adds the EXTUNDERLINE option to the inherited text style or specifies a combination of text styles (for example, STYLE=BOLD+UNDERLINE).

\[\text{-EXTUNDERLINE}\]

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
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:

<table>
<thead>
<tr>
<th>Date</th>
<th>City</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997/01/01</td>
<td>Los Angeles</td>
<td>159935</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>162712</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>159804</td>
</tr>
<tr>
<td>1997/02/01</td>
<td>Los Angeles</td>
<td>148915</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>143987</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>132505</td>
</tr>
<tr>
<td>1997/03/01</td>
<td>Los Angeles</td>
<td>165902</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>145129</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>165847</td>
</tr>
<tr>
<td>1997/04/01</td>
<td>Los Angeles</td>
<td>146106</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>158799</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>144169</td>
</tr>
<tr>
<td>1997/05/01</td>
<td>Los Angeles</td>
<td>178336</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>144534</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>190208</td>
</tr>
</tbody>
</table>
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
  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:

<table>
<thead>
<tr>
<th>Date</th>
<th>City</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997/01/01</td>
<td>Los Angeles</td>
<td>159935</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>162712</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>159804</td>
</tr>
<tr>
<td>1997/02/01</td>
<td>Los Angeles</td>
<td>148915</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>143987</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>132505</td>
</tr>
<tr>
<td>1997/03/01</td>
<td>Los Angeles</td>
<td>165902</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>145129</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>165847</td>
</tr>
<tr>
<td>1997/04/01</td>
<td>Los Angeles</td>
<td>146106</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>158799</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>144169</td>
</tr>
<tr>
<td>1997/05/01</td>
<td>Los Angeles</td>
<td>178336</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>144534</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>190208</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Date</th>
<th>City</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997/01/01</td>
<td>Los Angeles</td>
<td>159935</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>162712</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>159804</td>
</tr>
<tr>
<td>1997/02/01</td>
<td>Los Angeles</td>
<td>148915</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>143987</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>132505</td>
</tr>
<tr>
<td>1997/03/01</td>
<td>Los Angeles</td>
<td>165902</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>145129</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>165847</td>
</tr>
<tr>
<td>1997/04/01</td>
<td>Los Angeles</td>
<td>146106</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>158799</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>144169</td>
</tr>
<tr>
<td>1997/05/01</td>
<td>Los Angeles</td>
<td>178336</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>144534</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>190208</td>
</tr>
</tbody>
</table>
**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:

<table>
<thead>
<tr>
<th>AMOUNT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH ON HAND</td>
<td>8,784</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
<td>4,494</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
<td>7,961</td>
</tr>
</tbody>
</table>

| TOTCASH         | 21,239 |

1014 WebFOCUS
**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.

```plaintext
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:

<table>
<thead>
<tr>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH ON HAND</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
</tr>
<tr>
<td>TOTCASH</td>
</tr>
</tbody>
</table>
```
Adding an Image to a Report

How to:
Add an Image to an HTML Report
Add a Background Image
Add an Image to a PDF, PS, or HTML Report With Internal Cascading Style Sheet

Reference:
Image Attributes
Specifying a URL
File Size and Compression Considerations For Images in BLOB Fields

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). PDF and PowerPoint reports support GIF, JPEG, and SVG files. 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.

Reference: Image Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAGE</td>
<td>Adds an image.</td>
</tr>
<tr>
<td>IMAGEALIGN</td>
<td>Positions an image. This applies only to HTML reports.</td>
</tr>
<tr>
<td>POSITION</td>
<td>Positions an image.</td>
</tr>
<tr>
<td>IMAGEBREAK</td>
<td>Controls generation of a line break after an image. This applies only to HTML reports without internal Cascading Style Sheets.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SIZE</td>
<td>Sizes an image.</td>
</tr>
<tr>
<td>ALT</td>
<td>Supplies a description of an image for compliance with Section accessibility (Workforce Investment Act of 1998). ALT only applies to HTML reports. The description also displays as a pop-up description when your mouse or cursor hovers over the image in the report output.</td>
</tr>
<tr>
<td>PRESERVERATIO</td>
<td>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.</td>
</tr>
<tr>
<td>BACKIMAGE</td>
<td>Adds a background image.</td>
</tr>
</tbody>
</table>

**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 an HTML report with internal CSS, see *How to Add an Image to a PDF, PS, or HTML Report With Internal Cascading Style Sheet* on page 1026.

\[
\text{TYPE} = \{\text{REPORT} | \text{heading}\}, \ \text{IMAGE} = \{\text{url} | (\text{column})\} \ [,\text{IMAGEALIGN} = \text{position}\] \\
\[,\text{IMAGEBREAK} = \{\text{ON} | \text{OFF}\} \ [,\text{ALT} = '\text{description}'\], \$
\]

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.
Adding an Image to a Report

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. 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 1019.

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 849. Enclose column in parentheses.

This option enables you to add different images to a heading or footing, depending on the value of the field. See Using a File Name in a Data Source Field in an HTML Report on page 1022.

position
Is the position of the image. 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.

The description also displays as a pop-up description when your mouse or cursor hovers over the image in the report output.
Specifying a URL

Reference: 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_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=D:\ibi\apps\SESSION\
  ...
  ...
  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.

```sql
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=TABHEADING, IMAGE=/ibi_html/ggdemo/gotham.gif, IMAGEBREAK=ON,$
ENDSTYLE
END
```
IMAGEBREAK, set to ON, generates a line break between the logo and the heading text:

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
```
# EMPLOYEE SALARY INFORMATION AND SKILLS

<table>
<thead>
<tr>
<th>COURSE NAME</th>
<th>CURR SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADVANCED TECHNIQUES</td>
<td>$18,480.00</td>
</tr>
<tr>
<td>BASIC REPORT PREP DP MGRS</td>
<td>$27,062.00</td>
</tr>
<tr>
<td>BASIC REPORT PREP FOR PROG</td>
<td>$13,200.00</td>
</tr>
<tr>
<td></td>
<td>$18,480.00</td>
</tr>
<tr>
<td>BASIC REPORT PREP NON-PROG</td>
<td>$21,780.00</td>
</tr>
<tr>
<td>BASIC RPT NON-DP MGRS</td>
<td>$9,500.00</td>
</tr>
<tr>
<td>DECISION SUPPORT WORKSHOP</td>
<td>$21,780.00</td>
</tr>
<tr>
<td>FILE DESC &amp; MAINT NON-PROG</td>
<td>$21,780.00</td>
</tr>
<tr>
<td>FILE DESCRPT &amp; MAINT</td>
<td>$11,000.00</td>
</tr>
<tr>
<td></td>
<td>$13,200.00</td>
</tr>
<tr>
<td></td>
<td>$18,480.00</td>
</tr>
<tr>
<td></td>
<td>$16,100.00</td>
</tr>
<tr>
<td>FOCUS INTERNALS</td>
<td>$18,480.00</td>
</tr>
<tr>
<td>HOST LANGUAGE INTERFACE</td>
<td>$27,062.00</td>
</tr>
<tr>
<td>TIMESHARING WORKSHOP</td>
<td>$21,780.00</td>
</tr>
<tr>
<td>WHAT'S NEW IN FOCUS</td>
<td>$21,780.00</td>
</tr>
</tbody>
</table>
**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.

```
SET BASEURL=http://ibi/apps/session/gifs/
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, GRIDL=OFF, $
TYPE=HEADING, SIZE=12, STYLE=BOLD, $
TYPE=SUBHEAD, STYLE=BOLD, $
ENDSTYLE
END
```
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 DX AUTO
Example: Supplying an Image Description Using the ALT Attribute

The following illustrates how to use the ALT attribute. The ALT attribute supplies a description of an image that screen readers can interpret to comply with Section 508 accessibility (Workforce Investment Act of 1998). The description also displays as a pop-up description when your mouse or cursor hovers over the image in the report output.

SET BASEURL=D:\ibi\apps\images\ 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=TABHEADING, IMAGE=gglogo, IMAGEBREAK=ON, POSITION=(.25 .25), SIZE=(.5 .5), ALT='Gotham Grinds Logo Image', $ GRID=OFF, $ ENDSTYLE END

The output is:

![Gotham Grinds Logo Image]

Report on Units Sold

<table>
<thead>
<tr>
<th>Product</th>
<th>Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscoth</td>
<td>421377</td>
</tr>
<tr>
<td>Capuccino</td>
<td>189217</td>
</tr>
<tr>
<td>Coffee Grinder</td>
<td>186534</td>
</tr>
<tr>
<td>Coffee Pot</td>
<td>190595</td>
</tr>
<tr>
<td>Croissant</td>
<td>630054</td>
</tr>
<tr>
<td>Espresso</td>
<td>308986</td>
</tr>
<tr>
<td>Latte</td>
<td>878063</td>
</tr>
<tr>
<td>Mug</td>
<td>360570</td>
</tr>
<tr>
<td>Scone</td>
<td>323414</td>
</tr>
<tr>
<td>Thermos</td>
<td>190081</td>
</tr>
</tbody>
</table>
**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. 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 1016.
  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_HTML/TEMPLATE/CALM_BKG.GIF, $
ENDSTYLE
END
```
The background is tiled across the report area:

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>

**Syntax:**  
*How to Add an Image to a PDF, PS, or HTML Report With Internal Cascading Style Sheet*

This syntax applies to a PDF, PS, or HTML report with internal Cascading Style Sheet. The image can be in a separate file, or, for data sources that support the Binary Large Object (BLOB) data type (Microsoft SQL Server, DB2, Oracle, Informix, and PostgreSQL using its BYTEA data type), it can be stored in a BLOB field in the data source.

```
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=(w h), [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).

**heading**

Embeds an image in a heading or footing. Valid values are TABHEADING (not supported if the image is in a BLOB field), TABFOOTING (not supported if the image is in a BLOB field), FOOTING, HEADING, SUBHEAD, and SUBFOOT.
If the image is in a BLOB field that 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.

**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. 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 1019.

**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 849. Enclose column in parentheses.

For a PDF report against a data source that supports the Binary Large Object (BLOB) data type, it can be a BLOB field that contains an exact binary copy of a GIF or JPEG image. Images of different formats (GIF, JPEG) 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 field containing the file name or image must be a display field or BY field referenced in the request.

If the field is a BLOB field that 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 or footing. 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, subheading, footing, or subfooting itself.
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 doesn’t 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. See Using a File Name in a Data Source Field in an HTML Report on page 1022.

**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.

**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.
For a BLOB field in a heading or footing component. If omitted, the original dimensions of the image are used (any GIF or JPEG image has an original, unscaled size based on the dimensions of its bitmap).

\[
\text{[PRESERVERATIO=\{ON|\textit{OFF}\}]}
\]

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.

**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=TABTHEADING, IMAGE=HTTP://WEBSRVR1/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

<table>
<thead>
<tr>
<th>Product</th>
<th>Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscotti</td>
<td>421377</td>
</tr>
<tr>
<td>Capuccino</td>
<td>189217</td>
</tr>
<tr>
<td>Coffee Grinder</td>
<td>186534</td>
</tr>
<tr>
<td>Coffee Pot</td>
<td>190695</td>
</tr>
<tr>
<td>Croissant</td>
<td>630054</td>
</tr>
<tr>
<td>Espresso</td>
<td>308986</td>
</tr>
<tr>
<td>Latte</td>
<td>878063</td>
</tr>
<tr>
<td>Mug</td>
<td>360370</td>
</tr>
<tr>
<td>Scone</td>
<td>333414</td>
</tr>
<tr>
<td>Thermos</td>
<td>190081</td>
</tr>
</tbody>
</table>
```

**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=TABHEADING, IMAGE=GOTHAM.GIF, POSITION=(.25 .25), SIZE=(.5 .5), $
ENDSTYLE
END
```
The report is:

Report on Units Sold

<table>
<thead>
<tr>
<th>Product</th>
<th>Unit Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscotti</td>
<td>421377</td>
</tr>
<tr>
<td>Capuccino</td>
<td>189217</td>
</tr>
<tr>
<td>Coffee Grinder</td>
<td>186534</td>
</tr>
<tr>
<td>Coffee Pot</td>
<td>190695</td>
</tr>
<tr>
<td>Croissant</td>
<td>630054</td>
</tr>
<tr>
<td>Espresso</td>
<td>308986</td>
</tr>
<tr>
<td>Latte</td>
<td>878063</td>
</tr>
<tr>
<td>Mug</td>
<td>360570</td>
</tr>
<tr>
<td>Scone</td>
<td>333414</td>
</tr>
<tr>
<td>Thermos</td>
<td>190081</td>
</tr>
</tbody>
</table>
**Example:**  **Displaying an Image From a BLOB Field in a Report Column**

The following Master File describes the Microsoft SQL Server data source named retaildetail, which contains product information for a sports clothing and shoe retailer:

```plaintext
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:

```plaintext
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.

```
JOIN PRODUCTID IN RETAILDETAIL TO PRODUCTID IN RETAILIMAGE
TABLE FILE RETAILDETAIL
HEADING CENTER
"Product List"
""
PRINT NAME/A30 PRICE PRODIMAGE AS 'PICTURE'
BY PRODUCTID NOPRINT
BY NAME NOPRINT
ON NAME UNDER-LINE
ON TABLE SET PAGE NOPAGE
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT,COLOR=BLUE, $
TYPE=HEADING, SIZE = 18, FONT = ARIAL, 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),$ END
```
### Product List

<table>
<thead>
<tr>
<th>NAME</th>
<th>PRICE</th>
<th>PICTURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mizia 9 Spike Classic Mid G4 M</td>
<td>94.00</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td>Kine Cal Ripken Mid Molded Bas</td>
<td>49.00</td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>Mizia 9 Spike Classic Low G4 M</td>
<td>99.00</td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td>Adonas Excelsior Low Metal Bas</td>
<td>49.00</td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
</tbody>
</table>
**Example:** Displaying an Image From a BLOB Field in a SUBHEAD

The following example joins the RETAILDETAIL Master File to the RETAILIMAGE Master File and displays a product catalog in which all of the data is displayed in a subhead. The subhead has extra blank lines after the text of the subheading to accommodate the image, and the vertical offset of 0.6 inches allows the top of the image to fall below the text. Note that the image field is specified in the request with the NOPRINT option and is not referenced in the subheading at all:

```
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
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, COLOR=BLUE, $
TYPE=HEADING, COLOR = RED, SIZE = 16, FONT = ARIAL, JUSTIFY=CENTER, $
TYPE=SUBHEAD, BY=PRODUCTID, IMAGE=(PRODIMAGE), SIZE=(1 1), POSITION=(+2 +.6),$
END

The StyleSheet declaration that places the image in the subheading is:

```
TYPE=SUBHEAD, BY=PRODUCTID, IMAGE=(PRODIMAGE), SIZE=(1 1),
POSITION=(+2 +.6),$
```
Example: Adjusting Image Size

The following example joins the RETAILDETAIL Master File to the RETAILIMAGE Master File and displays the same images three different ways:

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
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT,COLOR=BLUE,$
TYPE=DATA,COLUMN=P1,IMAGE=(PRODIMAGE),SIZE=(1 1),$  
TYPE=DATA,COLUMN=P2,IMAGE=(PRODIMAGE),SIZE=(2 3),PRESERVERATIO=OFF,$
TYPE=DATA,COLUMN=P3,IMAGE=(PRODIMAGE),SIZE=(2 3),PRESERVERATIO=ON,$
END
The size of the first displayed image is 1 inch by 1 inch:

```
TYPE=DATA, COLUMN=P1, IMAGE=(PRODIMAGE), SIZE=(1 1), $
```

The size of the second displayed image is 2 inches by 3 inches with PRESERVERATIO=OFF. This image looks distorted on the output:

```
TYPE=DATA, COLUMN=P2, IMAGE=(PRODIMAGE), SIZE=(2 3), PRESERVERATIO=OFF, $
```

The size of the third displayed image is 2 inches by 3 inches with PRESERVERATIO=ON. This image is not distorted on the output, but it is not exactly the size specified. It is the largest size that fits in the bounding box and maintains the aspect ratio:

```
TYPE=DATA, COLUMN=P3, IMAGE=(PRODIMAGE), SIZE=(2 3), PRESERVERATIO=ON, $
```

The partial output is:

---

**Example:** Displaying 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 following request joins the RETAILDETAIL data source to the RETAILIMAGE data source. It includes a SUM command and a PRINT command, but uses the NOPRINT option so that the fields can be displayed in a subheading and subfooting rather than in report columns. Prices are summed by category, and a subhead displays the category name and total price for the category. Prices are then printed by category and product ID, and a subfoot displays the image and product data for each product ID:

```
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 CATEGORY PAGE-BREAK
ON PRODUCTID SUBFOOT
""
 " <16 Product #: <PRODUCTID"
 " <16 Name: <NAME"
 " <16 Price: <FST.PRICE"
 ""
""
""
""
""
ON TABLE SET PAGE NOPAGE
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET STYLE *
TYPE=REPORT, COLOR=BLUE, $
TYPE=HEADING, COLOR = RED, SIZE = 14, FONT = ARIAL, STYLE=BOLD, 
JUSTIFY=CENTER, $
TYPE=SUBHEAD, COLOR = RED, SIZE = 12, FONT = ARIAL, 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, $
END
```
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.

PDF 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

In this section:
Controlling Bar Graph Scaling in Horizontal (ACROSS) Sort Fields
Formatting Data Visualization Bar Graphs

How to:
Associate Data Visualization Bar Graphs With Report Columns

To make HTML, PDF 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 a:

- **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.

<table>
<thead>
<tr>
<th>Region</th>
<th>Midwest</th>
<th>Northeast</th>
<th>Southeast</th>
<th>West</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difference</td>
<td>Difference</td>
<td>Difference</td>
<td>Difference</td>
<td>Difference</td>
</tr>
<tr>
<td>Dollar Sales from Budget</td>
<td>Dollar Sales from Budget</td>
<td>Dollar Sales from Budget</td>
<td>Dollar Sales from Budget</td>
<td></td>
</tr>
<tr>
<td>11400665</td>
<td>11392310</td>
<td>11710379</td>
<td>11652957</td>
<td></td>
</tr>
<tr>
<td>-$206,292.00</td>
<td>$184,622.00</td>
<td>$37,602.00</td>
<td>-$11,444.00</td>
<td></td>
</tr>
</tbody>
</table>

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 1042.
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.

<table>
<thead>
<tr>
<th>City</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
<th>DIFFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlanta</td>
<td>$4,247,597.00</td>
<td>$4,100,107.00</td>
<td>$147,490.00</td>
</tr>
<tr>
<td>Boston</td>
<td>$3,818,397.00</td>
<td>$3,707,986.00</td>
<td>$110,411.00</td>
</tr>
<tr>
<td>Chicago</td>
<td>$3,866,856.00</td>
<td>$3,924,401.00</td>
<td>-$57,545.00</td>
</tr>
<tr>
<td>Houston</td>
<td>$3,680,679.00</td>
<td>$3,714,978.00</td>
<td>-$34,299.00</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>$3,669,484.00</td>
<td>$3,772,014.00</td>
<td>$102,530.00</td>
</tr>
<tr>
<td>Memphis</td>
<td>$3,689,979.00</td>
<td>$3,687,057.00</td>
<td>$2,922.00</td>
</tr>
<tr>
<td>New Haven</td>
<td>$3,832,202.00</td>
<td>$3,782,049.00</td>
<td>$50,153.00</td>
</tr>
<tr>
<td>New York</td>
<td>$3,926,333.00</td>
<td>$3,902,275.00</td>
<td>$24,058.00</td>
</tr>
<tr>
<td>Orlando</td>
<td>$3,870,405.00</td>
<td>$3,923,215.00</td>
<td>$52,810.00</td>
</tr>
<tr>
<td>San Francisco</td>
<td>$3,916,863.00</td>
<td>$3,870,258.00</td>
<td>$46,605.00</td>
</tr>
<tr>
<td>Seattle</td>
<td>$4,055,166.00</td>
<td>$4,010,685.00</td>
<td>$44,481.00</td>
</tr>
<tr>
<td>St. Louis</td>
<td>$3,646,838.00</td>
<td>$3,761,286.00</td>
<td>$114,448.00</td>
</tr>
</tbody>
</table>

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.
**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 849.

You can define WHEN conditions and bar graph features associated with those conditions using StyleSheet syntax. For details, see Formatting Report Data on page 1205.

**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, $.

```
DEFINE FILE GGSALES
DIFFERENCE/D7M=DOLLARS-BUDDOLLARS;
END

TALE 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, $
ENDSTYLE
END
```
### Controlling Bar Graph Scaling in Horizontal (ACROSS) SortFields

#### How to:
Apply Vertical Scaling to Data Visualization Bar Graphs

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.

<table>
<thead>
<tr>
<th>City</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
<th>DIFFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlanta</td>
<td>$4,247,587</td>
<td>$4,100,107</td>
<td>$147,480</td>
</tr>
<tr>
<td>Boston</td>
<td>$3,818,397</td>
<td>$3,707,986</td>
<td>$110,411</td>
</tr>
<tr>
<td>Chicago</td>
<td>$3,866,856</td>
<td>$3,924,401</td>
<td>-$57,545</td>
</tr>
<tr>
<td>Houston</td>
<td>$3,680,679</td>
<td>$3,714,978</td>
<td>-$34,299</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>$3,669,484</td>
<td>$3,772,003</td>
<td>-$102,519</td>
</tr>
<tr>
<td>Memphis</td>
<td>$3,689,979</td>
<td>$3,687,057</td>
<td>$2,922</td>
</tr>
<tr>
<td>New Haven</td>
<td>$3,832,202</td>
<td>$3,782,049</td>
<td>$50,153</td>
</tr>
<tr>
<td>New York</td>
<td>$3,926,322</td>
<td>$3,902,265</td>
<td>$24,057</td>
</tr>
<tr>
<td>Orlando</td>
<td>$3,870,405</td>
<td>$3,923,215</td>
<td>-$52,810</td>
</tr>
<tr>
<td>San Francisco</td>
<td>$3,916,863</td>
<td>$3,870,258</td>
<td>$46,605</td>
</tr>
<tr>
<td>Seattle</td>
<td>$4,055,166</td>
<td>$4,010,685</td>
<td>$44,481</td>
</tr>
<tr>
<td>St. Louis</td>
<td>$3,646,838</td>
<td>$3,761,286</td>
<td>-$114,448</td>
</tr>
</tbody>
</table>
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 figure 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 each column's set of values, 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 Apply Vertical 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.

**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 AS 'Difference'
END

TABLE FILE GGSALES
SUM DOLLARS/D12.2M DIFFERENCE
ACROSS REGION
ON TABLE SET STYLE *
GRAPHTYPE=DATA, ACROSSCOLUMN=N1,$
GRAPHTYPE=DATA,ACROSSCOLUMN=N2,$
ON TABLE SET ONLINE-FMT HTML
END
```

This request produces the following report output:

<table>
<thead>
<tr>
<th>Region</th>
<th>Midwest</th>
<th>Northeast</th>
<th>Southeast</th>
<th>West</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dollar Sales</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Difference</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 AS 'Difference'
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,$
ON TABLE SET ONLINE-FMT HTML
END
```
Notice the difference in the output:

<table>
<thead>
<tr>
<th>Region</th>
<th>Midwest</th>
<th>Northeast</th>
<th>Southeast</th>
<th>West</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dollar Sales</td>
<td>Difference</td>
<td>Dollar Sales</td>
<td>Difference</td>
<td>Dollar Sales</td>
</tr>
<tr>
<td>$11,400,665.00</td>
<td>$206,292.00</td>
<td>$11,392,300.00</td>
<td>-$184,621.00</td>
<td>$11,710,379.00</td>
</tr>
</tbody>
</table>

Now each bar graph is plotted based on the set of values for each field.

**Formatting Data Visualization Bar Graphs**

You can specify optional formatting attributes for data visualization bar graphs. These attributes are graph color, patterns, length, and width. For complete details, see *Visualizing Trends in Reports* in the *Creating Reporting Applications With Developer Studio* manual.

**Working With Mailing Labels and Multi-Pane Pages**

**How to:**
- Set Up a Report to Print Mailing Labels
- Print Mailing Labels or a Multi-Pane Report

**Reference:**
- Attributes for Mailing Labels and Multi-Pane Printing

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.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGEMATRIX</td>
<td>Sets the number of columns and rows of labels on a page.</td>
<td>PDF, PS</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>Sets the width and height of each label, expressed in the unit of measurement specified by the UNITS parameter.</td>
<td>PDF, PS</td>
</tr>
<tr>
<td>GUTTER</td>
<td>Sets the horizontal and vertical distance between each label, expressed in the unit of measurement specified by the UNITS parameter.</td>
<td>PDF, PS</td>
</tr>
<tr>
<td>MATRIXORDER</td>
<td>Sets the order in which the labels are printed.</td>
<td>PDF, PS</td>
</tr>
<tr>
<td>LABELPROMPT</td>
<td>Sets the position of the first label on the mailing label sheet.</td>
<td>PDF, PS</td>
</tr>
</tbody>
</table>

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 <TABPAGENO"
ON TABLE SET STYLE *
UNITS=IN, PAGESIZE=LETTER, PAGEMATRIX=(2 1), ELEMENT=(3.5 8.0),
MATRIXORDER=VERTICAL, $
TYPE=REPORT, SIZE=8, $
END
```

The report prints as:

<table>
<thead>
<tr>
<th>PAGE 1</th>
<th>DEPARTMENT</th>
<th>LAST_NAME</th>
<th>CURR_SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MIS</td>
<td>SMITH</td>
<td>$13,200.00</td>
</tr>
<tr>
<td></td>
<td>JONES</td>
<td>$18,480.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MCCOY</td>
<td>$18,480.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLACKWOOD</td>
<td>$21,780.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GREENSPAN</td>
<td>$9,000.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CROSS</td>
<td>$27,062.00</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PAGE 2</th>
<th>DEPARTMENT</th>
<th>LAST_NAME</th>
<th>CURR_SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PRODUCTION</td>
<td>STEVENS</td>
<td>$11,000.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SMITH</td>
<td>$9,500.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BANNING</td>
<td>$29,700.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IRVING</td>
<td>$26,862.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ROMANS</td>
<td>$21,120.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MCKNIGHT</td>
<td>$16,180.00</td>
</tr>
</tbody>
</table>
After you have designed a report with the desired data, you can make it more meaningful by adding components such as headings, footings, titles, and labels. Headings and footings supply key information about a report, such as its purpose and audience. They also provide structure, helping the user navigate to the detail sought. Titles and labels identify individual pieces of data, ensuring correct interpretation.

Together or singly, these components supply context for data. They also enhance the visual appeal of a report and communicate a sense of completeness.

For information about adding contextual information to specialized report types, see Creating a Graph on page 1245, Creating Financial Reports With Financial Modeling Language (FML) on page 1307, and Creating a Free-Form Report on page 1397.
Creating Headings and Footings

In this section:

- Limits for Headings and Footings
- Extending Heading and Footing Code to Multiple Lines in a Report Request
- Creating a Custom Report or Worksheet Title
- Creating a Report Heading or Footing
- Creating a Page Heading or Footing
- Freezing HTML Headings, Footings, and Column Titles
- Creating a Sort Heading or Footing

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 1059.

- 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 1061.

- 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 1069.

- 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 1078.
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.

Limits for Headings and Footings

The following limitations apply to report headings and footings, page headings and footings, and sort headings and footings:

- In a single report, there can be a maximum of 32K characters for all types of heading and footing text.
- 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 <0X spot marker. For more information, see Extending Heading and Footing Code to Multiple Lines in a Report Request on page 1056.

The maximum number of objects per line in a heading or footing is 128.

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

**How to:**
Extend Heading or 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 <0X 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 <0X 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 <0X 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. For an illustration, see Extending Heading and Footing Code to Multiple Lines in a Report Request on page 1057.

**Example:** Extending Heading and Footing Code to Multiple Lines in a Report Request

This request creates a sort heading coded on two lines. The <0X 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

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Product Number#</th>
<th>Quantity</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>74610</td>
<td>1012</td>
<td>268</td>
<td>$222,234.35</td>
</tr>
<tr>
<td>1028</td>
<td>323</td>
<td>$29,215.62</td>
<td></td>
</tr>
<tr>
<td>1032</td>
<td>339</td>
<td>$27,208.12</td>
<td></td>
</tr>
<tr>
<td>1034</td>
<td>339</td>
<td>$154,943.38</td>
<td></td>
</tr>
<tr>
<td>1036</td>
<td>339</td>
<td>$93,266.39</td>
<td></td>
</tr>
</tbody>
</table>

Century Corporation orders for store Audio Expert (store # 1003CA) in CA.

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Product Number#</th>
<th>Quantity</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>39274</td>
<td>1006</td>
<td>179</td>
<td>$54,201.50</td>
</tr>
<tr>
<td>1008</td>
<td>179</td>
<td>$29,034.16</td>
<td></td>
</tr>
<tr>
<td>1020</td>
<td>179</td>
<td>$45,457.59</td>
<td></td>
</tr>
<tr>
<td>1032</td>
<td>400</td>
<td>$30,861.24</td>
<td></td>
</tr>
<tr>
<td>1034</td>
<td>400</td>
<td>$143,744.49</td>
<td></td>
</tr>
</tbody>
</table>

Century Corporation orders for store Audio Expert (store # 1003CO) in CO.

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Product Number#</th>
<th>Quantity</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>35995</td>
<td>1008</td>
<td>13</td>
<td>$2,132.45</td>
</tr>
<tr>
<td>1020</td>
<td>13</td>
<td>$3,895.38</td>
<td></td>
</tr>
<tr>
<td>1022</td>
<td>146</td>
<td>$52,159.13</td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>146</td>
<td>$45,306.84</td>
<td></td>
</tr>
<tr>
<td>1030</td>
<td>146</td>
<td>$18,525.49</td>
<td></td>
</tr>
</tbody>
</table>

Century Corporation orders for store Audio Expert (store # 1003CT) in CT.

**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

**How to:**

Create a Custom Report 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.

**Syntax:**

**How to Create a Custom Report Title**

Add the following declaration to your WebFOCUS StyleSheet:

```plaintext
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 128. Any text that exceeds 128 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 <, >, &, etc. 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.
Creating Headings and Footings

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:

![1999 Sales Report](image)

<table>
<thead>
<tr>
<th>Region</th>
<th>Annualized</th>
</tr>
</thead>
<tbody>
<tr>
<td>CENTRAL AMERICA</td>
<td>9.520</td>
</tr>
<tr>
<td>EASTERN EUROPE</td>
<td>16.150</td>
</tr>
<tr>
<td>FAR EAST</td>
<td>9.100</td>
</tr>
<tr>
<td>MIDDLE EAST</td>
<td>7.980</td>
</tr>
<tr>
<td>NORTH AMERICA</td>
<td>12.460</td>
</tr>
<tr>
<td>SOUTH AMERICA</td>
<td>8.400</td>
</tr>
<tr>
<td>WESTERN EUROPE</td>
<td>7.980</td>
</tr>
</tbody>
</table>
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:

![Image of a report with regions and returns]

Creating a Report Heading or Footing

**How to:**

Create a Report Heading

Create a Report 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 1201.

**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 1055.
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 1093.

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 873.

</n specifies skipped lines. For details, see Controlling the Vertical Positioning of a Heading or Footing on page 1193.

<-n to position the next character on the line. For details, see Using Spot Markers to Refine Positioning on page 1188.

<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 1056.

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.)
Creating Headings and Footings

**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 STYLESHEET *
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.

<table>
<thead>
<tr>
<th>Store ID</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1019</td>
<td>18560</td>
<td>25680</td>
</tr>
<tr>
<td></td>
<td>21132</td>
<td>24010</td>
</tr>
<tr>
<td></td>
<td>15880</td>
<td>22919</td>
</tr>
<tr>
<td></td>
<td>24450</td>
<td>22605</td>
</tr>
<tr>
<td></td>
<td>14556</td>
<td>18942</td>
</tr>
<tr>
<td></td>
<td>13860</td>
<td>18915</td>
</tr>
<tr>
<td></td>
<td>22806</td>
<td>18837</td>
</tr>
<tr>
<td></td>
<td>19248</td>
<td>18535</td>
</tr>
<tr>
<td></td>
<td>15120</td>
<td>18466</td>
</tr>
<tr>
<td></td>
<td>15444</td>
<td>17784</td>
</tr>
<tr>
<td></td>
<td>15190</td>
<td>17644</td>
</tr>
<tr>
<td></td>
<td>12386</td>
<td>15996</td>
</tr>
<tr>
<td></td>
<td>12720</td>
<td>14546</td>
</tr>
<tr>
<td></td>
<td>17370</td>
<td>14328</td>
</tr>
<tr>
<td></td>
<td>13212</td>
<td>13200</td>
</tr>
<tr>
<td></td>
<td>6910</td>
<td>13095</td>
</tr>
<tr>
<td></td>
<td>11791</td>
<td>12372</td>
</tr>
</tbody>
</table>
Subsequent pages do not contain a heading:

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 1201.

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 1055.

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 1093.

**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 873.

`</n` specifies skipped lines. For details, see *Controlling the Vertical Positioning of a Heading or Footing* on page 1193.

`<+n` to position the next character on the line. For details, see *Using Spot Markers to Refine Positioning* on page 1188.

`<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 1056.

**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 STYLESHEET *
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

How to:

- Create a Page Heading
- Create a Page 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 1397 for details.

**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 1135.

**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 1055.

**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 1093.

**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 873.
- `</n` specifies skipped lines. For details, see Controlling the Vertical Positioning of a Heading or Footing on page 1193.
- `<n` to position the next character on the line. For details, see Using Spot Markers to Refine Positioning on page 1188.
- `<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 1056.

**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.)
Creating Headings and Footings

**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 [Including a Field Value in a Heading or Footing](#) on page 1093 for information on embedded field values. The first page of data applies to the MIS department.

**ACCOUNT REPORT FOR DEPARTMENT MIS**

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>BANK_ACCT</th>
<th>EMP_ID</th>
<th>CURR_SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>163800144</td>
<td>818692173</td>
<td>$27,062.00</td>
</tr>
<tr>
<td></td>
<td>122850108</td>
<td>326179357</td>
<td>$21,780.00</td>
</tr>
<tr>
<td></td>
<td>40950036</td>
<td>117593129</td>
<td>$18,480.00</td>
</tr>
<tr>
<td></td>
<td>112847612</td>
<td></td>
<td>$13,200.00</td>
</tr>
<tr>
<td></td>
<td>219984371</td>
<td></td>
<td>$18,480.00</td>
</tr>
<tr>
<td></td>
<td>543729165</td>
<td></td>
<td>$9,000.00</td>
</tr>
</tbody>
</table>

The second page of data applies to the PRODUCTION department.

**ACCOUNT REPORT FOR DEPARTMENT PRODUCTION**

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>BANK_ACCT</th>
<th>EMP_ID</th>
<th>CURR_SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTION</td>
<td>819000702</td>
<td>123764317</td>
<td>$26,862.00</td>
</tr>
<tr>
<td></td>
<td>136500120</td>
<td>451123478</td>
<td>$16,100.00</td>
</tr>
<tr>
<td></td>
<td>160633</td>
<td>119329144</td>
<td>$29,700.00</td>
</tr>
<tr>
<td></td>
<td>071382660</td>
<td></td>
<td>$11,000.00</td>
</tr>
<tr>
<td></td>
<td>119265415</td>
<td></td>
<td>$9,500.00</td>
</tr>
<tr>
<td></td>
<td>126724188</td>
<td></td>
<td>$21,120.00</td>
</tr>
</tbody>
</table>
**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 1135.

- **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 1198.

- **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 1055.

  - **variable**
    
    Can be any one of, or a combination of the following:
Creating Headings and Footings

**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 1093.

**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 873.

</n specifies skipped lines. For details, see *Controlling the Vertical Positioning of a Heading or Footing* on page 1193.

<-n positions the next character on the line. For details, see *Using Spot Markers to Refine Positioning* on page 1188.

<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 1056.

**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.

PRELIMINARY SALES FIGURES
FINAL TO COME END OF MONTH
Tip: If you do not see the navigation arrows, click the maximize button.

Freezing HTML Headings, Footings, and Column Titles

How to:
Create a Scrollable Area in an HTML Report

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.
How to Create a Scrollable Area in an HTML Report

Syntax:

```plaintext
TYPE=REPORT, HFREEZE={ON|TOP|BOTTOM|OFF}, [ SCROLLHEIGHT={4|nn[.n]}], $
```

where:

- **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.

- **nn[.n]**
  - Is the height, in inches, of the scrollable area. The default is 4 inches.

Reference: Usage Notes for Freezing Areas of HTML Report Output

- The request must include the setting ON TABLE SET HTMLCSS ON.
- The output format for the report must be HTML. If the format in the request is not HTML, the Stylesheet attributes to set aside a scrollable area for HTML format will be ignored.
- Accordion reports do not support column freezing.

Creating a Sort Heading or Footing

How to:

- Create a Sort Heading
- Create a Sort 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 825. For an illustration of this technique, see Creating a Conditional Sort Footing With Multiple Sort Options on page 1088.
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 261.

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. For an illustration, see *Omitting a Display Command in a Sort Footing* on page 1092. 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 1055.

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.
### 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>.

```plaintext
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.
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.)

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 1055.

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.

You can include images in a heading or footing.

For details, see Including an Element in a Heading or Footing on page 1093.

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.

<divides a heading or footing into items for formatting. For details, see Identifying a Heading, Footing, Title, or FML Free Text on page 873.

specifies skipped lines. For details, see Controlling the Vertical Positioning of a Heading or Footing on page 1193.

<n to position the next character on the line. For details, see Using Spot Markers to Refine Positioning on page 1188.

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 1056.

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 825. For related information, see Using Expressions on page 315.

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 STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```
The sort heading identifies the product that the next line of data applies to.

<table>
<thead>
<tr>
<th>Product Code</th>
<th>Package</th>
<th>Unit Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>G 110</td>
<td>Case</td>
<td>125.00</td>
</tr>
<tr>
<td>G121</td>
<td>Case</td>
<td>140.00</td>
</tr>
<tr>
<td>B142</td>
<td>Pounds</td>
<td>81.00</td>
</tr>
<tr>
<td>B141</td>
<td>Pounds</td>
<td>58.00</td>
</tr>
<tr>
<td>B144</td>
<td>Pounds</td>
<td>76.00</td>
</tr>
<tr>
<td>G104</td>
<td>Case</td>
<td>96.00</td>
</tr>
</tbody>
</table>

See *Including a Field Value in a Heading or Footing* on page 1093 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:

```plaintext
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 STYLESHEET *
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.

<table>
<thead>
<tr>
<th>Category</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>17231455</td>
</tr>
<tr>
<td>Food</td>
<td>17229333</td>
</tr>
<tr>
<td>Gifts ALERT: SALES FALL BELOW $17,000,000</td>
<td>11695502</td>
</tr>
</tbody>
</table>
See *Including a Field Value in a Heading or Footing* on page 1093, 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

```plaintext
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

```plaintext
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 1055.

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 1093.
**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 873.

</n specifies skipped lines. For details, see *Controlling the Vertical Positioning of a Heading or Footing* on page 1193.

<-n to position the next character on the line. For details, see *Using Spot Markers to Refine Positioning* on page 1188.

<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 1056.

**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 825. For related information, see *Using Expressions* on page 315.

**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 STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The sort footing identifies the product that the previous line of data applies to.

<table>
<thead>
<tr>
<th>Product Code</th>
<th>Package</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>G110</td>
<td>Case</td>
<td>125.00</td>
</tr>
<tr>
<td></td>
<td>Summary for Coffee Grinder</td>
<td></td>
</tr>
<tr>
<td>G121</td>
<td>Case</td>
<td>140.00</td>
</tr>
<tr>
<td></td>
<td>Summary for Coffee Pot</td>
<td></td>
</tr>
<tr>
<td>B142</td>
<td>Pounds</td>
<td>81.00</td>
</tr>
<tr>
<td></td>
<td>Summary for French Roast</td>
<td></td>
</tr>
<tr>
<td>B141</td>
<td>Pounds</td>
<td>58.00</td>
</tr>
<tr>
<td></td>
<td>Summary for Hazelnut</td>
<td></td>
</tr>
<tr>
<td>B144</td>
<td>Pounds</td>
<td>76.00</td>
</tr>
<tr>
<td></td>
<td>Summary for Kona</td>
<td></td>
</tr>
<tr>
<td>G104</td>
<td>Case</td>
<td>96.00</td>
</tr>
<tr>
<td></td>
<td>Summary for Thermos</td>
<td></td>
</tr>
</tbody>
</table>

See *Including a Field Value in a Heading or Footing* on page 1093 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**

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Date Of Order</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>94710</td>
<td>2001/01/02</td>
<td>$406,964.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+++ Order total is greater than or equal to $200,000 +++</td>
</tr>
<tr>
<td>94680</td>
<td>2001/01/02</td>
<td>$421,916.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+++ Order total is greater than or equal to $200,000 +++</td>
</tr>
<tr>
<td>94670</td>
<td>2001/01/02</td>
<td>$513,868.76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+++ Order total is greater than or equal to $200,000 +++</td>
</tr>
<tr>
<td>94550</td>
<td>2001/01/02</td>
<td>$496,323.64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+++ Order total is greater than or equal to $200,000 +++</td>
</tr>
<tr>
<td>94530</td>
<td>2001/01/02</td>
<td>$3,472.41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--- Order total is less than $200,000 ---</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Product</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazelnut</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td>117</td>
</tr>
</tbody>
</table>

Hazelnut has multiple orders.

<table>
<thead>
<tr>
<th>Product</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscotti</td>
<td>399</td>
</tr>
<tr>
<td>Croissant</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>433</td>
</tr>
</tbody>
</table>

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.

```plaintext
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).

<table>
<thead>
<tr>
<th>Region</th>
<th>Dollars</th>
<th>Sterling</th>
</tr>
</thead>
<tbody>
<tr>
<td>CENTRAL AMERICA</td>
<td>727,810</td>
<td>443,964</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Balance of investments for GUATEMALA in Euros is 749,644.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>39,472,857</td>
<td>24,078,443</td>
</tr>
<tr>
<td>Balance of investments for HONDURAS in Euros is 40,657,043.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAR EAST</td>
<td>88,068,897</td>
<td>53,722,027</td>
</tr>
<tr>
<td>Balance of investments for HONG KONG in Euros is 90,710,964.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>114,591,945</td>
<td>69,901,086</td>
</tr>
<tr>
<td>Balance of investments for JAPAN in Euros is 118,029,703.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WESTERN EUROPE</td>
<td>125,757,198</td>
<td>76,711,891</td>
</tr>
<tr>
<td>Balance of investments for ENGLAND in Euros is 129,529,914.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>84,793,176</td>
<td>51,723,837</td>
</tr>
<tr>
<td>Balance of investments for FRANCE in Euros is 87,336,971.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 1093 and Including a Text Field in a Heading or Footing on page 1101.
- A page number. See Including a Page Number in a Heading or Footing on page 1103.
- A Dialogue Manager variable. See Including a Dialogue Manager Variable in a Heading or Footing on page 1104.
- An image. See Including an Image in a Heading or Footing on page 1106.

Including a Field Value in a Heading or Footing

How to:
Include 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 1397.

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. 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. For an example, see *Aligning a Field Value Across Multiple Columns* on page 1165.

- 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.
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.

<table>
<thead>
<tr>
<th>BANK</th>
<th>ACCT</th>
<th>LAST NAME</th>
<th>FIRST NAME</th>
<th>CURR SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
<td>$9,000.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td>$18,480.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td>$13,200.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40950036 JONES</td>
<td>DIANE</td>
<td>$18,480.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>122850108 BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>$21,780.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16380144 CROSS</td>
<td>BARBARA</td>
<td>$27,062.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MIS EMPLOYEES WITH ELECTRONIC TRANSFER ACCOUNTS
On the second page of output, the value of DEPARTMENT is PRODUCTION.

PRODUCTION: BANK, EMPLOYEES AND SALARIES

<table>
<thead>
<tr>
<th>BANK_ACCT LAST_NAME</th>
<th>FIRST_NAME</th>
<th>CURR SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>$21,120.00</td>
</tr>
<tr>
<td>SMITH</td>
<td>RICHARD</td>
<td>$8,500.00</td>
</tr>
<tr>
<td>STEVENS</td>
<td>ALFRED</td>
<td>$11,000.00</td>
</tr>
<tr>
<td>160633 BANNING</td>
<td>JOHN</td>
<td>$29,700.00</td>
</tr>
<tr>
<td>136500120 MCKNIGHT</td>
<td>ROGER</td>
<td>$16,100.00</td>
</tr>
<tr>
<td>319000702 IRVING</td>
<td>JOAN</td>
<td>$26,862.00</td>
</tr>
</tbody>
</table>

PRODUCTION EMPLOYEES WITH ELECTRONIC TRANSFER ACCOUNTS

Note: If you do not see the navigation arrows, click the maximize button.
**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
UNIT S0LD  645
RETURNS  58
TOTAL SOLD  587.00
```

<table>
<thead>
<tr>
<th>PRODUCT_CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT_SOLD</td>
</tr>
<tr>
<td>RETURNS</td>
</tr>
<tr>
<td>ACTUAL_SALES</td>
</tr>
<tr>
<td>B10</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>B12</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>B17</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>B20</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>C13</td>
</tr>
<tr>
<td>C17</td>
</tr>
<tr>
<td>C7</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>D12</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>E1</td>
</tr>
<tr>
<td>E2</td>
</tr>
<tr>
<td>E3</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

**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 STYLESHEET *
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 STYLESHEET *
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

**How to:**

Include a Text Field in a Heading or Footing

**Reference:**

Limits for Text Fields 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 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. You can use the graphical development tools of Developer Studio to create this example. For details, see your Developer Studio documentation.

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
"<TXTFLD"
ON TABLE SET PAGE-NUM OFF
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF,$
ENDSTYLE
END
```

5. Run the report request.

The output is:

```
CATALOG
COURSE 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. For illustrations, see Aligning a Text Field With a Column in a Sort Footing on page 1159 and Aligning and Styling a Text Field in a Sort Footing on page 1160.

**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 929.
Including a Dialogue Manager Variable in a Heading or Footing

**How to:**

Include 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

```c
&[&]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.

```c
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=TABHEADING, LINE=1, FONT='TIMES', SIZE=10, STYLE=BOLD, $
TYPE=TABHEADING, LINE=2, COLOR=BLUE, $
ENDSTYLE
END
```
Including an Element in a Heading or Footing

The output is:

Sales Report for Store Code R1019
06/07/02

<table>
<thead>
<tr>
<th>Store ID</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1019</td>
<td>18560</td>
<td>25680</td>
</tr>
<tr>
<td></td>
<td>21132</td>
<td>24010</td>
</tr>
<tr>
<td></td>
<td>15880</td>
<td>22919</td>
</tr>
<tr>
<td></td>
<td>24450</td>
<td>22605</td>
</tr>
<tr>
<td></td>
<td>14556</td>
<td>18942</td>
</tr>
<tr>
<td></td>
<td>13860</td>
<td>18915</td>
</tr>
<tr>
<td></td>
<td>22806</td>
<td>18837</td>
</tr>
<tr>
<td></td>
<td>19248</td>
<td>18535</td>
</tr>
<tr>
<td></td>
<td>15120</td>
<td>18466</td>
</tr>
<tr>
<td></td>
<td>15444</td>
<td>17784</td>
</tr>
</tbody>
</table>

Note: You can modify the format of the date. Some formats are:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Display Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;DATEtrMMDYY</td>
<td>2002, December 11</td>
</tr>
<tr>
<td>&amp;DATEMDYY</td>
<td>12/11/2002</td>
</tr>
<tr>
<td>&amp;DATEtrMDYY</td>
<td>December 11, 2002</td>
</tr>
<tr>
<td>&amp;DATEQYY</td>
<td>Q4 2002</td>
</tr>
</tbody>
</table>

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 929.
Customizing a Column Title

In this section:
Customizing a Column Title in a Master File
Distinguishing Between Duplicate Field Names

How to:
Customize a Column Title in a Request

Reference:
Limits for Column Titles

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:

```plaintext
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 STYLESHEET *
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:

<table>
<thead>
<tr>
<th>DEPT</th>
<th>LASTNAME</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNTING</td>
<td>ANDERSON</td>
<td>$32,400.00</td>
</tr>
<tr>
<td></td>
<td>LOPEZ</td>
<td>$26,400.00</td>
</tr>
<tr>
<td></td>
<td>SANCHEZ</td>
<td>$83,000.00</td>
</tr>
<tr>
<td></td>
<td>SOPENA</td>
<td>$79,000.00</td>
</tr>
<tr>
<td></td>
<td>WANG</td>
<td>$62,500.00</td>
</tr>
<tr>
<td>CONSULTING</td>
<td>ELLNER</td>
<td>$35,900.00</td>
</tr>
<tr>
<td></td>
<td>WANG</td>
<td>$49,500.00</td>
</tr>
<tr>
<td></td>
<td>WHITE</td>
<td>$40,900.00</td>
</tr>
<tr>
<td>SALES</td>
<td>ADDAMS</td>
<td>$54,100.00</td>
</tr>
<tr>
<td></td>
<td>CASSANOVA</td>
<td>$70,000.00</td>
</tr>
<tr>
<td></td>
<td>CHISOLM</td>
<td>$43,000.00</td>
</tr>
<tr>
<td></td>
<td>DUBOIS</td>
<td>$43,600.00</td>
</tr>
<tr>
<td></td>
<td>LASTRA</td>
<td>$115,000.00</td>
</tr>
<tr>
<td></td>
<td>MEDINA</td>
<td>$39,000.00</td>
</tr>
<tr>
<td></td>
<td>OLSON</td>
<td>$30,500.00</td>
</tr>
</tbody>
</table>

**Reference:** Limits for Column Titles

- The width allotted for column titles has no limit other than the memory available.
- A column title can contain up to five 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 193.

**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.

**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).

```c
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"
END
```
**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 STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```
The output is:

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th>DEPARTMENT</th>
<th>NUMBER</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARY SMITH</td>
<td>MIS</td>
<td>112847612</td>
<td></td>
</tr>
<tr>
<td>DIANE JONES</td>
<td></td>
<td>117593129</td>
<td></td>
</tr>
<tr>
<td>JOHN MCCOY</td>
<td></td>
<td>219984371</td>
<td></td>
</tr>
<tr>
<td>ROSEMARIE BLACKWOOD</td>
<td>326179357</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARY GREENSPAN</td>
<td></td>
<td>543729165</td>
<td></td>
</tr>
<tr>
<td>BARBARA CROSS</td>
<td></td>
<td>818692173</td>
<td></td>
</tr>
<tr>
<td>ALFRED STEVENS</td>
<td>PRODUCTION</td>
<td>071382660</td>
<td></td>
</tr>
<tr>
<td>RICHARD SMITH</td>
<td></td>
<td>119265415</td>
<td></td>
</tr>
<tr>
<td>JOHN BANNING</td>
<td></td>
<td>119329144</td>
<td></td>
</tr>
<tr>
<td>JOAN IRVING</td>
<td></td>
<td>123764317</td>
<td></td>
</tr>
<tr>
<td>ANTHONY ROMANS</td>
<td></td>
<td>126724188</td>
<td></td>
</tr>
<tr>
<td>ROGER MCKNIGHT</td>
<td></td>
<td>451123478</td>
<td></td>
</tr>
</tbody>
</table>

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 STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```
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. For an illustration, see Justifying a Report Heading on page 1132.

See the Describing Data With WebFOCUS Language 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.

<table>
<thead>
<tr>
<th>PROD</th>
<th>CODE</th>
<th>UNIT</th>
<th>SOLD</th>
<th>RETAIL</th>
<th>PRICE</th>
<th>GENERATED</th>
<th>REVENUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>B10</td>
<td>30</td>
<td></td>
<td></td>
<td>$0.85</td>
<td></td>
<td></td>
<td>$25.50</td>
</tr>
<tr>
<td>B17</td>
<td>20</td>
<td></td>
<td></td>
<td>$1.89</td>
<td></td>
<td></td>
<td>$37.80</td>
</tr>
<tr>
<td>B20</td>
<td>15</td>
<td></td>
<td></td>
<td>$1.99</td>
<td></td>
<td></td>
<td>$29.85</td>
</tr>
<tr>
<td>C17</td>
<td>12</td>
<td></td>
<td></td>
<td>$2.09</td>
<td></td>
<td></td>
<td>$25.08</td>
</tr>
<tr>
<td>D12</td>
<td>20</td>
<td></td>
<td></td>
<td>$2.09</td>
<td></td>
<td></td>
<td>$41.80</td>
</tr>
<tr>
<td>E1</td>
<td>30</td>
<td></td>
<td></td>
<td>$0.89</td>
<td></td>
<td></td>
<td>$26.70</td>
</tr>
<tr>
<td>E3</td>
<td>35</td>
<td></td>
<td></td>
<td>$1.09</td>
<td></td>
<td></td>
<td>$38.15</td>
</tr>
</tbody>
</table>
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.

Creating Labels to Identify Data

In this section:

- Creating a Label for a Row or Column Total
- Creating a Label for a Subtotal and a Grand Total
- Creating a Label for a Row in a Financial Report

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 1113.
- Subtotals for sort groups. See Creating a Label for a Subtotal and a Grand Total on page 1116.

Creating a Label for a Row or Column Total

How to:

Create 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 261.
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 1143.

`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:

<table>
<thead>
<tr>
<th>State</th>
<th>Male Population</th>
<th>Female Population</th>
<th>Total Population by State</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT</td>
<td>426357</td>
<td>438836</td>
<td>865,193</td>
</tr>
<tr>
<td>WY</td>
<td>245023</td>
<td>245897</td>
<td>490,920</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Population by Gender</td>
<td>671380</td>
<td>684733</td>
<td>1,356,113</td>
</tr>
</tbody>
</table>

**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 STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```
Creating Labels to Identify Data

Creating a Label for a Subtotal and a Grand Total

How to:
Create a Label for a Subtotal or a Grand Total
Create a Label for the Subtotal of a Calculated Value

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 261.

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. MULTILINES 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.

The output is:

<table>
<thead>
<tr>
<th></th>
<th>MT</th>
<th>WY</th>
<th></th>
<th>MT</th>
<th>WY</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Male Population</td>
<td>426,357</td>
<td>438,836</td>
<td>Female Population</td>
<td>245,023</td>
<td>245,897</td>
<td>Male Population</td>
</tr>
</tbody>
</table>

Total by Gender 

Including Totals and Subtotals
WHEN expression

Specifies a conditional subtotal as determined by a logical expression. For details, see Using Expressions on page 315.

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.

```sql
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.

<table>
<thead>
<tr>
<th>DED_CODE</th>
<th>DEPARTMENT</th>
<th>BANK_ACCT</th>
<th>DED_AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CITY</td>
<td>MIS</td>
<td>40950036</td>
<td>$14.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>122850108</td>
<td>$31.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>163800144</td>
<td>$82.70</td>
</tr>
</tbody>
</table>

Total City Deduction for MIS $128.45

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>DED_AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTION</td>
<td>$7.42</td>
</tr>
</tbody>
</table>

Total City Deduction for PRODUCTION $85.87

**GRAND TOTAL** $214.32
Creating Labels to Identify Data

**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 STYLESHEET *
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```

The output is:

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Product Number</th>
<th>Quantity</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>28003</td>
<td>1006</td>
<td>1</td>
<td>$501.60</td>
</tr>
<tr>
<td></td>
<td>1008</td>
<td>1</td>
<td>$239.18</td>
</tr>
<tr>
<td></td>
<td>1020</td>
<td>1</td>
<td>$324.06</td>
</tr>
<tr>
<td></td>
<td>1032</td>
<td>1</td>
<td>$103.23</td>
</tr>
<tr>
<td></td>
<td>1034</td>
<td>1</td>
<td>$520.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Total</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Order Total 28003</strong></td>
</tr>
<tr>
<td>28004</td>
<td>1006</td>
<td>1</td>
<td>$471.83</td>
</tr>
<tr>
<td></td>
<td>1016</td>
<td>1</td>
<td>$282.19</td>
</tr>
<tr>
<td></td>
<td>1018</td>
<td>1</td>
<td>$462.96</td>
</tr>
<tr>
<td></td>
<td>1022</td>
<td>1</td>
<td>$401.30</td>
</tr>
<tr>
<td></td>
<td>1028</td>
<td>1</td>
<td>$110.34</td>
</tr>
<tr>
<td></td>
<td>1036</td>
<td>1</td>
<td>$295.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Total</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Order Total 28004</strong></td>
</tr>
</tbody>
</table>

**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. MULTILINES 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 315.

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.

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>BANK ACCT</th>
<th>GROSS DED AMT</th>
<th>DG RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>40950036</td>
<td>$6,099.50</td>
<td>$2,866.18</td>
</tr>
<tr>
<td></td>
<td>122850108</td>
<td>$9,075.00</td>
<td>$6,307.00</td>
</tr>
<tr>
<td></td>
<td>163800144</td>
<td>$22,013.75</td>
<td>$15,377.40</td>
</tr>
<tr>
<td>SUBTOTAL FOR MIS</td>
<td></td>
<td>$37,188.25</td>
<td>$24,550.58</td>
</tr>
<tr>
<td>PRODUCTION</td>
<td>160633</td>
<td>$2,475.00</td>
<td>$1,427.24</td>
</tr>
<tr>
<td></td>
<td>136500120</td>
<td>$9,130.00</td>
<td>$3,593.92</td>
</tr>
<tr>
<td></td>
<td>819000702</td>
<td>$17,094.00</td>
<td>$11,949.44</td>
</tr>
<tr>
<td>SUBTOTAL FOR PRODUCTION</td>
<td>$28,699.00</td>
<td>$16,970.60</td>
<td>.59</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>$65,887.25</td>
<td>$41,521.18</td>
</tr>
</tbody>
</table>

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 1307.

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. See Applying Font Attributes to a Heading, Footing, Title, or Label on page 1122.

- Add borders around headings and footings and grid lines around headings, footings, and column titles. See Laying Out the Report Page on page 929.

  You can also add space between heading or footing content and grid lines. See Controlling the Vertical Positioning of a Heading or Footing on page 1193.

- Left-justify, right-justify, or center a heading, footing, or individual lines in a multi-line heading or footing, column titles, and subtotals. See Justifying a Heading, Footing, Title, or Label on page 1130. See also How to Center a Page Heading or Footing Using Legacy Formatting on page 1135 and How to Justify a Column Title for a Display or BY Field Using Legacy Formatting on page 1142.

  You can also justify a label for a row or column total. See Justifying a Label for a Row or Column Total on page 1143.

- 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. See Aligning a Heading or Footing Element in an HTML Report on page 1149.

  - 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. See Aligning Content in a Multi-Line Heading or Footing on page 1166.

  - 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. See Positioning Headings, Footings, or Items Within Them on page 1180.

  For assistance in determining which of these approaches best suits your needs, see Choosing an Alignment Method for Heading and Footing Elements on page 1147.

- Control the vertical positioning of a heading or footing. See Controlling the Vertical Positioning of a Heading or Footing on page 1193.

- 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 1201.
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 849. For details on font options, including size, color, and style, see *Formatting Report Data* on page 1205.

**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 1205.

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
```
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 1205 and Controlling Report Formatting on page 825.

This request generates an internal Cascading Style Sheet and specifies font characteristics for the report heading.

```sql
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
```
Applying Font Attributes to a Heading, Footing, Title, or Label

The output is:

Sales Report

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>

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 &lt;+0&gt; 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 &lt;+0&gt;&amp;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:

**Sales Report for Store Code R1019 06/13/02**

<table>
<thead>
<tr>
<th>Store ID</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1019</td>
<td>18560</td>
<td>25680</td>
</tr>
<tr>
<td></td>
<td>21132</td>
<td>24010</td>
</tr>
<tr>
<td></td>
<td>15880</td>
<td>22919</td>
</tr>
<tr>
<td></td>
<td>24450</td>
<td>22605</td>
</tr>
<tr>
<td></td>
<td>14556</td>
<td>18942</td>
</tr>
<tr>
<td></td>
<td>13860</td>
<td>18915</td>
</tr>
<tr>
<td></td>
<td>22806</td>
<td>18837</td>
</tr>
<tr>
<td></td>
<td>19248</td>
<td>18535</td>
</tr>
<tr>
<td></td>
<td>15120</td>
<td>18466</td>
</tr>
<tr>
<td></td>
<td>15444</td>
<td>17784</td>
</tr>
<tr>
<td></td>
<td>15190</td>
<td>17644</td>
</tr>
<tr>
<td></td>
<td>12386</td>
<td>15996</td>
</tr>
<tr>
<td></td>
<td>12720</td>
<td>14546</td>
</tr>
<tr>
<td></td>
<td>17370</td>
<td>14328</td>
</tr>
</tbody>
</table>
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 929.

**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.

```plaintext
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:

<table>
<thead>
<tr>
<th>Category</th>
<th>Budget Units</th>
<th>Unit Sales</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1385923</td>
<td>1376266</td>
<td>17293886</td>
<td>17231455</td>
</tr>
<tr>
<td>Food</td>
<td>1377564</td>
<td>1384845</td>
<td>17267160</td>
<td>17229333</td>
</tr>
<tr>
<td>Gifts</td>
<td>931007</td>
<td>927880</td>
<td>11659732</td>
<td>11695502</td>
</tr>
</tbody>
</table>
**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

<table>
<thead>
<tr>
<th>Category</th>
<th>Budget Units</th>
<th>Unit Sales</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1385923</td>
<td>1376266</td>
<td>17293886</td>
<td>17231455</td>
</tr>
<tr>
<td>Food</td>
<td>1377564</td>
<td>1384845</td>
<td>17267160</td>
<td>17229333</td>
</tr>
<tr>
<td>Gifts</td>
<td>931007</td>
<td>927880</td>
<td>11659732</td>
<td>11695502</td>
</tr>
</tbody>
</table>
```
**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.

```plaintext
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=TABLE, STYLE=BOLD, JUSTIFY=CENTER, BORDER=HEAVY, $
  BORDER-COLOR=RED, BORDER-STYLE=DOTTED, $
ENDSTYLE
END
```

The output is:

```
Sales Report
**CONFIDENTIAL**
December 2002
```

<table>
<thead>
<tr>
<th>Category</th>
<th>Budget Units</th>
<th>Unit Sales</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1385923</td>
<td>1376266</td>
<td>17293886</td>
<td>17231455</td>
</tr>
<tr>
<td>Food</td>
<td>1377564</td>
<td>1384845</td>
<td>17267160</td>
<td>17229333</td>
</tr>
<tr>
<td>Gifts</td>
<td>931007</td>
<td>927880</td>
<td>11659732</td>
<td>11695502</td>
</tr>
</tbody>
</table>
```

**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.

```plaintext
TABLE FILE GGSALES
SUM BUDDUNITS 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=TABHEADING, 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

<table>
<thead>
<tr>
<th>Category</th>
<th>Budget Units</th>
<th>Unit Sales</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1385923</td>
<td>1376266</td>
<td>17293886</td>
<td>17231455</td>
</tr>
<tr>
<td>Food</td>
<td>1377564</td>
<td>1384845</td>
<td>17267160</td>
<td>17229333</td>
</tr>
<tr>
<td>Gifts</td>
<td>931007</td>
<td>927880</td>
<td>11659732</td>
<td>11695502</td>
</tr>
</tbody>
</table>
```

**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 1130.
- A column title. See Justifying a Column Title on page 1138.
- A label for a row or column total. See Justifying a Label for a Row or Column Total on page 1143.
- A label for a subtotal or grand total. See Justifying a Label for a Subtotal or Grand Total on page 1145.

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 1147.

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 1147.

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 1135.
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 STYLESHEET *
TYPE = REPORT, GRID=OFF, $
TYPE = REPORT, COLUMN = VENDOR_NAME, JUSTIFY = CENTER, $
TYPE = TABHEADING, JUSTIFY = CENTER, $
ENDSTYLE
END
```

The output is:

```
PRODUCT REPORT

<table>
<thead>
<tr>
<th>Product</th>
<th>Code</th>
<th>Vendor Name</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscotti</td>
<td>F102</td>
<td>Delancey Bakeries</td>
<td>17.00</td>
</tr>
<tr>
<td>Coffee Grinder</td>
<td>G110</td>
<td>Appliance Craft</td>
<td>125.00</td>
</tr>
<tr>
<td>Coffee Pot</td>
<td>G121</td>
<td>Appliance Craft</td>
<td>140.00</td>
</tr>
<tr>
<td>Croissant</td>
<td>F103</td>
<td>West Side Bakers</td>
<td>28.00</td>
</tr>
<tr>
<td>French Roast</td>
<td>B142</td>
<td>European Specialities,</td>
<td>81.00</td>
</tr>
<tr>
<td>Hazelnut</td>
<td>B141</td>
<td>Coffee Connection</td>
<td>58.00</td>
</tr>
<tr>
<td>Kona</td>
<td>B144</td>
<td>Evelina Imports, Ltd</td>
<td>76.00</td>
</tr>
<tr>
<td>Mug</td>
<td>G100</td>
<td>NY Ceramic Supply</td>
<td>26.00</td>
</tr>
<tr>
<td>Scone</td>
<td>F101</td>
<td>Ridgewood Bakeries</td>
<td>13.00</td>
</tr>
<tr>
<td>Thermos</td>
<td>G104</td>
<td>ThermoTech, Inc</td>
<td>96.00</td>
</tr>
</tbody>
</table>
```

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 1138.
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 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, 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

<table>
<thead>
<tr>
<th>Category</th>
<th>Budget Units</th>
<th>Unit Sales</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1385923</td>
<td>1376266</td>
<td>17293886</td>
<td>17231455</td>
</tr>
<tr>
<td>Food</td>
<td>1377564</td>
<td>1384845</td>
<td>17267160</td>
<td>17229333</td>
</tr>
<tr>
<td>Gifts</td>
<td>931007</td>
<td>927880</td>
<td>11659732</td>
<td>11695502</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Budget Units</th>
<th>Unit Sales</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1385923</td>
<td>1376266</td>
<td>17293886</td>
<td>17231455</td>
</tr>
<tr>
<td>Food</td>
<td>1377564</td>
<td>1384845</td>
<td>17267160</td>
<td>17229333</td>
</tr>
<tr>
<td>Gifts</td>
<td>931007</td>
<td>927880</td>
<td>11659732</td>
<td>11695502</td>
</tr>
</tbody>
</table>
```

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 default), 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**

```plaintext
{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.
Justifying a Heading, Footing, Title, or Label

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 1055.

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 TAPBAGENO). 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 1093.

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 873.

</n specifies skipped lines. For details, see Controlling the Vertical Positioning of a Heading or Footing on page 1193.

<-n to position the next character on the line. For details, see Using Spot Markers to Refine Positioning on page 1188.

<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 1056.

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
Justifying a Heading, Footing, Title, or Label

The page heading is centered over the report data, as shown in the first page of output.

![ACCOUNT REPORT FOR DEPARTMENT MIS](image)

DEPARTMENT BANK ACCT EMP_ID CURR_SAL
MIS 163300144 812862173 $27,062.00
122850108 326179357 $21,780.00
40850036 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

**How to:**

Justify a Column Title Using a StyleSheet

Justify a Column Title for a Display or BY Field Using Legacy Formatting

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 1107.
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 1142.

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

\[
\begin{align*}
\text{TYPE=} & \text{TITLE, [COLUMN=} & \text{column,]} & \text{JUSTIFY=} & \text{option, } & \$
\text{TYPE=} & \text{ACROSSTITLE, [ACROSS=} & \text{column,]} & \text{JUSTIFY=} & \text{option, } & \$
\text{TYPE=} & \text{ACROSSVALUE, [COLUMN=} & \text{column,]} & \text{JUSTIFY=} & \text{option, } & \$
\end{align*}
\]

To justify a horizontal sort column title (generated by ACROSS), the StyleSheet syntax is

\[
\text{TYPE=} & \text{ACROSSTITLE, [ACROSS=} & \text{column,]} & \text{JUSTIFY=} & \text{option, } & \$
\]

To justify an ACROSS value or a ROW-TOTAL column title in an HTML report, use

\[
\text{TYPE=} & \text{ACROSSVALUE, [COLUMN=} & \text{column,]} & \text{JUSTIFY=} & \text{option, } & \$
\]

where:

\begin{itemize}
\item **TITLE**
  Specifies a vertical sort (BY) title or a display field title.
\item **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 849.
\item **ACROSSTITLE**
  Specifies a horizontal sort (ACROSS) title.
\item **ACROSSVALUE**
  Specifies a horizontal sort (ACROSS) value or a ROW-TOTAL column title.
\end{itemize}
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.

**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.

```plaintext
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 STYLESHEET *
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:

<table>
<thead>
<tr>
<th>Contact</th>
<th>Store Name</th>
<th>St</th>
</tr>
</thead>
<tbody>
<tr>
<td>JEFF DARFELL</td>
<td>GOTHAM GRINDS #1244</td>
<td>CA</td>
</tr>
<tr>
<td>PAT SMILEY</td>
<td>GOTHAM GRINDS #1040</td>
<td>CA</td>
</tr>
</tbody>
</table>
**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.

```plaintext
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 STYLESHEET *
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:

<table>
<thead>
<tr>
<th>State</th>
<th>Total by Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT</td>
<td>WY</td>
</tr>
<tr>
<td>Male Population</td>
<td>Female Population</td>
</tr>
</tbody>
</table>

```plaintext
<table>
<thead>
<tr>
<th>Male Population</th>
<th>Female Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>426357</td>
<td>438836</td>
</tr>
<tr>
<td>245023</td>
<td>245897</td>
</tr>
<tr>
<td>671380</td>
<td>684733</td>
</tr>
</tbody>
</table>
```

**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.

```plaintext
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 STYLESHEET *
TYPE=REPORT, GRID=OFF, $
  TYPE=TITLE, COLUMN=REV, STYLE=BOLD, JUSTIFY=LEFT, $
ENDSTYLE
END
```
The output is:

<table>
<thead>
<tr>
<th>PROD</th>
<th>CODE</th>
<th>UNIT</th>
<th>SOLD</th>
<th>RETAIL</th>
<th>PRICE</th>
<th>REV</th>
</tr>
</thead>
<tbody>
<tr>
<td>B10</td>
<td></td>
<td>30</td>
<td></td>
<td>$ .85</td>
<td>$25.50</td>
<td></td>
</tr>
<tr>
<td>B17</td>
<td></td>
<td>20</td>
<td></td>
<td>$1.89</td>
<td>$37.80</td>
<td></td>
</tr>
<tr>
<td>B20</td>
<td></td>
<td>15</td>
<td></td>
<td>$1.99</td>
<td>$29.85</td>
<td></td>
</tr>
<tr>
<td>C17</td>
<td></td>
<td>12</td>
<td></td>
<td>$2.09</td>
<td>$25.08</td>
<td></td>
</tr>
<tr>
<td>D12</td>
<td></td>
<td>20</td>
<td></td>
<td>$2.09</td>
<td>$41.80</td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td></td>
<td>30</td>
<td></td>
<td>$ .89</td>
<td>$26.70</td>
<td></td>
</tr>
<tr>
<td>E3</td>
<td></td>
<td>35</td>
<td></td>
<td>$1.09</td>
<td>$38.15</td>
<td></td>
</tr>
</tbody>
</table>

**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 1139.

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 STYLESHEET *
TYPE = REPORT, GRID=OFF, $
ENDSTYLE
END

The output is:

<table>
<thead>
<tr>
<th>Contact</th>
<th>Store Name</th>
<th>St</th>
</tr>
</thead>
<tbody>
<tr>
<td>JEFF DARFELL GOTHAM GRINDS #1244 CA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAT SMILEY   GOTHAM GRINDS #1040 CA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

How to:

Justify a Label for a Row or Column Total Using Legacy Formatting

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 1113.
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
```
Justifying a Label for a Subtotal or Grand Total

Although you cannot directly justify a customized label for a subtotal, 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 849.

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 STYLESHEET *
TYPE=REPORT, GRID=OFF, $
TYPE=SUBTOTAL, STYLE=BOLD, JUSTIFY=RIGHT, $
TYPE=GRANDTOTAL, STYLE=BOLD, JUSTIFY=RIGHT, $
ENDSTYLE
END
```
The output is:

<table>
<thead>
<tr>
<th>DED CODE</th>
<th>DEPARTMENT</th>
<th>BANK</th>
<th>ACCT</th>
<th>DED AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CITY</td>
<td>MIS</td>
<td>40950036</td>
<td>$14.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>122850108</td>
<td>$31.75</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>163800144</td>
<td>$82.70</td>
<td></td>
</tr>
</tbody>
</table>

**Total City Deduction for MIS**  $128.45

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>160633</th>
<th>136500120</th>
<th>819000702</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTION</td>
<td>$7.42</td>
<td>$18.25</td>
<td>$60.20</td>
</tr>
</tbody>
</table>

**Total City Deduction for PRODUCTION**  $85.87

**TOTAL**  $214.32
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.

<table>
<thead>
<tr>
<th>Alignment Method</th>
<th>Applies to ...</th>
<th>When to use...</th>
<th>Related Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1) StyleSheet Attributes:</strong></td>
<td>HTML EXL2K</td>
<td><strong>To align heading or footing items in HTML and EXL2K reports:</strong> 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. <strong>To specify a heading or footing item that spans multiple columns:</strong> You can combine HEADALIGN syntax with the COLSPAN attribute to achieve this result. For details, see Aligning a Heading or Footing Element Across Columns in an HTML Report on page 1162.</td>
<td><strong>Details:</strong> See <a href="#">Aligning a Heading or Footing Element in an HTML Report</a> on page 1149.</td>
</tr>
</tbody>
</table>
### Alignment Method

<table>
<thead>
<tr>
<th>Alignment Method</th>
<th>Applies to ...</th>
<th>When to use...</th>
<th>Related Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>2) <strong>StyleSheet Attributes:</strong></td>
<td>HTML</td>
<td><strong>For portability between HTML and PDF:</strong> 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.</td>
<td>For an HTML or EXL2K report, you can align specific items with HEADALIGN options.</td>
</tr>
<tr>
<td></td>
<td>PDF</td>
<td><strong>To align heading or footing items:</strong> 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.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PS</td>
<td><strong>To align decimal points in a multi-line heading or footing:</strong> 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.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Alignment Method

<table>
<thead>
<tr>
<th>3) StyleSheet Attribute: POSITION</th>
<th>Applies to ...</th>
<th>When to use...</th>
<th>Related Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details: See Positioning Headings, Footings, or Items Within Them on page 1180 and Laying Out the Report Page on page 929.</td>
<td>PDF PS HTML (limited)</td>
<td><strong>To set starting positions for headings or footings, or items within them:</strong> Use POSITION syntax to specify absolute and relative starting positions. 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. <strong>To align heading and footing items with columns:</strong> Use POSITION syntax to align a heading item with a column position. For example, the syntax TYPE=SUBHEAD, LINE=1, ITEM=3,POSITION=SALES, $ places ITEM 3 of the sort heading at the horizontal position where the column SALES is.</td>
<td>For a PDF report, you can accomplish most positioning with WIDTH and JUSTIFY. For an HTML report, you can align a heading item with a column by setting the HEADALIGN attribute to BODY.</td>
</tr>
</tbody>
</table>

---

**Aligning a Heading or Footing Element in an HTML Report**

#### How to:

Align a Heading or Footing Element in an HTML 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.

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 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. 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.

To compare sample output, see Comparing Output Generated With HEADALIGN Options on page 1151.

To break a text string into multiple parts for manipulation across columns in an HTML table, you can use <+0> spot markers in the request. For details, see Identifying a Report Component in a WebFOCUS StyleSheet on page 849.

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 in an HTML Report on page 1149.

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.

HEADALIGN is not supported for PDF reports.

**Tip:** For a summary of other alignment methods, see Choosing an Alignment Method for Heading and Footing Elements on page 1147.

**Syntax:**

```
How to Align a Heading or Footing Element in an HTML 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 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. 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.

- **BODY** which aligns heading items with data columns by placing 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 related information, see *Comparing Output Generated With HEADALIGN Options* on page 1151. You can combine HEADALIGN options with the COLSPAN attribute to allow heading items to span multiple HTML table columns. For details, see *Aligning a Heading or Footing Element in an HTML Report* on page 1149.

For related information, see *Comparing Output Generated With HEADALIGN Options* on page 1151. You can combine HEADALIGN options with the COLSPAN attribute to allow heading items to span multiple HTML table columns. For details, see *Aligning a Heading or Footing Element in an HTML Report* on page 1149.

**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 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.

```
```
OUNTRY  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, $  

<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>CAR</th>
<th>MODEL</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is my subhead

Country is: ENGLAND  Car is: JAGUAR  
Model is: V12XKE AUTO

<table>
<thead>
<tr>
<th>ENGLAND</th>
<th>JAGUAR</th>
<th>V12XKE AUTO</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>XJ12L AUTO</td>
<td>12000</td>
</tr>
<tr>
<td>JENSEN</td>
<td>INTERCEPTOR III</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TRIUMPH</td>
<td>TR7</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

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, $**

<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>CAR</th>
<th>MODEL</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is my subhead

Country is: ENGLAND  Car is: JAGUAR  
Model is: V12XKE AUTO

<table>
<thead>
<tr>
<th>ENGLAND</th>
<th>JAGUAR</th>
<th>V12XKE AUTO</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>XJ12L AUTO</td>
<td>12000</td>
</tr>
<tr>
<td>JENSEN</td>
<td>INTERCEPTOR III</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TRIUMPH</td>
<td>TR7</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>CAR</th>
<th>MODEL</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGLAND</td>
<td>JAGUAR</td>
<td>V12XKE AUTO</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>XJ12L AUTO</td>
<td>12000</td>
</tr>
<tr>
<td>JENSEN</td>
<td>INTERCEPTOR III</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TRIUMPH</td>
<td>TR7</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>CAR</th>
<th>MODEL</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGLAND</td>
<td>JAGUAR</td>
<td>V12XKE AUTO</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>XJ12L AUTO</td>
<td>12000</td>
</tr>
<tr>
<td>JENSEN</td>
<td>INTERCEPTOR III</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TRIUMPH</td>
<td>TR7</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

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.

---

**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:

<table>
<thead>
<tr>
<th>Product</th>
<th>Ordered Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazelnut</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td>117</td>
</tr>
<tr>
<td>TOTAL IS:</td>
<td>417</td>
</tr>
<tr>
<td>French Roast</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>49</td>
</tr>
<tr>
<td>TOTAL IS:</td>
<td>261</td>
</tr>
<tr>
<td>Scone</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>325</td>
</tr>
<tr>
<td>TOTAL IS:</td>
<td>362</td>
</tr>
</tbody>
</table>
**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 849.

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.

```plaintext
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.
# Aligning a Heading or Footing Element in an HTML Report

The output is:

<table>
<thead>
<tr>
<th>Product Code</th>
<th>Product</th>
<th>Order Number</th>
<th>Order Date</th>
<th>Store Code</th>
<th>Ordered Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>B141</td>
<td>Hazelnut</td>
<td>1</td>
<td>01/01/96</td>
<td>R1019</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>01/01/96</td>
<td>R1019</td>
<td>117</td>
</tr>
<tr>
<td>B142</td>
<td>French Roast</td>
<td>3</td>
<td>01/01/96</td>
<td>R1019</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>01/01/96</td>
<td>R1019</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>01/01/96</td>
<td>R1019</td>
<td>49</td>
</tr>
<tr>
<td>G100</td>
<td>Mug</td>
<td>6</td>
<td>01/01/96</td>
<td>R1019</td>
<td>289</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>01/01/96</td>
<td>R1019</td>
<td>231</td>
</tr>
<tr>
<td>G104</td>
<td>Thermos</td>
<td>8</td>
<td>01/01/96</td>
<td>R1019</td>
<td>347</td>
</tr>
<tr>
<td>G110</td>
<td>Coffee Grinder</td>
<td>9</td>
<td>01/01/96</td>
<td>R1019</td>
<td>265</td>
</tr>
<tr>
<td>G121</td>
<td>Coffee Pot</td>
<td>10</td>
<td>01/01/96</td>
<td>R1019</td>
<td>309</td>
</tr>
</tbody>
</table>
Notice that the positioning is maintained when the grid is hidden (off):

**Example:** **Aligning a Text Field With a Column in a Sort Footing**

This example uses a Master File and the MODIFY procedure created in *Including a Text Field in a Sort Footing* on page 1102. 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 STYLESHEET *
TYPE = REPORT, GRID = ON, $
TYPE = SUBFOOT, HEADALIGN = BODY, $
ENDSTYLE
END
```
Aligning a Heading or Footing Element in an HTML Report

The output displays a new value for the text field each time the value of CATALOG changes.

<table>
<thead>
<tr>
<th>CATALOG</th>
</tr>
</thead>
<tbody>
<tr>
<td>COURSE 100</td>
</tr>
<tr>
<td>Evening Course</td>
</tr>
<tr>
<td>Course Description: This course provides the junior programmer with the skills needed to code simple reports.</td>
</tr>
<tr>
<td>COURSE 200</td>
</tr>
<tr>
<td>Evening Course</td>
</tr>
<tr>
<td>Course Description: This course provides the advanced programmer with techniques helpful in developing complex applications.</td>
</tr>
</tbody>
</table>

**Example:** Aligning and Styling a Text Field in a Sort Footing

This example uses a Master File and the MODIFY procedure created in *Including a Text Field in a Sort Footing* on page 1102. 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 TXTFLD
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 STYLESHEET *
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 Report

How to:
Align a Heading or Footing Element Across Columns

With HEADALIGN=BODY, each heading or footing element is aligned with a data column in an HTML 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 Report on page 1149.

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.

This feature is supported for HTML only. PDF display format does not support the COLSPAN attribute.

Syntax: How to Align a Heading or Footing Element Across Columns

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. See *Identifying a Report Component in a WebFOCUS StyleSheet* on page 849.

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=CENTRAL, $
TYPE = HEADING, LINE=3, ITEM=3, JUSTIFY=RIGHT, $
ENDSTYLE
END
```
The output is:

**Gotham Grinds, Inc.**

<table>
<thead>
<tr>
<th>Orders Report</th>
<th>Run on: 06/19/02</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Code</td>
<td>Product</td>
</tr>
<tr>
<td>B141</td>
<td>Hazelnut</td>
</tr>
<tr>
<td>B142</td>
<td>French Roast</td>
</tr>
<tr>
<td>G100</td>
<td>Mug</td>
</tr>
<tr>
<td>G104</td>
<td>Thermos</td>
</tr>
<tr>
<td>G110</td>
<td>Coffee Grinder</td>
</tr>
<tr>
<td>G121</td>
<td>Coffee Pot</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 STYLESHEET *
TYPE = REPORT, GRID = OFF, $
TYPE = SUBFOOT, HEADALIGN = BODY, JUSTIFY = RIGHT, STYLE = BOLD, $
TYPE = SUBFOOT, OBJECT = TEXT, COLSPAN = 5, $
ENDSTYLE
END
```

The partial output is:

<table>
<thead>
<tr>
<th>Product Code</th>
<th>Product</th>
<th>Order Number</th>
<th>Order Date</th>
<th>Store Code</th>
<th>Ordered Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>B141</td>
<td>Hazelnut</td>
<td>1 01/01/96</td>
<td>R1019</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 01/01/96</td>
<td>R1019</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total:</strong> 417</td>
</tr>
<tr>
<td>B142</td>
<td>French Roast</td>
<td>3 01/01/96</td>
<td>R1019</td>
<td>126</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 01/01/96</td>
<td>R1019</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 01/01/96</td>
<td>R1019</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total:</strong> 261</td>
</tr>
<tr>
<td>F101</td>
<td>Scone</td>
<td>11 01/01/96</td>
<td>R1019</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 01/01/96</td>
<td>R1019</td>
<td>325</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total:</strong> 362</td>
</tr>
</tbody>
</table>
Aligning Content in a Multi-Line Heading or Footing

In this section:
Aligning Decimals in a Multi-Line Heading or Footing
Combining Column and Line Formatting in Headings and Footings

How to:
Align Heading Text and Data in Columns

Reference:
Line and Item Formatting in a Multi-Line Heading or Footing

The HEADALIGN and COLSPAN syntax described in Aligning a Heading or Footing Element in an HTML Report on page 1149 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. For an illustration, see Aligning Data and Text in a Multi-Line Heading or Footing on page 1169.

- Combine columnar formatting with line-by-line formatting. See Combining Column and Line Formatting to Align Items in a Sort Heading on page 1173.

- 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 1172.

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 1147.
Syntax: How to Align Heading Text and Data in Columns

For a multi-line report or page heading or footing, use the syntax:

```plaintext
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:

```plaintext
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.

`sorfld`  
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. For an illustration, see Combining Column and Line Formatting to Align Items in a Sort Heading on page 1173.

`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 1169 for information about acceptable variations.

You can use either or both approaches for a single heading or footing. For an illustration, see Combining Column and Line Formatting to Align Items in a Sort Heading on page 1173.
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 849. 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 1172.

**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 1172.
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,

<table>
<thead>
<tr>
<th>Line 1</th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line 2</td>
<td>Item 1</td>
<td>Item 2</td>
<td>Item 3</td>
</tr>
</tbody>
</table>

- 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.

<table>
<thead>
<tr>
<th>Line 1</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Line 2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 effect, 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.

<table>
<thead>
<tr>
<th>Country</th>
<th>ARGENTINA</th>
<th>Exchange Rate:</th>
<th>35.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>ST. NOTES</td>
<td>Projected Return:</td>
<td>4.200</td>
</tr>
<tr>
<td>Holder</td>
<td>COMM</td>
<td>Balance:</td>
<td>42,167,880.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>BRAZIL</th>
<th>Exchange Rate:</th>
<th>39.55</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>ST. NOTES</td>
<td>Projected Return:</td>
<td>4.200</td>
</tr>
<tr>
<td>Holder</td>
<td>COMM</td>
<td>Balance:</td>
<td>45,493,501.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>CANADA</th>
<th>Exchange Rate:</th>
<th>49.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>ST. NOTES</td>
<td>Projected Return:</td>
<td>3.990</td>
</tr>
<tr>
<td>Holder</td>
<td>COMM</td>
<td>Balance:</td>
<td>56,212,634.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>CZECH REP</th>
<th>Exchange Rate:</th>
<th>1,192.45</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>ST. NOTES</td>
<td>Projected Return:</td>
<td>12.110</td>
</tr>
<tr>
<td>Holder</td>
<td>COMM</td>
<td>Balance:</td>
<td>51,121,201.00</td>
</tr>
</tbody>
</table>

The chart below breaks out the structure of the previous report:

<table>
<thead>
<tr>
<th>Item 1:</th>
<th>Item 2:</th>
<th>Item 3:</th>
<th>Item 4:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Data values</td>
<td>Text</td>
<td>Values with decimal places</td>
</tr>
<tr>
<td>Country</td>
<td>ARGENTINA</td>
<td>Exchange Rate</td>
<td>nn.dd</td>
</tr>
<tr>
<td></td>
<td>BRAZIL, and so on</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>ST. NOTES</td>
<td>Projected Return</td>
<td>n.ddd</td>
</tr>
<tr>
<td>Holder</td>
<td>COMM</td>
<td>Balance</td>
<td>nn,nnn,nnn.dd</td>
</tr>
</tbody>
</table>
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 STYLESHEET *
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 1167 and *Aligning Decimals in a Multi-Line Heading or Footing* on page 1172.

**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:

<table>
<thead>
<tr>
<th>Floating decimal points</th>
<th>Aligned decimal points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bond</td>
<td>Face Value</td>
</tr>
<tr>
<td>Galosh Ltd.</td>
<td>22375.5784596</td>
</tr>
<tr>
<td>Mukluk Inc.</td>
<td>1212345.457</td>
</tr>
<tr>
<td>Overshoe Inc.</td>
<td>232.45484</td>
</tr>
</tbody>
</table>

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 1167.

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:

<table>
<thead>
<tr>
<th>Font</th>
<th>Font size</th>
<th>Comparison</th>
<th>Inches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helvetica</td>
<td>10</td>
<td>England</td>
<td>.5</td>
</tr>
<tr>
<td>Times New Roman</td>
<td>10</td>
<td>England</td>
<td>.44</td>
</tr>
<tr>
<td>Courier</td>
<td>10</td>
<td>England</td>
<td>.56</td>
</tr>
</tbody>
</table>

**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 849.
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 STYLESHEET *
   TYPE=REPORT, FONT='TIMES', $  
   TYPE=REPORT, GRID=OFF, $  
   -* Bottom section of subhead:
6. 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.

<table>
<thead>
<tr>
<th>Country</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARGENTINA</td>
<td>SOUTH AMERICA</td>
</tr>
<tr>
<td>Type: ST. NOTES</td>
<td>Exchange Rate: 35.00</td>
</tr>
<tr>
<td>Holder: COMM</td>
<td>Projected Return: 4.200</td>
</tr>
<tr>
<td>Risk class: Medium</td>
<td>Balance: 42,167,880.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRAZIL</td>
<td>SOUTH AMERICA</td>
</tr>
<tr>
<td>Type: ST. NOTES</td>
<td>Exchange Rate: 39.55</td>
</tr>
<tr>
<td>Holder: COMM</td>
<td>Projected Return: 4.200</td>
</tr>
<tr>
<td>Risk class: Medium</td>
<td>Balance: 45,493,501.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>CZECH. REP</td>
<td>EASTERN EUROPE</td>
</tr>
<tr>
<td>Type: ST. NOTES</td>
<td>Exchange Rate: 1,192.45</td>
</tr>
<tr>
<td>Holder: COMM</td>
<td>Projected Return: 12.110</td>
</tr>
<tr>
<td>Risk class: High</td>
<td>Balance: 51,121,201.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGLAND</td>
<td>WESTERN EUROPE</td>
</tr>
<tr>
<td>Type: ST. NOTES</td>
<td>Exchange Rate: 21.35</td>
</tr>
<tr>
<td>Holder: COMM</td>
<td>Projected Return: 4.060</td>
</tr>
<tr>
<td>Risk class: Medium</td>
<td>Balance: 125,757,198.00</td>
</tr>
<tr>
<td>Line #</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>1-2</td>
<td>Defines the content for the <em>top</em>, 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 &lt;+0&gt; spot marker), the second contains text, the third contains data values related to the text.</td>
</tr>
<tr>
<td>3-5</td>
<td>Defines the content for the <em>bottom</em>, 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.</td>
</tr>
<tr>
<td>6</td>
<td>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.</td>
</tr>
</tbody>
</table>
### Aligning Content in a Multi-Line Heading or Footing

<table>
<thead>
<tr>
<th>Line #</th>
<th>Description</th>
</tr>
</thead>
</table>
| **7-11** | 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.  
**Important:** 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.  
The formatting of the bottom, three-line section of the heading is controlled by the following specifications:  
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.  
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.  
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.  
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.  
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.  
The common width and justification definitions enforce the proper alignment of each item. |
**Line #** | **Description**
---|---
12 | Defines a variable called &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. 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 *Developing Reporting Applications* manual.

13-18 | Specifies line-by-line formatting for the top, 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. 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 &INDENT. Item 2 on each line refers to the text area. It has a defined width of 1 inch and the text is left justified. Item 3 on each line refers to the data values. It has a defined width of 2 inches and the data is left justified. The common width and justification definitions enforce the proper alignment of each item. 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.
Positioning Headings, Footings, or Items Within Them

In this section:
Using PRINTPLUS
Using Spot Markers to Refine Positioning

How to:
Set a Starting Position for a Heading or Footing
Set a Starting Position for a Heading or Footing Element

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 1147.

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 929.

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. For an illustration, see *Setting a Starting Position for a Report Heading in HTML* on page 1182.

**Example:**  
**Setting a Starting Position for a Report Heading in PDF**

This request positions the report heading 1.25 inches from the left margin.

```plaintext
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

<table>
<thead>
<tr>
<th>Store ID</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1019</td>
<td>28965</td>
<td>21905</td>
</tr>
<tr>
<td></td>
<td>25350</td>
<td>19305</td>
</tr>
<tr>
<td></td>
<td>26054</td>
<td>17360</td>
</tr>
<tr>
<td></td>
<td>27210</td>
<td>17380</td>
</tr>
</tbody>
</table>
```
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

<table>
<thead>
<tr>
<th>Store ID</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1019</td>
<td>28965</td>
<td>21905</td>
</tr>
<tr>
<td></td>
<td>25350</td>
<td>19305</td>
</tr>
<tr>
<td></td>
<td>26054</td>
<td>17360</td>
</tr>
<tr>
<td></td>
<td>27210</td>
<td>17380</td>
</tr>
</tbody>
</table>
```

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. See *Identifying a Report Component in a WebFOCUS StyleSheet* on page 849.

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.

  + 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.

  – 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** 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.

```plaintext
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**

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Cappuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>87863</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>
**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.

```plaintext
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 STYLESHET *
TYPE = TABHEADING, LINE=1, ITEM=2, POSITION=UNITS, $
ENDSTYLE
END
```

The output is:

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>

**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 849.
Using PRINTPLUS

How to:

Use 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 "  
END
```

The output is:

<table>
<thead>
<tr>
<th>SEATS</th>
<th>COUNTRY</th>
<th>CAR</th>
<th>MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>ENGLAND</td>
<td>TR7</td>
<td>TR7</td>
</tr>
<tr>
<td></td>
<td>ITALY</td>
<td>ALFA ROMEO 2000 GT VELOCE</td>
<td>ALFA ROMEO 2000 SPIDER VELOCE</td>
</tr>
<tr>
<td></td>
<td>ITALY</td>
<td>Maserati Dora 2 Door</td>
<td>Maserati Dora 2 Door</td>
</tr>
<tr>
<td>4</td>
<td>ENGLAND</td>
<td>Jaguar V12XKE Auto</td>
<td>Jaguar V12XKE Auto</td>
</tr>
<tr>
<td></td>
<td>ITALY</td>
<td>JENSEN Interceptor III</td>
<td>JENSEN Interceptor III</td>
</tr>
<tr>
<td>5</td>
<td>ENGLAND</td>
<td>Jaguar XJ12L AuRO</td>
<td>Jaguar XJ12L AuRO</td>
</tr>
<tr>
<td></td>
<td>FRANCE</td>
<td>Peugeot 504 4 Door</td>
<td>Peugeot 504 4 Door</td>
</tr>
</tbody>
</table>

SUMMARY OF CARS IN COUNTRY BY SEATING CAPACITY  
RELPMEK 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 1147.

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 849.
<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 1193.

<n controls the positioning of a character immediately following a field. For illustrations of this technique, see Positioning a Character Immediately After a Field in an HTML Report on page 1189 and Positioning a Character Immediately After a Field in a PDF or PS Report on page 1190.

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

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Product Number#</th>
<th>Quantity</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>74610</td>
<td>1012</td>
<td>268</td>
<td>$222,234.35</td>
</tr>
<tr>
<td>1028</td>
<td>323</td>
<td></td>
<td>$29,215.62</td>
</tr>
<tr>
<td>1032</td>
<td>339</td>
<td></td>
<td>$27,208.12</td>
</tr>
<tr>
<td>1034</td>
<td>339</td>
<td></td>
<td>$154,943.38</td>
</tr>
<tr>
<td>1036</td>
<td>339</td>
<td></td>
<td>$93,266.39</td>
</tr>
</tbody>
</table>

Century Corporation orders for store Audio Expert (Store # 1003CA) in CA.

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Product Number#</th>
<th>Quantity</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>39274</td>
<td>1006</td>
<td>179</td>
<td>$54,201.50</td>
</tr>
<tr>
<td>1008</td>
<td>179</td>
<td></td>
<td>$29,034.16</td>
</tr>
<tr>
<td>1020</td>
<td>179</td>
<td></td>
<td>$45,457.59</td>
</tr>
<tr>
<td>1032</td>
<td>400</td>
<td></td>
<td>$30,861.24</td>
</tr>
<tr>
<td>1034</td>
<td>400</td>
<td></td>
<td>$143,744.49</td>
</tr>
</tbody>
</table>

Century Corporation orders for store Audio Expert (Store # 1003CO) in CO.

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.

\begin{verbatim}
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
\end{verbatim}

The output is:

<table>
<thead>
<tr>
<th>Order Number:</th>
<th>Product Number#</th>
<th>Quantity</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>74610</td>
<td>1012</td>
<td>268</td>
<td>$222,234.35</td>
</tr>
<tr>
<td></td>
<td>1028</td>
<td>323</td>
<td>$29,215.62</td>
</tr>
<tr>
<td></td>
<td>1632</td>
<td>339</td>
<td>$27,208.12</td>
</tr>
<tr>
<td></td>
<td>1634</td>
<td>339</td>
<td>$154,943.38</td>
</tr>
<tr>
<td></td>
<td>1036</td>
<td>339</td>
<td>$93,266.39</td>
</tr>
</tbody>
</table>

Century Corporation orders for store Audio Expert (store # 1003CA) in CA.

<table>
<thead>
<tr>
<th>Order Number:</th>
<th>Product Number#</th>
<th>Quantity</th>
<th>Line Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>39274</td>
<td>1006</td>
<td>179</td>
<td>$54,201.50</td>
</tr>
<tr>
<td></td>
<td>1008</td>
<td>179</td>
<td>$29,034.16</td>
</tr>
<tr>
<td></td>
<td>1020</td>
<td>179</td>
<td>$45,457.59</td>
</tr>
<tr>
<td></td>
<td>1032</td>
<td>400</td>
<td>$30,861.24</td>
</tr>
<tr>
<td></td>
<td>1034</td>
<td>400</td>
<td>$143,744.49</td>
</tr>
</tbody>
</table>

Century Corporation orders for store Audio Expert (store # 1003CO) in CO.

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
      1036  329  $154,943.38
      1036  329  $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 in 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:

**Controlling the Vertical Positioning of a Heading or Footing**

**How to:**

- Add Blank Lines to a Heading or Footing
- Control Vertical Spacing in a Heading or Footing
- Position a Page Footing at the Bottom of a Page

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 1194.

- In a PDF report you can control the vertical space between heading and footing lines, or increase the distance between text and the surrounding grid lines in a one-line heading or footing, using the TOPGAP and BOTTOMGAP attributes in a StyleSheet. See *Adjusting Vertical Spacing Below a Sort Footing* on page 1197. For full information on TOPGAP and BOTTOMGAP, see *Laying Out the Report Page* on page 929.
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 1198.

**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

<table>
<thead>
<tr>
<th>Category</th>
<th>Budget Units</th>
<th>Unit Sales</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1385923</td>
<td>1376266</td>
<td>17293907</td>
<td>17231465</td>
</tr>
<tr>
<td>Food</td>
<td>1377564</td>
<td>1384845</td>
<td>17267160</td>
<td>17229344</td>
</tr>
<tr>
<td>Gifts</td>
<td>931007</td>
<td>927880</td>
<td>11659732</td>
<td>11695502</td>
</tr>
</tbody>
</table>

**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 between heading or footing lines and 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 relative to the report component directly below it.

**BOTTOMGAP**

Indicates how much space to add below a report component relative to the report component directly above it.
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  December 2001

<table>
<thead>
<tr>
<th>Category</th>
<th>Budget Units</th>
<th>Unit Sales</th>
<th>Budget Dollars</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1385923</td>
<td>1376266</td>
<td>17293886</td>
<td>17231455</td>
</tr>
<tr>
<td>Food</td>
<td>1377564</td>
<td>1384845</td>
<td>17267160</td>
<td>17229333</td>
</tr>
<tr>
<td>Gifts</td>
<td>931007</td>
<td>927880</td>
<td>11659732</td>
<td>11695502</td>
</tr>
</tbody>
</table>
```
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:

```
<table>
<thead>
<tr>
<th>Product Code</th>
<th>Package</th>
<th>Unit Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>G110</td>
<td>Case</td>
<td>125.00</td>
</tr>
<tr>
<td></td>
<td>Summary for Coffee Grinder</td>
<td></td>
</tr>
<tr>
<td>G121</td>
<td>Case</td>
<td>140.00</td>
</tr>
<tr>
<td></td>
<td>Summary for Coffee Pot</td>
<td></td>
</tr>
<tr>
<td>B142</td>
<td>Pounds</td>
<td>81.00</td>
</tr>
<tr>
<td></td>
<td>Summary for French Roast</td>
<td></td>
</tr>
<tr>
<td>B141</td>
<td>Pounds</td>
<td>58.00</td>
</tr>
<tr>
<td></td>
<td>Summary for Hazelnut</td>
<td></td>
</tr>
<tr>
<td>B144</td>
<td>Pounds</td>
<td>76.00</td>
</tr>
<tr>
<td></td>
<td>Summary for Kona</td>
<td></td>
</tr>
</tbody>
</table>
```
Syntax: How to Position a Page Footing at the Bottom of a Page

For a PDF report or an HTML report displayed in the WebFOCUS Viewer, 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.
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 1055.

**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 1093.

**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 849.

- `</n` specifies skipped lines. For details, see *Controlling the Vertical Positioning of a Heading or Footing* on page 1193.

- `<-n` to position the next character on the line. For details, see *Using Spot Markers to Refine Positioning* on page 1188.

- `<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 1056.

**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

How to:
Position 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.

```plaintext
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 1055.
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 TABBAGENO). 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 1093.

**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 849.

</n specifies skipped lines. For details, see [Controlling the Vertical Positioning of a Heading or Footing](#) on page 1193.

<-n positions the next character on the line. For details, see [Using Spot Markers to Refine Positioning](#) on page 1188.

<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 1056.

**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

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Date Of Order</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>94710</td>
<td>2001/01/02</td>
<td>$406,964.24</td>
</tr>
<tr>
<td>94680</td>
<td>2001/01/02</td>
<td>$421,916.60</td>
</tr>
<tr>
<td>94670</td>
<td>2001/01/02</td>
<td>$513,868.76</td>
</tr>
<tr>
<td>94550</td>
<td>2001/01/02</td>
<td>$496,323.64</td>
</tr>
<tr>
<td>94530</td>
<td>2001/01/02</td>
<td>$3,472.41</td>
</tr>
</tbody>
</table>
```

**Tip:** To produce comparable results in HTML, include the following code in the request to turn on the WebFOCUS Viewer. This syntax is not respected by standard browsers.

```
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET WEBVIEWER ON
ON TABLE SET STYLESHEET *
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

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Date Of Order</th>
<th>Order Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>94710</td>
<td>2001/01/02</td>
<td>$406,964.24</td>
</tr>
<tr>
<td>94680</td>
<td>2001/01/02</td>
<td>$421,916.60</td>
</tr>
<tr>
<td>94670</td>
<td>2001/01/02</td>
<td>$513,868.76</td>
</tr>
<tr>
<td>94550</td>
<td>2001/01/02</td>
<td>$496,323.64</td>
</tr>
<tr>
<td>94530</td>
<td>2001/01/02</td>
<td>$3,472.41</td>
</tr>
</tbody>
</table>
```

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. This syntax is not respected by standard browsers.

```
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET WEBVIEWER ON
ON TABLE SET STYLESHEET *
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.
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.

Topics:

- Specifying Font Format in a Report
- Specifying Background Color in a Report
- Specifying Data Format in a Report
- Positioning Data 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, that is needed to identify the report component that you are formatting. See *Identifying a Report Component in a WebFOCUS StyleSheet* on page 849 for more information on 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 1207.
**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:

```
<table>
<thead>
<tr>
<th>Category</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1376266</td>
<td>17231465</td>
</tr>
<tr>
<td>Food</td>
<td>1384845</td>
<td>17229344</td>
</tr>
<tr>
<td>Gifts</td>
<td>927880</td>
<td>11695502</td>
</tr>
</tbody>
</table>
```

**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:

<table>
<thead>
<tr>
<th>Size in Points</th>
<th>Corresponding HTML Font Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 or smaller</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td><strong>10</strong></td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
</tr>
<tr>
<td>14 or larger</td>
<td>7</td>
</tr>
</tbody>
</table>
**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, that is needed to identify the report component that you are formatting. See *Identifying a Report Component in a WebFOCUS StyleSheet* on page 849 for more information on 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 GSALES
SUM UNITS DOLLARS BY CATEGORY
ON TABLE SET STYLE *
TYPE=TITLE, STYLE=BOLD+ITALIC, $
TYPE=REPORT, GRID=OFF, $
ENDSTYLE
END
```
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:

Sales Report by Category

<table>
<thead>
<tr>
<th>Category</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>1376266</td>
<td>17231465</td>
</tr>
<tr>
<td>Food</td>
<td>1384845</td>
<td>17229344</td>
</tr>
<tr>
<td>Gifts</td>
<td>927880</td>
<td>11695502</td>
</tr>
</tbody>
</table>

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, that is needed to identify the report component that you are formatting. See *Identifying a Report Component in a WebFOCUS StyleSheet* on page 849 for more information on 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 1211.

**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 uppercase or lowercase 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:

<table>
<thead>
<tr>
<th>Color Name</th>
<th>Color Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEDIUM FOREST GREEN</td>
<td>MEDIUM FOREST GREEN (OLIVE)</td>
</tr>
<tr>
<td>AQUA (CYAN)</td>
<td>MEDIUM GOLDENROD</td>
</tr>
<tr>
<td>AQUAMARINE</td>
<td>MEDIUM ORCHID</td>
</tr>
<tr>
<td>BLACK</td>
<td>MEDIUM SLATE BLUE</td>
</tr>
<tr>
<td>BLUE VIOLET</td>
<td>MEDIUM SPRING GREEN</td>
</tr>
<tr>
<td>CADET BLUE</td>
<td>MEDIUM TURQUOISE</td>
</tr>
<tr>
<td>CORAL</td>
<td>MEDIUM VIOLET RED</td>
</tr>
<tr>
<td>CORNFLOWER BLUE</td>
<td>NAVY (NAVY BLUE)</td>
</tr>
<tr>
<td>CYAN (AQUA)</td>
<td>MIDNIGHT BLUE</td>
</tr>
<tr>
<td>DARK GREEN</td>
<td>NAVY (NAVY BLUE)</td>
</tr>
<tr>
<td>DARK OLIVE GREEN</td>
<td>OLIVE (MEDIUM FOREST GREEN)</td>
</tr>
<tr>
<td>DARK ORCHID</td>
<td>ORANGE</td>
</tr>
<tr>
<td>DARK SLATE BLUE (PURPLE)</td>
<td>ORANGE RED</td>
</tr>
<tr>
<td>DARK SLATE GREY</td>
<td>ORCHID</td>
</tr>
<tr>
<td>DARK TURQUOISE</td>
<td>PALE GREEN</td>
</tr>
<tr>
<td>DIM GREY (GRAY, GREY)</td>
<td>PINK</td>
</tr>
<tr>
<td>FIREBRICK</td>
<td>PLUM</td>
</tr>
<tr>
<td>FOREST GREEN (GREEN)</td>
<td>PURPLE (DARK SLATE BLUE)</td>
</tr>
<tr>
<td>FUCHSIA (MAGENTA)</td>
<td>RED</td>
</tr>
<tr>
<td>GOLD</td>
<td>SALMON</td>
</tr>
<tr>
<td>GOLDENROD</td>
<td>SEA GREEN</td>
</tr>
<tr>
<td>GRAY (DIM GREY, GREY)</td>
<td>SIENNA</td>
</tr>
<tr>
<td>GREEN (FOREST GREEN)</td>
<td>SILVER</td>
</tr>
<tr>
<td>GREEN YELLOW</td>
<td>SKY BLUE</td>
</tr>
<tr>
<td>GREY (DIM GREY, GRAY)</td>
<td>SLATE BLUE</td>
</tr>
<tr>
<td>INDIAN RED</td>
<td>STEEL BLUE (TEAL)</td>
</tr>
<tr>
<td>KHAKI</td>
<td>TAN</td>
</tr>
<tr>
<td>LIGHT BLUE</td>
<td>TEAL (STEEL BLUE)</td>
</tr>
<tr>
<td>LIGHT GREY</td>
<td>THISTLE</td>
</tr>
<tr>
<td>LIGHT STEEL BLUE</td>
<td>TURQUOISE</td>
</tr>
<tr>
<td>LIME</td>
<td>VIOLET</td>
</tr>
<tr>
<td>LIME GREEN</td>
<td>VIOLET RED</td>
</tr>
<tr>
<td>MAGENTA (FUCHSIA)</td>
<td>WHEAT</td>
</tr>
<tr>
<td>MAROON</td>
<td>WHITE</td>
</tr>
<tr>
<td>MEDIUM AQUAMARINE</td>
<td>YELLOW</td>
</tr>
<tr>
<td>MEDIUM BLUE</td>
<td>YELLOW GREEN</td>
</tr>
</tbody>
</table>
Specifying Fonts for Reports

How to:

Specify Fonts in a Report
Specify the Default Browser Fonts for HTML 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, that is needed to identify the report component that you are formatting. See Identifying a Report Component in a WebFOCUS StyleSheet on page 849 for more information on 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 monospace 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 monospace 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**

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381600</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263328</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>
Specifying Background Color in a Report

How to:
Specify Background Color in a Report
Specify Alternating Data 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. In addition, alternating colors can be specified for the background of the data lines in a report.

Syntax: Specifying 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. See Identifying a Report Component in a WebFOCUS StyleSheet on page 849 for more information on how to specify different report components.

- **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 uppercase or lowercase 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:

```sql
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:

```
<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
<th>Unit Sales</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Capuccino</td>
<td>189217</td>
<td>2381590</td>
</tr>
<tr>
<td></td>
<td>Espresso</td>
<td>308986</td>
<td>3906243</td>
</tr>
<tr>
<td></td>
<td>Latte</td>
<td>878063</td>
<td>10943622</td>
</tr>
<tr>
<td>Food</td>
<td>Biscotti</td>
<td>421377</td>
<td>5263317</td>
</tr>
<tr>
<td></td>
<td>Croissant</td>
<td>630054</td>
<td>7749902</td>
</tr>
<tr>
<td></td>
<td>Scone</td>
<td>333414</td>
<td>4216114</td>
</tr>
<tr>
<td>Gifts</td>
<td>Coffee Grinder</td>
<td>186534</td>
<td>2337567</td>
</tr>
<tr>
<td></td>
<td>Coffee Pot</td>
<td>190695</td>
<td>2449585</td>
</tr>
<tr>
<td></td>
<td>Mug</td>
<td>360570</td>
<td>4522521</td>
</tr>
<tr>
<td></td>
<td>Thermos</td>
<td>190081</td>
<td>2385829</td>
</tr>
</tbody>
</table>
```
**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 uppercase or lowercase 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
```
Specifying Data Format in a Report

In this section:
- Changing the Format of Values in a Report Column
- Controlling Missing Values for a Reformatted Field
- Using Commas vs. Decimals (Continental Decimal Notation)
- Setting Characters to Represent Null and Missing Values
- Using Conditional Grid Formatting in a Field

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

How to:
Change Format of Values in a Column

Reference:
Usage Notes for Changing Column Format

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:

<table>
<thead>
<tr>
<th>Unit</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$58.00</td>
</tr>
<tr>
<td></td>
<td>$81.00</td>
</tr>
<tr>
<td></td>
<td>$76.00</td>
</tr>
<tr>
<td></td>
<td>$13.00</td>
</tr>
<tr>
<td></td>
<td>$17.00</td>
</tr>
<tr>
<td></td>
<td>$28.00</td>
</tr>
<tr>
<td></td>
<td>$26.00</td>
</tr>
<tr>
<td></td>
<td>$96.00</td>
</tr>
<tr>
<td></td>
<td>$125.00</td>
</tr>
<tr>
<td></td>
<td>$140.00</td>
</tr>
</tbody>
</table>

**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:

```plaintext
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:

<table>
<thead>
<tr>
<th>CAR</th>
<th>MODEL</th>
<th>RJUST STANDARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAGUAR</td>
<td>V12XKE AUT</td>
<td>POWER STEERING</td>
</tr>
<tr>
<td></td>
<td>XJ12L AUTO</td>
<td>RECLINING BUCKE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WHITETWALL RADIA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WRAP AROUND BUM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 WHEEL DISC BR</td>
</tr>
<tr>
<td>TOYOTA</td>
<td>COROLLA 4</td>
<td>BODY SIDE MOLDI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MACPHERSON STRU</td>
</tr>
</tbody>
</table>

**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 825.

- 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 351.
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

**How to:**
Control Missing Values in Reformatted Fields

**Reference:**
Usage Notes for SET COMPMISS

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.)*

```plaintext
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:

<table>
<thead>
<tr>
<th>STORE_CODE</th>
<th>RETURNS</th>
<th>REFORMATTED RETURNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>14Z</td>
<td>2</td>
<td>2.00</td>
</tr>
<tr>
<td>2</td>
<td>2.00</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>.00</td>
<td></td>
</tr>
<tr>
<td>.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4.00</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>.00</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3.00</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4.00</td>
<td></td>
</tr>
<tr>
<td>.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4.00</td>
<td></td>
</tr>
</tbody>
</table>

With COMPMISS ON, the column for the reformatted version of RETURNS displays the missing data symbol when a value is missing:

```plaintext
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:

<table>
<thead>
<tr>
<th>STORE_CODE</th>
<th>RETURNS</th>
<th>REFORMATTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>14Z</td>
<td>2</td>
<td>2.00</td>
</tr>
<tr>
<td>2</td>
<td>2.00</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>.00</td>
<td></td>
</tr>
<tr>
<td>.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4.00</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>.00</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3.00</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4.00</td>
<td></td>
</tr>
<tr>
<td>.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4.00</td>
<td></td>
</tr>
</tbody>
</table>

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)

How to:

Set the CDN Parameter

The CDN parameter determines whether a numeric value displays with default notation or Continental Decimal Notation. Continental Decimal Notation uses a comma to mark the decimal position in a number, and periods (.) to mark off significant digits into groups of three. The default notation setting uses a period (.) to mark the decimal position in a number, and commas to mark off significant digits into groups of three.

Syntax: How to Set the CDN Parameter

To set the CDN parameter, use the following syntax

```
SET CDN=[ON|OFF]
```

where:

ON

Indicates the use of the Continental Decimal Notation. For example, a numeric value with CDN set to ON would display as follows: 3.045.000,76.
OFF

Turns CDN off. For example, the number above is represented as 3,045,000.76. OFF is the default value.

Note: CDN is supported for output requests only.

Setting Characters to Represent Null and Missing Values

How to:
Set Characters to Represent a Null or Missing Value
Set the NODATA Character as a SET Command

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.

calendar 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 NODATA Character as a StyleSheet Attribute

This request changes the NODATA character for missing data from a period (default) to the word NONE.

```plaintext
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, $
ENDSTYLE
END
```

This request produces the following report.

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>MIS</th>
<th>PRODUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTION</td>
<td>BANNING</td>
<td>JOHN</td>
<td>NONE</td>
<td>$29,700.00</td>
</tr>
<tr>
<td></td>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>$21,780.00</td>
<td>NONE</td>
</tr>
<tr>
<td></td>
<td>CROSS</td>
<td>BARBARA</td>
<td>$27,062.00</td>
<td>NONE</td>
</tr>
<tr>
<td></td>
<td>GREENSPAN</td>
<td>MARY</td>
<td>$9,000.00</td>
<td>NONE</td>
</tr>
<tr>
<td></td>
<td>IRVING</td>
<td>JOAN</td>
<td>NONE</td>
<td>$26,862.00</td>
</tr>
<tr>
<td></td>
<td>JONES</td>
<td>DIANE</td>
<td>$18,480.00</td>
<td>NONE</td>
</tr>
<tr>
<td></td>
<td>MCCOY</td>
<td>JOHN</td>
<td>$18,480.00</td>
<td>NONE</td>
</tr>
<tr>
<td></td>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>NONE</td>
<td>$16,100.00</td>
</tr>
<tr>
<td></td>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>NONE</td>
<td>$21,120.00</td>
</tr>
<tr>
<td></td>
<td>SMITH</td>
<td>MARY</td>
<td>$13,200.00</td>
<td>NONE</td>
</tr>
<tr>
<td></td>
<td>STEVENS</td>
<td>ALFRED</td>
<td>NONE</td>
<td>$9,500.00</td>
</tr>
</tbody>
</table>
```

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

```plaintext
TABLE FILE CAR
SUM SALES BY CAR
ON TABLE SET STYLE *
TYPE=DATA, COLUMN=SALES, GRID=HEAVY, WHEN=CAR EQ 'DATSUN', $
ENDSTYLE
END
```
Positioning Data in a Report

In this section:

- Controlling Wrapping of Report Data
- Controlling Wrapping With Alternative Methods
- Justifying Report Columns
- Field-Based Reformatting
- Data Visualization
- Displaying Multi-Line An and AnV Fields

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 929.

<table>
<thead>
<tr>
<th>CAR</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALFA ROMEO</td>
<td>30200</td>
</tr>
<tr>
<td>AUDI</td>
<td>7800</td>
</tr>
<tr>
<td>BMW</td>
<td>80390</td>
</tr>
<tr>
<td>DATSUN</td>
<td>43000</td>
</tr>
<tr>
<td>JAGUAR</td>
<td>12000</td>
</tr>
<tr>
<td>JENSEN</td>
<td>0</td>
</tr>
<tr>
<td>MASERATI</td>
<td>0</td>
</tr>
<tr>
<td>PEUGEOT</td>
<td>0</td>
</tr>
<tr>
<td>TOYOTA</td>
<td>30030</td>
</tr>
<tr>
<td>TRIUMPH</td>
<td>0</td>
</tr>
</tbody>
</table>
Controlling Wrapping of Report Data

How to:

Control Wrapping of Report Data Using a StyleSheet Attribute
Control Wrapping of Report Data Using a SET Command
Control Spacing Between Wrapped Lines

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.

In PDF and PostScript report output, you can control the line spacing in wrapped lines by using the WRAPGAP attribute.

Syntax: How to Control Wrapping of Report Data Using a StyleSheet Attribute

To suppress wrapping of text inside a report, use the following syntax within a StyleSheet.

```plaintext
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, that is needed to identify the report component that you are formatting. See Identifying a Report Component in a WebFOCUS StyleSheet on page 849 for more information on how to specify different report components.

- **value**  
  Is one of the following:

  - **ON** turns on data wrapping. ON is the default value. If you do not specify the WRAP attribute in an HTML report, it defaults to ON. For HTML reports, WRAP is supported with all fields. For PDF reports, WRAP is supported only with embedded fields, not text.

  - **OFF** turns off data wrapping.

  - **n** represents a specific numeric value that the column width can be set to. The value represents the measure specified with the UNITS parameter.
Syntax: **How to Control Wrapping of Report Data Using a SET Command**

To suppress wrapping of text inside a report, use the following syntax

```plaintext
SET WRAP=value, $
```

where:

**value**

Is one of the following:

- **ON** turns on data wrapping. ON is the default value.
- **OFF** turns off data wrapping.
- **n** represents a specific numeric value that the column width can be set to. The value represents the measure specified with the UNITS parameter.

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
```

The output is:

```
<table>
<thead>
<tr>
<th>Product Code</th>
<th>Product</th>
<th>Size</th>
<th>Unit Price</th>
<th>Package</th>
<th>Vendor ID</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>B141</td>
<td>Hazelnut</td>
<td>16</td>
<td>58.00</td>
<td>Pounds</td>
<td>V082</td>
<td>Coffee Connection</td>
</tr>
<tr>
<td>B142</td>
<td>French Roast</td>
<td>12</td>
<td>81.00</td>
<td>Pounds</td>
<td>V083</td>
<td>European Specialities,</td>
</tr>
<tr>
<td>B144</td>
<td>Kona</td>
<td>12</td>
<td>76.00</td>
<td>Pounds</td>
<td>V081</td>
<td>Evelina Imports, Ltd</td>
</tr>
</tbody>
</table>
```

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

<table>
<thead>
<tr>
<th>Product Code</th>
<th>Product</th>
<th>Size</th>
<th>Unit Price</th>
<th>Package</th>
<th>Vendor ID</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>B141</td>
<td>Hazelnut</td>
<td>16</td>
<td>58.00</td>
<td>Pounds</td>
<td>V082</td>
<td>Coffee Connection</td>
</tr>
<tr>
<td>B142</td>
<td>French Roast</td>
<td>12</td>
<td>81.00</td>
<td>Pounds</td>
<td>V083</td>
<td>European Specialities,</td>
</tr>
<tr>
<td>B144</td>
<td>Kona</td>
<td>12</td>
<td>76.00</td>
<td>Pounds</td>
<td>V081</td>
<td>Evelina Imports, Ltd</td>
</tr>
</tbody>
</table>
```

**Syntax:**  **How to Control Spacing Between Wrapped Lines**

You can use the WRAPGAP attribute in a StyleSheet to control spacing between wrapped lines in PDF and PostScript report output.

```
type=component, WRAPGAP={ON|OFF|n}
```

where:

- **component**
  - Is the component with wrapped lines.

- **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,$
END
```
With WRAPGAP=OFF, each wrapped line is placed on the next report line:

<table>
<thead>
<tr>
<th>LAST NAME</th>
<th>FIRST NAME</th>
<th>ADDRESS LINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANNING</td>
<td>JOHN</td>
<td>FREEPORT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NY 11520</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>NEW YORK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NY 10001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EDISON NJ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>08817</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BROOKLYN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NY 11210</td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>NEW YORK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NY 10001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLUSHING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NY 11354</td>
</tr>
</tbody>
</table>
With WRAPGAP=ON, the wrapped lines are placed directly under each other:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>ADDRESS_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANNING</td>
<td>JOHN</td>
<td>FREEPORT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NY 11520</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>NEW YORK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NY 10001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EDISON NJ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>08817</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BROOKLYN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NY 11210</td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>NEW YORK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NY 10001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLUSHING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NY 11354</td>
</tr>
</tbody>
</table>

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 or these attributes makes a decrease in spacing between wrapped lines more noticeable.
Controlling Wrapping With Alternative Methods

How to:
Control Wrapping Using SQUEEZE
Control Wrapping Using SQUEEZE as a SET Command

Reference:
Set Commands That Affect Wrapping in a Report
Usage Notes for WRAP and SQUEEZE

In addition to the WRAP parameter, whether or not text wraps within a cell is affected by the SQUEEZE and GRID attributes. Both SQUEEZE and GRID can be used as SET commands or StyleSheet attributes. By specifying a setting other than the default value for these two attributes, the wrapping behavior in a browser is affected.

Reference:  Set Commands That Affect Wrapping in a Report

The following SET commands affect the wrapping of data.

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET SQUEEZE=OFF</td>
<td>Forcing the browser to display the entire column width and suppressing the wrapping of column data.</td>
</tr>
<tr>
<td>SET SQUEEZE=ON</td>
<td>Allowing the browser to follow its own display behavior. If a browser settings are specified, columns will be compressed to the length of the longest data value, and column data with embedded blanks will be wrapped.</td>
</tr>
<tr>
<td>SET GRID=OFF</td>
<td>Automatically suppressing the wrapping of data.</td>
</tr>
<tr>
<td>SET GRID=ON</td>
<td>Has no effect on either wrapping or compressing column width.</td>
</tr>
</tbody>
</table>
Syntax: **How to Control Wrapping Using SQUEEZE**

SQUEEZE can be used as either a SET parameter or a StyleSheet attribute. To control wrapping using SQUEEZE, use the following syntax within a StyleSheet.

```
TYPE =type, SQUEEZE=value, $
```

where:

- **type**
  - Is the report component you wish to affect, such as REPORT, HEADING, or TITLE.

- **value**
  - Is one of the following:
    - **ON** turns on data wrapping. This is the default for HTML output. Has no effect on PDF output.
    - **OFF** turns off data wrapping. This is the default for PDF and PostScript.
    - **n** represents a specific numeric value to which the column width can be set (valid only in PDF and PostScript). The value represents the measure specified with the UNITS parameter.

**Note:** For PDF reports, if a data value is wider than the specified width of the column, it will be hidden from view. For Excel reports, column width can be adjusted after the spreadsheet has been generated.

Syntax: **How to Control Wrapping Using SQUEEZE as a SET Command**

To suppress the wrapping feature in a report by setting the SQUEEZE attribute to OFF, use the following syntax.

```
SET SQUEEZE=value, $
```

where:

- **value**
  - Is one of the following:
    - **ON** turns on data wrapping. ON is the default value.
    - **OFF** turns off data wrapping.
    - **n** represents a specific numeric value to which the column width can be set (valid only in PDF and PS). The value represents the measure specified with the UNITS parameter.
Reference: Usage Notes for WRAP and SQUEEZE

- Most web browsers, by default, determine the width of a report column using an internal algorithm. If a data value exceeds the column width, the browser wraps that data onto the next line of the column. If you want to prevent these line breaks, you can override this browser behavior by using the WRAP StyleSheet attribute.

- Specifying a width is supported in HTML reports that generate an internal Cascading Style Sheet, as well as in PDF and PostScript reports.

- The WRAP and SQUEEZE attributes are incompatible. You should not apply both of them to the same report component.

- The OVER attribute is not supported with WRAP.

- The WRAP attribute is for alphanumeric columns, and is not supported for text (TX) columns.

Justifying Report Columns

**How to:** Justify a Report Column

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 1053.

**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, that is needed to identify the report component that you are formatting. See Identifying a Report Component in a WebFOCUS StyleSheet on page 849 for more information on how to specify different report components.
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 specifies that the column will be left justified.

RIGHT specifies that the column will be right justified.

CENTER specifies that the column will be centered.

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
Field-Based Reformatting

How to:
Define and Apply a Format Field

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.

<table>
<thead>
<tr>
<th>Product</th>
<th>Code</th>
<th>Vendor Name</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscotti</td>
<td>F102</td>
<td>Delancey Bakeries</td>
<td>17.00</td>
</tr>
<tr>
<td>Coffee Grinder</td>
<td>G110</td>
<td>Appliance Craft</td>
<td>125.00</td>
</tr>
<tr>
<td>Coffee Pot</td>
<td>G121</td>
<td>Appliance Craft</td>
<td>140.00</td>
</tr>
<tr>
<td>Croissant</td>
<td>F103</td>
<td>West Side Bakers</td>
<td>28.00</td>
</tr>
<tr>
<td>French Roast</td>
<td>B142</td>
<td>European Specialties</td>
<td>81.00</td>
</tr>
<tr>
<td>Hazelnut</td>
<td>B141</td>
<td>Coffee Connection</td>
<td>58.00</td>
</tr>
<tr>
<td>Kona</td>
<td>B144</td>
<td>Evelina Imports, Ltd</td>
<td>76.00</td>
</tr>
<tr>
<td>Mug</td>
<td>G100</td>
<td>NY Ceramic Supply</td>
<td>26.00</td>
</tr>
<tr>
<td>Scone</td>
<td>F101</td>
<td>Ridgewood Bakeries</td>
<td>13.00</td>
</tr>
<tr>
<td>Thermos</td>
<td>G104</td>
<td>ThermoTech, Inc</td>
<td>96.00</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>SALES</th>
<th>DEALER_COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGLAND</td>
<td>12,000.0</td>
<td>37,853.0</td>
</tr>
<tr>
<td>FRANCE</td>
<td>.00</td>
<td>4,631.00</td>
</tr>
<tr>
<td>ITALY</td>
<td>30,200.0</td>
<td>41,235.00</td>
</tr>
<tr>
<td>JAPAN</td>
<td>78,030</td>
<td>5,512</td>
</tr>
<tr>
<td>W GERMANY</td>
<td>88,190.0</td>
<td>54,563.00</td>
</tr>
</tbody>
</table>
Data Visualization

How to:  
Incorporate Data Visualization Formatting Attributes

Reference:  
Formatting Options for Data Visualization Bar Graphs

To make HTML 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. For details see Displaying Report Data on page 43.

Reference:  Formatting Options for Data Visualization Bar Graphs

You can specify optional formatting attributes for data visualization bar graphs in the GRAPHTYPE declaration. The following table lists the formatting attributes and a description of each:

<table>
<thead>
<tr>
<th>Formatting Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRAPHCOLOR</td>
<td>Sets the color of the bar graphs.</td>
</tr>
<tr>
<td>GRAPHLENGTH</td>
<td>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. 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.</td>
</tr>
<tr>
<td>GRAPHWIDTH</td>
<td>Sets the width of the bar graphs. The width value is expressed in the current units. See GRAPHLENGTH, above, for more information about units.</td>
</tr>
</tbody>
</table>

Note: Graphcolor=orange is not supported.
**Syntax:**

How to Incorporate Data Visualization Formatting Attributes

```
GRAPHTYPE=DATA, {COLUMN|ACROSSCOLUMN|FIELD}=identifier,
           [GRAPHCOLOR=graphcolor,] [GRAPHLENGTH=lengthvalue,]
           [GRAPHWIDTH=widthvalue,] $
```

Where:

- **GRAPHCOLOR**
  - Specifies the color of the bar graphs. Black is the default color if you omit this attribute from the declaration.

- **graphcolor**
  - Is one of the following values:

<table>
<thead>
<tr>
<th>AQUA</th>
<th>NAVY</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLACK</td>
<td>OLIVE</td>
</tr>
<tr>
<td>BLUE</td>
<td>PURPLE</td>
</tr>
<tr>
<td>FUCHSIA</td>
<td>RED</td>
</tr>
<tr>
<td>GRAY</td>
<td>SILVER</td>
</tr>
<tr>
<td>GREEN</td>
<td>TEAL</td>
</tr>
<tr>
<td>LIME</td>
<td>WHITE</td>
</tr>
<tr>
<td>MAROON</td>
<td>YELLOW</td>
</tr>
</tbody>
</table>

- **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.

- **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.
Displaying Multi-Line An and AnV Fields

**How to:**
Display An and AnV Fields Containing Line Breaks on Multiple Lines

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.
- **LFCR** 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:** The report output must be formatted as PDF or PostScript.
**Example:** Displaying an Alphanumeric Field With Line Breaks in a PDF Report

The following request defines an alphanumeric named ANLB field with a semi-colon (in an EDCDIC environment) or a circumflex (in an ASCII environment) in the middle. The CTRAN function then replaces the semi-colon 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:

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>ANLBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLACKWOOD</td>
<td>THIS IS AN An FIELD WITH A LINE BREAK.</td>
</tr>
</tbody>
</table>
Example: Using an Alphanumeric Field With a Line Break in a Subfoot

The following request defines an alphanumeric named ANLB field with a semi-colon in the middle. The CTRAN function then replaces the semi-colon (in an EDCDIC environment) or a circumflex (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:

```
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 output is:

```
LAST_NAME   FIRST_NAME
BLACKWOOD    ROSEMARIE

THIS IS AN An FIELD
WITH A LINE BREAK.
```
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.

**Topics:**
- Content Analysis: Determining Graphing Objectives
- The GRAPH Command
- 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/connection that you want them to make.

3. **Begin developing your graph.**

4. **Refine your graph:**
   - Are the labels meaningful/useful?
   - How can the data be organized in a meaningful way? Consider customizing the scales you use with your graph.
The GRAPH Command

In this section:
- Similarities Between GRAPH and TABLE
- Differences Between GRAPH and TABLE

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. For an example, see Converting a TABLE Request to a GRAPH Request on page 1248.

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 783.

- 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:

<table>
<thead>
<tr>
<th>Coffee Types</th>
<th>Ordered Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>French Roast</td>
<td>285689</td>
</tr>
<tr>
<td>Hazelnut</td>
<td>100427</td>
</tr>
<tr>
<td>Kona</td>
<td>61498</td>
</tr>
</tbody>
</table>

The same request with a GRAPH command in place of the TABLE command is:

```sql
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:

**Sample Graph**

![Graph showing the ordered units for Coffee Types: French Roast, Hazelnut, and Kona]
Selecting a Graph Type

In this section:
Graph Types
Selecting Scales
Determining Graph Styles With Display Commands and Sort Phrases
Determining Graph Styles Using LOOKGRAPH

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 1250) 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 1261.

- **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

How to:
Select 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:

```plaintext
ON GRAPH SET GRAPHSTYLE *
setY1LogScale(value);
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**

**How to:**

Create a Line Graph
Create a Horizontal Bar Graph
Create a Vertical Bar Graph
Create a Pie Graph
Create a Scatter Graph

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 81.

By default, a particular combination of display commands and sort phrases determines the graph format. The combinations are:

<table>
<thead>
<tr>
<th>Graph Type</th>
<th>Display Command and Sort Phrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line graph</td>
<td>PRINT A {ACROSS</td>
</tr>
<tr>
<td>Vertical bar graph</td>
<td>SUM A ACROSS B (where B is alphanumeric)</td>
</tr>
<tr>
<td>Horizontal bar graph</td>
<td>SUM A BY B</td>
</tr>
<tr>
<td>Pie graph</td>
<td>SET LOOKGRAPH=PIE SUM A {ACROSS</td>
</tr>
<tr>
<td>Scatter graph (without connecting lines)</td>
<td>PRINT A ACROSS B (where B is numeric)</td>
</tr>
<tr>
<td>Scatter graph (with connecting lines)</td>
<td>SUM A ACROSS B (where B is numeric)</td>
</tr>
</tbody>
</table>

You can override the default graph format using the LOOKGRAPH parameter. For details, see *Determining Graph Styles Using LOOKGRAPH* on page 1261.
**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 1261.

**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"
PRINT QUANTITY
ACROSS PRODUCT_DESC
WHERE PRODUCT_DESC EQ 'French Roast' OR 'Hazelnut' OR 'Kona'
END
```
The output is:

**Sample Line Graph**

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.
Selecting a Graph Type

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:

![Sample Horizontal Bar Graph]

**Example:** Creating a Vertical Bar Graph

The following illustrates how to create a vertical bar graph:

```webfocus
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
```
**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:

![Sample Pie Chart](image)

**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:

![Sample Scatter Graph](image-url)
Determining Graph Styles Using LOOKGRAPH

How to:
Specify a Graph Style Using LOOKGRAPH

Reference:
Style Options for Line Graphs
Style Options for Bar Graphs
Style Options for Pie Graphs
Style Options for Scatter Graphs
Style Options for Three-Dimensional Graphs
Style Options for Area Graphs
Style Options for Stock Charts
Style Options for Polar Charts
Style Options for Radar Charts
Style Options for Bubble Charts
Style Options for Spectral Charts
Other Graph Types

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 1262.
- Style Options for Bar Graphs on page 1263.
Selecting a Graph Type

- **Style Options for Pie Graphs** on page 1264.
- **Style Options for Scatter Graphs** on page 1264.
- **Style Options for Three-Dimensional Graphs** on page 1265.
- **Style Options for Area Graphs** on page 1266.
- **Style Options for Stock Charts** on page 1267.
- **Style Options for Polar Charts** on page 1268.
- **Style Options for Radar Charts** on page 1268.
- **Style Options for Bubble Charts** on page 1269.
- **Style Options for Spectral Charts** on page 1269.
- **Other Graph Types** on page 1269.

**Reference: Style Options for Line Graphs**

Choose one of the following LOOKGRAPH values to change the style of connected point plots:

<table>
<thead>
<tr>
<th>SET LOOKGRAPH=</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE</td>
<td>A vertical connected point plot graph.</td>
</tr>
<tr>
<td>HLINE</td>
<td>A horizontal connected point plot graph.</td>
</tr>
<tr>
<td>HLINE2</td>
<td>A horizontal connected point plot graph with two axes.</td>
</tr>
<tr>
<td>HLINE2S</td>
<td>A horizontal connected point plot graph with two separate axes.</td>
</tr>
<tr>
<td>HLINSTK</td>
<td>A stacked horizontal connected point plot graph.</td>
</tr>
<tr>
<td>HLINSTK2</td>
<td>A stacked horizontal connected point plot graph with two axes.</td>
</tr>
<tr>
<td>HLINSTK2S</td>
<td>A stacked horizontal connected point plot graph with two separate axes.</td>
</tr>
<tr>
<td>HLNSTKPC</td>
<td>A stacked horizontal connected point plot graph showing percentages.</td>
</tr>
<tr>
<td>VLINE</td>
<td>A vertical connected point plot graph.</td>
</tr>
<tr>
<td>VLINE2</td>
<td>A vertical connected point plot graph with two axes.</td>
</tr>
<tr>
<td>SET LOOKGRAPH=</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>VLINE2S</td>
<td>A vertical connected point plot graph with two separate axes.</td>
</tr>
<tr>
<td>VLINSTK</td>
<td>A stacked vertical connected point plot graph.</td>
</tr>
<tr>
<td>VLINSTK2</td>
<td>A stacked vertical connected point plot graph with two axes.</td>
</tr>
<tr>
<td>VLNSTK2S</td>
<td>A stacked vertical connected point plot graph with two separate axes.</td>
</tr>
<tr>
<td>VLNSTKPC</td>
<td>A stacked vertical connected point plot graph showing percentages.</td>
</tr>
</tbody>
</table>

**Reference:**  **Style Options for Bar Graphs**

Choose one of the following LOOKGRAPH values to change the style of bar graphs:

<table>
<thead>
<tr>
<th>SET LOOKGRAPH=</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAR</td>
<td>A bar graph with the bars displayed beside each other.</td>
</tr>
<tr>
<td>STACK</td>
<td>A bar graph with stacked bars.</td>
</tr>
<tr>
<td>VBAR</td>
<td>A vertical bar graph.</td>
</tr>
<tr>
<td>VBAR2AX</td>
<td>A vertical bar graph with two axes.</td>
</tr>
<tr>
<td>VBAR2AXS</td>
<td>A vertical bar graph with two separate axes.</td>
</tr>
<tr>
<td>VBRSTK1</td>
<td>A stacked vertical bar graph.</td>
</tr>
<tr>
<td>VBRSTK2</td>
<td>A stacked vertical bar graph with two axes.</td>
</tr>
<tr>
<td>VBRSTK2S</td>
<td>A stacked vertical bar graph with two separate axes.</td>
</tr>
<tr>
<td>VBRSTKPC</td>
<td>A stacked vertical bar graph that shows percentages.</td>
</tr>
<tr>
<td>HBAR</td>
<td>A horizontal bar graph.</td>
</tr>
<tr>
<td>HBAR2AX</td>
<td>A horizontal bar graph with two axes.</td>
</tr>
<tr>
<td>HBAR2AXS</td>
<td>A horizontal bar graph with two separate axes.</td>
</tr>
<tr>
<td>HBRSTK1</td>
<td>A stacked horizontal bar graph.</td>
</tr>
</tbody>
</table>
Selecting a Graph Type

**Reference: Style Options for Pie Graphs**

Choose one of the following LOOKGRAPH values to change the style of pie graphs:

<table>
<thead>
<tr>
<th>SET LOOKGRAPH=</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIE</td>
<td>A pie graph.</td>
</tr>
<tr>
<td>PIESINGL</td>
<td>A single pie graph.</td>
</tr>
<tr>
<td>PIEMULTI</td>
<td>Multiple pie graphs.</td>
</tr>
<tr>
<td>PIERING</td>
<td>A ring-shaped pie graph.</td>
</tr>
<tr>
<td>PIEMULPR</td>
<td>Multiple, ring-shaped pie graphs of proportional size.</td>
</tr>
<tr>
<td>PIEMULTP</td>
<td>Multiple pie graphs of proportional size.</td>
</tr>
<tr>
<td>PIEMULTR</td>
<td>Multiple, ring-shaped pie graphs.</td>
</tr>
</tbody>
</table>

**Reference: Style Options for Scatter Graphs**

Choose one of the following LOOKGRAPH values to change the style of scatter graphs:

<table>
<thead>
<tr>
<th>SET LOOKGRAPH=</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCATTER</td>
<td>Produces a scatter graph.</td>
</tr>
<tr>
<td>SCATTERD</td>
<td>A dual scatter graph. Values from an additional data set are displayed on a second value (Y) axis.</td>
</tr>
<tr>
<td>SCATTRLS</td>
<td>A scatter graph that labels each data point with its exact numeric value.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th><code>SET LOOKGRAPH=</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCATTRLD</td>
<td>A dual scatter graph that labels each data point with its exact numeric value.</td>
</tr>
<tr>
<td>3DAREAG</td>
<td>A three-dimensional connected group area chart.</td>
</tr>
<tr>
<td>3DAREAS</td>
<td>A three-dimensional connected series area chart.</td>
</tr>
<tr>
<td>3DBAR</td>
<td>A two-dimensional bar graph with three-dimensional bars.</td>
</tr>
<tr>
<td>3D_BAR</td>
<td>A three-dimensional chart with bars.</td>
</tr>
<tr>
<td>3DCUBE</td>
<td>A three-dimensional bar graph in which all data points are blocks of identical size, hovering at the position that shows their data value.</td>
</tr>
<tr>
<td>3DGROUP</td>
<td>A three-dimensional chart with bars.</td>
</tr>
<tr>
<td>3DOCTAGN</td>
<td>A three-dimensional bar graph with octagon-shaped bars that have no roots.</td>
</tr>
<tr>
<td>3DPYRAMD</td>
<td>A three-dimensional pyramid chart.</td>
</tr>
<tr>
<td>3DRIBBNG</td>
<td>A three-dimensional connected group ribbon chart.</td>
</tr>
<tr>
<td>3DRIBBNS</td>
<td>A three-dimensional connected series ribbon chart.</td>
</tr>
<tr>
<td>3DSPHERE</td>
<td>A three-dimensional bar graph in which all data points are spheres of identical size, hovering at the position that shows their data value.</td>
</tr>
<tr>
<td>3DSTACK</td>
<td>A two-dimensional stack chart with three-dimensional type bars.</td>
</tr>
<tr>
<td>3DSURFCE</td>
<td>A three-dimensional surface chart that graphs all data points as a three-dimensional surface, like a rolling wave.</td>
</tr>
</tbody>
</table>
**Selecting a Graph Type**

<table>
<thead>
<tr>
<th><code>SET LOOKGRAPH=</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>3DSURFHC</code></td>
<td>A three-dimensional honeycomb surface chart that graphs all data points as a three-dimensional surface using a honeycomb effect.</td>
</tr>
<tr>
<td><code>3DSURFSD</code></td>
<td>A three-dimensional surface chart with sides that graphs all data points as a three-dimensional surface with solid sides.</td>
</tr>
</tbody>
</table>

**Reference: Style Options for Area Graphs**

Choose one of the following `LOOKGRAPH` values to change the style of area graphs:

<table>
<thead>
<tr>
<th><code>SET LOOKGRAPH=</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>VAREA</code></td>
<td>A vertical area graph.</td>
</tr>
<tr>
<td><code>VARESTK</code></td>
<td>A stacked vertical area graph.</td>
</tr>
<tr>
<td><code>VAREAR2</code></td>
<td>A vertical area graph with two axes.</td>
</tr>
<tr>
<td><code>VARESTK2</code></td>
<td>A stacked vertical area graph with two axes.</td>
</tr>
<tr>
<td><code>VARESTKP</code></td>
<td>A stacked vertical area graph that shows percentages.</td>
</tr>
<tr>
<td><code>HAREA</code></td>
<td>A horizontal area graph.</td>
</tr>
<tr>
<td><code>HAREAR2</code></td>
<td>A horizontal area graph with two axes.</td>
</tr>
<tr>
<td><code>HARESTK</code></td>
<td>A stacked horizontal area graph.</td>
</tr>
<tr>
<td><code>HARESTK2</code></td>
<td>A stacked horizontal area graph with two axes.</td>
</tr>
<tr>
<td><code>HARESTKP</code></td>
<td>A stacked horizontal area graph that shows percentages.</td>
</tr>
</tbody>
</table>
Reference: Style Options for Stock Charts

Choose one of the following LOOKGRAPH values to change the style of stock charts:

<table>
<thead>
<tr>
<th>SET LOOKGRAPH=</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STOCK</td>
<td>A stock chart.</td>
</tr>
<tr>
<td>STOCKH</td>
<td>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. 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.</td>
</tr>
<tr>
<td>STOCKHB</td>
<td>A bipolar high-low stock chart. Values from different data sets are displayed on separate poles.</td>
</tr>
<tr>
<td>STOCKHD</td>
<td>A dual high-low stock chart. Values from an additional data set are displayed on a second value (Y) axis.</td>
</tr>
<tr>
<td>STOCKHCL</td>
<td>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.</td>
</tr>
<tr>
<td>STOCKHCB</td>
<td>A bipolar high-low-close stock chart. Values from different data sets are displayed on separate poles. 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.</td>
</tr>
<tr>
<td>STOCKHCD</td>
<td>A dual high-low-close stock chart. Values from an additional data set are displayed on a second value (Y) axis.</td>
</tr>
<tr>
<td>STOCKHOC</td>
<td>A high-low-open-close stock chart. 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.</td>
</tr>
<tr>
<td>STOCKHOB</td>
<td>A bipolar high-low-open-close stock chart. Values from different data sets are displayed on separate poles.</td>
</tr>
</tbody>
</table>
### Style Options for Polar Charts

Choose one of the following `LOOKGRAPH` values to change the style of polar charts:

<table>
<thead>
<tr>
<th>SET LOOKGRAPH=</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POLAR</td>
<td>A polar chart that displays data points on a circle.</td>
</tr>
<tr>
<td>POLAR2</td>
<td>A dual polar chart. Values from an additional data set are displayed on a second value (Y) axis.</td>
</tr>
</tbody>
</table>

### Style Options for Radar Charts

Choose one of the following `LOOKGRAPH` values to change the style of radar charts:

<table>
<thead>
<tr>
<th>SET LOOKGRAPH=</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADARA</td>
<td>A radar area chart.</td>
</tr>
<tr>
<td>RADARL</td>
<td>A radar line chart.</td>
</tr>
<tr>
<td>RADARL2</td>
<td>A dual radar line chart. Values from an additional data set are displayed on a second value (Y) axis.</td>
</tr>
</tbody>
</table>
**Reference: Style Options for Bubble Charts**

Choose one of the following `LOOKGRAPH` values to change the style of bubble charts:

<table>
<thead>
<tr>
<th>SET LOOKGRAPH=</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUBBLE</td>
<td>A bubble chart.</td>
</tr>
<tr>
<td>BUBBLED</td>
<td>A bubble chart with a dual axis.</td>
</tr>
<tr>
<td>BUBBLEDL</td>
<td>A bubble chart with a dual axis and labels.</td>
</tr>
<tr>
<td>BUBBLEL</td>
<td>A bubble chart with labels.</td>
</tr>
</tbody>
</table>

**Reference: Style Options for Spectral Charts**

Choose one of the following `LOOKGRAPH` values to change the style of spectral charts:

<table>
<thead>
<tr>
<th>SET LOOKGRAPH=</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPECTRAL</td>
<td>A spectral map chart. This is a chart with a row or column matrix of markers that is colored according to the data values.</td>
</tr>
</tbody>
</table>

**Reference: Other Graph Types**

<table>
<thead>
<tr>
<th>SET LOOKGRAPH=</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GANTT</td>
<td>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.</td>
</tr>
<tr>
<td>POSITION</td>
<td>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.</td>
</tr>
<tr>
<td>VWATERFL</td>
<td>Vertical waterfall graph.</td>
</tr>
<tr>
<td>HWATERFL</td>
<td>Horizontal waterfall graph.</td>
</tr>
<tr>
<td>PARETO</td>
<td>Displays data following Pareto 80:20 rule. Pareto charts require only one display field.</td>
</tr>
</tbody>
</table>
Selecting Values for the X and Y Axes

<table>
<thead>
<tr>
<th>SET LOOKGRAPH=</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULTI3Y</td>
<td>Stacks charts in order to make it easier to read, analyze and manage them.</td>
</tr>
<tr>
<td>MULTI4Y</td>
<td></td>
</tr>
<tr>
<td>MULTI5Y</td>
<td></td>
</tr>
</tbody>
</table>

In this section:
- Hiding the Display of a Y-Axis Field
- Interpolating X and Y Axis Values Using Linear Regression

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 1272.

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 1272.

- Temporarily hide the display of a Y-axis field. For details, see Hiding the Display of a Y-Axis Field on page 1271.

- Interpolate X and Y axis values using linear regression. For details, see Interpolating X and Y Axis Values Using Linear Regression on page 1271.

**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
```
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:

![Graph with linear regression line](image)

```
y = 1 * x + 559
```

### Creating Multiple Graphs

**In this section:**
- Merging Multiple Graphs
- Displaying Multiple Graphs in Columns

You can create multiple graphs by selecting a second horizontal category (X-axis).

The number of graphs created depends on the number of values in the field you designate in the sort (BY, ACROSS) phrases. 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.

Multiple graphs can be displayed in either merged format or in columns. For details, see *Merging Multiple Graphs* on page 1273 and *Displaying Multiple Graphs in Columns* on page 1274.

**Merging Multiple Graphs**

**How to:**
Merge Multiple Graphs

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.

To do this, use the SET command GRMERGE.

**Syntax:**

**How to Merge Multiple Graphs**

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

where:

- **ON**
  - Turns on the merge graph option.
- **OFF**
  - Turns off the merge graph option. This is the default.

**Example:**

**Merging Multiple Graphs**

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
```
Displaying Multiple Graphs in Columns

**How to:**

Display 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.

\[
\text{SET GRWIDTH=2} \\
\text{GRAPH FILE GGORDER} \\
\text{SUM UNIT_PRICE ORDER_NUMBER} \\
\text{ACROSS PRODUCT_ID} \\
\text{BY PACKAGE_TYPE} \\
\text{END}
\]

The output is:

Plotting Dates in Graphs

In this section:
- Basic Date Support for X and Y Axes
- Formatting Dates for Y-Axis Values

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:

![Graph example](image_url)
Basic Date Support for X and Y Axes

Reference: Date Support Limitations

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 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):

```plaintext
setTextFormatPreset(getX1Label(),xx); // for X Axis
setTextFormatPreset(getY1Label(),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:

```plaintext
setDataTextFormat(xx);
```

For more information, see Customizing Graphs Using the Graph API (ON GRAPH SET GRAPHSTYLE *) on page 1298.

- 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:

<table>
<thead>
<tr>
<th>Display Format</th>
<th>Corresponding WebFOCUS Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>yy/mm/dd</td>
<td>YMD</td>
</tr>
<tr>
<td>yy/mm</td>
<td>YM</td>
</tr>
<tr>
<td>mm/dd/yyyy</td>
<td>MDYY</td>
</tr>
<tr>
<td>mm/dd/yy</td>
<td>MDY</td>
</tr>
<tr>
<td>mm/dd</td>
<td>MD</td>
</tr>
</tbody>
</table>

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 143.
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

How to:
Display Missing 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.
Displaying Missing Data Values in a Graph

- **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 697.

**Syntax:**

**How to Display Missing Values in a Graph**

```
GRAPH FILE filename
.
.
.
SET VZERO={ON|OFF}
ON GRAPH SET GRAPHSTYLE *
API call
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.

```plaintext
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 GRAPHSTYLE *
setTextRotation(getO1Label(),0);
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.

```plaintext
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 GRAPHSTYLE *
setFillMissingData(0);
setTextRotation(get01Label(),0);
ENDSTYLE
END
```

The output is:

![Graph showing missing data values]

The CURR_SAL for Seay is represented as a gap, as are the RAISE values for Bryant and Huntley.
**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 GRAPHPEDIT=SERVER
SET GRID=ON
SET VZERO=OFF
GRAPH FILE MSFATIA
SUM CUR_SAL
RAISE
ACROSS LAST_NAME
ON GRAPH SET GRAPHSTYLE *
setFillMissingData(1);
setTextRotation(getO1Label(),0);
ENDSTYLE
END
```

The output is:

![Graph showing missing values as a dotted line to zero](image)

- **CUR_SAL**
- **RAISE**
**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 GRAPHSTYLE *
setFillMissingData(2);
setTextRotation(getO1Label(),0);
ENDSTYLE
END
```

The output is:

![Graph Example](image)

**Applying Conditional Styling to a Graph**

**How to:**

Apply 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 849.

**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 1205.

**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 315.

**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

**Syntax: How to Link a Graph to Other Resources**

```
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 1205.

**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 849.

**FOCEXEC=fex**
- Identifies the file name of the linked procedure to run when a user selects the report object.

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 or URL.
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 315.

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
```
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 519.

Adding Labels to a Graph

**In this section:**

Adding Vertical (Y-axis) and Horizontal (X-axis) 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 1053.
**Note:** If your graph labels or legends are not appearing correctly when you run your graph, see *How to Change Color Settings* on page 1306 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
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:

**Total Coffee Orders: 447614**

**For year-end orders**

**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

- **In this section:**
  - Setting the Graph Height and Width
  - Customizing Graphs Using SET Parameters
  - Setting Fixed Scales for the X-Axis
  - Setting Fixed Scales for the Y-Axis
  - Customizing Graphs With the Graph Editor
  - Customizing Graphs Using the Graph API (ON GRAPH SET GRAPHSTYLE *)

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 1053.

### Setting the Graph Height and Width

- **How to:**
  - Set the Graph Width
  - Set the Graph Height

The width (or horizontal axis) of each graph, which includes any surrounding text, is automatically set to 770 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
```

where:

```
 nn
```

Is a numeric value between 20 and 985 pixels.

**Syntax:** How to Set the Graph Height

```
SET VAXIS=nn
```

where:

```
 nn
```

Is a numeric value between 20 and 518 pixels.

### Customizing Graphs Using SET Parameters

#### How to:

Use SET Parameters With GRAPH Requests

#### Reference:

Values and Functions of SET Parameters for Graphs

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 1293.

**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

<table>
<thead>
<tr>
<th>Graph SET Parameter</th>
<th>Values</th>
<th>Parameter Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>AUTOTICK</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>BARNUMB</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>BSTACK</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>GRAPHEDIT</td>
<td>graphedit</td>
<td>Determines whether the Graph Editor is enabled at run time. For details, see &quot;Customizing Graphs With the Graph Editor&quot; on page 1297.</td>
</tr>
<tr>
<td>Graph SET Parameter</td>
<td>Values</td>
<td>Parameter Function</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------</td>
<td>---------------------</td>
</tr>
<tr>
<td>GRID</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>HAUTO</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>HAXIS</td>
<td>20–770</td>
<td></td>
</tr>
<tr>
<td>HCLASS</td>
<td>nnn</td>
<td></td>
</tr>
<tr>
<td>HISTOGRAM</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>HMAX</td>
<td>nnn</td>
<td></td>
</tr>
<tr>
<td>HMIN</td>
<td>nnn</td>
<td></td>
</tr>
<tr>
<td>HSTACK</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>HTICK</td>
<td>nnn</td>
<td></td>
</tr>
<tr>
<td>LOOKGRAPH</td>
<td>option</td>
<td></td>
</tr>
<tr>
<td>PIE</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>Graph SET Parameter</td>
<td>Values</td>
<td>Parameter Function</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------</td>
<td>--------------------</td>
</tr>
<tr>
<td>VAUTO</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>VAXIS</td>
<td>20–405</td>
<td></td>
</tr>
<tr>
<td>VCLASS</td>
<td>nnn</td>
<td></td>
</tr>
<tr>
<td>VGRID</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>VMAX</td>
<td>nnn</td>
<td></td>
</tr>
<tr>
<td>VMIN</td>
<td>nnn</td>
<td></td>
</tr>
<tr>
<td>VTICK</td>
<td>nnn</td>
<td></td>
</tr>
<tr>
<td>VZERO</td>
<td>ON</td>
<td>OFF</td>
</tr>
</tbody>
</table>
Setting Fixed Scales for the X-Axis

**How to:**
Set 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*

```plaintext
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

**How to:**
Set 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 With the Graph Editor

How to:
Enable the Graph Editor

The Graph Editor is an interactive graphical tool that allows you to modify a graph without having to edit the code manually. The Graph Editor applies and displays the changes you make to the graph.

When it is turned on in a procedure, you will be able to invoke the Graph Editor and make changes at run time without affecting your procedure.

For details on using the Graph Editor, see the Creating Charts With Graph Tools manual.
**Syntax:**

**How to Enable the Graph Editor**

The SET GRAPHEdit command must be placed before the GRAPH FILE command in a procedure.

```
SET GRAPHEdit = graphedit
```

where:

`graphedit`

Determines whether the Graph Editor is enabled at run time. Possible values are:

- **ON** activates the Graph Editor. A button is displayed in the upper-right corner of the window that contains the graph. Click this button to access the Graph Editor.

- **OFF** is where the Graph Editor will not be enabled at run time and a Java-rendered graph is returned.

- **SERVER** generates a server-side graph. A GIF graph is created on the Web server and the image is returned to the client in the browser.

This is the default.

**Customizing Graphs Using the Graph API (ON GRAPH SET GRAPHESTYLE *)**

**How to:**

Customize a Graph Using the Graph API

You can further enhance your graph output by manually adding API calls inside the ON GRAPH SET GRAPHESTYLE * and ENDSTYLE commands in the GRAPH request. When you save changes in the Graph Editor (if invoked from the Graph Assistant), the corresponding API calls will be written to the graph StyleSheet.

For reference information about the Graph API, see the *WebFOCUS Graphics* manual.
**Syntax:** How to Customize a Graph Using the Graph API

```
GRAPH FILE filename
<graph commands>
ON GRAPH SET GRAPHSTYLE *
<API calls>
ENDSTYLE
[ON GRAPH SET STYLE *
<WEBFOCUS StyleSheet commands>
ENDSTYLE]
END
```

where:

- **filename**
  - Specifies a data source for the graph.

- **API calls**
  - For reference information about the Graph API, see the *WebFOCUS Graphics* manual.

- **WEBFOCUS StyleSheet commands**
  - For details on StyleSheet commands, see *Using an External Cascading Style Sheet* on page 891.

**Example:** Customizing Graphs Using the Graph API

The following annotated example illustrates how to customize a graph using `ON GRAPH SET GRAPHSTYLE *`. 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 GRAPHSTYLE *
1. setLegendMarkerPosition(4);
2. setO1LabelRotate(0);
3. setTitleString("Sales Report");
4. setTextJustHoriz(getTitle(),1);
ENDSTYLE
ON GRAPH SET STYLE *
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:

![Sales Report Chart]

**Saving a Graph as an Image File**

**In this section:**

Saving a Graph as an Image File Using GRAPHSERVURL

You can save graph output to an image file using the GRAPHSERVURL 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 GRAPHSERVURL* on page 1301.
Saving a Graph as an Image File Using GRAPHSERVURL

**How to:**

Save a Graph as an Image File Using GRAPHSERVURL
Save a Graph as an Image File

**Reference:**

Usage Notes for Saving a Graph

The GRAPHSERVURL 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 GRAPHSERVURL 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 GRAPHSERVURL**

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 GRAPHSERVURL in the procedure to the URL that invokes the WebFOCUS Graph Servlet. For example,

   ```
   SET GRAPHSERVURL=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 1302.

3. Run the procedure directly on the WebFOCUS Reporting Server, using a browser, or from Developer Studio.
How to Save a Graph as an Image File

Syntax:

```
[FILEDEF filename DISK drive:\...\filename.fmt]
SET GRAPHSERVURL= graph_servlet_URL
GRAPH FILE file
graph commands
ON GRAPH HOLD AS filename FORMAT fmtEND
```

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, JPEG, or JPG.

**graph_servlet_URL**

Is the URL to invoke the WebFOCUS Graph Servlet.

**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 GRAPHSERVURL= http://localhost/ibi_apps/IBIGraphServlet
GRAPH FILE CENTORD
SUM LINEPRICE
ACROSS PLANTLING AS 'Plant'
ON GRAPH HOLD AS PLANT FORMAT GIF
END

TABLE FILE CENTORD
SUM LINEPRICE
BY PLANTLING 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 GRAPHSERVURL 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:

![Launch Page Screenshot](image)

You can now run the report from Developer Studio or 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:

![Report Screenshot](image)

**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, JPEG, 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 .jpeg, .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

**How to:**

- Print Your Graph
- Change Color Settings
- Send Graph Output Directly to a Printer

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 1306 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.

Topics:
- Reporting With FML
- Creating Rows From Data
- Supplying Data Directly in a Request
- Performing Inter-Row Calculations
- Referring to Rows in Calculations
- Referring to Columns in Calculations
- Referring to Rows and Columns in Calculations
- Referring to Cells in Calculations
- Using Functions in RECAP Calculations
- Inserting Rows of Free Text
- Adding a Column to an FML Report
- Creating a Recursive Model
- Reporting Dynamically From a Hierarchy
- Customizing a Row Title
- Formatting an FML Report
- Suppressing the Display of Rows
- Saving and Retrieving Intermediate Report Results
- 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 be used to 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/column formatting:** You can specify results in a row-by-row, column-by-column fashion (see *Performing Inter-Row Calculations* on page 1324).

- **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 (see *Saving and Retrieving Intermediate Report Results* on page 1391).

- **Inline data entry:** FML enables you to specify constants from within the procedure, in addition to the data values retrieved from your data source (see *Supplying Data Directly in a Request* on page 1323).

- **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 (see *Creating a Recursive Model* on page 1346).

- **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. See *Reporting Dynamically From a Hierarchy* on page 1348.
**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 OVER
  4. Recap UTPNET=UTP-UTPAD; AS 'TOTAL PLANT-NET' OVER
     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 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:

### COMPARATIVE ASSET SHEET

<table>
<thead>
<tr>
<th></th>
<th>1983</th>
<th>1982</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UTILITY PLANT</strong></td>
<td>1,430,903</td>
<td>1,294,611</td>
</tr>
<tr>
<td><strong>LESS ACCUMULATED DEPRECIATION</strong></td>
<td>249,504</td>
<td>213,225</td>
</tr>
<tr>
<td><strong>TOTAL PLANT-NET</strong></td>
<td>1,181,399</td>
<td>1,081,386</td>
</tr>
<tr>
<td><strong>INVESTMENTS</strong></td>
<td>818</td>
<td>5,639</td>
</tr>
<tr>
<td><strong>CURRENT ASSETS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CASH</strong></td>
<td>4,938</td>
<td>4,200</td>
</tr>
<tr>
<td><strong>ACCOUNTS RECEIVABLE-NET</strong></td>
<td>28,052</td>
<td>23,758</td>
</tr>
<tr>
<td><strong>INTEREST RECEIVABLE</strong></td>
<td>15,945</td>
<td>10,206</td>
</tr>
<tr>
<td><strong>FUEL INVENTORY</strong></td>
<td>35,158</td>
<td>45,643</td>
</tr>
<tr>
<td><strong>MATERIALS AND SUPPLIES</strong></td>
<td>16,099</td>
<td>12,909</td>
</tr>
<tr>
<td><strong>OTHER</strong></td>
<td>1,264</td>
<td>1,743</td>
</tr>
<tr>
<td><strong>TOTAL CURRENT ASSETS</strong></td>
<td>101,456</td>
<td>98,459</td>
</tr>
<tr>
<td><strong>DEFERRED DEBITS</strong></td>
<td>30,294</td>
<td>17,459</td>
</tr>
<tr>
<td><strong>TOTAL ASSETS</strong></td>
<td>1,313,967</td>
<td>1,202,943</td>
</tr>
</tbody>
</table>

*** 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:

```plaintext
FOR fieldname [NOPRINT]
value [OR value OR...] [AS 'text'] [LABEL label] OVER
.
[value [OR value ...] [AS 'text'] [LABEL label]
END
```

where:

- **fieldname**
  - Is a field name in the data source.

- **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, etc,) 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). See *How to Use the Same FOR Field Value in Multiple Rows* on page 1319.

See the *Using Functions* manual for information about the FMLFOR, FMLLIST, and FMLINFO functions that return the tag values used in an FML request.
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:

<table>
<thead>
<tr>
<th>CHART OF ACCOUNTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNT</td>
</tr>
<tr>
<td>1010</td>
</tr>
<tr>
<td>1020</td>
</tr>
<tr>
<td>1030</td>
</tr>
<tr>
<td>1100</td>
</tr>
<tr>
<td>1200</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

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, etc.), 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:

<table>
<thead>
<tr>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1010  8,784</td>
</tr>
<tr>
<td>1020  4,494</td>
</tr>
<tr>
<td>1030  7,961</td>
</tr>
<tr>
<td>1100 18,829</td>
</tr>
<tr>
<td>1200 27,307</td>
</tr>
</tbody>
</table>
Creating Rows From Multiple Records

**How to:**

- Sum Values in Rows With the OR Phrase
- Identify a Range of Values With the TO Phrase
- Use Masking Characters to Retrieve Tag Values
- Use the Same FOR Field Value in Multiple Rows

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. See *How to Sum Values in Rows With the OR Phrase* on page 1316.

- The TO phrase to identify a range of tag values on which to report. See *How to Identify a Range of Values With the TO Phrase* on page 1317.

- A mask to specify a group of tag values without having to name each one. See *How to Use Masking Characters to Retrieve Tag Values* on page 1318.

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. See *How to Use the Same FOR Field Value in Multiple Rows* on page 1319.

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, etc.) 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:

```
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, etc.) 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:

```
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:

```text
SET FORMULTIPLE={ON|OFF}
```

or

```text
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.)

See Reporting Dynamically From a Hierarchy on page 1348.
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 are displayed 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:

<table>
<thead>
<tr>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>CASH ON HAND</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
</tr>
<tr>
<td>TOTAL CASH</td>
</tr>
<tr>
<td>ACCOUNTS RECEIVABLE</td>
</tr>
<tr>
<td>INVENTORY</td>
</tr>
<tr>
<td>TOTAL NON-CASH ASSETS</td>
</tr>
</tbody>
</table>
**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 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:

```sql
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):

```ml
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:

```
CURRENT ASSETS FOR REGION EAST

<table>
<thead>
<tr>
<th></th>
<th>CUR_YR</th>
<th>LAST_YR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH</td>
<td>9,511.00</td>
<td>7,903.64</td>
</tr>
<tr>
<td>ACCOUNTS RECEIVABLE</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>INVENTORY</td>
<td>.</td>
<td></td>
</tr>
</tbody>
</table>
<pre><code>                | -------- |----------|
</code></pre>
<p>| CUR_ASSET        | 9,511   | 7,903    |</p>
```

---

**Creating Rows From Data**
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'
```

See *Performing Inter-Row Calculations* on page 1324.

### Supplying Data Directly in a Request

**How to:**
Supply Data Directly in a Request

In certain cases, you may need to include additional constants (such as exchange rates, inflation rates, etc.) 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:

```plaintext
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:

<table>
<thead>
<tr>
<th></th>
<th>EUROPE</th>
<th>ENGLISH</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH--LOCAL CURRENCY</td>
<td>8,784</td>
<td>28,108</td>
</tr>
<tr>
<td>EXCHANGE RATE</td>
<td>.87</td>
<td>1.67</td>
</tr>
<tr>
<td>US_DOLLARS</td>
<td>7,642</td>
<td>46,940</td>
</tr>
</tbody>
</table>

**Performing Inter-Row Calculations**

**How to:**
Define Inter-Row Calculations

**Reference:**
Usage Notes for RECAP

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, etc). However, you may assign more meaningful labels. See *Referring to Rows in Calculations* on page 1326.
**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. See *Referring to Rows in Calculations* on page 1326.

**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. See *Using Functions in RECAP Calculations* on page 1339 and the Using Functions manual.

The expression can include references to specific rows using the default FML positional labels (R1, R2, etc), or it can refer to rows, columns, and cells using a variety of flexible notation techniques. See *Referring to Rows in Calculations* on page 1326, *Referring to Columns in Calculations* on page 1329, and *Referring to Cells in Calculations* on page 1338.

**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.
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,

  \[
  \text{RECAP TOTVAL/D6.2S=IF R1 GT R4 THEN R4 ELSE R1; AS 'REDUCED VALUE'}
  \]

  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.

Referring to Rows in Calculations

How to:
Assign an Explicit Row Label

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, etc. You can use these labels to refer to rows in RECAP expressions. (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.

- Mix positional and explicit row labels.

If you assign an explicit label, the positional label (R1, R2, etc.) 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. See Formatting an FML Report on page 1367.
**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.

- Even if you assign an explicit label, the positional label (R1, R2, etc.) 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:

```
AMOUNT
-------
CASH ON HAND       8,784
DEMAND DEPOSITS    4,494
TIME DEPOSITS      7,961
                  -------
TOTAL CASH         21,239
```
**Example:** 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:

```
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.

**Example:** 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. See *Creating Rows From Multiple Records* on page 1315.
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), to represent columns 2 through 5.
- Column addressing.
- A factor to represent every other column, or every third column, etc.
- 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:

<table>
<thead>
<tr>
<th>CURRENT</th>
<th>LAST</th>
<th>CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>YEAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CASH ON HAND</td>
<td>8,784</td>
<td>7,216</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
<td>4,494</td>
<td>3,483</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
<td>7,961</td>
<td>6,499</td>
</tr>
<tr>
<td>TOTAL CASH</td>
<td>21,239</td>
<td>17,198</td>
</tr>
</tbody>
</table>

For example, you could use the value 6,499 in another FML report by referring to column 2, row 3. See *Referring to Cells in Calculations* on page 1338.
Referring to Column Numbers in Calculations

A calculation may be performed 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

```ml
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(%)' OVER
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:

<table>
<thead>
<tr>
<th>CURRENT YEAR</th>
<th>LAST YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH ON HAND</td>
<td>8,784</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
<td>4,494</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
<td>7,961</td>
</tr>
<tr>
<td>TOTAL CASH</td>
<td>21,239</td>
</tr>
<tr>
<td>CASH GROWTH(%)</td>
<td>23.50</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th></th>
<th>NEXT_YR</th>
<th>CUR_YR</th>
<th>LAST_YR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH</td>
<td>25,992</td>
<td>21,239</td>
<td>17,198</td>
</tr>
<tr>
<td>ACCOUNTS RECEIVABLE</td>
<td>21,941</td>
<td>18,829</td>
<td>15,954</td>
</tr>
<tr>
<td>INVENTORY</td>
<td>31,522</td>
<td>27,307</td>
<td>23,329</td>
</tr>
<tr>
<td>ASSETS--ACTUAL</td>
<td></td>
<td>67,375</td>
<td>56,481</td>
</tr>
</tbody>
</table>
Referring to Column Addresses in Calculations

**How to:**

Use Column Addressing in a RECAP Expression

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,

\[ \text{COMP}=\text{FIX}(*)-\text{FIX}(*-1); \]

can refer to the change in fixed assets from one period to the next. The reference to \( \text{COMP} = \text{FIX}(*) \) is equivalent to \( \text{COMP} = \text{FIX} \).

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)/I5SC = TOTCASH(*) - TOTCASH(*+1); AS 'CHANGE IN CASH'
END
```

The output is:

<table>
<thead>
<tr>
<th>TOTAL CASH</th>
<th>NEXT_YR</th>
<th>CUR_YR</th>
<th>LAST_YR</th>
</tr>
</thead>
<tbody>
<tr>
<td>25,992</td>
<td>21,239</td>
<td>17,198</td>
<td></td>
</tr>
</tbody>
</table>

| CHANGE IN CASH | 4,753 | 4,041 |
Controlling the Creation of Column Reference Numbers

**How to:**
Control 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**

```plaintext
SET CNOTATION={$\text{ALL}|\text{PRINTONLY}|\text{EXPLICIT}$}
```

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;            OVER
BAR                                        OVER
RECAP PROFIT = CASH + RECEIVE - INVENTORY_COST;
END
```

The output is:

<table>
<thead>
<tr>
<th>YEAR</th>
<th>1985</th>
<th>1986</th>
<th>1987</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH</td>
<td>5,663</td>
<td>7,001</td>
<td>8,575</td>
</tr>
<tr>
<td>ACCOUNTS RECEIVABLE</td>
<td>5,295</td>
<td>6,250</td>
<td>7,284</td>
</tr>
<tr>
<td>INVENTORY VALUE</td>
<td>7,754</td>
<td>9,076</td>
<td>10,477</td>
</tr>
<tr>
<td>INVENTORY COST FACTOR</td>
<td>1.10</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>INVENTORY COST</td>
<td>8,529</td>
<td>11,345</td>
<td>13,096</td>
</tr>
<tr>
<td>PROFIT</td>
<td>2,429</td>
<td>1,906</td>
<td>2,763</td>
</tr>
</tbody>
</table>
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**

```plaintext
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, $
ENDSTYLE
END
```

The output is:

<table>
<thead>
<tr>
<th>Region</th>
<th>Midwest</th>
<th>Northeast</th>
<th>Southeast</th>
<th>West</th>
<th>TTL1</th>
<th>TTL2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Biscotti</td>
<td>86105</td>
<td>145242</td>
<td>119594</td>
<td>70436</td>
<td>421377</td>
<td>526721.25</td>
</tr>
<tr>
<td>Capuccino</td>
<td>70485</td>
<td>73264</td>
<td>71168</td>
<td></td>
<td>189217</td>
<td>236521.25</td>
</tr>
<tr>
<td>Coffee Grinder</td>
<td>50393</td>
<td>40977</td>
<td>47083</td>
<td>48081</td>
<td>196534</td>
<td>233167.50</td>
</tr>
<tr>
<td>Coffee Pot</td>
<td>47156</td>
<td>46135</td>
<td>49922</td>
<td>47432</td>
<td>190695</td>
<td>238368.75</td>
</tr>
<tr>
<td>Croissant</td>
<td>139182</td>
<td>137394</td>
<td>156456</td>
<td>197022</td>
<td>630054</td>
<td>787567.50</td>
</tr>
<tr>
<td>Espresso</td>
<td>101154</td>
<td>68127</td>
<td>68030</td>
<td>71675</td>
<td>308986</td>
<td>386232.50</td>
</tr>
<tr>
<td>Latte</td>
<td>231623</td>
<td>222866</td>
<td>209654</td>
<td>213920</td>
<td>878063</td>
<td>1097578.75</td>
</tr>
<tr>
<td>Mug</td>
<td>86718</td>
<td>91497</td>
<td>88474</td>
<td>93381</td>
<td>360570</td>
<td>450712.50</td>
</tr>
<tr>
<td>Scone</td>
<td>116127</td>
<td>70732</td>
<td>73779</td>
<td>72776</td>
<td>333414</td>
<td>416767.50</td>
</tr>
<tr>
<td>Thermos</td>
<td>46587</td>
<td>48870</td>
<td>43976</td>
<td>45648</td>
<td>190081</td>
<td>237601.25</td>
</tr>
</tbody>
</table>
```
Referring to Cells in Calculations

How to:

Use Cell Notation for Rows and Columns in a RECAP Expression

You can refer to columns and rows using a form of cell notation that identifies the intersection of a row and a column as $\text{E}(r, c)$.

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:

$$\text{E}(r, c)$$

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).

```plaintext
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
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
```

1338 WebFOCUS
The output is:

<table>
<thead>
<tr>
<th></th>
<th>E_ACTUAL</th>
<th>E_BUDGET</th>
<th>W_ACTUAL</th>
<th>W_BUDGET</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALES</td>
<td>6,000</td>
<td>4,934</td>
<td>7,222</td>
<td>7,056</td>
</tr>
<tr>
<td>COST</td>
<td>4,650</td>
<td>3,760</td>
<td>5,697</td>
<td>5,410</td>
</tr>
<tr>
<td>PROFIT</td>
<td>1,350</td>
<td>1,174</td>
<td>1,525</td>
<td>1,646</td>
</tr>
<tr>
<td>EAST--VARIANCE</td>
<td>176</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WEST--VARIANCE</td>
<td></td>
<td></td>
<td>-121</td>
<td></td>
</tr>
</tbody>
</table>

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

**How to:**

Call a Function in a RECAP Command

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. See the Using Functions manual for information about creating your own functions.

**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: =, -, / ().

input
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:

```sql
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
    RECAP TOTCAS = CASH+ACR+ACI+FUEL+MAT; AS 'TOTAL ASSETS'
    BAR
    RECAP THISDATE/A8 = &YMD; NOPRINT
    RECAP INVAIL = INVEST(CASH,TOTCAS,THISDATE,'D12.2'); AS
      'AVAIL. FOR INVESTMENT'
    BAR AS '='
END
```

The output is:

```
ASSETS AND MONEY AVAILABLE FOR INVESTMENT

<table>
<thead>
<tr>
<th></th>
<th>1986</th>
<th>1985</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH</td>
<td>2,100</td>
<td>1,684</td>
</tr>
<tr>
<td>ACCOUNTS RECEIVABLE</td>
<td>875</td>
<td>619</td>
</tr>
<tr>
<td>INTEREST RECEIVABLE</td>
<td>4,026</td>
<td>3,360</td>
</tr>
<tr>
<td>FUEL INVENTORY</td>
<td>6,250</td>
<td>5,295</td>
</tr>
<tr>
<td>MATERIALS AND SUPPLIES</td>
<td>9,076</td>
<td>7,754</td>
</tr>
<tr>
<td>TOTAL ASSETS</td>
<td>22,327</td>
<td>18,712</td>
</tr>
<tr>
<td>AVAIL. FOR INVESTMENT</td>
<td>3,481</td>
<td>2,994</td>
</tr>
</tbody>
</table>
```
Inserting Rows of Free Text

**How to:**
Insert Data Variables in Text Rows

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:

```
    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

\[
\text{"text } <\text{label}[(c)]>[\text{>}]"
\]

where:

\(<\)

Is a required left caret to bracket the label.

\(\text{label}\)

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. This identifier, however, is required whenever there is more than one column in the report. If you use it, enclose it in parentheses.

\(>\)

Is an optional right bracket 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:

```
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:

<table>
<thead>
<tr>
<th>CURRENT</th>
<th>LAST</th>
<th>CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>YEAR</td>
<td>YEAR</td>
<td>CHANGE</td>
</tr>
<tr>
<td>CASH ON HAND</td>
<td>8,784</td>
<td>7,216</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
<td>4,494</td>
<td>3,483</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
<td>7,961</td>
<td>6,499</td>
</tr>
<tr>
<td>TOTAL CASH</td>
<td>21,239</td>
<td>17,198</td>
</tr>
</tbody>
</table>

**Note:** The designated calculation is performed on each tag or RECAP row of the report. The RECAP rows, however, may change the calculation.
**Example:** 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:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH ON HAND</td>
<td>1,684</td>
<td>2,100</td>
<td>5,000</td>
<td>4,210</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
<td>619</td>
<td>875</td>
<td>3,000</td>
<td>1,548</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
<td>3,360</td>
<td>4,026</td>
<td>575</td>
<td>8,400</td>
</tr>
<tr>
<td>TOTAL CASH</td>
<td>5,663</td>
<td>7,001</td>
<td>8,575</td>
<td>14,158</td>
</tr>
<tr>
<td>CHANGE</td>
<td>1,338</td>
<td>1,574</td>
<td>5,583</td>
<td></td>
</tr>
</tbody>
</table>
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, an error 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 below.

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.

```sql
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,*1) = 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:

<table>
<thead>
<tr>
<th>LAST_YR</th>
<th>CUR_YR</th>
<th>NEXT_YR</th>
</tr>
</thead>
<tbody>
<tr>
<td>7,903.64</td>
<td>9,024.00</td>
<td>10,374.00</td>
</tr>
</tbody>
</table>

| SALES   | 4,986.00 | 6,000.00 | 7,378.20 |
| COST    | 3,864.15 | 4,650.00 | 5,718.11 |

| PROFIT  | 1,121    | 1,350    | 1,660    |

| ENDCASH | 9,024    | 10,374   | 12,034   |
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. See the Describing Data With WebFOCUS Language manual for information.

   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.
**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

How to: Display 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] [ASCaption|'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.

See the *Using Functions* manual for information about the FMLFOR, FMLLIST, FMLCAP, and FMLINFO functions that return the tag values and captions used in an FML request.
Example: Displaying an FML Hierarchy

The following request displays two levels of account numbers, starting from account 3000:

```sql
SET BLANKINDENT=ON
TABLE FILE CENTGL
PRINT GL_ACCOUNT_PARENT
FOR GL_ACCOUNT
3000 WITH CHILDREN 2
END
```

The output is:

<table>
<thead>
<tr>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
</tr>
<tr>
<td>3100</td>
</tr>
<tr>
<td>3110</td>
</tr>
<tr>
<td>3120</td>
</tr>
<tr>
<td>3130</td>
</tr>
<tr>
<td>3140</td>
</tr>
<tr>
<td>3200</td>
</tr>
<tr>
<td>3300</td>
</tr>
<tr>
<td>3400</td>
</tr>
<tr>
<td>3500</td>
</tr>
<tr>
<td>3600</td>
</tr>
<tr>
<td>3700</td>
</tr>
<tr>
<td>3800</td>
</tr>
<tr>
<td>3900</td>
</tr>
</tbody>
</table>

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
</tr>
<tr>
<td>3000</td>
</tr>
<tr>
<td>3100</td>
</tr>
<tr>
<td>3100</td>
</tr>
<tr>
<td>3100</td>
</tr>
<tr>
<td>3100</td>
</tr>
<tr>
<td>3100</td>
</tr>
<tr>
<td>3100</td>
</tr>
<tr>
<td>3100</td>
</tr>
<tr>
<td>3100</td>
</tr>
<tr>
<td>3200</td>
</tr>
<tr>
<td>3200</td>
</tr>
<tr>
<td>3200</td>
</tr>
<tr>
<td>3200</td>
</tr>
<tr>
<td>3200</td>
</tr>
</tbody>
</table>

Note that if the request specifies GET CHILDREN instead of WITH CHILDREN, the line for the parent value (3000) does not display on the report output.
**Example:**  **Displaying an FML Hierarchy With Captions**

The following request displays two levels of a charts of accounts hierarchy, starting with account 1000 (the top of the hierarchy), and displays the caption field values instead of the account numbers:

```
SET BLANKINDENT=ON
TABLE FILE CENTGL
PRINT GL_ACCOUNT_PARENT
FOR GL_ACCOUNT
  1000 WITH CHILDREN 2 AS CAPTION
END
```

The output is:

```
Parent
------
Profit Before Tax  1000
  Gross Margin     2000
  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 that 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**

**How to:**

Create One Summary Row for an FML Hierarchy
Consolidate FML Hierarchy Data to Any Level and Depth

The ADD command consolidates multiple levels of the hierarchy on one line of the FML report output. ADD can be used 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.
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.

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 ...

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;

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 3100 (Selling Expenses). 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:

```sql
SET BLANKINDENT=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:

<table>
<thead>
<tr>
<th>Selling Expenses</th>
<th>Month Actual</th>
<th>YTD Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advertising</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TV/Radio</td>
<td>1,049,146.</td>
<td>2,954,342.</td>
</tr>
<tr>
<td>Print Media</td>
<td>244,589.</td>
<td>721,448.</td>
</tr>
<tr>
<td>Internet Advertising</td>
<td>9,542.</td>
<td>29,578.</td>
</tr>
<tr>
<td>Promotional Expenses</td>
<td>53,719.</td>
<td>151,732.</td>
</tr>
<tr>
<td>Joint Marketing</td>
<td>97,135.</td>
<td>289,799.</td>
</tr>
<tr>
<td>Bonuses/Commissions</td>
<td>100,188.</td>
<td>304,199.</td>
</tr>
<tr>
<td>Selling Expenses</td>
<td>1,554,319.</td>
<td>4,451,098.</td>
</tr>
</tbody>
</table>
Syntax: How to Consolidate FML Hierarchy Data to Any Level and Depth

```
TABLE FILE filename
SUM...
FOR hierarchyfld
parentvalue {GET|WITH} CHILDREN [n|ALL] ADD [m|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.

- `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.
Consolidating FML Hierarchy Data

Example:

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 BLANKINDENT = 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 AS =
" "
  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 3100 (Selling Expenses): 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.

<table>
<thead>
<tr>
<th>Selling Expenses</th>
<th>Month Actual</th>
<th>YTD Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advertising</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TV/Radio</td>
<td>1,049,146</td>
<td>2,954,342</td>
</tr>
<tr>
<td>Print Media</td>
<td>244,589</td>
<td>721,448</td>
</tr>
<tr>
<td>Internet Advertising</td>
<td>9,542</td>
<td>29,578</td>
</tr>
<tr>
<td>Promotional Expenses</td>
<td>53,719</td>
<td>151,732</td>
</tr>
<tr>
<td>Joint Marketing</td>
<td>97,135</td>
<td>289,799</td>
</tr>
<tr>
<td>Bonuses/Commissions</td>
<td>100,188</td>
<td>304,199</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Selling Expenses</th>
<th>Month Actual</th>
<th>YTD Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advertising</td>
<td>1,303,277</td>
<td>3,705,368</td>
</tr>
<tr>
<td>Promotional Expenses</td>
<td>53,719</td>
<td>151,732</td>
</tr>
<tr>
<td>Joint Marketing</td>
<td>97,135</td>
<td>289,799</td>
</tr>
<tr>
<td>Bonuses/Commissions</td>
<td>100,188</td>
<td>304,199</td>
</tr>
</tbody>
</table>
Using GET CHILDREN instead of WITH CHILDREN eliminates the top line from each portion of the output. The remaining lines are the same:

<table>
<thead>
<tr>
<th></th>
<th>Month Actual</th>
<th>YTD Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advertising</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TV/Radio</td>
<td>1,049,146</td>
<td>2,954,342</td>
</tr>
<tr>
<td>Print Media</td>
<td>244,589</td>
<td>721,448</td>
</tr>
<tr>
<td>Internet Advertising</td>
<td>9,542</td>
<td>29,578</td>
</tr>
<tr>
<td>Promotional Expenses</td>
<td>53,719</td>
<td>151,732</td>
</tr>
<tr>
<td>Joint Marketing</td>
<td>97,135</td>
<td>289,799</td>
</tr>
<tr>
<td>Bonuses/Commisions</td>
<td>100,188</td>
<td>304,199</td>
</tr>
</tbody>
</table>

|                  |              |            |
| Advertising      | 1,303,277    | 3,705,368  |
| Promotional Expenses | 53,719    | 151,732    |
| Joint Marketing  | 97,135       | 289,799    |
| Bonuses/Commisions | 100,188   | 304,199    |
The following request displays a consolidated line for account 2000 and each of its direct children and grandchildren.

```sql
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:

<table>
<thead>
<tr>
<th>Gross Margin</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Month</strong></td>
<td></td>
<td><strong>YTD</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Actual</td>
<td></td>
<td>Actual</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-4,513,659</td>
<td></td>
<td>-13,080,549</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales Revenue</td>
<td>-10,398,305</td>
<td></td>
<td>-30,877,546</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retail Sales</td>
<td>-8,237,253</td>
<td></td>
<td>-24,539,197</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mail Order Sales</td>
<td>-1,138,414</td>
<td></td>
<td>-3,403,387</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internet Sales</td>
<td>-1,022,638</td>
<td></td>
<td>-2,934,962</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost Of Goods Sold</td>
<td>5,884,646</td>
<td></td>
<td>17,796,997</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable Material Costs</td>
<td>4,415,560</td>
<td></td>
<td>13,410,629</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direct Labor</td>
<td>961,143</td>
<td></td>
<td>2,920,449</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed Costs</td>
<td>507,943</td>
<td></td>
<td>1,465,919</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Loading a Hierarchy Manually

**How to:**

Load a Hierarchy From One Master File for Use With a Separate Master File

**Reference:**

Usage Notes for FML Hierarchies

In most cases, a hierarchy is loaded 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 be loaded is limited by available memory. Charts are automatically unloaded when the session ends.
The chart is loaded 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:

<table>
<thead>
<tr>
<th>parentfield</th>
<th>hierarchyfield</th>
</tr>
</thead>
<tbody>
<tr>
<td>parentvalue1</td>
<td>child1</td>
</tr>
<tr>
<td>parentvalue1</td>
<td>child2</td>
</tr>
<tr>
<td>parentvalue1</td>
<td>child3</td>
</tr>
</tbody>
</table>

**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.

The number of charts that can be loaded is limited by available memory. Charts are automatically unloaded when FOCUS 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.
Reporting Dynamically From a Hierarchy

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. 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. 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.
Customizing a Row Title

**How to:**

Customize a Row Title in FML

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.

- **calcname**
  
  Is the value that is derived by the RECAP calculation.

- **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.
Changing the Titles of Tag Rows

Example:  
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:

```
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

Example:  
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:

```
AMOUNT
-------
CASH ON HAND     8,784
DEMAND DEPOSITS  4,494
TIME DEPOSITS    7,961
TOTAL CASH       21,239
```

If no AS phrase are included, the name of the RECAP value (TOTCASH) is displayed in the report.
Improving 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 1386.
How to Add an Underline Character for Columns

The syntax is:

```plaintext
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 (-).

Underlining Columns

This example uses the default underscore character (_):

```sql
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:

```
AMOUNT
------
CASH ON HAND  8,784
DEMAND DEPOSITS  4,494
TIME DEPOSITS    7,961
      ------
TOTCASH       21,239
```

Notice that the BAR ... OVER phrases underline only the column containing the display field.

How to Specify a Page Break in an FML Report

Include the following syntax in the FML request in place of a tag value:

```plaintext
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
```
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 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 1367.

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
Example: 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 STYLESHEET *
TYPE=REPORT, GRID=OFF, $
TYPE = REPORT, LABEL = CA, STYLE = BOLD, $
ENDSTYLE
END
```

The output is:

```
    AMOUNT
    CASH ON HAND      8,784
    DEMAND DEPOSITS   4,494
    TIME DEPOSITS     7,961
    TOTAL CASH       21,239
```

```
    AMOUNT
     CASH         21,239
 ACCOUNTS RECEIVABLE   18,829
     INVENTORY     27,307
     CURASST      67,375
```
Example: 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/5C = CA + AR + INV;
ON TABLE SET STYLESHEET *
  TYPE=REPORT, GRID=OFF, $
  TYPE = REPORT, COLUMN=N2, LABEL = CA, STYLE = BOLD, $
ENDSTYLE
END
```

The output is:

<table>
<thead>
<tr>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH</td>
</tr>
<tr>
<td>ACCOUNTS RECEIVABLE</td>
</tr>
<tr>
<td>INVENTORY</td>
</tr>
<tr>
<td>CURASST</td>
</tr>
</tbody>
</table>
**Example:** 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'
ON TABLE SET STYLESHEET *
TYPE=REPORT, GRID=OFF, $
**TYPE=REPORT, COLUMN=AMOUNT, STYLE=BOLD,** $
ENDSTYLE
END
```

The output is:

```
---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
```
**Example: 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 STYLESHEET *
TYPE=REPORT, GRID=OFF, $
**TYPE = FREETEXT, STYLE = BOLD, **$
ENDSTYLE
END
```

The output is:

```
<table>
<thead>
<tr>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>---CASH ACCOUNTS---</td>
</tr>
<tr>
<td>CASH ON HAND</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
</tr>
<tr>
<td>---OTHER CURRENT ASSETS---</td>
</tr>
<tr>
<td>ACCOUNTS RECEIVABLE</td>
</tr>
<tr>
<td>INVENTORY</td>
</tr>
</tbody>
</table>
```
**Example:** 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:

```
<table>
<thead>
<tr>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>---CASH ACCOUNTS---</td>
</tr>
<tr>
<td>CASH ON HAND</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
</tr>
<tr>
<td>---OTHER CURRENT ASSETS---</td>
</tr>
<tr>
<td>ACCOUNTS RECEIVABLE</td>
</tr>
<tr>
<td>INVENTORY</td>
</tr>
</tbody>
</table>
```
**Example:** 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.

```fml
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 STYLESHEET *
  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:

<table>
<thead>
<tr>
<th>AMOUNT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH ON HAND</td>
<td>8,784</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
<td>4,494</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
<td>7,961</td>
</tr>
<tr>
<td>TOTAL CASH</td>
<td>21,239</td>
</tr>
</tbody>
</table>
**Example: 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 SETstylesheet *
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:

<table>
<thead>
<tr>
<th>AMOUNT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH ON HAND</td>
<td>8,784</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
<td>4,494</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
<td>7,961</td>
</tr>
<tr>
<td>TOTAL CASH</td>
<td><strong>21,239</strong></td>
</tr>
</tbody>
</table>
**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(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)}], $
```

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:

```
<table>
<thead>
<tr>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH ON HAND</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
</tr>
<tr>
<td>TOTAL CASH</td>
</tr>
</tbody>
</table>
```
**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.

```qml
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-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:

```
<table>
<thead>
<tr>
<th>AMOUNT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH ON HAND</td>
<td>8,784</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
<td>4,494</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
<td>7,961</td>
</tr>
<tr>
<td>TOTAL CASH</td>
<td>21,239</td>
</tr>
</tbody>
</table>
```
**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 STYLESHEET *
TYPE=REPORT, GRID=OFF,$
TYPE=REPORT, LABEL=TOTCASH, COLUMN=N2, BORDER=MEDIUM, $
ENDSTYLE
END
```

The output is:

```
<table>
<thead>
<tr>
<th>AMOUNT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH ON HAND</td>
<td>8,784</td>
</tr>
<tr>
<td>DEMAND DEPOSITS</td>
<td>4,494</td>
</tr>
<tr>
<td>TIME DEPOSITS</td>
<td>7,961</td>
</tr>
<tr>
<td>TOTAL CASH</td>
<td>21,239</td>
</tr>
</tbody>
</table>
```
How to Specify an Indent for an FML Label, Tag, or Caption

**Syntax:**

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:

`k`
Is the starting column for the FOR value in an FML report.

`forfield`
Is a field in the data source whose values are included in the 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 1351.

`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.

eexpression
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):

```sql
SET FORMULTIPLE = ON
TABLE FILE CENTGL
PRINT GL_ACCOUNT_PARENT
FOR GL_ACCOUNT
3000 AS 'Not Indented' OVER
3000 INDENT 5 AS 'Indented 5' OVER
3000 GET CHILDREN 2 INDENT 5 AS 'Hierarchy Indented 5'
END
```

The output is:

<table>
<thead>
<tr>
<th>Parent</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Indented</td>
<td>3000</td>
</tr>
<tr>
<td>Indented 5</td>
<td>3000</td>
</tr>
<tr>
<td>Hierarchy Indented 5</td>
<td>3000</td>
</tr>
<tr>
<td>Hierarchy Indented 5</td>
<td>3100</td>
</tr>
<tr>
<td>Hierarchy Indented 5</td>
<td>3100</td>
</tr>
<tr>
<td>Hierarchy Indented 5</td>
<td>3100</td>
</tr>
<tr>
<td>Hierarchy Indented 5</td>
<td>3100</td>
</tr>
<tr>
<td>Hierarchy Indented 5</td>
<td>3000</td>
</tr>
<tr>
<td>Hierarchy Indented 5</td>
<td>3200</td>
</tr>
<tr>
<td>Hierarchy Indented 5</td>
<td>3200</td>
</tr>
<tr>
<td>Hierarchy Indented 5</td>
<td>3200</td>
</tr>
<tr>
<td>Hierarchy Indented 5</td>
<td>3200</td>
</tr>
<tr>
<td>Hierarchy Indented 5</td>
<td>3200</td>
</tr>
<tr>
<td>Hierarchy Indented 5</td>
<td>3200</td>
</tr>
<tr>
<td>Hierarchy Indented 5</td>
<td>3200</td>
</tr>
</tbody>
</table>
**Example: 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 OVER
Analog OVER
BAR OVER
RECAP TOTAL = R1 + R2; INDENT 5 AS 'Total: ' OVER
BAR OVER
RECAP PROFIT(2) = TOTAL(1) - TOTAL(2); AS 'Profit: ' INDENT 10
END
```

The output is:

<table>
<thead>
<tr>
<th>Price:</th>
<th>Our</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Digital</td>
<td>4,080.00</td>
<td>3,052.00</td>
</tr>
<tr>
<td>Analog</td>
<td>1,883.00</td>
<td>1,371.00</td>
</tr>
<tr>
<td></td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>5,963.00</td>
<td>4,423.00</td>
</tr>
<tr>
<td></td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>Profit:</td>
<td>1,540.00</td>
<td></td>
</tr>
</tbody>
</table>

**Indenting Row Titles in an FML Hierarchy**

**How to:**

Indent FML Hierarchy Captions in an HTML Report

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 increase the width of the indented column if it is not wide enough to accommodate the indented fields. While this is no problem in an HTML report, in a PDF or PostScript report it can cause data in the columns that follow the indented column to shift out of alignment. You may need to use StyleSheet syntax to make the column wide enough for the indented values or to move the columns that follow it. Change the width of a column using the StyleSheet SQUEEZE attribute, and specify a starting position for a column using the POSITION attribute. You can also move a column in a PostScript report with the IN phrase.

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. See Formatting an FML Report on page 1367.

**Syntax:**

**How to Indent FML Hierarchy Captions in an HTML Report**

```plaintext
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:

<table>
<thead>
<tr>
<th>Parent</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Operating Expenses</td>
<td>1000</td>
</tr>
<tr>
<td>Selling Expenses</td>
<td>3000</td>
</tr>
<tr>
<td>Advertising</td>
<td>3100</td>
</tr>
<tr>
<td>Promotional Expenses</td>
<td>3100</td>
</tr>
<tr>
<td>Joint Marketing</td>
<td>3100</td>
</tr>
<tr>
<td>Bonuses/Commissions</td>
<td>3100</td>
</tr>
<tr>
<td>General + Admin Expenses</td>
<td>3000</td>
</tr>
<tr>
<td>Salaries-Corporate</td>
<td>3200</td>
</tr>
<tr>
<td>Company Benefits</td>
<td>3200</td>
</tr>
<tr>
<td>Depreciation Expenses</td>
<td>3200</td>
</tr>
<tr>
<td>Gain/(Loss) Sale of Equipment</td>
<td>3200</td>
</tr>
<tr>
<td>Leasehold Expenses</td>
<td>3200</td>
</tr>
<tr>
<td>Interest Expenses</td>
<td>3200</td>
</tr>
<tr>
<td>Utilities</td>
<td>3200</td>
</tr>
</tbody>
</table>
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, 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:

<table>
<thead>
<tr>
<th>Parent</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Operating Expenses</td>
<td>1000</td>
</tr>
<tr>
<td>Selling Expenses</td>
<td>3000</td>
</tr>
<tr>
<td>Advertising</td>
<td>3100</td>
</tr>
<tr>
<td>Promotional Expenses</td>
<td>3100</td>
</tr>
<tr>
<td>Joint Marketing</td>
<td>3100</td>
</tr>
<tr>
<td>Bonuses/Commisions</td>
<td>3100</td>
</tr>
<tr>
<td>General + Admin Expenses</td>
<td>3000</td>
</tr>
<tr>
<td>Salaries-Corporate</td>
<td>3200</td>
</tr>
<tr>
<td>Company Benefits</td>
<td>3200</td>
</tr>
<tr>
<td>Depreciation Expenses</td>
<td>3200</td>
</tr>
<tr>
<td>Gain/(Loss) Sale of Equipment</td>
<td>3200</td>
</tr>
<tr>
<td>Leasehold Expenses</td>
<td>3200</td>
</tr>
<tr>
<td>Interest Expenses</td>
<td>3200</td>
</tr>
<tr>
<td>Utilities</td>
<td>3200</td>
</tr>
</tbody>
</table>
Suppressing the Display of Rows

In this section:
Suppressing Rows With No Data

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.
See Suppressing Rows With No Data on page 1391.

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
" "
RECAP ROS/F6.2=100*PROFIT/SLS;
AS 'RETURN ON SALES'
END

The output is:

<table>
<thead>
<tr>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALES</td>
</tr>
<tr>
<td>PROFIT</td>
</tr>
<tr>
<td>RETURN ON SALES</td>
</tr>
</tbody>
</table>
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

In this section: Posting Data

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, UNIX, and CMS, 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**

**How to:**

- Post Data to a File
- Pick Up Data From a Work File

You can save any TAG, RECAP, or DATA row by posting the output to a file. These rows can then be used 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 ($). See *Posting Rows to a Work File* on page 1393.

**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 re-format 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 work file, LEDGEOUT:

```
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:

<table>
<thead>
<tr>
<th></th>
<th>CUR_YR</th>
<th>LAST_YR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1100</td>
<td>18.829</td>
<td>15.954</td>
</tr>
<tr>
<td>1200</td>
<td>27.307</td>
<td>23.329</td>
</tr>
</tbody>
</table>

**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 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 1393.)

Tip: You must assign a logical name to the file by issuing a FILEDEF command on Windows, UNIX, and CMS, 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:

<table>
<thead>
<tr>
<th></th>
<th>CUR YR</th>
<th>LAST YR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH</td>
<td>21,239</td>
<td>17,198</td>
</tr>
<tr>
<td>ACCOUNTS RECEIVABLE</td>
<td>18,829</td>
<td>15,954</td>
</tr>
<tr>
<td>INVENTORY</td>
<td>27,307</td>
<td>23,329</td>
</tr>
<tr>
<td>CUR.Asset</td>
<td>67,375</td>
<td>56,481</td>
</tr>
</tbody>
</table>

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: RECAP rows cannot be reformatted 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

<table>
<thead>
<tr>
<th>FIELDNAME</th>
<th>ALIAS</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E01</td>
<td>A 19</td>
</tr>
<tr>
<td>AMOUNT</td>
<td>E02</td>
<td>I5C</td>
</tr>
</tbody>
</table>
```

Then report from the HOLD file as:

```
TABLE FILE HOLD
PRINT E01 E02
END
```
The output is:

<table>
<thead>
<tr>
<th>ACCOUNT</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASH</td>
<td>21,239</td>
</tr>
<tr>
<td>ACCOUNTS RECEIVABLE</td>
<td>18,829</td>
</tr>
<tr>
<td>INVENTORY</td>
<td>27,307</td>
</tr>
<tr>
<td>CURRENT ASSETS</td>
<td>67,375</td>
</tr>
</tbody>
</table>
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.

**Topics:**
- 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:

- **Heading**: Contains the body of the report. It displays the text characters, graphic characters, and data fields that make up the report.

- **Footing**: 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.

- **Prefix operators**: Indicates field calculations and manipulation.

- **Temporary fields**: Derives new values from existing fields in a data source.

- **BY phrases**: Specifies the report sort order, and determines how many records are included on each page.

- **WHERE criteria**: 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 1402.

- Customize the layout of your report. See *Laying Out a Free-Form Report* on page 1404.

- 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 1404.

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 Request for EMPLOYEE EDUCATION HOURS REPORT on page 1400.
**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

In this section:
- Incorporating Text in a Free-Form Report
- Incorporating Data Fields in a Free-Form Report
- Incorporating Graphic Characters in a Free-Form Report
- Laying Out a Free-Form Report
- Sorting and Selecting Records in 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 Request for EMPLOYEE EDUCATION HOURS REPORT on page 1400) 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 1400) 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 1398) 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 Request for EMPLOYEE EDUCATION HOURS REPORT on page 1400) 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.
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.

Topics:

- 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 1406 and Unsupported SQL Statements on page 1408.

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.

### 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 1412.
- 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 1419.
- 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.
Supported and Unsupported SQL Statements

- 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.
- =ALL, <>ANY, and <>SOME 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 1412.

**Syntax:**

```sql
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 1406.

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 1410.

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
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 1053.

For details on ON TABLE HOLD or ON TABLE PCHOLD, see Saving and Reusing Your Report Output on page 351.

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 351.

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 143.

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 769.

For related information, see Supported SQL Statements on page 1406 and How to Use SQL Translator Commands on page 1409.
SQL Joins

How to:
Create an Inner Join
Create an Outer Join

Reference:
Join Name Assignments From the SQL Translator
SQL Join Considerations

When performing SQL joins, the formats of the joined fields must be the same. Join fields need not be indexed, and non-equijoins 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
SQL
SELECT fieldlist FROM file1 {LEFT|RIGHT} JOIN file2
ON join_condition [{LEFT|RIGHT} 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.

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 1424.

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:

```sql
-- 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**

**How to:**

Create a View

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 351.

**Syntax:**  How to Create a View

The SQL Translator supports the following SQL statement:

```sql
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.
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.

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 769.

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 1412. 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 769.

**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**

In this section:

Extracting Date-Time Components Using the SQL Translator

Reference:

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 number of seconds difference. 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:

<table>
<thead>
<tr>
<th>Format</th>
<th>USAGE Attribute in Master File</th>
<th>Date Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>YYMD</td>
<td>YYYY-MM-DD</td>
</tr>
<tr>
<td>Hour</td>
<td>HH</td>
<td>HH</td>
</tr>
<tr>
<td>Hour through minute</td>
<td>HHI</td>
<td>HH.MM</td>
</tr>
<tr>
<td>Hour through second</td>
<td>HHIS</td>
<td>HH.MM.SS</td>
</tr>
<tr>
<td>Hour through millisecond</td>
<td>HHISs</td>
<td>HH.MM.SS.fff</td>
</tr>
<tr>
<td>Hour through microsecond</td>
<td>HHISsm</td>
<td>HH.MM.SSffffff</td>
</tr>
<tr>
<td>Year through hour</td>
<td>HYYMDH</td>
<td>YYYY-MM-DD HH</td>
</tr>
<tr>
<td>Year through minute</td>
<td>HYYMDI</td>
<td>YYYY-MM-DD HH.MM</td>
</tr>
<tr>
<td>Year through second</td>
<td>HYYMDS</td>
<td>YYYY-MM-DD HH.MM.SS</td>
</tr>
<tr>
<td>Year through millisecond</td>
<td>HYYMDS</td>
<td>YYYY-MM-DD HH.MM.SS.fff</td>
</tr>
<tr>
<td>Year through microsecond</td>
<td>HYYMDS</td>
<td>YYYY-MM-DD HH.MM.SSffffff</td>
</tr>
</tbody>
</table>

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

**How to:**

Use Date, Time, and Timestamp Functions Accepted by the SQL Translator
Use the SQL Translator EXTRACT Function to Extract Date-Time Components

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:

<table>
<thead>
<tr>
<th>Function</th>
<th>Return Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{YEAR}(x) )</td>
<td>year</td>
</tr>
<tr>
<td>( \text{MONTH}(x) )</td>
<td>month number</td>
</tr>
<tr>
<td>( \text{DAY}(x) )</td>
<td>day number</td>
</tr>
<tr>
<td>( \text{HOUR}(x) )</td>
<td>hour</td>
</tr>
<tr>
<td>( \text{MINUTE}(x) )</td>
<td>minute</td>
</tr>
<tr>
<td>( \text{SECOND}(x) )</td>
<td>second</td>
</tr>
<tr>
<td>( \text{MILLISECOND}(x) )</td>
<td>millisecond</td>
</tr>
<tr>
<td>( \text{MICROSECOND}(x) )</td>
<td>microsecond</td>
</tr>
</tbody>
</table>
**Example:** Using SQL Translator Date, Time, and Timestamp Functions

Using the timestamp column TS whose value is '1999-11-23 07:32:16.123456':

- \( \text{YEAR(TS)} = 1999 \)
- \( \text{MONTH(TS)} = 11 \)
- \( \text{DAY(TS)} = 23 \)
- \( \text{HOUR(TS)} = 7 \)
- \( \text{MINUTE(TS)} = 32 \)
- \( \text{SECOND(TS)} = 16 \)
- \( \text{MILLISECOND(TS)} = 123 \)
- \( \text{MICROSECOND(TS)} = 123456 \)

**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.

\[
\text{SQL} \\
\text{SELECT TRANSDATE,} \\
\text{YEAR(TRANSDATE),} \\
\text{MONTH(TRANSDATE),} \\
\text{MINUTE(TRANSDATE)} \\
\text{FROM VIDEOTR2;} \\
\text{FILE VIDSQL} \\
\text{END} \\
\]

The SQL Translator produces the following virtual fields for functions, followed by a TABLE request to display the output:

\[
\text{SET COUNTWIDTH=ON} \\
\text{-SET SQLERRNUM = 0;} \\
\text{DEFINE FILE} \\
\text{VIDEOTR2 TEMP} \\
\text{SQLDEF01/I4 MISSING ON NEEDS ALL = HPART(TRANSDATE,'YEAR','I4');} \\
\text{SQLDEF02/I2 MISSING ON NEEDS ALL = HPART(TRANSDATE,'MONTH','I2');} \\
\text{SQLDEF03/I2 MISSING ON NEEDS ALL = HPART(TRANSDATE,'MINUTE','I2');} \\
\text{END} \\
\text{TABLEF FILE VIDEOTR2} \\
\text{PRINT TRANSDATE SQLDEF01 AS 'SQLDEF01' SQLDEF02 AS 'SQLDEF02' SQLDEF03 AS 'SQLDEF03'} \\
\text{ON TABLE SET CARTESIAN ON} \\
\text{ON TABLE SET ASNAMES ON} \\
\text{ON TABLE SET HOLDLIST PRINTONLY} \\
\text{END} \\
\]
The output is:

<table>
<thead>
<tr>
<th>TRANSDATE</th>
<th>SQLDEF02</th>
<th>SQLDEF04</th>
<th>SQLDEF05</th>
</tr>
</thead>
<tbody>
<tr>
<td>1999/06/20 04:14</td>
<td>1999</td>
<td>6</td>
<td>14</td>
</tr>
<tr>
<td>1991/06/27 02:45</td>
<td>1991</td>
<td>6</td>
<td>45</td>
</tr>
<tr>
<td>1996/06/21 01:16</td>
<td>1996</td>
<td>6</td>
<td>16</td>
</tr>
<tr>
<td>1991/06/21 07:11</td>
<td>1991</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>1991/06/20 05:15</td>
<td>1991</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>1999/06/26 12:34</td>
<td>1999</td>
<td>6</td>
<td>34</td>
</tr>
<tr>
<td>1919/06/26 05:45</td>
<td>1919</td>
<td>6</td>
<td>45</td>
</tr>
<tr>
<td>1991/06/24 04:43</td>
<td>1991</td>
<td>6</td>
<td>43</td>
</tr>
<tr>
<td>1991/06/21 04:43</td>
<td>1991</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>1999/10/06 02:51</td>
<td>1999</td>
<td>10</td>
<td>51</td>
</tr>
<tr>
<td>1991/06/25 01:17</td>
<td>1991</td>
<td>6</td>
<td>17</td>
</tr>
</tbody>
</table>

**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:

\[
\text{EXTRACT}(\text{component} \text{ FROM value})
\]

where:

- **component**
  
  Is one of the following: YEAR, MONTH, DAY, 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:

<table>
<thead>
<tr>
<th>TRANSDATE</th>
<th>SQLDEF02</th>
<th>SQLDEF04</th>
<th>SQLDEF05</th>
</tr>
</thead>
<tbody>
<tr>
<td>1999-01-01</td>
<td>1999</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2000-03-03</td>
<td>2000</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Index Optimized Retrieval

In this section:

- Optimized Joins

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**

**How to:**

Improve Performance Using SQLTOPTTF

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 1427.

**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.
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 does not support subqueries, such as:

  \[
  \text{INSERT...INTO...SELECT...FROM...}
  \]

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.
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.
Rotating a Data Structure for Enhanced Retrieval

**How to:**
Request an Alternate View

**Reference:**
Usage Notes for Restructuring Data

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:

```
AREA

DATE

PROD_CODE
UNIT_CODE
RETAIL_PRICE
DELIVER_AMT
```

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**

```plaintext
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.

**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. This value is the default. However, the default may have been changed in a supported profile. You can check your setting by issuing the `? SET` command.

**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.
**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:

```plaintext
TABLEF FILE HOLD
PRINT CURR_SAL AND LAST_NAME AND FIRST_NAME
END
```

**Preserving the Internal Matrix of Your Last Report**

**How to:**

Save an Internal Matrix

**Note:** The following feature is only available during the development phase in Developer Studio.

An internal matrix is generated with each TABLE, FML, GRAPH, and MATCH request. These requests are available for the duration of your session, or until you generate a new report or graph that overwrites it.

While a report (or graph) request is available, you can:

- Extract and save data from it using the HOLD, SAVE, and SAVB commands.
- Redisplay it using the RETYPE or REPLOT commands.

If you wish to save the matrix from your last request to protect it from being overwritten when using Dialogue Manager commands, you can activate the SET SAVEMATRIX feature.

**Note:** SET SAVEMATRIX is not available with the TABLEF command.

**Syntax:  **

**How to Save an Internal Matrix**

```plaintext
SET SAVEMATRIX = {ON|OFF}
```

where:

- **ON**
  
  Saves the internal matrix from the last report request, preventing it from being overwritten.

- **OFF**
  
  Overwrites the internal matrix for each request. OFF is the default value.
Example: Saving the Internal Matrix of a Report

The following request creates a report, then executes a procedure that contains a Dialogue Manager command (which would otherwise overwrite the internal matrix), and recalls the report using the RETYPE command:

```
SET SAVEMATRIX = ON
TABLE FILE EMPLOYEE
. .
END
EX DMFEX
RETYPE
```

Compiling Expressions

In this section:

- Compiling Expressions Using the DEFINES Parameter
- Compiling Expressions Using the COMPUTE Parameter

Compiling expressions into machine code provides faster processing.

Compiling Expressions Using the DEFINES Parameter

How to:

- Compile DEFINE Expressions
- Query Compiled DEFINE Expressions

On z/OS and z/VM, two expression compilers are available. By issuing the appropriate command, you can select one of them or disable compilation of expressions. Both compilers cannot be active for the same request:

- The DEFINE compiler compiles only those expressions that are found in DEFINE fields referenced in TABLE requests, but it provides much faster execution of those expressions than the other compiler. It compiles expressions using the arithmetic operations built into the underlying operating system, and is, therefore, referred to as the native compiler. Compilation takes place at TABLE run time. This compiler is invoked by issuing the command SET DEFINES = COMPILED.
The other compiler is invoked with the command SET COMPUTE = NEW. This compiler provides expression compilation for DEFINE, IF, and WHERE commands in TABLE procedures. Under this compiler, expressions are compiled at DEFINE time. Therefore, compilation may be invoked for expressions that are never actually used in a request.

Among the benefits of the DEFINE compiler 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.

After the native DEFINE compiler is invoked, any request that uses a DEFINE expression causes the expression to be compiled and then loaded into the system. For each record of the request that needs computation, the system executes the generated code. This compiler is most effective with TABLE requests that include a large number of DEFINE fields and read a large number of records because the speed of evaluation per record in such requests offsets the extra compilation and load steps.

**Syntax:** How to Compile DEFINE Expressions

Issue the following command FOCPARM, a FOCEXEC, or at the command line:

```
SET DEFINES = {COMPILED|OLD}
```

where:

- **COMPILED**
  
  Implements expression compilation at request run time, compiling only those DEFINEs that are used in the request.

- **OLD**
  
  Leaves expression compilation up to the control of the current SET COMPUTE value. OLD is the default value. If you issue the SET DEFINES = OLD command, the COMPUTE parameter is automatically set to NEW.

**Syntax:** How to Query Compiled DEFINE Expressions

Issue the following command to query the current setting:

```
? SET DEFINES
```
Compiling Expressions Using the COMPUTE Parameter

How to:
Control Expression Compilation Using the COMPUTE Parameter

Reference:
Usage Notes for SET COMPUTE
Interaction Between SET DEFINES and SET COMPUTE
Usage Notes for Compiled DEFINEs

Syntax: How to Control Expression Compilation Using the COMPUTE Parameter

SET COMPUTE = {NEW|OLD|NATV}
where:

NEW
Compiles DEFINE calculations when a request is executed.

OLD
Does not compile DEFINE calculations when a request is executed. The old logic is used.

NATV
Compiles DEFINE calculations using the native compiler. This setting is also activated by the SET DEFINES=COMPILED command, which is the default setting.

Reference: Usage Notes for SET COMPUTE

The following calculations are not compiled with SET COMPUTE = NEW:

- Calculations that involve any function (for example, user functions), except for EDIT, DECODE, and LAST.
- Calculations that test for existing data (IF field IS-NOT MISSING) or that result in a missing field (TEMP/A4 MISSING ON= ...).
- Calculations that involve fields with date formats. (See the table of date formats in the Describing Data With WebFOCUS Language manual.)
- Calculations that use exponentiation (10**2).
Reference: Interaction Between SET DEFINES and SET COMPUTE

Two expression compilers are available, but only one can be activated for any request. Activating either compiler automatically deactivates the other compiler:

- Issuing the SET DEFINES=COMPILED command activates the new compiler and deactivates the old compiler by automatically setting the value of the COMPUTE parameter to NATV (native compiler).

- Issuing the SET DEFINES=OLD command deactivates the new compiler and automatically activates the old compiler by automatically setting the value of the COMPUTE parameter to NEW.

- Issuing the SET COMPUTE command with either the OLD or NEW setting deactivates the new compiler. The OLD setting also deactivates the old compiler.

Therefore, you can select either compiler by issuing the SET DEFINES command. DEFINES=COMPILED selects the new compiler, DEFINES=OLD selects the old compiler. To turn compilation off, issue SET COMPUTE=OLD.

The new compiler is recommended for TABLE requests that include a large number of DEFINE fields (especially those that use packed arithmetic or date expressions) and read a large number of records.

If a TABLE request retrieves a large number of records or if the DEFINE fields use packed arithmetic (especially with long packed fields) or date expressions, the new compiler is likely to provide the most benefit.

Reference: Usage Notes for Compiled DEFINEs

- 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 with the new compiler:
  - Functions. However, expressions that use the following functions can be compiled: YMD, DMY, INT, and DECODE.
  - CONTAINS, OMITS, LAST.

- The SET DEFINES command is not supported in an ON TABLE phrase.

- SET DEFINES creates a pool boundary when used in conjunction with Pooled Tables.

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.
This appendix contains descriptions and structure diagrams for the sample data sources used throughout the documentation.

**Topics:**
- EMPLOYEE Data Source
- JOBFILE Data Source
- EDUCFILE Data Source
- SALES Data Source
- CAR Data Source
- LEDGER Data Source
- FINANCE Data Source
- REGION Data Source
- EMPDATA Data Source
- TRAINING Data Source
- COURSE Data Source
- JOBHIST Data Source
- JOBLIST Data Source
- LOCATOR Data Source
- PERSINFO Data Source
- SALHIST Data Source
- VIDEOTRK, MOVIES, and ITEMS Data Sources
- VIDEOTR2 Data Source
- Gotham Grinds Data Sources
- Century Corp Data Sources
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 JOBFILE and EDUCFILE files, also described in this appendix. The segments are:

**JOBSEG (from JOBFILE)**
Describes the job positions held by each employee.

**SKILLSEG (from JOBFILE)**
Lists the skills required by each position.
SECSEG (from JOBFILE)

Specifies the security clearance needed for each job position.

ATTNDSEG (from EDUCFILE)

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,$
EMPLOYEE Structure Diagram

The EMPLOYEE structure follows:
JOBFILE Data Source

In this section:

JOBFILE Master File

JOBFILE Structure Diagram

JOBFILE contains sample data about company job positions. Its segments are:

JOBSEG

Describes what each position is. The field JOBCODE in this segment is indexed.

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

<table>
<thead>
<tr>
<th>JOBSEGS</th>
<th>JOBSEG</th>
<th>01</th>
<th>S1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>JOBCODE</em></td>
<td><strong>I</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>*JOB_DESC</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>**</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>**</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>**</td>
<td></td>
</tr>
<tr>
<td></td>
<td>***********</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SECSEG</th>
<th>SKILLSEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>02 I U</td>
<td>03 I S1</td>
</tr>
<tr>
<td>***********</td>
<td>***********</td>
</tr>
<tr>
<td><em>SEC_CLEAR</em></td>
<td><em>SKILLS</em></td>
</tr>
<tr>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>***********</td>
<td>***********</td>
</tr>
</tbody>
</table>

**EDUCFILE Data Source**

**In this section:**

EDUCFILE Master File

EDUCFILE Structure Diagram

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, SECTYPE=S1
  FIELDNAME=COURSE_CODE, ALIAS=CC, FORMAT=A6, $
  FIELDNAME=COURSE_NAME, ALIAS=CD, FORMAT=A30, $
SEGNAME=ATTNDSEG, SECTYPE=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
ATTNDSEG
02 I SH2
****************
*DATE_ATTEND **
*EMP_ID      **I
*              *
*              *
*              *
****************
****************
SALES Data Source

In this section:
SALES Master File
SALES Structure Diagram

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
          DATE_SEG
02 I SH1
**************
*DATE        **
*            **
*            **
*            **
*            **
**************

          I
          I
          I
          PRODUCT
03 I S1
**************
*PROD_CODE  **I
*UNIT_SOLD   **
*RETAIL_PRICE**
*DELIVER_AMT **
*            **
**************
CAR Data Source

In this section:
CAR Master File
CAR Structure Diagram

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, D7, $
    FIELDNAME=RETAIL_COST, D7, $
    FIELDNAME=SALES, 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, $

**LEDGER Data Source**

In this section:

- LEDGER Master File
- LEDGER Structure Diagram

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

In this section:
FINANCE Master File
FINANCE Structure Diagram

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

In this section:
TRAINING Master File
TRAINING Structure Diagram

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

**In this section:**
- COURSE Master File
- COURSE Structure Diagram

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       **
   *               **
   **************
```

Creating Reports With WebFOCUS Language 1459
JOBHIST Data Source

In this section:

JOBHIST Master File
JOBHIST Structure Diagram

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

In this section:

JOBLIST Master File
JOBLIST Structure Diagram

JOBLIST contains information about jobs. The JOBCLASS field is indexed.
JOBLIST Master File

FILENAME=JOBLIST, SUFFIX=FOC
SEGNAME=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

In this section:
LOCATOR Master File
LOCATOR Structure Diagram

JOBHIST contains information about employee location and phone number. The PIN field is indexed.

LOCATOR Master File

FILENAME=LOCATOR, SUFFIX=FOC
SEGNAME=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=BTEL, FORMAT=A5, $
LOCATOR Structure Diagram

SECTION 01

LOCATOR
01   S1
**************
*PIN    **I
*SITE   **
*FLOOR  **
*ZONE   **
*       **
**************
**************

PERSINFO Data Source

In this section:
PERSINFO Master File
PERSINFO Structure Diagram

PERSINFO contains employee personal information. The PIN field is indexed.

PERSINFO Master File

FILENAME=PERSINFO, SUFFIX=FOC
SEGNAME=PERSONAL, SEGTYPE=S1

FILENAME=PIN, ALIAS=ID, FORMAT=A9, INDEX=I, $  
FILENAME=INCAREOF, ALIAS=ICO, FORMAT=A35, $  
FILENAME=STREETNO, ALIAS=STR, FORMAT=A20, $  
FILENAME=APT, ALIAS=APT, FORMAT=A4, $  
FILENAME=CITY, ALIAS=CITY, FORMAT=A20, $  
FILENAME=STATE, ALIAS=PROV, FORMAT=A4, $  
FILENAME=POSTALCODE, ALIAS=ZIP, FORMAT=A10, $  
FILENAME=COUNTRY, ALIAS=CTRY, FORMAT=A15, $  
FILENAME=HOMEPHONE, ALIAS=TEL, FORMAT=A10, $  
FILENAME=EMERGENCYNO, ALIAS=ENO, FORMAT=A10, $  
FILENAME=EMERGCONTACT, ALIAS=ENAME, FORMAT=A35, $  
FILENAME=RELATIONSHIP, ALIAS=REL, FORMAT=A8, $  
FILENAME=BIRTHDATE, ALIAS=BDAT, FORMAT=YMD, $
PERSINFO Structure Diagram

SECTION 01

PERSONAL

01 S1

**************

*PIN  **I
*INCAREOF  **
*STREETNO  **
*APT  **
*  **

**************

SALHIST Data Source

In this section:

SALHIST Master File
SALHIST Structure Diagram

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

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 facility.

**VIDEOTRK Master File**

```plaintext
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

SECTION 01

STRUCTURE OF FOCUS FILE VIDEOTRK ON 05/15/03 AT 12.25.19

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
************** **************
*PRODCODE ** MOVIECODE ** I
*TRANSCODE ** COPY **
*QUANTITY ** RETURNDATE **
*TRANSTOT ** FEE **
*
************** **************

Creating Reports With WebFOCUS Language
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 **
* **
**************
**************

VIDEOTRK, MOVIES, and ITEMS Data Sources
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 contains sample data about customer, rental, and purchase information for a video rental business. It consists of four segments.

**VIDEOTR2 Master File**

FILENANE=VIDEOTR2, SUFFIX=FOC
SEGNAME=CUST,    SECTYPE=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=HYYMDI,      $
SEGNAME=SALES,    SECTYPE=S2,   PARENT=TRANSDAT
   FIELDNAME=TRANSCODE,    ALIAS=TCOD,         FORMAT=I3,          $
   FIELDNAME=QUANTITY,     ALIAS=NO,           FORMAT=I3S,         $
   FIELDNAME=TRANSTOT,     ALIAS=TTOT,         FORMAT=F7.2S,       $
SEGNAME=RENTALS,  SECTYPE=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,       $
A. Master Files and Diagrams

```plaintext
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 TRANSDAT
02 I SH1

***************
*TRANSDATE  **
*           **
*           **
*           **
*           **

***************

I
+-----------------+
I

I SALES I RENTALS
03 I S2 04 I S2

***************  ***************
*TRANSCODE  **  *MOVIECODE  ** I
*QUANTITY  **  *COPY  **
*TRANSLOT  **  *RETNURDATE  **
*           **  *FEE  **
*           *  *

***************  ***************

Creating Reports With WebFOCUS Language 1469
```
Gotham Grinds Data Sources

In this section:
GGDEMOG Master File
GGDEMOG Structure Diagram
GGORDER Master File
GGORDER Structure Diagram
GGPRODS Master File
GGPRODS Structure Diagram
GGSALES Master File
GGSALES Structure Diagram
GGSTORES Master File
GGSTORES Structure Diagram

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 **
*MEDHHI98 **
* 
**************
GGORDER Master File

FILENAME=GGORDER, SUFFIX=FOC,$
SEGNAME=ORDER01, SEGTYPE=S1,$
FIELD=ORDER_NUMBER, ALIAS=ORDNO1, 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 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_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

GGSALES 01  1
**************
*SEQ_NO      **
*CATEGORY    **I
*PCD         **I
*PRODUCT     **I
  **
**************
**************

1474 WebFOCUS
GGSTORES Master File

FILENAME=GGSTORES, SUFFIX=FOC
SEGNAME=STORES01, SECTYPE=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
Century Corp Data Sources

<table>
<thead>
<tr>
<th>In this section:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CENTCOMP Master File</td>
</tr>
<tr>
<td>CENTCOMP Structure Diagram</td>
</tr>
<tr>
<td>CENTFIN Master File</td>
</tr>
<tr>
<td>CENTFIN Structure Diagram</td>
</tr>
<tr>
<td>CENTHR Master File</td>
</tr>
<tr>
<td>CENTHR Structure Diagram</td>
</tr>
<tr>
<td>CENTINV Master File</td>
</tr>
<tr>
<td>CENTINV Structure Diagram</td>
</tr>
<tr>
<td>CENTORD Master File</td>
</tr>
<tr>
<td>CENTORD Structure Diagram</td>
</tr>
<tr>
<td>CENTQA Master File</td>
</tr>
<tr>
<td>CENTQA Structure Diagram</td>
</tr>
<tr>
<td>CENTGL Master File</td>
</tr>
<tr>
<td>CENTGL Structure Diagram</td>
</tr>
<tr>
<td>CENTSYSF Master File</td>
</tr>
<tr>
<td>CENTSYSF Structure Diagram</td>
</tr>
<tr>
<td>CENTSTMT Master File</td>
</tr>
<tr>
<td>CENTSTMT Structure Diagram</td>
</tr>
<tr>
<td>CENTGLL Master File</td>
</tr>
<tr>
<td>CENTGLL Structure Diagram</td>
</tr>
</tbody>
</table>

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**

```plaintext
FILE=CENTCOMP, SUFFIX=FOC, FDFC=19, FYRT=00
SEGNAME=COMPINFO, SECTYPE=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**

```plaintext
SECTION 01

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 MTEXT/MT=MONTH;,$

CENTFIN Structure Diagram

SECTION 01

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',$
Creating Reports With WebFOCUS Language 1481
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 PLANTNG/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 '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 Cateogory:', 
DEFINE PRODTYPE/A19 = IF PRODTYPE CONTAINS 'Digital' OR 'DVD' OR 'QX'
   THEN 'Digital' ELSE 'Analog';,WITHIN='*Product Dimension',
   TITLE='Product Type:',

CENTINV Structure Diagram

SECTION 01

INVINFO
 01   S1
  *************
  *PROD_NUM   **I
  *PRODNAME   **
  *QTY_IN_STOCK**
  *PRICE      **
  *            **
  *************
  *************

Creating Reports With WebFOCUS Language 1483
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/A11=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 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', 
  WITHIN=*Problem,
  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/A11=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, 

Creating Reports With WebFOCUS Language 1487
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=F0C
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', $

Creating Reports With WebFOCUS Language
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
SEGNNAME=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 CMS, the file type is ERRORS.
- For z/OS, the ddname is ERRORS.

Topics:
- Displaying Messages
Displaying Messages

To display the text and explanation for any message, issue the following query command in a stored procedure

```plaintext
? n
```

where:

```plaintext
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

```plaintext
? 210
```

displays the following:

```plaintext
(FOC210) THE DATA VALUE HAS A FORMAT ERROR:
    An alphabetic character has been found where all numerical digits are required.
```
Table Syntax Summary

This appendix summarizes WebFOCUS reporting commands and options.

Topics:
- TABLE Syntax Summary
- TABLEF Syntax Summary
- MATCH Syntax Summary
- FOR Syntax Summary
In this section:
Hierarchical Reporting Syntax Summary

The syntax of a TABLE request is:

```
DEFINE FILE filename    CLEAR|ADD
tempfield  [/[o]format] [WITH realfield] = expression;
tempfield  [/[o]format] REDEFINES qualifier.fieldname = expression;
.
.
.
END
TABLE  FILE  filename
HEADING  [CENTER] "text"
   {display_command}  [SEG.] field  [/R|/L|/C]  [/[o]format]
   {display_command}  [prefixop.] [field]  [/R|/L|/C]  [/[o]format]
   [NOPRINT|AS 'title1,...,title5']  [AND|OVER]  [obj2...obj1024]
   [WITHIN field]  [IN n]
COMPUTE field  [/[o]format] = expression;  [AS 'title,...,title5']  [IN n]
[AND]  ROW-TOTAL  [/R|/L|/C]  [/[o]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  fldl  [/fmt] = FORECAST (fld2, intvl, npredct,
   '{MOVAVE|EXPANE}', npnt);

ON  sfld  Recap  fldl[/fmt] = FORECAST(fld2, interval, npredct, 'DOUBLEXP',
   npoint1, npoint2);

ON  sfld  Recap  fldl[/fmt] = FORECAST(fld2, interval, npredct, 'SEASONAL',
   nperiod, npoint1, npoint2, npoint3);
ON  sfld  Recap  fldl[/fmt] = FORECAST (fld2, intvl, npredct,
   'REGRESS');
```
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:

```plaintext
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., PCT., RPCT., PCT.CNT., FST., LST., CNT., SUM., or TOT. TABLEF requests cannot use prefix operators PCT.CNT., RPCT., and TOT.
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] [FORMAT FOCUS] matchtype]
MORE
FILE file4
subrequest
END
```

where:

```
macthype
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]]```
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 1502.

- **Refer to fields using long and qualified field names.** For details, see *Referring to Fields Using Long and Qualified Field Names* on page 1502.

- **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 1504.

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.

**Topics:**

- Referring to an Individual Field
- Referring to Fields Using Long and 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.

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 Long and Qualified Field Names

How to:
Activate Long and Qualified Field Names

Reference:
Usage Notes for Long and Qualified Field Names

Field names and aliases have a maximum length of 66 characters, including up to two qualifiers and qualification characters. The names you assign to temporary fields may also be up to 66 characters. However, text fields and indexed field names in Master Files are limited to 12 characters, although the aliases for text and indexed fields may be up to 66 characters. 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 1053.
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 66 characters) and qualified field names.

```plaintext
SET FIELDNAME = fieldname
```

where:

- **fieldname**
  
  Specifies the activation status of long and qualified field names. Valid identifiers include:

  - **NEW** specifies that 66-character and qualified field names are supported. The maximum length is 66 characters. NEW is the default value.
  - **NOTRUNC** supports the 66-character maximum. It does not permit unique truncations of field names.
  - **OLD** specifies that 66-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. The maximum of 66 characters includes the name of the field or alias, plus an eight-character maximum for field qualifiers (Master File name and segment name) and two delimiting characters (periods).

**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

In this section:

Listing Field Names, Aliases, and Format Information

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 Command Console in Developer Studio or 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 Command Console in Developer Studio or 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 167
-HTMLFORM 542
-n spot marker 1188
-SET command 316
? DEFINE command 204
? FILTER command 186, 187
? JOIN command 765
? STAT command 132
?F command 1505
?FF command 1505
[F]DEFCENT attribute 333
[F]YRTHRESH attribute 333
* multiplication operator 319, 321
** exponentiation operator 321
- subtraction operator 321
/ division operator 319, 321
/n spot marker 1188
&FOCFIELDNAME variable 1503
% masking character 167, 169, 170
+ addition operator 321
$ masking character 167, 169
$* masking character 167, 169, 171

0X spot marker 1056
3D graphs 1250

508 compliance 1024

A

absolute starting positions 1184
ACCEPT attribute 370, 371
accordion reports 662
Accordion Reports 678, 679, 680, 681
Accordion Reports and ACROSS phrase 678
Accordion Reports and BY phrase 678
Accordion Reports and distributing with ReportCaster 679
Accordion Reports and drill-downs 679
Accordion Reports and HTML report 678
ACROSS attribute 864, 865
ACROSS COLUMNS AND phrase 103, 105, 106
ACROSS phrase 45, 65, 81, 91, 92, 94, 217, 678
ACROSS subtype 864, 875
ACROSS summary commands 306
ACROSS values 269, 270, 874, 876
ACROSS with ROW-TOTAL 266
ACROSS-TOTAL component 269, 270
ACROSSCOLUMN attribute 853, 855, 1042
ACROSSCOLUMN subtype 1042, 1285, 1287
ACROSSLINE parameter 92
ACROSSPRT parameter 95
Index

ACROSSTITLE component 875, 876
ACROSSVALUE and drill-downs 876
ACROSSVALUE component 864, 875, 876
active cache 668
ADD command 53
ADD option 203
ADD parameter 1355
adding attributes to WebFOCUS StyleSheets 652, 653
adding blank lines to headings and footings 1194
adding blank rows 1342
adding blank spaces around a report 942
adding borders for cells 1379, 1383
adding borders for rows 1381, 1382
adding calculated values in financial reports 1344
adding cell borders 1379, 1383
adding color to graphs 1285, 1286, 1290
adding columns to financial reports 1344
adding comments to WebFOCUS StyleSheets 815
adding conditional styling to graphs 1286
adding data to financial reports 1312, 1323
adding declarations to font map files 445
adding dynamic tables of contents to headings 653
adding dynamic tables of contents to reports 640, 641, 642
adding dynamic tables of contents to WebFOCUS StyleSheets 641, 642
adding fields to footings and headings 1289
adding fields to graph footings 1289
adding fields to graph headings 1289
adding fields to headings and footings 1289
adding financial data to reports 1323
adding fonts for PostScript (PS) format 443
adding fonts in PS (PostScript) format 438
adding footings to graphs 1289, 1290
adding headings to graphs 1289, 1290
adding HTML reports to dynamic tables of contents 640
adding HTML tables of contents to headings 653
adding HTML tables of contents to reports 641, 642
adding HTML tables of contents to WebFOCUS StyleSheets 641, 642
adding HTML TOCs (tables of contents) to headings 652
adding images to a report 1016, 1019
adding JavaScript for DRILLMENUITEM 542
adding JavaScript for DRILLMENUITEM attribute 542
adding JavaScript for HTMTABLE 542
adding labels to graphs 1289, 1290
adding line breaks to text fields 1242
adding PostScript fonts 443
adding PostScript Type 1 fonts 443
adding PostScript Type1 fonts in Unix 445
adding PostScript Type1 fonts in Windows 445
adding PostScript Type1 fonts in z/OS 447
adding rows to financial reports 1312, 1315
adding tables of contents (TOCs) to HTML reports 640
adding tag rows to financial reports 1312, 1313, 1351
adding text rows to financial reports 1342
adding TOCs (tables of contents) to HTML reports 640
adding underlines 1008, 1009
adding underlines to columns 1014, 1015
adding underlines to financial reports 1368
adding values 54
adding values for numeric fields 54
adding virtual fields 204
addition operator 321
addressing columns 1333
aggregate values 153
aggregation 138
aggregation and external sorting 139, 140
AHTML display format 659
AHTML format 387
AHTMLTAB format 388
aliases 1414, 1501, 1502, 1505
displaying 1505
aliases and SQL Translator 1414
aligning decimal points 1169, 1172, 1173
aligning footings 1173
aligning heading items 1186
aligning headings 1169, 1173
aligning headings and footings 1147, 1149, 1150, 1157, 1162
aligning sort headings 1173
aligning text fields 1159, 1160
aligning using HTML tables 1157
alignment methods 1147
ALL parameter 153, 717, 728
ALL parameter and JOIN command 727, 728
ALL parameter and missing values 717, 718
ALL parameters 717
ALL prefix 716
ALL prefix and missing values 716
ALLOCATE command 352
ALPHA format 388
alphanumeric fields 165, 166, 167, 168, 169, 170, 171, 172, 173, 280
alphanumeric fields and text fields 254
ALT attribute 1016, 1017, 1024
alternate file views 1428, 1429
alternate indexes 191
alternating background color 1217
alternative methods for wrapping data 1233
amper variables 1287, 1289
AND operator 155, 346
AnV fields 341, 342
APDF display format 388
applying columns to graphs 1240
applying FORECAST command to conditional formatting 840
applying FORECAST command to conditional styling 829
applying graphs to columns 1040
applying graphs to report columns 1040
applying grids to a report 998, 999
applying macros 817, 818
applying selection criteria in graph requests 1278, 1279, 1289
area graphs 1250, 1266
AREXPIRE parameter 677
arithmetic expressions 319
arithmetic operators 319, 321
around headings and footings of border 992
ARPASSWORD parameter 678
AS CAPTION phrase 1351, 1352
AS phrase 353, 365
ascending sort order 100, 101
ASNAMES command 363, 364, 365, 366, 367
ASQ calculations on field values 64
ASQ prefix operators 64
assigning Cascading Style Sheet classes to report components 905, 906
assigning classes in Cascading Style Sheets to ACROSS values 898
assigning classes in Cascading Style Sheets to report components 903, 905, 906
assigning external Cascading Style Sheet classes to report components 906
assigning row labels 1326
assigning row titles 1365
assigning row titles PICKUP rows 1394
assigning row titles RECAP rows 1366
assigning row titles TAG rows 1366
assigning titles to rows 1365, 1366
associating bar graphs with report columns 1040, 1042
associating graphs with columns 1042
associating graphs with report columns 1042
attribute inheritance 819, 820
attributes 520, 532, 535, 562, 813, 1044, 1367
augmenting attributes 820
AUTOINDEX parameter 1431, 1432, 1433
AUTOPATH parameter 1431
AUTOTABLEF parameter 67, 69
AVE field 63
AVE prefix operator 63
AVE prefix operators 63

B
BACKCOLOR 1215, 1217
background color 1215
background color, alternating 1217
background images 1016, 1025
BACKIMAGE attribute 1016, 1025
BAR command 1014, 1015, 1368
bar graphs 1250, 1264
bar rows 1368
base dates 326, 327
BASEURL SET parameter 561
basic date support 1277
basic date support in graphs 1277
BINARY format 389
BINARY HOLD format 389
BINARY output file format 389
BINARY PCHOLD format 389
BINS parameter 131
bipolar graphs 1250
blank lines 1000, 1001, 1002, 1193
blank rows 1342
blank spaces 939
blank spaces above data values 944
blank spaces between columns 945
blanks 707
BLOB image in reports 1026
BODY element 903
Boolean expressions 345, 347
Boolean operator 346
Boolean operators 345
BORDER attribute 991, 992, 993, 995, 1128, 1367
border colors 992
border styles 992
borders 991, 995, 1126, 1379
borders for headings 1128, 1129
borders in PDF reports 1126
bottom margins 935
BOTTOMGAP attribute 939, 942, 1195
BOTTOMMARGIN attribute 936, 937
breaks 968
browser fonts 1213, 1214
browser fonts in HTML Reports 1213
browser support for Cascading Style Sheets 921, 922, 925
browser support for EXL2K display format 457
browser support for HTML tables of contents 657
browser titles 1059
bubble charts 1269
bursting reports 692, 694
BY 87
BY phrase 45, 55, 65, 81, 83, 84, 86, 97, 857, 1321
BY phrase using with financial reports 1321, 1322
BY ROWS OVER phrase 103, 104
BY subtype 857, 867, 880
BY TOTAL phrase 119, 120, 121
BYDISPLAY parameter 86
BYLASTPAGE system variable 979, 983
byte precision 57, 58

C
CACHEFIELDS phrase 480, 482, 483, 487
CACHEFIELDS phrase and PivotTables 484, 487
CACHEFIELDS phrase and PivotTables requests 487
calculated values 194, 196, 217, 218, 262, 266, 276, 277, 278, 279, 1107, 1344
calculated values and native Excel formulas 477
calculated values and row totals 262
calculated values for column titles 1111
calculated values in column totals 266
calculated values in row totals 266
calculating column and row totals 262
calculating column percents 65
calculating column totals 262, 264, 265, 266
calculating dates 328
calculating MAX field values 64
calculating maximum field values 64
calculating maximum values for field values 64
calculating MIN field values 64
calculating minimum field values 64
calculating minimum values for field values 64
calculating row percents 65
calculating row totals 262, 264, 265
calculating trends with FORECAST 229
calculating values for temporary fields 213
calculation on field values 67
calculations 666, 1339
calculations and functions 1339, 1340
calculations counting field values 72
calculations on counting field values 72
calculations on field values 60, 63, 64, 97
calculations on sum numeric field values 72
calculations on SUM numeric field values 72
calculations on TOT field values 72
calculations on total field values 72
calling a spreadsheet from a hyperlink 508, 509
calling a spreadsheet from hyperlinks 508
calling functions 1339, 1340
CAPTION parameter 1350, 1355, 1362, 1363
captions 1059
captions in a hierarchy 1354
captions in Master Files 1350, 1362, 1363
CAR data source 1451, 1452
carriage-returns 1242
Cartesian product 794, 795, 796
Cartesian product answer sets 1417
Cartesian product answer sets and SQL Translator 1417
Cascading Style Sheet class for ACROSS values 898
Cascading Style Sheet classes 893, 903, 922
Cascading Style Sheet FAQS 922
Cascading Style Sheet rules 893, 898, 903, 922
Cascading Style Sheets (CSS) 803, 826, 827, 891, 892, 901
Cascading Style Sheets (CSS) browser support 921, 922, 925
Cascading Style Sheets (CSS) example 899
Cascading Style Sheets (CSS) images 921
Cascading Style Sheets (CSS) multiple output formats 916, 917, 919
Cascading Style Sheets (CSS) report formatting 895, 897
Cascading Style Sheets (CSS) requirements 921
Cascading Style Sheets (CSS) troubleshooting 925
Cascading Style Sheets (CSS), benefits 894
Cascading Style Sheets (CSS), choosing 902
Cascading Style Sheets (CSS), multiple 902
CDN (Continental Decimal Notation) 1223, 1417
CDN (Continental Decimal Notation) and SQL Translator 1418
CDN parameter 1223
cell formatting 1367, 1370, 1373, 1379
cell notation 1338
cells 1338
CENTCOMP data source 1478
CENTER command 1130
CENTFIN data source 1476
CENTGL data source 1487, 1488
CENTGLL data source 1490, 1491
CENTHR data source 1476
CENTINV data source 1476
CENTORD data source 1476
CENTQA data source 1476
CENTSTMT data source 1489, 1490
CENTSYSF data source 1488
Century Corp data sources 1476, 1478, 1479, 1480, 1482, 1483, 1484, 1485, 1486, 1487
changing color settings 1289
changing column order 953, 954
changing fonts 1289
changing row titles 1365
changing row titles PICKUP rows 1394
changing row titles RECAP rows 1366
changing row titles TAG rows 1366
character expressions 317, 338
character strings 165, 340
chart legend 672
chart of accounts hierarchies 1348, 1350
charts 1250
charts of accounts 1351
CHECK FILE command 764
CHECK FILE command and join structures 763, 764
CHECK PICTURE command 49
CHECK STYLE command 810, 815
choosing formatting reports style sheets 803
choosing report formatting style sheets 803
CLASS attribute 898, 905, 906, 922
classes in Cascading Style Sheets 893
clearing conditional join structures 766, 767
clearing join structures 766, 767
clearing virtual fields 204, 205
CLIP format 389
CLIP HOLD format 389
CLIP output file format 389
CMS requirements 131
CNOTATION SET parameter 218, 219, 1335
CNT prefix operator 72
collapsing PRINT with ACROSS 95
COLOR attribute 458, 1209
color graph settings 1306
color settings 934
color settings in graphs 1306
color values 1211
Index

color, background 1215
color, background, alternating 1217
COLSPAN attribute 1162
column addresses 1333, 1334
column and row totals in calculated values 262
COLUMN attribute 853, 854, 863, 1042
column formats 1219
column formatting 1370
column notation 218, 219, 1335
column numbers 1331
column reference numbers 218, 1335
COLUMN subtype 1139, 1236, 1371
column titles 77, 1107, 1109, 1219
column titles for calculated values 1111
column total labels 1114
column totals 262, 265, 266
column values 1336
column width 467
column width and EXCEL 468
column width and Excel 2000 468, 469
COLUMN-TOTAL phrase 262, 264
columns 853, 1218, 1370
   formatting 1370
columns in financial reports 1312
columns numbers 1331
COM format 390
COM HOLD format 390
COM output file format 390
COM PCHOLD format 390
COM SAVE format 390

combination of summary commands 300
combinations of subtotals 299
combining expressions 156
combining external Cascading Style Sheets with other formatting methods 907, 908
combining fields in date expressions 330, 331
combining mixed format reports 604, 605, 607, 608
combining multiple values 1315
combining PDF reports 604, 605
combining range of records 1317
combining range of values 1317, 1318
combining records 1315, 1316
combining report formats 608
combining report formatting Cascading Style Sheets and other formatting methods 922
combining values 1315, 1316, 1318
COMMA format 390
COMMA HOLD format 390
COMMA output file format 390
COMMA SAVE format 390
comma-delimited files 726
command support for Accordion Reports 680
commands 816
commands in PivotTables 482
comments 812, 815
comments in WebFOCUS StyleSheets 812, 815
common high-order sort fields 776, 777, 779, 790, 791, 793

1514
comparing characters with masks 167, 168, 169, 170, 171
comparing decimal values 1172
comparing records 773, 774, 775
compiling calculated values 1438
compiling expressions 1436, 1437
compiling virtual fields 1436
complex expressions 315
COMPMISS parameter 711, 1221
compound display formats for reports 565, 604
Compound Excel reports 515
compound expressions 155, 156
COMPOUND parameter 604
compound report display formats 565, 604
Compound report syntax 566, 567, 568, 569, 570, 578, 581, 586
COMPONENT 569
displaying a grid 586
draw objects 581
example 570
page overflow 578
PAGELAYOUT 568
SECTION 567
compound reports 400, 565, 604
compound reports in PDF format 605
compressing PDF output files 440
COMPUTE command 55, 213, 214, 215, 217, 276, 295, 308, 309, 311, 316
COMPUTE command expressions 316
COMPUTE component 872
computing the average field values 63
COMT format 391
COMT HOLD format 391
COMT output file format 391
COMT PCHOLD format 391
COMT SAVE format 391
concatenated data sources and MATCH FILE command 788
concatenating character strings 340
concatenating data sources 783, 784, 785
concatenating data sources and field names 787
concatenation 340
concatenation data sources 785, 788
concatenation for data sources 783
concatenation operators 340
concatenation usage formats 786
conditional drill-down 554, 555
conditional drill-down and multiple links 540
conditional expression types 347
conditional expressions 317, 347, 348, 349
conditional formatting 1088
conditional formatting and WHEN phrase 841, 843, 844, 1088
conditional grid formatting 1225, 1226
conditional join structures 724, 726, 735, 753, 754, 755, 762
conditional operators 157, 162, 163
conditional sort headings 1083
conditional styling 540, 825, 828, 841, 843, 844, 921
conditional styling and graphs 1285, 1286
conditional styling and style sheets 831, 838
conditional styling and WebFOCUS StyleSheets 828, 829, 831, 833, 834, 835, 836, 838
conditional styling in Cascading Style Sheets (CSS) 921
conditional text 313, 314
conditionally displaying page breaks 841
conditionally displaying skipped lines 841
conditionally displaying summary lines 841
conditionally displaying underlines 841
configuring PostScript fonts 445, 447
configuring PostScript fonts in z/OS 447
configuring PostScript Type 1 fonts 445
connected point plot 1263
connected point plot graphs 1250, 1263
consolidating financial data 1351, 1354, 1355, 1357, 1358
    single row 1357
consolidating financial data in multiple rows 1358
consolidating financial data single row 1355
constant dates 325, 329, 330
constants 1324
constants in date expressions 329
constants in Financial Modeling Language (FML) 1323
constants in financial reports 1323
constants in FML (Financial Modeling Language) 1323
constants in FML requests 1323, 1324
CONTAINS operator 165, 166
contiguous columns 1332
contiguous columns in financial reports 1332
Continental Decimal Notation (CDN) 1223, 1417
Continental Decimal Notation (CDN) and SQL Translator 1417, 1418
controlling attributes 363, 368, 369, 370, 371
controlling attributes and HOLD Master Files 369
controlling column order 946, 953
controlling column reference numbers 218, 1335
controlling column spacing 946, 952, 953
controlling column width 467, 469
controlling column width Excel 2000 467
controlling column width in EXCEL 2000 469
controlling display of sort field values 678
controlling field names 364, 365, 366, 367
controlling field names HOLD Master Files 364
controlling fields 368
converting data types for join structures 753
converting formulas to Excel 2000 474
converting TABLE requests 1247, 1248
converting TABLE requests to GRAPH requests 1247, 1248
converting totals to Excel 2000 formulas 474
COUNT * command 56, 57
COUNT command 55, 56, 82
COUNT command for unique segments 56
count of occurrences 72
counting field values 55
COUNTWIDTH SET parameter 55, 57, 58
COURSE data source 1459
CREATE TABLE command 1415, 1416
CREATE VIEW command 1416, 1417
creating bar graphs 1255, 1256, 1257
creating calculated values 196, 213, 214, 215
creating EXL2K reports 458
creating financial reports 1309, 1315
creating FOCUS data sources 358, 361, 362, 363
creating free-form reports 1398, 1400, 1402, 1403, 1404
creating graphs 1246
creating HOLD files 353, 354, 355, 356, 358, 359, 361
creating horizontal bar graphs 1255, 1256
creating links 520
creating matrix report with OVER 955
creating matrix reports with OVER 956, 957
creating multiple virtual fields 203
creating numeric expressions 319
creating output files 353
creating parameters 544, 546, 548, 550
creating parameters for drill-down reports 544
creating parameters for drill-downs 1289
creating PCHOLD files 385
creating pie graphs 1258, 1259
creating RECAP expressions 1324
creating reports 32, 33, 35, 37, 38, 458
  financial 33
  free-form 33
creating reports for EXL2K display format 458
creating rows 1312, 1313, 1314
creating rows from multiple records 1315
creating rows in financial reports 1312
creating SAVB files 383, 384
creating SAVE files 381, 382, 383
creating scatter graphs 1259, 1260
creating single-line footings 1068
creating single-line headings 1064, 1072
creating sort headings 1080
creating style sheets 808, 809
creating tag rows 1313, 1314
creating temporary fields 35
creating temporary fields with COMPUTE phrases 213
creating temporary fields with DEFINE FUNCTION 255, 256, 258
creating vertical bar graphs 1256, 1257
creating virtual fields 196, 197, 198, 199, 201, 203, 207
creating Web Query (.iqy) file 493, 494
creating WebFOCUS reports for Excel 2000 492, 503
creating WebFOCUS reports for Excel 97 494
creating WebFOCUS StyleSheets 808, 809
cross-century dates 329
cross-referenced fields 745
cross-referenced files 727, 735
CSS (Cascading Style Sheets) 803, 826, 827, 891, 892, 901
CSS (Cascading Style Sheets) browser support 921, 922, 925
CSS (Cascading Style Sheets) example 899
CSS (Cascading Style Sheets) images 908, 921
CSS (Cascading Style Sheets) multiple output formats 916, 917, 919
CSS (Cascading Style Sheets) report formatting 895, 897
CSS (Cascading Style Sheets) requirements 921
CSS (Cascading Style Sheets) troubleshooting 925
CSS (Cascading Style Sheets), benefits 894
CSS (Cascading Style Sheets), choosing 902
CSS (Cascading Style Sheets), editing 903
CSS (Cascading Style Sheets), multiple 902
CSS location 902
CSSURL (StyleSheets) attribute 898
CSSURL attribute 898, 902, 909, 910, 911
CSSURL attribute example 911
CSSURL attribute HTMTABLE format 909
CSSURL parameter 898, 902, 909, 910, 913
CSSURL parameter HTMTABLE format 909
CT. prefix operator 1093
custom report titles 1059
custom sort order 103
custom worksheet names 1059
customizing column titles in a Master File 1112
customizing graphs 1275, 1285, 1286, 1289, 1290, 1291, 1292, 1295, 1296, 1298, 1299
customizing reports 39, 1402
customizing sort order 103

D
data 861, 862
DATA component 861, 862
data descriptions 32
data extraction 353, 356
data fields 167
data formats 1218
data formats for cross-referenced fields 751
data formats for host fields 751
data formats for join structures 751
data report component 860
data retrieval 148, 150, 1312, 1393, 1394, 1427, 1428, 1429, 1431, 1432, 1433, 1435
data retrieval and TABLEF command 1434, 1435
data retrieval using TABLEF command 1434
data source 49
data source types 32
data sources 35, 770, 783, 1441
joining 35
merging 770
data structures 1430
data type conversion 752
data type conversions 752, 753
data types 255, 752, 753
data visualization 1040, 1042, 1046, 1240, 1241
formatting 1046
data visualization and conditional formatting 835
data visualization and WebFOCUS StyleSheets 1040, 1042, 1241

data visualization controlling scaling 1043, 1044, 1045

data wrapping 467

date constants 325

date expressions 325, 331

date fields 325

date formats 1419, 1420

date formats and SQL Translator 1419, 1420

date formats not supported in Excel 2000 466

date separators 467

date support 1277

date support in graphs 1277

date value formats 326, 327

date values 326

date-time data types 332

date-time expression types 325

date-time expressions 325

date-time field formats 332

date-time format 336

date-time format and display fields 332

date-time values 325, 1421, 1422, 1423

date-time values and SQL Translator 1421, 1422, 1423

DATE FORMAT parameter 331, 332
DATE FORMAT setting 332

dates 328

dates in graphs 1277, 1278

DB2 format 392

DB2 HOLD format 392

DB2 output file format 392

DBAFILE attribute 727

DBASE format 392

DBASE HOLD format 392

DBASE output file format 392

decimal points 1169, 1172, 1173

decimal values 1169, 1173

declarations 812, 813, 814

declaring filters 183

default browser type 1213

default column titles 1107

default font types 1212

default proportional fonts 1212

default StyleSheet values 847

DEFAULT- FIXED attribute 1213

DEFAULT- PROPORTIONAL attribute 1213

DEFINE and dates 209, 211, 329

DEFINE attribute 316

DEFINE command 199, 201, 203, 316

DEFINE command and join structures 747, 748, 749, 750, 757, 758, 760

DEFINE command and missing values 700, 701, 702, 703, 704

DEFINE command expressions 316

DEFINE compiler 1436, 1437

DEFINE FILE RETURN command 760

DEFINE FILE SAVE command 208, 760
Index

- DEFINE FUNCTION command 255, 256, 258
- DEFINE function deleting 255
- DEFINE function displaying 255, 258
- DEFINE function limitations 255, 257
- DEFINE function querying 255
- DEFINE functions 255, 256, 258
- DEFINE parameter 208
- defining custom groups 111, 112
- defining field formats 1219
- defining filters 182, 183, 184
- defining macros 818
- defining virtual fields 201
- DEFMACRO command 816, 818
- DEFMACRO commands 816
- DELETE command 1426
- deleting DEFINE functions 259
- deleting underlines 1008, 1009
- delimited file, creating 409
- delimited output files 392
- deploying a Web query to a Web server 508, 509
- descending sort order 100, 102
- designating missing values 720
- determining column width 946, 947, 948, 949, 950, 951
- DFIX 409
- DFIX format 392
- DFSORT utility 130, 131
- DHTML HOLD format 393
- DHTML output format 393
- Dialect Translation 1406
- Dialogue Manager 316, 1308
- Dialogue Manager variables 1093, 1503
  - FOCFIELDNAME 1503
- Dialogue Manager variables in footings 1093
- Dialogue Manager variables in headings 1093
- Dialogue Manager variables in headings and footings 1093, 1104, 1105
- DIF format 393
- DIF output file format 393
- DIF PCHOLD format 393
- DIF SAVE format 393
- direct percent 67
- direct percent of counts (PCT.CNT) 67
- display ADD command 53
- display commands 43, 82, 780
- display commands and graph format 1253
- display commands and MATCH FILE command 780
- display COUNT * command 56
- display COUNT command 55, 82
- display DOC formats for reports 454
- display field values 45, 47
- display fields 60, 61, 1220
- display fields and graph format 1253
- display fields for prefix operators 60
- display formats for compound reports 608
- display formats for EXCEL reports 431, 455
- display formats for EXL2K PIVOT reports 431
- display formats for EXL2K reports 431, 432, 455, 457
display formats for HTML reports 432
display formats for PDF reports 432, 438, 439
display formats for PostScript reports 432, 438, 440, 442
display formats for reports 429, 430, 431, 432, 433, 477, 480
display formats for reports for Excel 97 490
display LIST * command 47
display LIST command 82
display LIST commands 43, 45
display PRINT * command 47
display PRINT command 82
display PRINT commands 43, 45
display SUM command 53, 54, 82
display SUM commands 43
display values 43
display WRITE command 53
displaying ADD command 53
displaying all fields in a segment 48
displaying all fields in segments 47
displaying captions 1351
displaying children 1351, 1353, 1354
displaying compound reports 604
displaying dates 465
displaying dates for EXL2K display format 464
displaying dates in EXL2K 465
displaying empty reports 846
displaying error messages 135, 136
displaying excluded values 87
displaying field descriptions 77
displaying field names 1505
displaying field values 43, 45, 46, 47
displaying FML hierarchies 1348, 1351
displaying formats for reports for EXL2K FORMULA 474
displaying formatted dates 464, 465
displaying grand totals 270, 271
displaying graph data 1270
displaying hierarchies 1348
displaying hierarchy values as captions 1354
displaying HOLD Master Files 353, 356
displaying join structures 763, 764, 765
displaying LIST * command 47
displaying LIST command 45
displaying LIST commands 43
displaying missing values 1279, 1282, 1289 in graphs 1282
displaying missing values in graphs 1280, 1281, 1283, 1284, 1289
displaying multiple graphs in columns 1275
displaying numeric values 464
displaying numeric values in EXL2K 464
displaying parents and children 1351, 1353, 1354
displaying PRINT * command 47
displaying PRINT command 45
displaying PRINT commands 43
displaying reports 143, 429, 430, 431, 432, 435, 438, 439, 440, 454, 455, 467, 468, 474, 475, 477, 479, 490
Index

displaying reports in a browser 435
displaying reports in PDFs 565, 604
displaying reports in PostScripts 565, 604
displaying reports in WebFOCUS Viewer 683
displaying retrieval order 49
displaying retrieval order for multi-path data sources 51
displaying structure for multi-path data sources 49
displaying sub-totals 274, 275, 276
displaying subtotals 270, 271, 272, 273, 274, 275, 276
displaying SUM command 53
displaying summary lines 314
displaying values 43
displaying virtual fields 204
displaying WRITE command 53

distinct prefix operators 67, 68
distinguishing virtual fields and calculated values 196
division operator 319, 321
DOC format 394, 431, 454
DOC output file format 394
DOC PCHOLD format 394
DOC report display formats 454
DOC SAVE format 394
double exponential smoothing 241, 242
double exponential smoothing FORECAST 241, 242
drill through compound reports in PDF format 623, 624, 625, 626, 628, 629, 630, 634, 637
drill through PDF compound reports 623, 624, 625, 626, 628, 629, 630, 634, 637
drill through reports 623, 624, 625, 626, 628, 629, 630, 634, 637
drill-down reports 519, 520, 534, 542, 554, 555
drill-down reports compared 624
drill-downs 534, 1289
drill-downs and ACROSSVALUE 876
drill-downs and graphs 1287
drilling down 1287
drilling down graphs 1287, 1288
drilling down reports 520

DRILLMENIITEM attribute 535, 542
DRILLTHROUGH command 626
DROP VIEW command 1416
DROP VIEW command and SQL Translator 1417
DST prefix operator 68
DST prefix operator restrictions 67
DST prefix operators 67
duplicate field names as column titles 1112, 1113
DUPLICATECOL parameter 125
dynamic reformatting 1237, 1238
dynamic reporting 1348
dynamic tables of contents 639, 640
dynamic tables of contents for multiple sort groups 645
dynamically formatting virtual fields 209, 211, 212
E

editing Cascading Style Sheets 903
editing font map files in Windows 445
editing font map files in z/OS 447
editing metrics files in Windows 445
editing metrics files in z/OS 447
EDUCFILE data source 1447, 1448
ELEMENT attribute 1047, 1048
elements in footings 1093
elements in headings 1093
embedded fields 886
embedded fields in footings 886
embedded quotation marks 339, 340
embedding compound reports graphs 607
embedding compound reports images 607
embedding Excel 97 format for WebFOCUS reports 492
embedding graphics 607
embedding images 607
embedding reports in Excel 2000 501, 503
embedding reports in Excel 97 492, 494, 501
embedding reports in Excel spreadsheets 492, 501, 508
embedding reports in spreadsheets 492, 494, 501, 503
EMPDATA data source 1457
EMPLOYEE data source 1442, 1444, 1445
empty reports 846
EMPTYREPORT SET parameter 846
END command 37
ending a report request 37
EQ operator 165, 346
equijoins 724, 726, 737
error files 1493
error messages 1493, 1494
escape characters 171, 172, 173
ESSBASE hierarchies 1348
establishing segment locations 207
estimating number of records 133
ESTLINES parameter 133
ESTRECORDS parameter 133
evaluation of temporary fields 195
ex_forecast_dist 250
ex_forecast_mov 249
ex_forecast_mult 248
ex_regress_mult 253
Excel 2000 457, 517
TOCs (tables of contents) 517
Excel 2000 alignment 1149
Excel 2000 and PivotTables 480
Excel 2000 column width 467
Excel 2000 data wrapping 468
Excel 2000 displaying formatted dates 464, 465
Excel 2000 numeric values 464
EXCEL 2000 report display formats 455
Excel 2000 TOCs (tables of contents) 517
Excel 97 490
Excel 97 and Excel 2000 490
Excel 97 and Internet Explorer 490
Excel 97 format 490, 492
Excel 97 limitations 492
Excel 97 unsupported date formats 492
EXCEL data wrapping 468
EXCEL display format 431, 455
EXCEL display format and data wrapping 468
EXCEL format 394
Excel formats 431
Excel formats supported with AnV fields 394
excel locking 471
Excel Named Ranges 514, 515
EXCEL output file format 394
EXCEL PCHOLD format 394
EXCEL report display format 431
EXCEL report display formats 455, 468, 474
Excel reports 517
EXCEL SAVE format 394
Excel workbook 514, 515
EXCEPT operator 1418
EXCLUDES operator 174, 175
excluding missing values from tests 708
existing data 165
EXL2K alignment 1149
EXL2K and tables of contents (TOCs) 460
EXL2K and TOCs (tables of contents) 460
EXL2K display format 395, 431, 432, 455, 457, 459, 466, 474, 475, 476, 517
saving reports 474
TOCs (tables of contents) 517
unsupported numeric formats 466
EXL2K display format and drill-downs 459, 460
EXL2K display format and font support 459
EXL2K display format and formatted dates 464, 465, 492
EXL2K display format and Internet Explorer 458
EXL2K display format and numeric values 464
EXL2K display format and separators 467
EXL2K display format and TABLE syntax 483
EXL2K display format column width 467
EXL2K display format displaying dates 465
EXL2K display format tables of contents 459
EXL2K display format TOCs (tables of contents) 517
EXL2K display formats for reports 475
EXL2K format and column totals 475
EXL2K FORMULA 396
EXL2K FORMULA display format 396, 431, 455, 474
EXL2K FORMULA display formats for reports 455, 476
EXL2K FORMULA report display formats 455, 474, 476
EXL2K output file format 395
EXL2K output formats 457
EXL2K PCHOLD format 395
EXL2K PIVOT 396
EXL2K PIVOT display format 431, 455, 480, 484, 486
EXL2K PIVOT display format and CACHEFIELDS phrase 487
EXL2K PIVOT display format and toolbars 481
EXL2K PIVOT display formats for reports 455
EXL2K PIVOT format 396
EXL2K PIVOT report display format 431
EXL2K PIVOT report display formats 455
EXL2K report display format 431
EXL2K report display formats 455, 457, 458, 467, 476
EXL2K SAVE format 395
EXL97 display format 396, 490
EXPANDABLE command 678, 680, 681
expandable reports 662
expanding byte precision 58
expanding byte precision for COUNT command 57
expanding precision 58
explicit labels 1326, 1327, 1328
EXPN and numeric functions 320, 321
EXPN function 320, 321
exponential moving average 229, 230, 231, 239
exponential moving average FORECAST 229, 231, 239
exponentiation operator 321
exporting from data sources 353, 356
expression dates 317, 329
expression types 317
expressions 194, 315, 316, 1419
expressions and SQL Translator 1419
expressions IF phrase 316
expressions, relational 347

EXTAGGR parameter 138
extending heading and footing code 1056, 1057
extending underlines 1010
external Cascading Style Sheet class for ACROSS values 898
external Cascading Style Sheet class names 904
external Cascading Style Sheet classes 893, 903, 922
external Cascading Style Sheet rules 893, 898, 903, 922
external Cascading Style Sheet rules BODY element 903
external Cascading Style Sheet rules TD element 903
external Cascading Style Sheets 803, 891, 892, 901
external Cascading Style Sheets and other formatting methods 907
external Cascading Style Sheets and WebFOCUS StyleSheets 908
external Cascading Style Sheets browser support 921, 922, 925
external Cascading Style Sheets hyperlinks 908
external Cascading Style Sheets images 921
external Cascading Style Sheets multiple output formats 916, 917, 919
external Cascading Style Sheets report formatting 895
external Cascading Style Sheets requirements 921
external Cascading Style Sheets troubleshooting 925
external Cascading Style Sheets, benefits 894
external Cascading Style Sheets, choosing 902
external Cascading Style Sheets, editing 903
external Cascading Style Sheets, linking to 909
external Cascading Style Sheets, multiple 902
external files 1321
external report formatting Cascading Style Sheets 897
external sorting 130, 132, 135, 136
external sorting and aggregation 138, 139
external sorting and HOLD files 141
external sorting by aggregation 138, 139, 140
external sorting requirements 131
EXTHOLD parameter 141
extract files 352, 376, 379, 380, 381, 417
extract files and missing values 709
EXTRACT function 1423
extracting date components 330
EXTSORT parameter 130, 132
EXTUNDERLINE attribute 1010

field names 365, 366, 367, 1418, 1501, 1502, 1503, 1505
   aliases 1501, 1502
displaying 1505
long 1502, 1503
qualified 1501, 1502, 1503
truncated 1501, 1502
field names and concatenating data sources 787
field names and SQL Translator 1418
field names and universal concatenation 786, 787
field padding 414, 415
field references for COMPUTE command 216
field reformatting and missing values 711, 1221
field types not supported for EXCEL format 394
field values 43, 97, 1083, 1095
field values in headings and footings 1093, 1094, 1095
field-based reformatting 209, 211, 212, 1237, 1238, 1239

FIELDNAME command 1503
fields 194, 318, 1219, 1501, 1502
   in report requests 1501, 1502
file location of CSS (Cascading Style Sheets) 902
file location of external Cascading Style Sheets 902
file names 520, 522

FILECOMPRESS parameter 440
FILEDEF command 352, 374
FILTER parameter 182, 183, 184, 185, 186
FILTER query command 182, 186, 187
filters 182, 184, 185, 186, 187, 188
FINANCE data source 1455
financial data 1354
financial data values 1312
Financial Modeling Language (FML) 856, 1308, 1500
Financial Modeling Language (FML) and Dialogue Manager 1308
financial report hierarchies 1351
Financial Report Painter 33
financial reports 33, 797, 1308, 1348
financial reports and external files 1321
financial reports and HOLD files 1395
financial reports and inter-row calculations 1325, 1326
financial reports and recursive models 1346, 1347
financial reports charts of accounts 1348, 1351, 1353, 1354
financial reports hierarchies 1355, 1362, 1363, 1364
financial reports inserting text rows 1342
financial reports inserting variables in text rows 1343
financial reports records in multiple rows 1320
financial reports repeating rows 1328
financial reports saving intermediate results 1391
financial reports sorting with BY 1321
financial reports sorting with FOR 1321
financial reports supplying data as constants 1323
fixed scales 1291, 1296
FIXRETRIEVE parameter 372, 373
FLEX display format 396
FML (Financial Modeling Language) 856, 1014, 1308, 1309, 1500
FML (Financial Modeling Language) and Dialogue Manager and 1308
FML hierarchies 1348, 1350, 1351, 1353, 1354
FML hierarchies indenting captions 1387
FML hierarchies indenting row titles 1386
FML hierarchies indenting text or numbers 1367
FML hierarchies loading into memory 1362, 1363
FML Painter 33
FOCEXEC attribute 520, 521
FOCEXEC attributes 520
FOCFIELDNAME variable 1503
FOCFIRSTPAGE SET parameter 979, 980, 986, 987
FOCHTMLURL parameter 79, 679
pop-up field descriptions and 79
FOCPOST files 1391
FOCUS data sources 145, 358
FOCUS file structure 358, 360
FOCUS format 397
FOCUS StyleSheets 1367
FOLD-LINE command 946, 955, 956
FONT attribute 1212, 1213
font attributes 1122, 1192
font colors 1206, 1209, 1215, 1217
font file 445, 447
font files 443
font inherited styles 1209
font map file entries 449
font map files 443, 444, 445
font sizes 1206, 1207
font styles 1208
fonts 443, 1120, 1205, 1206, 1212
fonts and labels 1122
fonts and system variables 1124
fonts and titles 1122
fonts in headings and footings 1122, 1123
fonts in HTML reports 1213
fonts in HTML Reports 1213
footing code 1057
FOOTING command 1073, 1130, 1402
FOOTING component 879, 880
footing limitations 1055
footings 873, 1053, 1054, 1069, 1073
footings for bursted reports 693
footings for graphs 1290
FOR phrase 67, 69, 103, 111, 112, 1308, 1309, 1313, 1321, 1322, 1500
syntax 1500
FOR phrase reusing values 1319
FORECAST 233, 247, 248, 249, 250
FORECAST command 831
FORECAST double exponential smoothing 241, 242
FORECAST exponential moving average 229, 231, 239
FORECAST limit 234
FORECAST linear regression analysis 229, 231
FORECAST linear regression equation 245, 246
FORECAST processing 230
FORECAST simple moving average 229, 231, 234, 235, 237
FORECAST triple exponential smoothing 242, 244
format ALPHA 388
format dates 326, 327
FORMAT DFIX 409
format DHTML 393
FORMAT EXL2K PIVOT command 480, 484, 486
FORMAT EXL97 command 491, 492
formatting blank lines 1003, 1004
formatting carriage-returns for text fields 1242
formatting Cascading Style Sheets (CSS) 907
formatting cells 1367, 1370, 1373
formatting cells and labels 1373
formatting columns 1218, 1219, 1220, 1367, 1370, 1374
formatting dates in graphs 1278
formatting fields 318, 329
formatting financial reports 1014, 1015, 1367, 1368, 1370, 1373
formatting footings 1120
formatting free text 1370, 1376, 1377
formatting graphs 1261, 1263, 1264, 1265, 1266, 1268, 1269, 1272
formatting heading and footing lines 1169
formatting headings 1120
formatting HOLD files 353
formatting in Cascading Style Sheets (CSS) 905
formatting in CSS (Cascading Style Sheets) 905
formatting labels 1120
formatting line-feeds for text fields 1242
formatting options for financial reports 1367, 1370
formatting output files 386
formatting PCHOLD files 385
formatting RECAP rows 1378
formatting reports 797, 798, 799, 891, 895, 897, 922, 1240, 1241, 1410
formatting reports and SQL Translator 1410
formatting reports in FML (Financial Modeling Language) 1014, 1015
formatting rows 1367, 1370, 1371, 1372, 1375
formatting rows and labels 1370, 1371, 1372
formatting SAVB files 383
formatting SAVE files 381, 382
formatting skipped lines 1003, 1004
formatting text 1377
formatting text fields 1160, 1242
formatting text rows 1376
formatting titles 1120
FORMULTIPLE parameter 1319, 1320
free text 873, 1342, 1376
free text formatting 1375
free text in Financial Modeling Language (FML) 877
free text in FML (Financial Modeling Language) 877
free-form reports 33, 797, 922, 1169, 1173, 1397, 1404
FREETEXT component 877, 878, 1376, 1377
freeze columns 669
freezing HTML headings 1077
FROM ... TO operator 162, 163
FST prefix operator 70, 71
functions 315, 1339
FYRTHRESH attribute
date-time data type and 332
G
gantt charts 1269
GE operator 163, 164, 346
generating calculated values as EXL2K formulas 477
generating calculated values for EXL2K FORMULA display format 477
generating formulas in Excel 2000 475
generating native Excel formulas 476
generating native formulas for EXL2K display format 474
generating PivotTables 486
generating row totals with EXL2K FORMULA display format 476
generating TABLEF commands 1425
GET CHILDREN parameter 1351, 1355
GGDEMOG data source 1470
GGORDER data source 1470
GGPRODS data source 1470
GGSALES data source 1470
GGSTORES data source 1470
GIF files 397
GIF format 397, 399
Gotham Grinds data sources 1470, 1471, 1472, 1473, 1474, 1475
grand totals 270, 1116
GRANDTOTAL component 857, 858, 866, 867, 869
GRAPH command 1247
GRAPH command compared to TABLE command 1247, 1248
Graph Editor 1298
graph footings 1289, 1290
graph formats 1253, 1261
graph formats and display commands 1253
graph formats and display fields 1253
graph formats and sort phrases 1253
graph formatting 897
graph formatting BODY element 903
graph formatting in external Cascading Style Sheets 905, 922
graph formatting TD element 903
graph formatting with external Cascading Style Sheets 897
graph headings 1289, 1290
graph height 1291, 1292, 1293
graph SET parameters 1275, 1292
graph styling 1289, 1291
graph titles 1059
graph types 1250
graph width 1291, 1292, 1293
GRAPHCOLOR attribute 835, 1240, 1241
GRAPHEDIT command 1298
graphic elements 557, 559
graphics 557, 1106
graphics in footings 1106
graphics in headings 1106
GRAPHLENGTH attribute 1240, 1241
graphs 33, 1245, 1246, 1247, 1274
displaying multiple graphs in columns 1274
graphs and missing values 1279, 1281, 1282, 1283, 1284
graphs and WebFOCUS StyleSheets 1285
GRAPHSCALE attribute 1043, 1044, 1045
GRAPHSERVURL SET parameter 1301, 1302, 1303
GRAPHSTYLE * parameter 1296, 1298, 1299
GRAPHTYPE attribute 835, 1040, 1042, 1240
GRAPHWIDTH attribute 1240, 1241
GRID attribute 992, 996, 997
GRID parameter 1233
grids 991, 992, 996, 998, 999, 1126
GRMERGE parameter 1273, 1278
group fields 746
group fields and join structures 746
group key values 175, 176
grouping numeric data 109, 110, 111, 112, 113
grouping numeric data into tiles 113, 114, 115, 116, 117
grouping sort fields 640, 645
groups of values 1318
GRWIDTH parameter 1274, 1275
GT operator 163, 164, 346
GTREND parameter 1271, 1272
GUTTER attribute 1047, 1048

H

H data type 332
HAUTO parameter 1296
HAXIS parameter 1291, 1292
HEADALIGN attribute 1149, 1150, 1151, 1155
heading code 1057
HEADING command 1070, 1130, 1402
HEADING component 879, 880, 881
heading limitations 1055
headings 873, 1053, 1054, 1069
headings and footings for HTML index pages 693
headings for bursted reports 693
headings for graphs 1289, 1290
helper applications 433
HGRID attribute 992, 998, 999, 1127
hiding columns 669, 962, 963, 965
hiding display fields 961
hiding fields 962, 963, 965
hiding fields in y-axis 1271
hiding rows 1390, 1391
hiding sort field values 122, 123, 678
hiding y-axis fields 1271
hierarchies 1348, 1355, 1363, 1364
hierarchies in MASTER Files 1363
hierarchy of sort fields 641
high-order sort fields 776, 777, 779, 790, 791, 793
HMAX parameter 1291, 1296
HMIN parameter 1291, 1296
HOLD AT CLIENT command 353, 385
HOLD command 352, 353
HOLD file INTERNAL format 416
HOLD file keys and indexes 381
HOLD file structured 417
HOLD file suppressing field padding 414, 415
HOLD file text fields 407, 408
HOLD files 352, 353, 359, 372, 373, 417, 770
HOLD files and external sorting 141
HOLD files and merge phrases 773, 774, 775
HOLD files and missing values 709
HOLD files for financial reports 1395
HOLD files text fields 386
HOLD format ALPHA 388
HOLD format DFIX 392
HOLD format GIF 397, 399
HOLD format INGRES 398
HOLD format INTERNAL 398, 414, 415
HOLD format PowerPoint 401
HOLD format Red Brick 401
HOLD format SQL 401
HOLD format SQLDBC 402
HOLD format SQLINF 402
HOLD format SQLMAC 402
HOLD format SQLMSS 402
HOLD format SQLODBC 403
HOLD format SQLORA 403
Index

HOLD format SQLSYB 403
HOLD format SYLK 403
HOLD format TAB 404
HOLD format TABT 404
HOLD FORMAT VISDIS 405
HOLD format WK1 406
HOLD format WP 406
HOLD format XFOCUS 407
HOLD formats 386
HOLD formats AHTML 387
HOLD formats AHTMLTAB 388
HOLD formats APDF 388
HOLD formats EXL97 396
HOLD formats FLEX 396
HOLD formats FOCUS 397
HOLD Master Files 353, 357, 363, 365, 366, 367, 368, 369, 370, 371, 374
HOLDATTR command 363, 370
HOLDATTR parameter 371
HOLDLIST command 363, 368, 369
horizontal bar graphs 1040, 1240
horizontal labels 1290
horizontal menu HTML Active Reports 658
horizontal waterfall graphs 1269
host fields 751
host files 727, 735
HTML Active Reports 658, 659, 660, 662, 663, 665, 666, 668, 669, 670, 671
HTML alignment 1149
HTML display formats for reports 435
HTML format 397, 431, 432, 435
HTML heading freezing 1077
HTML index pages 692
HTML report display formats 435
HTML reports 77, 79, 640, 685, 1213
pop-up field descriptions 79
HTML tables of contents 639, 640, 641
HTML tables of contents for multiple sort groups 645
HTMLCSS SET parameter 826
HTMLFORM command 909
HTMTABLE format 398, 542, 909
hyperlinks 509, 685, 921
calling a spreadsheet from 509
hyperlinks in HTML reports 685
identifying across columns in a style sheet 855
identifying ACROSS phrase sort data 864
identifying across totals in a style sheet 865
identifying ACROSS-TOTAL values 865
identifying ACROSSVALUE component 865
identifying cells 1338, 1370
identifying column report components 853
identifying columns 854, 855, 863, 1330, 1331, 1332, 1333, 1334, 1370
identifying columns by address 1333
identifying columns by number 1331
identifying columns by relative address 1334
identifying columns by value 1336
identifying columns in a style sheet 854, 863
identifying columns in financial reports 1329, 1330, 1331
identifying contiguous columns 1332
identifying data 860
identifying data report components 861, 862
identifying embedded fields in report components 886
identifying Excel 2000 null values 461, 462
identifying footings in a style sheet 879, 880
identifying free text in FML reports 878
identifying free text in report components 878
identifying groups of values 1318
identifying headings in a style sheet 879, 880, 881
identifying null values 461, 462
identifying page numbers in report components 887, 888, 889
identifying ranges of multiple values 1317
identifying report components 849, 851
identifying report components for totals and subtotals 857, 858, 859, 866, 867, 869
identifying rows 1326, 1327, 1328, 1370
identifying rows and report components 856, 857, 864
identifying rows in a style sheet 864
identifying rows in financial reports 1327
identifying rows in WebFOCUS StyleSheets 1370, 1371, 1372
identifying skipped lines in report components 887, 888, 889
identifying sort value report components 876
identifying sort values 874, 876
identifying subtotal calculations in a style sheet 872
identifying subtotals in a style sheet 869, 870
identifying text strings in report components 884, 885
identifying title report components 874, 875, 876, 877
identifying totals in a style sheet 869
identifying underlines in report components 887, 888
IF command with LIKE or UNLIKE 168
IF operator 157, 159
IF phrase 144, 168, 177, 178, 316, 1278
IF phrase expressions 316
IF/THEN/ELSE statements 348, 479
IMAGE attribute 557, 607, 1016, 1017, 1019, 1025, 1026
IMAGEALIGN attribute 1016, 1017
IMAGEBREAK attribute 1016, 1017
images 557, 559, 1093, 1106
images and WebFOCUS StyleSheets 1017
images in footings 1093, 1106
images in headings 1093, 1106
improving performance 141, 736, 1425, 1427
IN command 958, 959
in reports 35
IN-GROUPS-OF option 1275
IN-GROUPS-OF phrase 109, 110
IN-RANGES-OF phrase 109, 110
INCLUDES operator 174, 175
including JavaScript in an HTML report 436
indenting captions 1354
independent paths 148, 150
index optimized retrieval 1424
index pages 692
index pages for headings and footings 693
INGRES formats 398
inheritance in Cascading Style Sheets (CSS) 913, 914, 925
inheritance in style sheets 913, 914, 925
inheritance style sheets 820
inheriting attributes 819, 820
inline StyleSheets 808, 809
inline WebFOCUS StyleSheets 814
inner join 724, 740, 743
inner join structures 743
INSERT command 1426
INSERT INTO command 1415
inserting blank lines 1000
inserting text in financial reports 1342
inserting underlines 1000
inserting variables in free text 1343
inter-row calculations 1324, 1325
internal Cascading Style Sheets (CSS) 825, 826, 827
internal Cascading Style Sheets (CSS) and WebFOCUS StyleSheets 908
INTERNAL format 398, 414, 415
internal matrixes 218, 1335, 1435
internal storage and field formats 327
interpolating X and Y axis values 1271, 1272
INTERSECT operator 1418
irrelevant report data 697, 698
IS NOT operator 167, 168
IS operator 167, 168
ISO standard date-time formats 336
ITEM attribute 884
ITEM subtype 884, 886, 1162
ITEMS data source 1467

J
JavaScript 79, 542
requirements for pop-up field descriptions 79
JavaScript functions 528, 529
JavaScript functions in HTML reports 436
JavaScript requirements 679
JavaScript requirements for Accordion Reports 679
JOBFILE data source 1446, 1447
JOBHIST data source 1460
JOBLIST data source 1460
JOIN CLEAR command 766, 767
JOIN command 726, 731, 732, 735, 737, 748, 754, 1412, 1413, 1414
JOIN command and ALL parameter 727, 728
JOIN command and SQL Translator 1412, 1413, 1414
join structures 726, 735, 737, 745, 752, 1412, 1413, 1414
join structures and CHECK FILE command 763, 764
join structures and DBA security 727
join structures and DEFINE command 747, 748, 749, 750, 757, 758, 760
join structures and group fields 746
join structures and numeric data types 753
join structures and qualified field names 1414
join structures and virtual fields 747, 748, 749, 750, 758, 760
join structures and WHERE phrase 762
join types 724
joining data sources 35, 188, 726, 727, 735, 752
joining fields 752, 753
joins 724
JSURL SET parameter 436
justification regions 1135
JUSTIFY attribute 1131, 1132, 1139, 1172, 1235
justifying column titles 1138, 1139, 1140, 1141, 1142, 1143
justifying columns 1235
justifying data 1226, 1236
justifying field values 1165
justifying footings 1130
justifying grand totals 1145
justifying headings 1130
justifying headings and footings 1130, 1131, 1132, 1133, 1134, 1135, 1137, 1163
justifying labels 1130
justifying report columns 1235
justifying row totals 1143, 1144
justifying subtotals 1145
justifying titles 1130

K
KEEPDEFINES parameter 757, 758
KEEPFILTERS SET parameter 188
key fields 358, 360
keyed retrieval 372, 373

L
LABEL attribute 856, 877, 1370, 1371, 1372, 1373
LABEL subtype 1371, 1379
LABELPROMPT attribute 1047, 1048
labels 1046, 1048, 1049, 1053, 1113, 1326
labels and formatting rows and 1376
labels for calculated values 1119
labels for column totals 1115
labels for grand totals 1117
labels for row totals 1115
labels for rows 1327, 1328
labels for subtotals 1116, 1117, 1118
labels for subtotals or grand totals 1116
labels in financial reports 1120
Index

lagging values 729
landscape orientation 933
last page number 981
last page number in a sort group 983
LE operator 163, 164, 346
LEDGER data source 1454
left margins 935
left outer join 724, 743
left outer join structures 743
LEFTGAP attribute 939, 942, 945
LEFTMARGIN attribute 936, 937, 938
legacy features 804
legacy report formatting 804
LIKE operator 167, 168
limit FORECAST 234
limitations for headings and footings 1055
limitations in display fields 59
limitations of dynamic tables of contents 657
limitations of HTML tables of contents 657
limitations on EXL2K FORMULA display format 479
limiting data for graphs 1278
limiting display fields 59
limits column titles 1108
limits for display fields 59
LINE attribute 882, 883
line breaks 1242
line graph 1263
line graphs 1250, 1254, 1263
LINE subtype 882, 884, 886, 1162
line termination characters 453
line-by-line formatting 1169
line-feeds 1242
linear regression 251, 253
linear regression analysis 229, 230, 231
linear regression analysis FORECAST 229, 231
linear regression equation 245, 246
linear regression equation FORECAST 245, 246
linear regression in graphs 1271, 1272
linear scales 1252
LINES SET parameter 969
LINK element 897, 898, 909, 910
linking from graphics 557
linking graphic elements 557, 559
linking graphs 1288
linking images 557, 559
linking report components 519, 520, 521, 522, 525, 526, 528, 529, 554, 557, 559, 561, 562
linking report pages 685
linking report pages and heading text 688
linking report pages and images 686
linking report pages and page numbers 688
linking report pages in HTML reports 685
linking reports 639, 685, 828
linking reports with WebFOCUS StyleSheets 685
linking summary and detail data 623, 624, 628
linking summary and detail data drill through reports 625
linking to a Maintain procedure 531, 532

1536 WebFOCUS
linking to Cascading Style Sheets (CSS) 897, 898, 909, 910, 911, 913
linking to CSS (Cascading Style Sheets) 897
linking to external Cascading Style Sheets 897, 898, 909, 910
linking to JavaScript functions 528, 529
linking to Maintain procedures 532
linking to Uniform Resource Locators (URLs) 525, 526
linking to URLs (Uniform Resource Locators) 525, 526
linking with conditions 554, 555
links 519, 520
LIST * command 47
LIST command 43, 45, 57, 82
list records 45
listing join structures 765
listing records 45, 46
literals 1419
LOAD CHART command 1362, 1363
load procedures 1441
loading a hierarchy into memory 1362, 1363
location of Cascading Style Sheets (CSS) 902
LOCATOR data source 1461
LOCKED attribute 471
locking excel spreadsheets 471
logarithmic scales 1252
logical expression types 345
logical expressions 155, 165, 317, 345
logical operator 165
logical operators 155, 156, 157, 346
long field names 1501, 1502, 1503
LOOKGRAPH parameter 1261, 1263, 1264, 1265, 1266, 1268, 1269, 1291
LOTUS format 399
LST prefix operator 70
LT operator 163, 164, 346
M
MACRO attribute 817
macros 805, 816, 818
macros style sheets 816, 817
mailing labels 1046, 1047, 1048, 1049
Maintain procedures 531, 532
maintaining across joins 188
maintaining filters 188
maintaining filters across joins 188
margins 935, 937
masked fields 167, 169
masking characters 167, 169, 1318
masks 167, 168
Master Files 32, 165, 182, 1350, 1362, 1441
hierarchies in 1362
Master Files and MISSING attribute 700, 701, 702
Master Files DESCRIPTION attribute 77
Master Files for financial reports 1350
Master Files for FML hierarchies 1350, 1362, 1363
Master Files for hierarchies 1350
Master Files HOLD files 353
MATCH command 770, 771, 772, 773, 1112, 1113, 1499
MATCH FILE command 771, 772, 773, 774, 775
MATCH FILE command and concatenated data sources 788
MATCH FILE command and display commands 780
MATCH FILE command and merge phrases 773, 774
MATCH FILE commands 770
matrix reports 99, 100, 262, 265
matrix type reports 99
matrixes 1435, 1436
MATRIXORDER attribute 1047, 1048
MAX prefix operator 64
MAX prefix operators 64
maximum prefix operators 64
measurement units 936
measuring fonts 1172
measuring for column width alignment 1172
measuring for decimal alignment 1172
merge phrases 773, 775
merge phrases and HOLD files 773, 774, 775
merge phrases and MATCH FILE command 773, 774
merging data sources 770, 771, 772, 773, 774, 775, 776, 777, 779, 780, 788, 790, 791, 793
merging data sources and display commands 780
merging data sources and PRINT command 780
merging data sources and SUM command 780
merging multiple graphs 1273
metrics file 443, 445, 447
Microsoft Web Query utility 501
MIME types 426, 427, 428, 433
MIN prefix operator 64
MIN prefix operators 64
minimum prefix operators 64
MISSING attribute 165, 699, 705, 706, 707, 708
MISSING attribute and extract files 709
MISSING attribute and Master Files 700, 701, 702
MISSING attribute and virtual fields 702
MISSING attribute limits 702
missing descendants 717, 718
missing instances 699
missing instances and ALL parameter 717, 718
missing instances and ALL prefix 716
missing instances and DEFINE command 700, 701, 702, 703, 704
missing instances and extract files 709
missing instances and segment instances 699, 713, 714, 715
missing instances and temporary fields 702
missing instances for reformatted fields 711, 1221
MORE phrase 783, 784, 785, 788
MORE phrase and universal concatenation 784, 785
MOVIES data source 1466
multi-pane reports 1046, 1051
multi-path data sources 49, 51, 53, 207
multi-segment data sources 174, 175
multi-segment files 174
multi-table HTML reports 968, 969, 971
MULTILINES command 272, 273, 277, 1116, 1119
multipath join structures 743
MULTIPATH parameter 148, 149, 150, 153
multiple Cascading Style Sheets 902
multiple display commands 124
multiple display commands and ROW-TOTAL 266
multiple drill-down links 534
multiple drill-down links and WHEN phrase 540
multiple drill-down links conditional styling 540
multiple drill-down reports 534
multiple drill-downs 534
multiple fields in PivotTables 485
multiple graphs 1272
multiple parameters 552
multiple records 1315
multiple sort fields 86, 87, 94, 124
multiple values 1315
multiple verbs 124
multiple virtual fields 203
multiple Web pages 968
multiple WHERE phrases 147
multiple Y-axis graphs 1269
multiple-line footings 1075
multiplication operator 319, 321
multivariate REGRESS 250, 251, 253

N
naming Cascading Style Sheet classes 904
naming classes in Cascading Style Sheets (CSS)
904, 906
naming extract files 352
naming output files 352
naming StyleSheet files 811
naming WebFOCUS StyleSheet files 811
National Language Support (NLS) 130
native Excel formulas 475
native-mode arithmetic 323
navigating between reports 657
NE operator 165, 346
NLS (National Language Support) 130
NOBREAK phrase 604
NODATA character 698, 699, 720, 721, 1224, 1225
non-numeric fields 53, 54
non-recursive models 1346
non-unique join structures 724, 727, 728
NOPRINT command 122, 123, 962, 963, 965, 1271, 1390
NOSPLIT command 973, 974
NOT FROM ... TO operator 162
NOT LIKE operator 167
NOT operator 346
NOTOTAL command 312
Index

null values 1224
numeric constants 1419
numeric data 109, 113
numeric data types 752
numeric data types and join structures 753
numeric expressions 317, 322, 323, 324
   evaluating 323
numeric fields 53
numeric formats not supported in Excel 2000 466
numeric functions 320
numeric operator expressions 319, 321
numeric values 464

O

OBJECT attribute 884
OBJECT subtype 1162, 1167
OMITS operator 165, 166
ON TABLE SET command 913
on-demand paging 682, 683
ONLINE-FMT parameter 432
operand formats 323, 324
operators 319, 345
operators prefix 61
optimized join structures 1424
optimizing join structures 1424
optimizing sorting data 130
OR operator 155, 346, 1316
order of evaluation 322
ORIENTATION attribute 930, 933, 934
outer join 724
output file AHTML format 387
output file AHTMLTAB format 388
output file format 388
output file format DFIX 392
output file format GIF 397, 399
output file format HTML 397
output file format HTMTABLE 398
output file format INGRES 398
output file format INTERNAL 398
output file format LOTUS 399
output file format PDF 399
output file format PDF OPEN/CLOSE 400
output file format PostScript (PS) 401
output file format PPT 401
output file format Red Brick 401
output file format SQL 401
output file format SQLDBC 402
output file format SQLINF 402
output file format SQLMAC 402
output file format SQLMSS 402
output file format SQLODBC 403
output file format SQLORA 403
output file format SQLSYB 403
output file format SYLK 403
output file format TAB 404
output file format TABT 404
output file format WK1 406
output file format WP 406
output file format XFOCUS 407
output file formats 386, 396
output file formats APDF 388
output file formats EXL97 396
output file formats FLEX 396
output file formats FOCUS 397
output file text fields 407, 408
output files 352, 376
output files and missing values 709
output files text fields 386
output format VISDIS 405
output formats 386, 457
OVER command 946, 956, 957
overriding attribute inheritance 822
overriding macros 818

P

padded fields 414, 415
PAGE BREAK command 1368
page breaks 968, 970
page breaks in financial reports 1368
page colors 930, 934, 935
page count 981, 983
page footings 1069, 1073, 1075
page headings 1069, 1070
page layout 929
page margins 935, 937
page numbers 887, 888, 889, 979, 980, 981, 983, 986, 989
page numbers in footings 1093
page numbers in headings 1093
page numbers in headings and footings 1103
page orientation 930, 933, 934
page size 930, 931
PAGE-BREAK command 969, 1201
PAGE-NUM SET parameter 979, 980, 989
PAGECOLOR attribute 930, 934, 935
PAGEFIELDS and FORMAT EXL2K PIVOT command 489
PAGEFIELDS phrase 482, 488
PAGEFIELDS phrase and PivotTables 488, 489
PAGEMATRIX attribute 1047, 1048
PAGENUM component 887, 888, 889
PAGESIZE attribute 930, 931
PAGESIZE option 440, 441
paginating a report 968
pagination options 663
paper size settings 440, 441, 442
parameters 544, 546, 548, 550, 552
parent instances 717, 718
parent segments in qualified field values 174
Pareto graphs 1269
PCHOLD AHTML format 387
PCHOLD command 352, 385, 431, 454, 642
PCHOLD command and Excel formats 491
PCHOLD command and PDF format 605
PCHOLD files 385
PCHOLD format 388
placing footings on a separate page 1201, 1203, 1204
placing headings on a separate page 1201, 1203, 1204
plotting dates 1275, 1276
plotting dates in graphs 1275, 1276
PLUS OTHERS 87
PLUS OTHERS phrase 87
polar charts 1268
pop-up field descriptions 77, 79
JavaScript requirements 79
ReportCaster and 79
portrait orientation 933
POSITION attribute 939, 940, 942, 1016, 1026, 1030, 1180
positional column referenced calculated values 216
positional field references for COMPUTE command 216
positional labels 1326, 1327
positional referencing for columns 216
positioning columns 939, 940, 942, 946, 957, 958, 959, 960
positioning columns for WebFOCUS StyleSheets 1235
positioning footings 1180, 1197, 1198, 1200
positioning headings 1180, 1196
positioning headings and footings 1180, 1181, 1182, 1200
positioning report components 939
positioning reports for headers and footers 1235
positioning with spot markers 1188, 1189, 1190
POST command 1392
posting data 1392, 1393
posting data in financial reports 1392
posting financial data 1391
PostScript (PS) format 401, 438
PostScript (PS) reports 440, 441
PostScript display format 431, 432, 440
PostScript display format for compound reports 565, 604
PostScript display formats for reports 440, 441
PostScript fonts 443, 445
PostScript fonts in UNIX 445
PostScript fonts in Windows 445
PostScript fonts in z/OS 447
PostScript format 431
PostScript formats 443
PostScript report display formats 432, 440, 441, 442
PostScript Type 1 fonts 449
PostScript Type1 fonts 443
PowerPoint format 401
PPT format 401
precision 57
predicting values with FORECAST 229
prefix operators 60, 61, 62, 280, 281, 283, 295, 297, 298, 303, 306, 1093
prefix operators for display fields 60
prefix operators in free-form reports 1403

Creating Reports With WebFOCUS Language 1543
prefix operators in headings and footings 1094, 1099, 1100, 1101
preparing reports for an Excel spreadsheet 501, 503, 508
preparing WebFOCUS reports for an Excel spreadsheet 508
preserving field names 363
preserving missing values 708, 709, 713
preserving virtual fields 208
preventing breaks 973
preventing page breaks 973
PRINT * command 47
PRINT command 43, 45, 47, 82, 780
PRINT command and merging data sources 780
PRINT command unique segments 53
print display formats 438
PRINT OFFLINE parameter 1306
printing graphs 1306
printing labels 1046, 1047, 1049
printing multi-pane reports 1051
PRINTONLY parameter 368, 369
PRINTPLUS parameter 1187
procedures sorting data 130
producing a direct percent of a count 67
product position graphs 1269
PROTECTED attribute 471
protecting virtual fields 208, 209
PS (PostScript) format 438

Q
qualified field names 1414, 1418, 1501, 1502, 1503
qualified field names and SQL join structures 1414
qualified field names and SQL Translator 1418
qualified field values 97, 174
QUALTITLES command 1112, 1113
query ? STAT command 132
query commands
  ? DEFINE 204
  ?F 1505
  ?FF 1505
querying HOLD files 353, 356
querying sort types 132
QUIT command 37
quotation marks 339
quote-delimited string 339, 340

R
radar graphs 1250, 1268
range of values 1317, 1318
range tests 162, 163, 164
ranges 109, 110, 111, 112
RANKED BY phrase 107, 108
RANKED BY TOTAL phrase 107, 121
ranking columns 121
ranking sort field values 74, 107, 108, 118, 119
reading selection values from a file 179, 180
reading values from a file 180, 181
READLIMIT operator 176
READLIMIT relational operator 176
RECAP and sort footings 1090
RECAP command 308, 309, 310, 311, 316, 1308, 1309, 1325, 1326
RECAP command and FML reports 1324
RECAP command expressions 316
RECAP component 857, 866, 867, 872
RECAP expressions 1326
RECAP rows 1324
RECOMPUTE command 276, 277, 279, 280, 281, 295, 299, 300, 301, 303, 304, 306
RECOMPUTE command and propagation to grand total 289
RECOMPUTE prefix operators 306
RECORDLIMIT operator 176, 177
RECORDLIMIT relational operator 176, 177
records 45, 145, 1315
records in multiple rows 1319, 1320
recursive join structures 731, 732, 733, 1414
recursive models 1346, 1347
recursive structures 731, 732
Red Brick format 401
REDBRICK format 401
redefining formats for fields 1219
reducing report width 955
ref_regress_usage 251
reformatting fields 209, 211, 212, 1237, 1238
refreshing Cascading Style Sheets 925
refreshing Cascading Style Sheets (CSS) 903, 925
REGION data source 1456
relational expressions 155, 345, 347
relational operator 167
relational operators 157, 159, 162, 163, 164, 165, 167, 169, 345
relative column addresses 1334
relative point sizes and HTML fonts 1207
relative starting positions 1185
removing grids 997
renaming column titles 1107, 1109
renaming column totals 262, 1113
renaming HOLD files AS phrase 353
renaming PCHOLD files 385
renaming row totals 262, 1113
REPAGE command 969, 979, 980
repeating fields 731
repeating fields in join structures 731
repeating rows 1328
repeating sort values 86
report columns 119
REPORT component 851, 852
report components 520, 521, 522, 525, 526, 528, 529, 554, 557, 559, 561, 562, 812, 849, 897, 1206
report components, column 850
report components, entire report 850
report components, row 850
report display EXL2K formats 432
report display formats 429, 430, 431, 432, 433, 477, 479, 480
report display formats for Excel 97 490, 492
report display PDF formats 438
report display PostScript formats 438
report footings 1061, 1067
report formatting 797, 798, 799, 804, 805
report formatting in external Cascading Style Sheets 905
report formatting methods 801
report formatting with external Cascading Style Sheets 891, 895, 897
report formatting, inheritance in style sheets 913
report headings 1061, 1062
report navigating 806
report output 35
report output formats 457, 916, 917, 919
report pagination 968
report requests 34, 35, 37, 1405
   creating 35
report requests and SQL statements 1405
report styling 891
report styling in external Cascading Style Sheets 905
report styling with external Cascading Style Sheets 891, 895, 897
report SUM columns 119
report titles 1059, 1060
report types 33
reporting against hierarchies 1348, 1351, 1353, 1354, 1355, 1364
reporting commands 1495
reporting options 1495
reporting tools 34
reports 31, 32, 33, 34, 35, 37, 38, 39, 41, 520, 639, 640, 692, 1205, 1206, 1212, 1501
   creating 33, 37, 38
   customizing 35, 39
   displaying 41
   displaying data 35
   printing 41
   running 37
   saving 41
   selecting data 35
   sorting data 35
   specifying fields 35
reports in Excel 97 491
reports in EXL97 display format 491
requirements for Cascading Style Sheets 921
requirements for external sorting 130
reserved words 1408
restricting sort field values 107, 108, 118, 119
restrictions for distinct prefix operators 67, 69
restrictions for DST prefix operators 69
restructuring data 1430
retrieval data 148, 150
retrieval limits 176, 177
retrieval logic 1428
retrieval order 139, 140
retrieving data 148, 150
retrieving data values 1314
retrieving financial data 1391
retrieving financial data values for rows 1313, 1314
retrieving HOLD Master Files 374
retrieving multiple values with masking characters 1318
retrieving posted data 1393, 1394
retrieving records 70, 176, 177, 736
retrieving values for rows 1313
retrieving values with masking characters 1318
returned fields 329
reusing FOR field values 1319
reusing output reports 351
reusing records 1319, 1320
reusing report output 351
reusing tag values 1313, 1319, 1320, 1355
reusing values 1320
right margins 935
RIGHTGAP attribute 939, 942
RIGHTMARGIN attribute 936, 937
RNK. prefix operator 74
rotating data sources 1428
rounding numeric values 319
row formatting 1367, 1370
row labels 1328
row percent (RPCT) 65
row selection colors 665
row titles 877, 1365, 1366
row total labels 1114
row totals 262, 265, 266, 269, 270, 476, 1113
ROW-TOTAL phrase 262, 264
ROW-TOTAL with ACROSS and multiple display commands 266
rows 856, 857
rows and calculations 1324, 1325, 1326
rows in financial reports 1312
ROWTOTAL attribute 859
RPCT prefix operator 65
RPCT row percent 65
rules in Cascading Style Sheets 893, 903, 922
rules in external Cascading Style Sheets 898
RUN command 37

S
SALES data source 1449, 1450
SALHIST data source 1463
SAME_DB 376
SAME_DB extract files 376, 377, 379, 380, 381
SAME_DB HOLD files 376, 377, 379, 380, 381
SAME_DB HOLD format 376, 377, 379
SAME_DB HOLD format columns 380
SAME_DB output files 376, 377, 379, 380, 381
sample data sources 1441, 1442, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452, 1454, 1455, 1456, 1458, 1459, 1460, 1461, 1462, 1463, 1464, 1465, 1466, 1467, 1468, 1469, 1470, 1471, 1472, 1473, 1474, 1475, 1476, 1478, 1479, 1480, 1482, 1483, 1484, 1485, 1486, 1487, 1488, 1489, 1490, 1491
CAR 1451, 1452
CENTGL 1487, 1488
CENTGLL 1490, 1491
CENTSTMT 1489, 1490
Index

dsampling data sources (continued)

Centurion 1476, 1478, 1479, 1480, 1482, 1483, 1484, 1485, 1486, 1487
COURSE 1459
EDUCFILE 1447, 1448
EMPLOYEE 1442, 1444, 1445
FINANCE 1455
Gotham Grinds 1470, 1471, 1472, 1473, 1474, 1475
ITEMS 1467
JOBFILE 1446, 1447
JOBHIST 1460
JOBLIST 1460
LEDGER 1454
LOCATOR 1461
MOVIES 1466
PERSINFO 1462
REGION 1456
SALES 1449, 1450
SALHIST 1463
TRAINING 1458
VIDEOTR2 1468, 1469
VideoTrk 1464, 1465

SAVB command 381
SAVB files 381
SAVE AHTML format 387
SAVE AHTMLTAB format 388
SAVE command 352, 381
SAVE files 381
SAVE format 388
SAVE format EXL2K 396
SAVE format HTML 397
SAVE format HTMTABLE 398
SAVE format LOTUS 399
SAVE format PDF 399
SAVE format SYLK 403
SAVE format TAB 404
SAVE format TABT 404
SAVE format WP 406
SAVE formats 386, 396
SAVEMATRIX parameter 1435, 1436
saving drill-down reports to a server 542
saving drill-down reports with HTMTABLE 534, 542
saving graphs as GIF files 1300, 1303
saving graphs as GIF files using SET
GRAPHSERVURL 1301
saving HOLD Master Files 374, 375
saving intermediate report results 1391
saving output files 352
saving report output 351, 352, 474
saving reports 351, 352, 474
saving rows 1392, 1393
saving virtual fields 757, 758, 760
scalar functions 1419
scale graphs 1252
scales 1252
scaling 1043, 1044
scaling and vertical bar graphs 1045
scatter graphs 1250, 1259, 1260, 1265
screening conditions 182
screening segments 762
screening values 218
scrollable area for HTML output 1077
search option of WebFOCUS Viewer 684
Section 508 compliance 1024

1548

WebFOCUS
SEG. operator 1504
segment instances 697, 717, 718
segment instances and missing values 713, 714, 715
segment locations 206
segment types 70, 71
segments 148, 153, 699, 1504
SEGTYPE parameter 372
selecting graph types 1250
selecting paper size 440, 441, 442
selecting paper size for PS (PostScript) format 438
selecting paper size for style sheets 442
selecting paper size PostScript (PS) format 440
selecting PostScript (PS) format paper size 441
selecting records 143, 144, 148, 150, 153, 154, 155, 156, 157, 159, 161, 165, 167, 174, 175, 176, 177, 178, 179, 180, 181, 191, 1404
selecting records with IF phrase 180, 181
selecting records with VSAM 191
selecting sort procedures 131
selecting sort types 131
selecting style sheets 803
selecting values for x- and y- axes 1270
selecting values using WHERE phrase 179
selecting values with IF phrase 181
selection criteria 143, 145, 146, 147, 148, 150, 157, 159, 161, 179, 180, 181, 313
selection values 181
selection values with IF phrase 181
sending graphs directly to a printer 1306
SEQUENCE attribute 953, 954
sequential conditional formatting 829
SET ACCESSIBLE parameter 492
SET ACROSSPRT 95
SET ALL parameter 717, 718
SET AREXPIRE parameter 677
SET ARPASSWORD parameter 678
SET ASNAMES parameter 363
SET AUTOINDEX parameter 1431
SET AUTOPATH parameter 1431
SET BLANKINDENT parameter 1386, 1387
SET BYTOC parameter 641
SET CDN parameter 1223
SET CNOTATION parameter 218, 219, 1335
SET commands 641, 642
SET COMPMISS parameter 711, 1221
SET COMPOUND parameter 604
SET COMPUTE = NEW command 1436
SET COMPUTE command 1439
SET COUNTWIDTH parameter 57
SET CSSURL command 913
SET DATEFORMAT parameter 332
SET DEFINES command 1439
SET DUPLICATECOL command 125
SET EMPTYCELLS parameter 461, 462
SET EMPTYREPORT parameter 846
SET EXPANDABLE parameter 678, 680, 681
SET EXTSORT parameter 130, 132
Index

SET FILECOMPRESS command 440
SET FILTER parameter 182, 185
SET FOCFIRSTPAGE parameter 979
SET FORMULTIPLE parameter 1319, 1320
SET GRAPHSERVURL parameter 1301, 1302, 1303
SET GRAPHSTYLE * parameter 1296, 1298, 1299
SET GRID parameter 1233
SET GRMERGE parameter 1273, 1278
SET GRWIDTH parameter 1274, 1275
SET GTREND parameter 1271, 1272
SET HAUTO parameter 1296
SET HAXIS parameter 1291, 1292
SET HMAX parameter 1291, 1296
SET HMIN parameter 1291, 1296
SET HOLDATTR parameter 363, 370, 371
SET HOLDLIST parameter 363, 368, 369
SET HOLDMISS parameter 709
SET HTMLCSS parameter 826
SET JSURL parameter 436
SET KEEPFILTER parameter 188
SET LINES parameter 969
SET LOOKGRAPH parameter 1261, 1263, 1264, 1265, 1266, 1268, 1269, 1291
SET NODATA command 1224
SET NODATA parameter 461, 462, 1224
SET ONLINE-FMT parameter 432
SET PAGE-NUM parameter 979
SET parameter ACROSSLINE 92
SET parameter ESTLINES 133
SET parameter ESTRECORDS 133
SET parameter EXTAGGR 138
SET parameter EXTHOLD 141
SET parameter for BYDISPLAY 86
SET parameter NULL=ON 405
SET parameters 92, 433, 452, 809, 1234, 1388, 1431, 1503
SET parameters and EXTHOLD 141
SET parameters for graphs 1293
SET PDFLINETERM parameter 452, 453, 454
SET PRINTPLUS parameter 1187
SET PSPAGESETUP parameter 440, 441
SET QUALITIES parameter 1112, 1113
SET SAVEMATRIX parameter 1435, 1436
SET SHOWBLANKS parameter 433
SET SPACES parameter 946
SET SQLTOPTTF parameter 1425
SET SQUEEZE parameter 1233, 1234
SET STYLEMODE parameter 969
SET STYLESHEET parameter 810, 847
SET SUMMARYLINES parameter 289
SET SUMMARYLINES parameter 288, 290, 300
SET UNITS parameter 827
SET VAUTO parameter 1291
SET VAXIS parameter 1291, 1292
SET VMAX parameter 1291, 1297
SET VMIN parameter 1291, 1297
SET WEBVIEWER parameter 968
SET WPMINWIDTH 407
SET WRAP parameter 1228
setting column width for EXCEL display format 468
setting conditions for linking reports 828
setting fixed scales 1291, 1296, 1297
setting graph height 1292
setting graph height and width 1291
setting height and width 1292
setting page colors 935
setting paper size for PostScript (PS) reports 442
setting retrieval order 140, 141
setting x-axis fixed scales 1291, 1296
setting y-axis fixed scales 1291, 1297
sheet names 1059
SHOWBLANKS SET parameter 433
simple moving average 229, 230, 231, 234, 235, 237
simple moving average FORECAST 231, 234, 235, 237
single-line footings 1068
single-line headings 1064
SIZE attribute 1016, 1026, 1030
SKIP-LINE option 1000, 1001, 1002
SKIPLINE attribute 1003, 1004
SKIPLINE component 887, 888, 889
skipped lines 887, 888, 889, 1001, 1002
sort field values 118
sort fields 81, 640, 642, 1321
sort fields for multi-path data sources 83
sort fields using multi-path data sources 91
sort footing limitations 1055
sort footings 1054, 1078, 1084, 1087
sort footings and RECAP 1090
sort footings omitting a display command 1092
sort heading limitations 1055
sort headings 1054, 1078, 1080, 1082, 1169
sort multiple fields 86, 94
sort order 86, 94, 109, 110, 111, 112, 113
sort phrases 45, 1253
sort phrases and graph format 1253
sort sequence 83, 91, 1321
sort temporary fields 83, 91
sort values 82, 83, 109
sorted by calculated values 121
sorting alphabetically 967
sorting by calculated values 119, 120, 121
sorting by columns 91, 92, 94, 121
sorting by rows 83
sorting columns 120
sorting columns by 91
sorting data 87, 130, 131
sorting data by columns 91, 92
sorting data by multiple fields 86, 94
sorting data by rows 83, 84, 87
sorting report columns 119, 120
sorting reports 81, 82, 1404
sorting reports by calculated values 218
sorting rows by 83, 84, 86, 87
sorting with COMPUTE command 121
SQLMSS formats 402
SQLODBC formats 403
SQLORA formats 403
SQLSYB formats 403
SQLTOPTTF parameter 1425
SQUEEZE attribute 946, 948, 949, 950, 1234, 1235
SQUEEZE command 1234
SQUEEZE parameter 1233
ST. prefix operator 1093
stacking columns 946, 955
stacking columns with FOLD-LINE 955, 956
stacking columns with OVER 956, 957
STAT query 132
stock charts 1268
storing StyleSheet files 811
storing WebFOCUS StyleSheet files 811
structure diagrams 1441
structured HOLD files 417
STYLE attribute 1008, 1208
style sheet attributes 897
style sheet CLASS attribute (StyleSheets) 905, 906
style sheet CSSURL attribute (StyleSheets) 909, 910, 911
style sheet types 826
style sheets 800, 807, 827, 1206, 1367
style sheets and conditional styling 831, 835, 836, 838
style sheets and data visualization 1241
style sheets and graphics 607, 1016, 1025, 1026
style sheets and graphs 1285
style sheets and links 520
style sheets and multi-pane reports 1046
style sheets and parameters 544, 546, 548, 550
style sheets declarations 813
style sheets default values 847
style sheets multiple output formats 916, 917, 919
style sheets troubleshooting 925
style sheets, external cascading 891
style sheets, FAQs 922
style sheets, report components in 849
style sheets, requirements 921
STYLEMODE parameter 921
STYLEMODE SET parameter 969
StyleSheet attributes 1367
StyleSheet declarations 814, 816
StyleSheet files 810, 811, 812
StyleSheet syntax 660, 662, 663, 665, 666, 668, 669, 670, 672
styling ACROSS-TOTAL component 865
styling free-form reports 922
styling graphs 1291
styling reports 891, 1235, 1240, 1241
styling reports in external Cascading Style Sheets 905
styling reports with external Cascading Style Sheets 891, 895, 897
sub-total calculated values 277
SUB-TOTAL command 272, 273, 274, 275, 280, 281, 283, 297, 298, 299, 300, 302, 303, 304, 306
SUB-TOTAL command and propagation to grand total 289
SUB-TOTAL prefix operators 306
SUBFOOT command 1067
SUBFOOT command report footings 1067
SUBFOOT command sort footings 1084
SUBFOOT component 879, 880
SUBHEAD command 1062, 1080
report headings 1062
SUBHEAD component 879, 880
subroutines 1339
subtotal calculated values 277
SUBTOTAL command 272, 273, 274, 275, 280, 281, 283, 297, 298, 299, 301, 302, 303, 304, 306
SUBTOTAL command and propagation to grand total 289
SUBTOTAL component 857, 866, 867, 870
subtotal labels 1116
SUBTOTAL prefix operators 306
SUBTOTAL SUMMARYLINES command 300
subtotals 276, 277, 278, 279, 299, 300, 308, 309, 310, 311, 857, 858, 859, 866, 867, 1113, 1116
subtraction operator 321
subtype attribute 521, 525, 528, 532, 535
subtypes 521, 1162, 1371
SUM command 43, 53, 54, 82, 780
SUM command and merging data sources 780
SUM prefix operator 72
SUMMARIZE command 276, 277, 278, 280, 281, 295, 297, 298, 299, 300, 306
SUMMARIZE command and propagation to grand total 289
SUMMARIZE prefix operators 306
summary commands 299, 303, 305
summary line processing 288
summary lines 313, 314
summary values 280, 281, 306
SUMMARYLINES SET parameter 288, 289, 290, 300
summing columns 120
summing field values 82
summing report columns 120
summing values 82
SUMPREFIX parameter 140
SUP-PRINT command 122, 123, 962, 963, 965
supplying data directly in FML 1323, 1324
supplying images descriptions for screen readers 1024
support for Excel Named Ranges 516
support for EXL2K FORMULA display format 479
supported data sources 726
suppressing column titles 1110
suppressing display fields 961
suppressing display in financial reports 1390, 1391
suppressing field padding 414, 415
suppressing grand totals 312
suppressing page numbers 990
suppressing rows 1390, 1391
suppressing rows in financial reports 1390, 1391
suppressing sort field values 122, 123
suppressing sort footings 1089
suppressing wrapping data 1229
SYLK format 403
syn_regress_mult 251
SyncSort utility 130, 131

TABLE command compared to GRAPH command 1247, 1248
TABLE command and data retrieval 1434, 1435
TABLEF command and SQL Translator 1425
tables of contents (TOCs) 639, 640
tables of contents (TOCs) and EXL2K 459
tables of contents EXL2K display format 460
TABPAGENO system variable 979, 980, 981
TABT format 404
tag names 732
TAG rows 1390, 1391
TAG rows suppressing display in financial reports 1390, 1391
tag values 1313
TARGET attribute 562
target frames 562, 563, 564
TARGET parameter 679
TARGETFRAME SET parameter 562, 563, 564
TD element 903
temporary field types 194
temporary fields 35, 194, 196, 197, 213
calculated values 213
creating 35
temporary fields and missing values 702
temporary sort fields 91
temporary tables 376, 379, 380, 381
temporary tables extract files 377
temporary tables HOLD files 379, 380, 381
temporary tables output files 377, 379, 380, 381
testing character strings 166, 167, 168, 169, 170, 171, 172, 173
testing data fields 167, 168
testing for blanks or zeros 707
testing for existing data 165, 707
testing for missing segment instances 719
testing for missing values 705, 706, 719
testing multi-segment files 174
TEXT component 884
text field output files 407, 408
text fields 83, 1160
text fields and alphanumeric fields 254
text fields in DEFINE and COMPUTE 254
text fields in headings and footings 1101, 1102
text fields output files 386
text rows 1342, 1343
text rows formatting 1375
text strings 885
three-dimensional graphs 1266
TILE column 113, 114, 117
tile fields 113, 114, 117
TILES phrase 113, 114, 117
time fields 1419
time fields and SQL Translator 1420
timestamp data type 332
timestamp fields 1419
timestamp fields and SQL Translator 1420
TITLE attribute 370, 371, 1109, 1112
TITLE component 875, 876
titles 873, 1053, 1054, 1059
titles of column 876
titles of columns 874, 875
TITLETEXT attribute 1059, 1060
TO phrase 1317, 1318
TOCs (tables of contents) 639, 640
TOCs (tables of contents) and EXL2K 459
TOCS (tables of contents) Excel 2000 459
tools for reporting 34
top margins 935
TOPGAP attribute 939, 942, 944, 1195
TOPMARGIN attribute 936, 937
TOT prefix operator 72
total page count 981
totals 262, 269, 270, 857, 858, 859, 866, 867
trailing blanks 1098
TRAINING data source 1458
translating formulas to Excel 2000 474
translation to EXL2K format 479
treating as literal masking characters 171, 172
treating as literal wildcard characters 171, 172
treating literal masking characters 173
treating literal wildcard characters 173
triple exponential smoothing 242, 244
triple exponential smoothing FORECAST 242, 244
truncated field names 1501, 1502
truncating decimal values 324
TYPE attribute 521, 525, 528, 532, 535, 812, 1008, 1044, 1122

U

UNDER-LINE option 1000, 1005
UNDERLINE attribute 1006, 1007
UNDERLINE component 887, 888
underlines 887, 888, 1000
underlines in Financial Modeling Language (FML) reports 1014, 1015
underlines in financial reports 1368
underlining values 1005, 1006, 1007
Uniform Resource Locators (URLs) 525, 526, 561, 1019
UNION operator 1418
unique join structures 724, 727, 728, 740
unique segments 47, 56
unique segments for PRINT command 49
UNITS attribute 827, 828, 936
units of measurement 827, 936
universal concatenation 783
universal concatenation and field names 786, 787
universal concatenation and MORE phrase 783, 784, 785
UNIX 452, 453, 454
UNIX PDF files 452, 453, 454
UNIX PDF format 454
UNLIKE 168
unsupported numeric formats 492
UPDATE command 1426
URLs (Uniform Resource Locators) 525, 526, 561, 1019
user menu options 665
user-controlled options 670
using a SET command for wrapping data 1228
using ACROSS phrase with Accordion Reports 678
using ACROSSVALUE component for a numeric column reference 865
using BY phrase with Accordion Reports 678
using concatenation with AnV fields 342
using CONTAINS and OMITS with AnV fields 342
using drill-down reports conditions 554, 555
using drill-downs with Accordion Reports 679
using EDIT function with AnV fields 342
using LIKE fields with AnV fields 343
using multiple parameters 552
using operators with AnV fields 343
validating StyleSheet files 815
validating WebFOCUS StyleSheet files 815
value dates 326
value format for dates 327
values 1316
values for columns 1336
values for x- and y- axes 1270
values in multiple rows 1319, 1320
varchar fields 394
variable length character expressions 341, 342
variables in WHERE tests 160
VAUTO parameter 1291, 1297
VAXIS parameter 1291, 1292
VBScript in an HTML report 436
verbs 43, 82
VERBSET attribute 127
verifying external sorting 132
vertical bar graphs 1040, 1240
vertical bar graphs and scaling 1043, 1044, 1045
vertical labels 1290
vertical menu for HTML Active Reports 658
vertical scaling 1043, 1044, 1045
vertical spacing 1193, 1195, 1197
vertical waterfall graphs 1269
VGRID attribute 992, 998
VIDEOTR2 data source 1468, 1469
VideoTrk data source 1464, 1465
viewing reports 490, 639
virtual fields 182, 184, 194, 196, 197, 206, 207
virtual fields and join structures 747, 748, 749, 750, 757, 758, 760
virtual fields and MISSING attribute 702
virtual fields and missing values 702
VISDIS 405
Visual Discovery format 405
visualization colors 668
VMAX parameter 1291, 1297
VMIN parameter 1291, 1297
VMSORT utility 131
VSAM data sources 191
VSAM record selection efficiencies 191
VZERO parameter 1280, 1281, 1282, 1283, 1284

W

waterfall graphs 1269
Web browser support for Cascading Style Sheets 921, 922, 925
web query
  deploying to a Web server 509
Web query 492, 501, 509
Web Query (.iqy) file 492
Web query utility 492, 501, 503
WEBARCHIVE parameter 457
WebFOCUS font map files 445, 449
WebFOCUS Font Map files 443
WebFOCUS inheritance StyleSheets 819, 820
WebFOCUS macros StyleSheets 816, 817
WebFOCUS reports 492
WebFOCUS style sheets 812, 814, 815, 819, 820
WebFOCUS StyleSheet CLASS attribute 898, 905, 906
WebFOCUS StyleSheet CSSURL attribute 898, 909, 910, 911
WebFOCUS StyleSheet declarations 813, 814, 816
WebFOCUS StyleSheet files 810, 811, 812
WebFOCUS StyleSheets 640, 652, 653, 685, 803, 807, 808, 828, 1042, 1206, 1235, 1240
WebFOCUS StyleSheets and adding graphics 1016, 1025
WebFOCUS StyleSheets and conditional styling 828, 829, 831, 833, 834, 835, 836, 838
Creating Reports With WebFOCUS Language

Index

WebFOCUS StyleSheets and data visualization 1040, 1042, 1240, 1241
WebFOCUS StyleSheets and external CSS 908
WebFOCUS StyleSheets and graphics 1017, 1026
WebFOCUS StyleSheets and graphs 1285
WebFOCUS StyleSheets and links 520
WebFOCUS StyleSheets and multi-pane reports 1046, 1051
WebFOCUS StyleSheets and parameters 544, 548, 550
WebFOCUS StyleSheets macros 816
WebFOCUS Viewer 639, 682
WebFOCUS Viewer search option 684
WEBVIEWER SET parameter 968
WHEN attribute 828, 829
WHEN EXISTS phrase 1391
WHEN phrase 273, 313, 316, 841
WHEN phrase and conditional formatting 841, 843, 844, 1088
WHEN phrase expressions 316
WHEN=FORECAST attribute 234, 831
WHERE operator 157, 159, 163, 164
WHERE phrase 144, 145, 146, 147, 153, 155, 167, 169, 179, 181, 316, 1278, 1279
WHERE phrase and existing data 707
WHERE phrase and join structures 762
WHERE phrase and missing values 705, 706, 708
WHERE phrase expressions 316
WHERE tests 160
WHERE TOTAL phrase 153, 154, 155, 1278
WHERE-based join structures 724
WIDTH attribute 1172
width of borders 992
width of columns 947, 948, 950, 951
wildcard characters 167, 169
window display 663, 668
window titles 1059
WITH CHILDREN parameter 1351
WITHIN phrase 97
WK1 format 406
worksheet names 1061
worksheet titles 1059
WP display formats for reports 454
WP format 406, 431, 454
WP report display formats 454
WPMINWIDTH parameter 407
WRAP attribute 1227, 1235
WRAP command 1228
wrapgap StyleSheet attribute 1229
wrapping data 1226, 1227
wrapping data by Web browser functionality 1228
wrapping data EXL2K display format 467
wrapping data reports 1227
wrapping reports with alternative methods 1233
WRITE command 53
x-axis 1270, 1272
Index

XFOCUS format 407

Y

y-axis 1270
y-axis fields 1271
Y2K attributes in Master Files 332

Year 2000 attributes in Master Files 332
YRTHRESH attribute 332

Z

z/OS requirements 131
zeros 707
Reader Comments

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. You can contact us through the following methods:

**Mail:**
Documentation Services - Customer Support
Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898

**Fax:**
(212) 967-0460

**E-mail:**
books_info@ibi.com

**Web form:**
http://www.informationbuilders.com/bookstore/derf.html

Name:______________________________________________________________
Company:________________________________________________________
Address:________________________________________________________________________
Telephone:_________________________ Date:_________________________ 
Email:______________________________________________________________________________
Comments:_______________
Creating Reports With WebFOCUS Language
Version 7 Release 6