Information Builders

IBM DB2 Web Query for IBM i

Version 2 Release 1.8

Contents

Preface

This documentation describes the functions that are available for IBM DB2 Web Query for IBM i.

How This Manual Is Organized

This manual includes the following chapters:

	Chapter/Appendix	Contents
1	Analyzing Data in an OLAP Report	Presents the terminology and benefits of using Online Analytical Processing (OLAP). Describes how to customize reports with the OLAP selections panel and the OLAP Control Panel. Describes how to sort and apply various selection criteria (to restrict your data) as well as how to troubleshoot an OLAP-enabled report. Explains how the OLAP Control Panel (OCP) provides you with a versatile way to gain more insight from your reports by dynamically manipulating report data. From the Control Panel, you can perform every function available to a DB2 Web Query OLAP user.
2	Describing and Accessing Data Overview	Introduces data source descriptions and explains how to use them.
3	Using the Synonym Editor	Describes the Synonym Editor and how you may use it to view and edit synonyms.
4	Analyzing Metadata and Procedures	Describes how to analyze procedures using Impact Analysis and how to view Data Profiling for the columns in a synonym.
5	Using Functions	Introduces functions and explains the different types of available functions.

	Chapter/Appendix	Contents
A	Running DB2 Web Query Reports Using the Java Batch Run Utility	Describes how to run DB2 Web Query Reports using the Java Batch Run Utility.
B	Describing an Individual Field	Documents how to describe specific field level information of a data source.

Documentation Conventions

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

Convention	Description
`THIS TYPEFACE` or `this typeface`	Denotes syntax that you must enter exactly as shown.
this typeface	Represents a placeholder (or variable), a cross-reference, or an important term.
<u>underscore</u>	Indicates a default setting.
this typeface	Highlights a file name or command. It may also indicate a button, menu item, or dialog box option you can click or select.
Key + Key	Indicates keys that you must press simultaneously.
`{ }`	Indicates two or three choices; type one of them, not the braces.
`[ ]`	Indicates a group of optional parameters. None is required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
`|`	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
`...`	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (...).

Convention	Description
. . .	Indicates that there are (or could be) intervening or additional commands.

IBM

1 Analyzing Data in an OLAP Report

If the OLAP license option has been activated, DB2 Web Query Online Analytical Processing (OLAP) enables you to view and quickly analyze data in order to make critical business decisions.

Topics:

We Do It Every Day: Typical DB2 Web Query

Suppose that you own a small business in New York and are exploring a partnership with a company in Oakland, California. You need to get to a Monday morning meeting. How do you go about arranging your flight?

Most likely, you go online.

First, you check available flights on the airline that holds your frequent flyer miles. You discover that your frequent flyer carrier requires a change of planes and you would prefer a direct flight, so you look at routes and fares for other airlines.

In New York, you can get to LaGuardia, JFK, and MacArthur airports on Long Island. In California, you can fly into Oakland or San Francisco.

While you would prefer to fly out on Sunday and return Tuesday morning, you could consider a Saturday flight to California and a return flight on the red-eye Monday night, if fares and schedules are better.

You begin your search by airline and then look at options for each departure point and destination, by day, time and price.

Another approach is to start with an online consolidator, enter the times you can fly, and see what flights and fares are available.

There are a lot of variables to play with, but in ½ hour you have done your research and can make a good decision based on all available factors.

The Web sites you access are designed to facilitate your queries. Various menus and selection boxes make it easy to pursue each line of inquiry. Required and optional information is identified for you. You can move forward down a path of choices, backtrack and start down a different path, or resume the original path with different selections.

You need to keep track of the question you want to answer, but a well-designed site makes your investigation easy. For most of us, this process has become intuitive.

The same process works when analyzing the data in an OLAP-enabled DB2 Web Query report.

OLAP Reporting Requirements

In this section:

OLAP reporting requires some preparation both of the data to be reported against and of the report itself. In many instances, this preparation is entirely transparent, having been done before a user encounters an OLAP report. However, for developers who are charged with OLAP-enabling data and reports and for users who wish, and are authorized, to OLAP-enable their personal reports, the following summary will be useful.

OLAP-Enabling Data

Behind the scenes of any DB2 Web Query OLAP report is a hierarchical data structure. For example, a typical hierarchy of sales regions might contain a GEOGRAPHY category including the fields (in descending order) Region, State, and City. Region, the highest level in this hierarchy, would contain a list of all available regions within GEOGRAPHY. State, the second highest level in the hierarchy, would contain a list of all available states within those regions, and so on.

In DB2 Web Query, the hierarchical structure is generally built into the Master File for a data source, where it becomes active for any report that uses that data source. Developers or administrators who are responsible for describing data in a Master File can use the underlying language. The keyword WITHIN defines the elements in each dimension in the hierarchy.

In addition, those working in Developer Workbench have access to a variety of graphical tools that make it easy to drag and drop fields into position to form a hierarchy. The hierarchy may be global to all procedures or local to one procedure. To define a:

- ❑ Global hierarchy in a Master File for use with multiple procedures, use the Dimension Builder.
- ❑ Local hierarchy as a component of a particular procedure, use the Dimension tool. The hierarchy you define with this tool does not affect the source Master File.

OLAP-Enabling a Report

Reference:

Setting OLAP Reporting Options

In addition to using OLAP-enabled data, a report must be enabled to support OLAP analysis. OLAP-enabling a report consists of specifying how a user will interact with and drill down on OLAP data.

The primary interactions occur in the report itself. In addition, you can choose to expose two supplementary tools, the OLAP Selections pane and the Control Panel.

Reference: Setting OLAP Reporting Options

OLAP Interface Options

Users can control the OLAP interfaces and following drill-down options.

- **OFF.** Provides an OLAP button to open the OLAP Control Panel.
- **Columns only.** Turns off the OLAP Control Panel and the OLAP Selections pane, but allows OLAP functionality from the report itself. You can access options on right-click menus, drag and drop columns within the report, and use up and down arrows to sort columns from high to low or vice versa. This is the default.
- **Columns with panel.** Provides access to the OLAP Selections pane from a square button to the left of the column titles.
- **Show filters on top.** Opens the OLAP Selections pane above the report. The Measures, Graph, and Dimension controls, as well as the band containing the OLAP, Run, and Reset buttons appear above the report output. You can open the Control Panel by clicking the OLAP button on the Selection pane.
- **Show filters on bottom.** Opens the OLAP Selections pane below the report. The Measures, Graph, and Dimension controls, as well as the band containing the OLAP, Run, and Reset buttons appear below the report output. You can open the Control Panel by clicking the OLAP button on the Selection pane.
- **Show Panel in Report.** Opens the OLAP report with the OLAP Selections pane hidden. You can perform a variety of analytic tasks from the report itself. Selection Criteria is shown next to the OLAP button.
- **Show Tabbed.** For OLAP reports that have multiple dimensions, this option groups the dimension elements under a tab labeled with the dimension name.

Drill Down options

These options enable you to sort instantly from high to low or low to high for selected report columns:

- **None**

  Disables automatic drill downs.

- **Dimensions**

  Enables automatic drill downs on dimensions in both reports and graphs.

- **Dimensions and Measures**

  Enables automatic drill downs on dimensions in both reports and graphs and, also, on measures in reports.

Note: Explicit drill downs in a StyleSheet (if they exist) take precedence over OLAP-enabled hyperlinks. If you click a hyperlink associated with an explicit drill-down, the behavior will be defined by the StyleSheet rather than by the AutoDrill On or All settings.

OLAP Terminology

The following table describes OLAP terms that may be useful as you work in the DB2 Web Query OLAP tools. Some of these terms are directly reflected in the interfaces of the OLAP Selections pane and the OLAP Control Panel. Others provide useful background information.

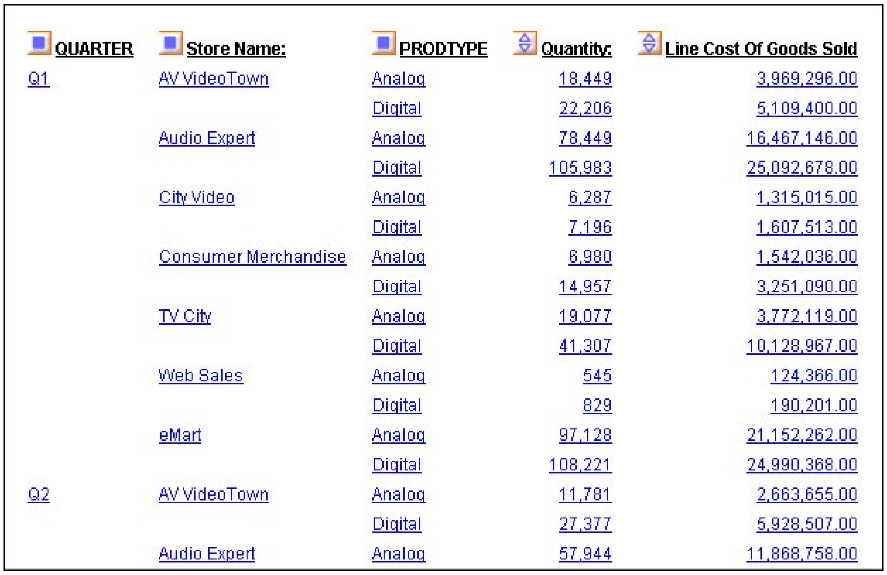
The first column of the following table provides the term and the second column provides the definition.

Term	**Definition**
Dimension	Group or list of related elements, usually structured in a hierarchy. For example, a Location dimension could include the elements Country, Region, State, and City arranged in a hierarchy where Country is the top level and City is the base level. Dimensional data usually describes the measured item.
Hierarchy	Logical parent-child structure of elements within a dimension.
Measure	Type of item that specifies the **quantity** of another element with which it is associated. A measure typically defines how much or how many. For example, Units, Revenue, and Gross Margin are measures in the Account dimension and specify how many units were sold, how much revenue was generated, and at what profit margin, respectively.
Pivot	Manipulating (or rotating) the view of a report by moving a field (or a group of fields) from a column to a row, or vice versa.

Characteristics of an OLAP Report

An OLAP-enabled report has a number of features that distinguish it from other DB2 Web Query reports.

A basic OLAP report is shown in the following image.



QUARTER	Store Name:	PRODTYPE	Quantity:	Line Cost Of Goods Sold
Q1	AV VideoTown	Analog	18,449	3,969,296.00
		Digital	22,206	5,109,400.00
	Audio Expert	Analog	78,449	16,467,146.00
		Digital	105,983	25,092,678.00
	City Video	Analog	6,287	1,315,015.00
		Digital	7,196	1,607,513.00
	Consumer Merchandise	Analog	6,980	1,542,036.00
		Digital	14,957	3,251,090.00
	TV City	Analog	19,077	3,772,119.00
		Digital	41,307	10,128,967.00
	Web Sales	Analog	545	124,366.00
		Digital	829	190,201.00
	eMart	Analog	97,128	21,152,262.00
		Digital	108,221	24,990,368.00
Q2	AV VideoTown	Analog	11,781	2,663,655.00
		Digital	27,377	5,928,507.00
	Audio Expert	Analog	57,944	11,868,758.00

Every OLAP user can take advantage of the analytic features that are built into the OLAP report:

❑ **Hyperlinks.** The values in an OLAP report are usually hyperlinks from which you can drill down to related information.

  Depending on your OLAP settings, the hyperlinks may be active for both the dimension fields (by which the report is sorted) and the measures fields (which display quantitative data), or only for the dimension fields. For related information, see *OLAP-Enabling a Report* on page 18.

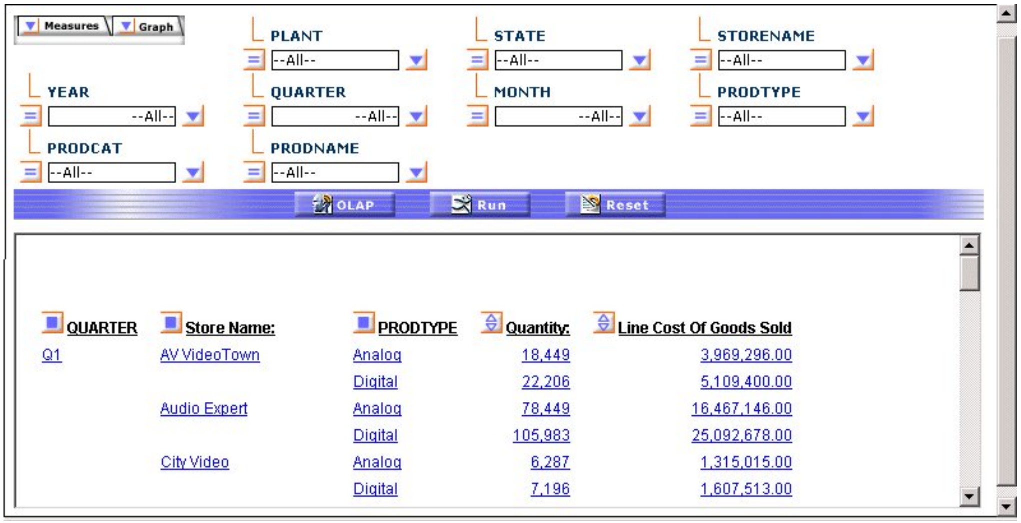
❑ **Context menus.** You can right-click any column title to access a menu of options that facilitate analysis. The options vary slightly to suit the tasks associated with dimensions and measures.

❑ **Sorting diamonds** ⬦ **.** The measures (fields that make up the body of the report) have blue diamonds adjacent to them. You can click either the top or bottom of the diamond to instantly sort data from high to low or low to high.

- **Drag and drop capabilities for dimensions and measures.**
    - You can drag and drop sort fields to shift sorting from vertical (By) to horizontal (Across) or vice versa.
    - You can change the order in which sorting occurs by dragging sort fields from inner to outer positions (and vice versa).
    - You can drag measures from one position to another to affect the order in which data appears.

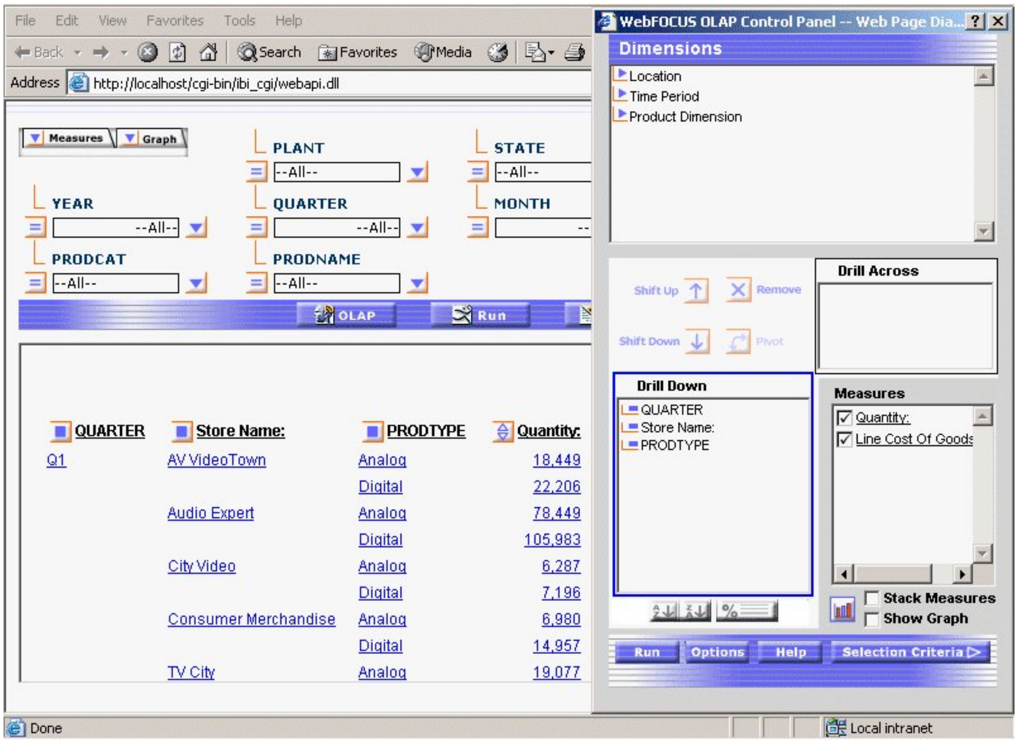
Beyond the features in the report itself, your OLAP options depend on the interface and drill down settings that are in effect for a particular report. Those choices determine whether you have access to the following tools:

- **Selections Pane.** When this tool is available, a pane may appear above, as shown in the following image, or below your report. The limit of entries in an OLAP drop down is 5000. For details, see *Selections Pane* on page 24.



QUARTER	Store Name:	PRODTYPE	Quantity:	Line Cost Of Goods Sold
Q1	AV VideoTown	Analog	18,449	3,969,296.00
		Digital	22,206	5,109,400.00
	Audio Expert	Analog	78,449	16,467,146.00
		Digital	105,983	25,092,678.00
	City Video	Analog	6,287	1,315,015.00
		Digital	7,196	1,607,513.00

- **OLAP Control Panel.** When this tool is available, the blue squares adjacent to the sort fields (By or Across) in the report become active. You can click a square or the *OLAP* button to open the Control Panel, as shown in the following image. For details, see *OLAP Control Panel* on page 26.



Address: http://localhost/cgi-bin/ibi_cgi/webapi.dll

QUARTER	Store Name:	PRODTYPE	Quantity:
Q1	AV VideoTown	Analog	18,449
		Digital	22,206
	Audio Expert	Analog	78,449
		Digital	105,983
	City Video	Analog	6,287
		Digital	7,196
	Consumer Merchandise	Analog	6,980
		Digital	14,957
	TV City	Analog	19,077

Three Ways of Working With OLAP Data

In this section:

The Report

Selections Pane

OLAP Control Panel

There are three ways to work with OLAP data: from the report itself, from the Selections pane, and from the Control Panel. This documentation is organized to help you understand what you can do from each location and which method is most suitable and efficient for your particular OLAP settings.

The Report

You can perform a wide range of basic analytic functions from the report itself. Changes you make in the report are implemented instantly. Every OLAP user can perform these tasks:

- Sort the data in measures in either ascending (lowest value to highest) or descending order (highest value to lowest).
- Drill down on measures, dimensions, or both (depending on the settings described in *Setting OLAP Reporting Options* on page 19).
- Hide fields in the current report.
- View hidden fields in the dimensions hierarchy and add them to the report.
- Change a vertical (By) sort field to a horizontal (Across) sort field and vice versa.
- Delete sort fields.
- Add a column of small bar graphs that help you visualize trends in numeric data (measures).
- Display a graphical representation of your data in a frame above the tabular report.

For an illustration of report-powered OLAP analysis, see *We Do It Every Day: Typical DB2 Web Query* on page 16.

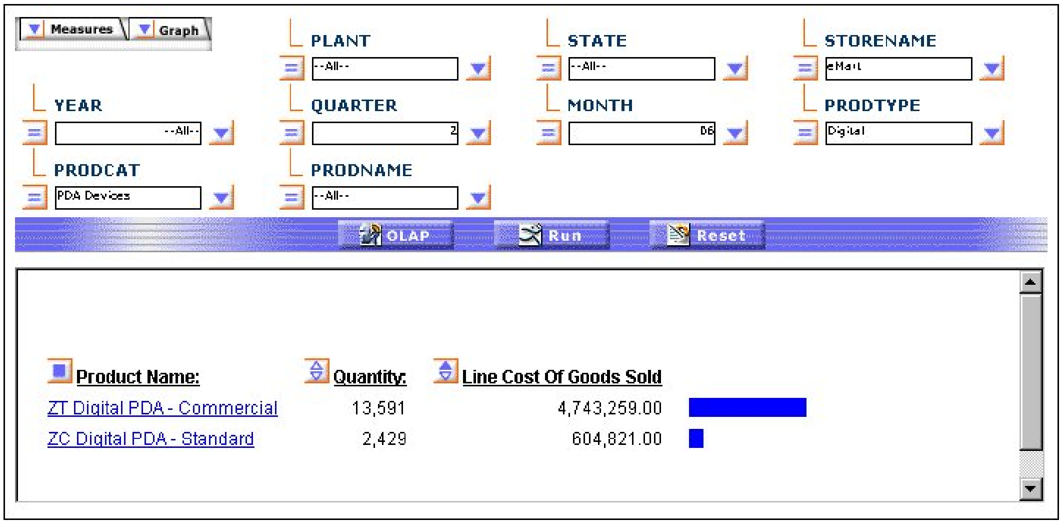
Selections Pane

When the OLAP Selections pane is turned on, you can quickly limit the data in the report by selecting specific values for the dimensions in the hierarchy. A drop-down list is available for each dimension. You can multi-select values from one or more dimension lists to refine your report output. The limit of entries in an OLAP drop down is 5000.

If you wish to add a dimension element to the report you can drag it from the Selections pane into the report frame. (The cursor changes to a + sign to indicate an acceptable location.)

Each dimension has a relational operator button located to its left. This button toggles through a selection of basic numeric operators that enable you to quickly define your selection criteria. The operators are: Equal to, Not equal to, Less than or equal to, Less than but not equal to, Greater than or equal to, Greater than but not equal to. For details, see *Selection Criteria Relational Operators* on page 42.

The following image shows the Equal to operator as the selection for each dimension in the Selection Pane.



PLANT: --All--
STATE: --All--
STORENAME: eMart
YEAR: --All--
QUARTER: 2
MONTH: 06
PRODTYPE: Digital
PRODCAT: PDA Devices
PRODNAME: --All--

Product Name:	Quantity:	Line Cost Of Goods Sold
ZT Digital PDA - Commercial	13,591	4,743,259.00
ZC Digital PDA - Standard	2,429	604,821.00

The name of the dimension field appears as defined in the Master File, even if an alternate column title has been specified.

In addition, you can customize the display of the measures in your report from the Selections pane. You can click either the Measures or the Graphs arrow in the upper left corner of the pane to list the measures.

- From the Measures arrow, you can display or hide the selected measure(s) or request a column of simple bar graphs to reveal trends.
- From the Graphs arrow, you can choose the measure(s) you wish to graph and specify one of seven basic graph types: vertical and horizontal bar, line, area charts and pie charts.

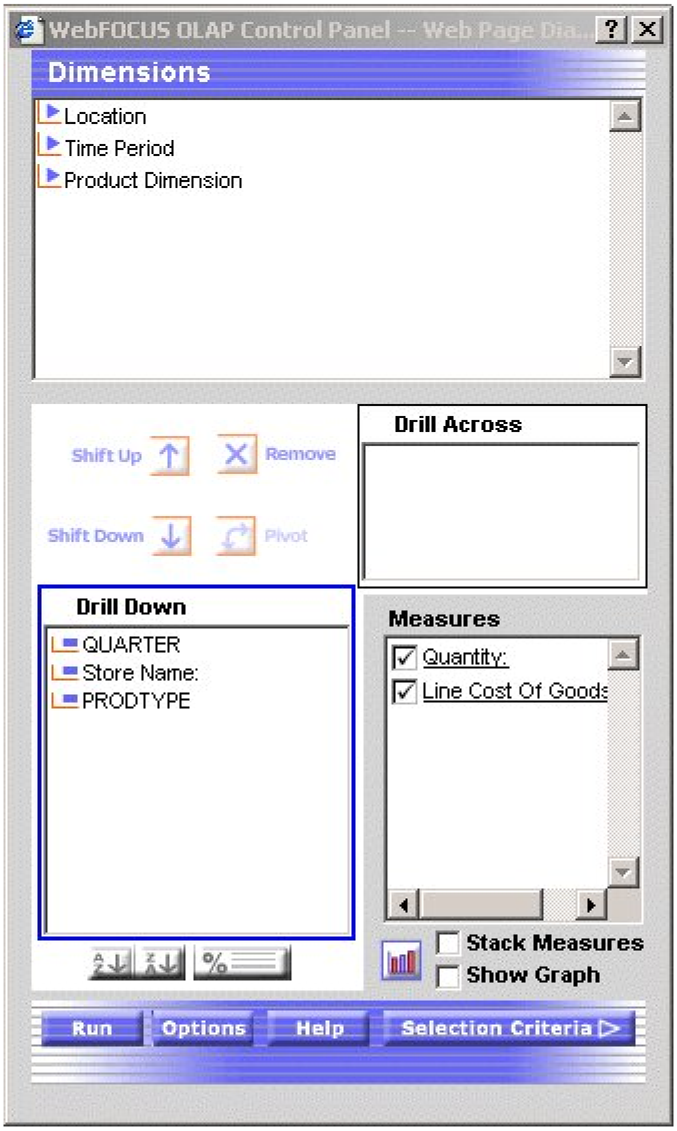
Note that the Selections pane is resizable; the controls for dimensions, measures, and graphs float as you resize the report window so that they continue to be visible in the frame.

Three buttons appear below the Selections pane: OLAP, Run, and Reset.

- **OLAP** opens the OLAP Control Panel (OCP).
- **Run** executes the report with the current set of selections.
- **Reset** resets all the controls in the report to their previous state (that is, before the current set of selections was made and after the last execution of the report).

OLAP Control Panel

From the Control Panel you can perform every analytic function available to a DB2 Web Query OLAP user as shown in the following image.



The main window of the Control Panel contains the following components:

- **Dimensions box** reflects the hierarchical structure of the data source being used by the current report (for example, the Location dimension contains the Region, State, and City fields; the Region is made up of several States, and each State contains several Cities). You click the arrow to the left of a dimension name to view the elements that comprise it. (The fields shown here are also listed in the Selections pane.)
- **Drill Down and Drill Across boxes** list the fields being used to sort the report. You can 'pivot' a Drill Down field to a Drill Across field and vice versa, and shift their positions in the report. (You can also accomplish these tasks by dragging fields within the report.)
- **Measures Properties box** contains the body of your report (usually numeric fields). You can change the display mode of a measure by clicking the check box next to the measure; the options are display, hide, and show a column of associated bar graphs. (This is equivalent to the options available from the Measures control in the Selections pane.)

Although the most frequently used functions are available directly from an OLAP report and/or from the Selections pane, several can only be performed from the Control Panel.

Unique Control Panel operations include:

- Sorting options for dimensions: from highest to lowest and vice versa (A>Z or Z>A); restricting sort field values to a specified number of either highest or lowest values; assigning a rank number to each row in the report. For details, see *Sorting Data* on page 28.
- Options for grouping numeric data by tile (for example, percentile, decile, quartile, etc.). For details, see *Grouping Numeric Data Into Tiles* on page 37.
- Defining selection criteria based on omitted or existing characters, dates, and range specifications. For details, see *Limiting Data* on page 42.
- Saving OLAP output in PDF and Excel formats. For details, see *Saving and Displaying OLAP Reports and Graphs in Other Formats* on page 65.
- Stacking multiple measures to limit the width of the report. For details, see *Stacking Measures* on page 57.

Drilling Down On Dimensions and Measures

You can drill down on dimensions in OLAP reports and graphs and, also, on measures in reports. The settings activate the required hyperlinks:

- **Dimensions** enables automatic drill downs on dimensions in reports and graphs.
- **Dimensions and Measures** enables automatic drill downs on dimensions in both reports and graphs and on measures in reports.

- **None** disables automatic drill downs. This is the default.

In Developer Workbench, you can set drill down options from the Report Options Format tab. For details about this setting, see *Setting OLAP Reporting Options* on page 19.

Sorting Data

In this section:

You can sort the data in an OLAP report based on the values of dimensions in the hierarchy and/or the values of the quantitative measures that constitute the body of the report. Sorting options vary depending on the nature of the data being sorted. For details, see *Sorting Measures* on page 28 and *Sorting Dimensions* on page 32.

You can also group numeric data into any number of tiles (percentiles, quartiles, deciles, etc.). See *Grouping Numeric Data Into Tiles* on page 37.

Sorting Measures

How to:

You can apply aggregation and sorting simultaneously to a numeric measure in an OLAP report, and sort the data from high to low (descending order) or from low to high (ascending order). All other columns are sorted correspondingly.

For the measure being sorted, you can restrict the report to a specified number of highest values (when sorting high to low) or lowest values (when sorting from low to high).

When you sort a measure, any subtotals, subheadings, or subfootings in the report are automatically suppressed since these elements relate to a specific sort field and are not meaningful when the report is resorted by the values in a measure column.

Note: Sorting by measures is not available in a report in which measures have been stacked. See *Hiding and Displaying Measures* on page 58.

Procedure: How to Sort Measures High to Low/Low to High in an OLAP Report

To sort the values of a measure from high to low:

- ❑ Click the top half of the diamond button.

  or

- ❑ Right-click the measure and select *Sort By Highest* from the menu.

The report runs automatically. The highest value is now first in the column. The top of the diamond button becomes solid blue to indicate the current sort direction.

To sort the values of a measure from low to high:

- ❑ Click the bottom half of the diamond button.

  or

- ❑ Right-click the measure and select *Sort By Lowest* from the menu.

The lowest value is first in the column. The bottom of the diamond button becomes solid blue.

Tip: After a measure has been sorted, clicking the upper or lower half of the diamond button inverts the sort order of that measure. Place your mouse over either half of the diamond to see a message that indicates the next sort order that will occur if you click that half of the diamond.

Procedure: How to Sort Measures High to Low/Low to High From the Control Panel

1. Open the OLAP Control Panel.
2. Click a measure in the Measures box to open the sort options pane (do not click the Measures check box which controls the display of a measure, not its sorting).

   Verify that the Sort box is checked (this setting is required to apply sorting specifications to the selected measure).

3. Select the *High-to-Low* or *Low-to-High* options button to specify the sort order you wish to apply. The default sort order is high to low.
4. Click *OK*.

   The sort pane is replaced by the Measures box, where the measure becomes blue to indicate that sorting specifications have been defined.

5. Click *Run* to display the report with sorting applied to the selected measure.

   The diamond button next to the sorted measure changes to reflect the sort order: if High to Low, the top half of the diamond is solid blue; if Low to High, the bottom half is solid blue.

Note:

- Report execution is automatic when you sort a measure in an OLAP report. However, if the Control Panel is open, all current changes in the Control Panel are applied.
- If an OLAP request contains a horizontal (Across) sort field, the measures appear several times in the report, once for each Across value. If you apply sorting to a measure, the sort is performed on the first column occurrence of the measure, and reflected in all subsequent instances. The appropriate half of the diamond button becomes solid only for the first instance. Any additional sorting you wish to perform must be done from the first occurrence of the measure.

Procedure: How to View a Subset of Data for Sorted Measures

You can select to view only a subset of the total number of records in your report.

1. Open the OLAP Control Panel.
2. Click a measure name in the Measures box to open the sort options pane (do not click the Measures check box which controls the display of a measure, not its sorting).

   Verify that the *Sort* check box is selected (this setting is required to apply sorting specifications to a measure).

3. Select the *Rank* check box, then specify the number of sort field values to be included in the report. (Notice that *Highest* or *Lowest* appears to the left of the input box to reflect the current sort order.)

   - Use the spin buttons located to the right of the word Highest or Lowest to increase or decrease the number of sort fields.

     or

   - Position the cursor in the input box and type a number.

   The default number of sort fields values is 5.

4. Click *OK*.

   The sort pane is replaced by the Measures box, where the measure becomes blue to indicate that sorting specifications have been defined.

5. Click *Run* to display the report with the designated number of sorted values.

Procedure: How to Remove Sorting Criteria for a Measure

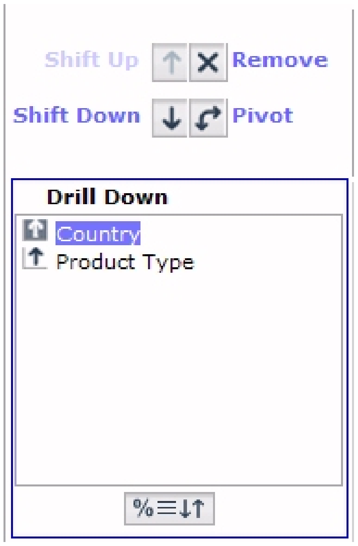
You can remove sorting specifications for a measure whether the measure is appears or hidden.

1. Open the OLAP Control Panel.
2. In the Measures box, click the measure for which you want to remove sorting specifications.
3. Clear the *Sort* check box.
4. Click *OK*.

Procedure: How to Hide a Sort Field

1. Use a Web Query development tool to build an OLAP report.
2. Open the OLAP Control Panel.
3. Double-click the *Country* field in the Drill Down panel of the OLAP Control Panel. In the resulting window panel, select the *Hide* check box.
4. Click *OK*.

   Notice that the color of the sort icon has been reversed. The Drill Down panel now appears as shown in the following image.



Sorting Dimensions

How to:

Change Sort Order for a Dimension

Restrict the Display of Sort Values

Rank Rows in a Vertically Sorted Report

Reposition Sort Fields in an OLAP Report

Reposition Sort Fields From the Control Panel

Pivot Rows and Columns In an OLAP Report

Pivot Rows and Columns From the Control Panel

Sort by a Field Without Displaying the Sort Column

There are several ways in which you can sort dimensions in an OLAP hierarchy. You can:

- Control the order in which data is sorted: ascending or descending.
- Restrict sort field values to a specified number of either highest or lowest values.
- Assign a rank number to each row in a vertically sorted report.
- Shift the positions of sort fields in the report. For example, you can change from sorting by State and then by Product to sorting by Product and then by State.
- Pivot a vertical (By) sort field to make it a horizontal (Across) sort field and vice versa.
- Hide a sort field in the report while retaining the sorting associated with it. For example, you can sort data by quarters without showing the Quarter column.
- Group numeric data in tiles (for example, percentile, decile, etc.).

Procedure: How to Change Sort Order for a Dimension

1. Open the Control Panel.
2. Select a field from the Drill Down or Drill Across box.
3. Click the *Sort* button.

   The sort pane opens.
4. Under Sort Order, choose the *Low to High* or *High to Low* options button (Low to High is the default for a dimension).

5. Click *OK*.

   The main Control Panel window reopens.

6. Click *Run* to execute the report.

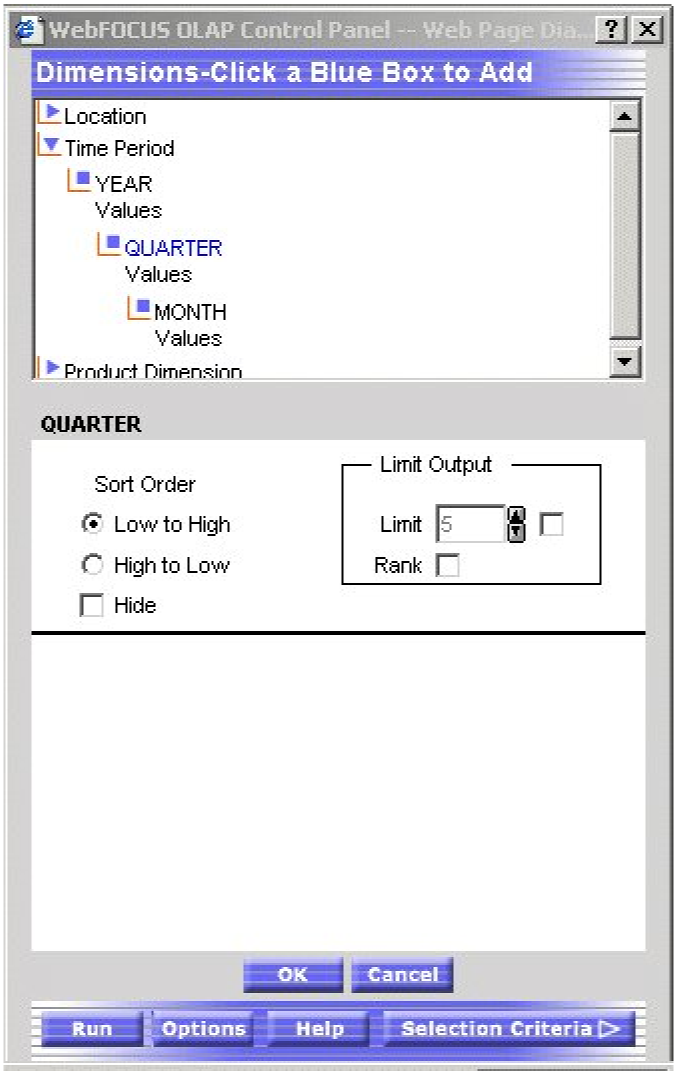
Procedure: How to Restrict the Display of Sort Values

To restrict the display of sort field values to a certain number of highest or lowest values:

1. Open the OLAP Control Panel.
2. Select a field from the Drill Down box.
3. Click the *Sort* button.

   The sorting pane opens.

4. Under Sort Order, choose the *Low to High* or *High to Low* options button as shown in the following image on the OLAP Control Panel.



QUARTER

Sort Order

[✓ Low to High]

High to Low

Hide

Limit Output

Limit 5

Rank

5. Under Limit Output, click the *Limit* check box and choose or type a value in the input area.

6. Click *OK*.

   The main Control Panel window reopens.

7. Click *Run* to execute your report.

Procedure: How to Rank Rows in a Vertically Sorted Report

1. Open the OLAP Control Panel.
2. Select a field from the Drill Down box.
3. Click the *Sort* button.

   The sort pane opens.
4. Under Sort Order, choose the *Low to High* or *High to Low* options button.
5. Click the *Rank* check box.
6. If you wish to place a restriction on the number of sort field values to rank, click the Limit check box, choose or type a value in the input area.
   - If the *High to Low* option button is selected, you can rank a specified number of Highest values.
   - If the *Low to High* option button is selected, you can rank a specified number of Lowest values.
7. Click *OK*.

   The main Control Panel window reopens.
8. Click *Run* to execute your report.

Procedure: How to Reposition Sort Fields in an OLAP Report

You can change the order in which data is sorted and presented in the report. For example, you can change from sorting by State and then by Product to sorting by Product and then by State.) If you want to reposition:

- Vertical (By) sort fields, drag and drop a field into a new column position.
- Horizontal (Across) sort fields, drag and drop the lower field above the higher one or vice versa.

In each case, the cursor changes to a plus (+) sign to indicate acceptable places into which you can drop the field. Unacceptable positions are shown by a circle with a slash across the center.

Procedure: How to Reposition Sort Fields From the Control Panel

1. Open the OLAP Control Panel.
2. Select a field in the Drill Down or Drill Across box.

3. Click the *Shift Up* or *Shift Down* arrow until the field is in the desired position.

   Repeat for other fields as needed.

4. Click *Run* to execute your report.

Procedure: How to Pivot Rows and Columns In an OLAP Report

You can quickly change a field from one that sorts data vertically, creating rows, to one that sorts data horizontally, creating columns, or vice versa. To change a:

- Vertical (By) sort field to a horizontal (Across) sort field, drag and drop a field above the row of column titles.
- Horizontal (Across) sort field to a vertical (By) sort field, drag and drop the field into the desired location in the row of column titles.

In each case, the cursor changes to a plus (+) sign to indicate acceptable places into which you can drop the field. (Unacceptable positions are shown by a circle with a slash across the center.)

Procedure: How to Pivot Rows and Columns From the Control Panel

You can change a field from one that sorts data vertically, creating rows, to one that sorts data horizontally, creating columns, or vice versa.

1. Open the OLAP Control Panel.
2. Select the title of the row or column you want to pivot in the Drill Down or Drill Across box.
3. Click the *Pivot* button. The title appears in the new location.
4. Click *Run* to execute your report.

Procedure: How to Sort by a Field Without Displaying the Sort Column

To use a field to sort your data, but not show the sort field as a column in the report:

1. Open the OLAP Control Panel.
2. Select a field in the Drill Down or Drill Across box.
3. Click the *Sort* button.

   The sort pane opens.

4. Under Sort Order, click the *Hide* check box.

5. Click *OK*.

   The main Control Panel window reopens.

6. Click *Run* to execute the report.

Tip: To expose the hidden sort field, repeat the process and deselect the *Hide* check box.

Grouping Numeric Data Into Tiles

How to:

Group Data Into Tiles in an OLAP Report

You can group numeric data into any number of tiles (percentiles, deciles, quartiles, etc.) in tabular reports. For example, you can group student's 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 title in the Tiles section of the OLAP Control Panel.

❑ 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's value changes.

❑ Instances are counted using the tile field. If the request displays 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:

Group	**Data Values**
1	1,5
2	5,5
3	8,9

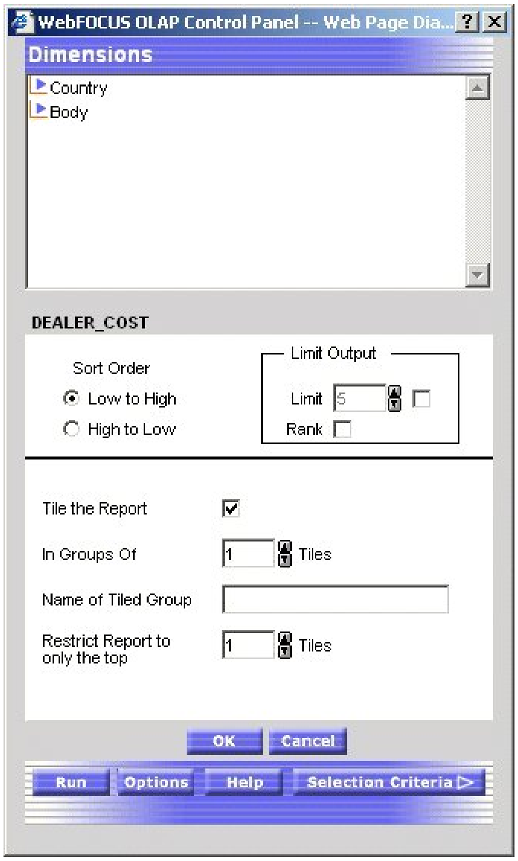
However, because all of the same data values must be in the same tile, the fives (5) that are in group 2 are moved to group 1. Group 2 remains empty. The final tiles look like the following table:

Tile Number	**Data Values**
1	1,5,5,5
2	
3	8,9

Procedure: How to Group Data Into Tiles in an OLAP Report

1. Open the OLAP Control Panel.
2. Select a numeric or date field from the Drill Down box.
3. Click the *Tiles* button.

The Tiles pane opens at the bottom of the OLAP Control Panel as shown in the following image.



DEALER_COST

Sort Order
Low to High [✓]
High to Low

Limit Output
Limit 5
Rank

Tile the Report ✓
In Groups Of 1 Tiles
Name of Tiled Group
Restrict Report to only the top 1 Tiles

4. Click the *Tile the Report* check box.
5. In the *In Groups Of* input area, select the number of tiles to be used in grouping the data. For example, 100 tiles produces percentiles, 10 tiles produces deciles, etc.
6. In the *Name of Tile Group* input box, type a name for the Tile column.
7. In the *Restrict Report to only the Top* input area, select the number of tile groups to display in the report.
8. Optionally, select a Sort Order option button:
   - Choose *High to Low* to sort data in descending order so that the highest data values are placed in tile 1.

- ❑ Choose *Low to High* to sort data in ascending order so that the lowest data values are placed in tile 1. This is the default.

9. If you wish to specify the highest tile value to appear in the report, select a value from the Limit input area. For example, if you enter a Limit of 3, the report will not display any data row that is assigned a tile number greater than 3.
10. Click *OK* to accept the selections and return to the main Control Panel window.
11. Click *Run* to re-execute and view the report.

Performing a Calculation on a Measure

How to:

Apply a Calculation to a Measure

Reference:

Calculations You Can Perform on a Measure

You can perform standard calculations, such as average, percent, and summarize, on the numeric data in measures on an OLAP report.

Procedure: How to Apply a Calculation to a Measure

1. Run the report.
2. Open the OLAP Control Panel.
3. Click a measure in the Measures box.

   The sort options pane opens. Do not click the Measures check box, which controls the display of a measure, not its sorting.

4. Click the arrow under Measure Calculations and select a calculation from the drop-down list.

   None is the default value. For details, see *Calculations You Can Perform on a Measure* on page 41.

5. Click *OK*.

   The sort pane is replaced by the Measures box, where the selected calculation appears as a prefix to the measure.

6. Click *Run*.

   The applied calculation is added to the column title.

Reference: Calculations You Can Perform on a Measure

The following table lists the types of calculations in the first column and describes their functions in the second column.

Calculation	Function
`Average Sum of Squares`	Computes the average sum of squares for standard deviation in statistical analysis.
`Average`	Computes the average value of the field.
`Count`	Counts the number of occurrences of the field.
`Count Distinct`	Counts the number of distinct values within a field when using -REMOTE. For other modes of operation, this behaves like Count.
`Maximum`	Generates the maximum value of the field.
`Minimum`	Generates the minimum value of the field.
`Percent`	Computes a field's percentage based on the total values for the field. The Percent can be used with detail as well as summary fields.
`Percent Count`	Computes a field's percentage based on the number of instances found.
`Row`	Computes a field's percentage based on the total values for the field across a row.
`Summarize`	Sums the number of occurrences of the field.
`Total`	Counts the occurrences of the field for use in a heading (includes footings, subheads, and subfoots).

Limiting Data

An OLAP report is limited to values belonging to the parent categories in the dimensions hierarchy. There are several ways to further limit the data that appears in the report.

From the Selections pane or the Control Panel, you can explicitly limit the data in an OLAP report by selecting dimension values and relational operators (such =, >, <). For a list of the relational operators, see *Selection Criteria Relational Operators* on page 42.

- The Selections pane provides the easiest approach since you can choose both dimension values and relational operators with a few mouse clicks, while the report is fully exposed to view.
- The Control Panel offers several options that are not available from the Selections pane, including record selection based on a date or date range and on the existence or absence of specific characters.

Note that changes you make in the Selections Pane are immediately implemented in the Control Panel (even if the Control Panel is closed). However, the inverse is not true; changes you make in the Control Panel are not reflected in the Selections pane until the report is run.

From the report, you can limit data indirectly by drilling down on measures and dimensions to hone in on a subset of information. For details, see *Drilling Down On Dimensions and Measures* on page 27.

Reference: Selection Criteria Relational Operators

You can define selection criteria using several relational operators, most of which are supported in both the Selections pane and the Control Panel. Those supported only in the Control Panel are noted in the following table which describes the operators.

The first column shows the operator and the second column describes the operator.

Operator	**Displays Records That...**
= - **Equal to**	Are equal to the criteria you specified. This is the default operator.
≠ - **Not Equal to**	Are not equal to the criteria you specified.
≤ - **Less than or equal to**	Are less than or equal to the criteria you specified.
< - **Less than**	Are less than, but not equal to, the criteria you specified.
≥ - **Greater than or equal to**	Are greater than or equal to the criteria you specified.
> - **Greater than**	Are greater than, but not equal to, the criteria you specified.
⊆ - **Contains**	Contain the criteria you specified. **Note:** This operator is available only for alphanumeric fields and is only supported in the Control Panel.
⊈ - **Not contain**	Do not contain the criteria you specified. **Note:** This operator is available only for alphanumeric fields and is only supported in the Control Panel.

Note: You can select more than one value using the same relational operator.

The following relational operators are available for selecting a range of dates. They are only available from the Control Panel.

The following table shows the relational operators in the first column and describes them in the second column.

Operator	Displays Records Where...
- **Within range**	The value in the indicated date field falls within the specified range. **Note:** To use this relational operator, you must select the Range check box in the Date Selection panel.
- **Not within range**	The value in the indicated date field does *not* fall within the specified range. **Note:** To use this relational operator, you must select the Range check box in the Date Selection panel.

Procedure: How to Apply Selection Criteria From the Selections Pane

When the Selections pane is turned on, there is one control (drop-down list) for every dimension in the OLAP hierarchy. (Note that the name of the dimension field appears as defined in the Master File, even if an alternate column title has been specified.)

To limit data for the dimensions that are included in the report:

1. Click the arrow to the right of the dimension to open the list of values.
2. Select one or more values from the list (*All* is the default value).

   To select multiple values, click the desired values while holding the *Ctrl* key on the keyboard.
3. Select a relational operator from the button to the left of the dimension to indicate the basis for selection (equals (=) is the default).

   You can toggle through a list of operators. See *Selection Criteria Relational Operators* on page 42.
4. Repeat steps 1-3 for each dimension whose values you wish to limit.
5. Click the *Run* button on the band below the Selections pane.

Tip: To change or eliminate selection criteria, reopen the values list and choose another value or choose *All*.

Procedure: How to Apply Selection Criteria From the Control Panel

1. Open the Control Panel.
2. Click the *Selection Criteria* button at the bottom right of the window.

   The Selection Criteria pane opens.
3. In the Dimensions box above the Selection Criteria pane, expand a dimension and click *Values*.

   A secondary window opens; select one or more values (press the *Ctrl* key to multi-select).
4. Click *OK* to return to the Selection Criteria pane, where the selected values appear in the drop-down lists.

   - If a developer has applied selection criteria to an OLAP report, you only see the selected acceptable values of the field.
   - If no selection criteria have been applied, you see all the values of the field in the drop-down lists.
5. In the *Selection Criteria* pane, click a relational operator next to the dimension to specify the relationship that you want to base selection on—for example, =, >, or <. For a complete list, see *Selection Criteria Relational Operators* on page 42.
6. Repeat the process for other dimensions whose values you wish to limit.
7. Click *Run* to execute your report.

Procedure: How to Change Selection Criteria From the Control Panel

Tip: If you have access to the Selections pane, it provides the easiest way to adjust or remove selection criteria. See *How to Apply Selection Criteria From the Selections Pane* on page 44.

From the Selections Criteria pane in the Control Panel:

1. Click the *Select* button next to the dimension value you wish to modify.

   A secondary window opens.

   To change a value, type the new value in the text box or select one or more values from the list. (The value you type must be in the same case as the value in the data source.)

   You can input only one value in the text box. If you select more than one value from the list, only the first value appears. However, all values appear in your report.

   To deselect a value, hold down the *Ctrl* key while clicking the value.

2. Click *OK* to return to the Selection Criteria pane where you can verify the revised value and/or change the relational operator if required.
3. Click *OK* again to confirm your choice and return to the main Control Panel window.
4. Click *Run* to execute your report.

Procedure: How to Remove Selection Criteria From the Control Panel

Tip: If you have access to the Selections pane, it provides the easiest way to adjust or remove selection criteria. See *How to Apply Selection Criteria From the Selections Pane* on page 44.

From the Selections Criteria pane in the Control Panel:

1. Select the criterion you want to remove.
2. Click the *Delete* ☒ button.

   The selection category is removed from the list.
3. Click *Run* to execute your report with all values.

Applying Selection Criteria to Date Elements

How to:

Apply Selection Criteria to a Date Field

Apply Selection Criteria to a Date Range

Add Dates to the Selections List Box

Delete Dates From the Selections List Box

Reference:

Date Format Limitations

You can apply selection criteria to date elements just as you apply them to other types of elements. The results are limited by the date(s) you select. For example, you can select to view data associated with a particular date or to exclude data from the specified date.

Note: Like other dimension elements, date fields must have been defined in the Master File by a developer. The Master File specifies the date formats available for selection criteria.

This feature is only supported from the Control Panel, where you choose the selection criteria from a Date selection pane that contains the appropriate controls for the date format.

You can also select a range of dates in a designated year by specifying a *From* and *To* date. Two relational operators are available for selecting a range of dates:

- The **Within range** operator displays records when the value in the indicated date field falls within the specified range.
- The **Not within range** operator displays records when the value in the indicated date field does *not* fall within the specified range.

For more information on supported date formats, see *Date Format Limitations* on page 49.

Procedure: How to Apply Selection Criteria to a Date Field

1. Open the Control Panel.
2. Click the *Selection Criteria* button.

   The Selection Criteria pane opens.
3. In the Dimensions box above the Selection Criteria pane, expand the dimension that includes the date field, and click the *Values* button.

   A secondary window displays controls for the dimension's date format. For example, if the date format is YYM, only the year and month controls appear. If the format is YYMD, year, month, and day controls appear.

   Note: The date selection pane appears only when a supported date format is provided. See *Date Format Limitations* on page 49.
4. Specify a date using the spin controls, drop-down lists, or by typing the value.

   If your date format includes edit masking such as Y.M.D, the date appears with forward slashes in the Date selection list box, the Selection Criteria pane, and the drop-down list at the bottom of the report. However, the date edit mask appears as specified within the body of the report.
5. Click *Add* to display the date in the Selections list box.
6. Click *OK* to return to the Selection Criteria pane and verify the selected date.
7. In the Selection Criteria pane, click a relations button to the left of the date field (for example, =, >, or <) to indicate a basis for record selection.
8. Optionally, define additional date selection criteria by repeating steps 2-7.
9. Click *Run* to execute your report.

Procedure: How to Apply Selection Criteria to a Date Range

1. Open the OLAP Control Panel.
2. Click the *Selection Criteria* button.

   The Selection Criteria pane opens.

3. In the Dimensions box above the Selection Criteria pane, expand the dimension that includes the date field, and click the *Values* button.

   A secondary window displays controls for the dimension's date format. For example, if the date format is YYM, only the year and month controls appear. If the format is YYMD, year, month, and day controls appear.

   Note: The Date selection pane appears only when a supported date format is provided. See *Date Format Limitations* on page 49.

4. Click the *Range* check box.

   Inclusive and Exclusive options buttons appear:

   - Choose *Inclusive* to show the range including the dates specified.
   - Choose *Exclusive* to show the range excluding the dates specified.

   Note:

   - You can select only one range of dates at a time.
   - You can apply selection criteria to a range of dates only if the date format contains a year. See *Date Format Limitations* on page 49.

   From and To drop-down lists open for all selectable options. By default, the current date appears.

5. Specify a *From* date and a *To* date by using the spin controls and drop-down lists.
6. Click *OK* to return to the Selection Criteria pane.
7. To view both the From and To dates of the range selected, click the down arrow on the drop-down list.
8. Click a relational operator to the left of the date element in the Selection Criteria pane:

   - Choose *Within range* operator to display records when the value falls within the specified range.
   - Choose *Not within range* operator to display records when the value does *not* fall within the specified range.

9. Click *Run* to execute your report.

Procedure: How to Add Dates to the Selections List Box

1. Open the Control Panel.
2. Click *Selection Criteria* to open the Selection Criteria pane.
3. Click the *Select* button to open the Date selection pane.
4. Specify the date you want to add by using the spin buttons, drop-down lists, or by typing the value.
5. Click *Add*.

   The date appears inside the Selections list box.

6. Click *OK* to return to the Selection Criteria pane.

Procedure: How to Delete Dates From the Selections List Box

1. Open the Control Panel.
2. Click *Selection Criteria* to open the Selection Criteria pane.
3. Click *Select* to open the Date selection pane.
4. Select one or more dates that you want to remove from the Selections list box.
5. Click *Delete*.

   The date is removed.

6. Click *OK* to return to the Selection Criteria pane.

Reference: Date Format Limitations

Note the following limitations when applying selection criteria to date elements:

- ❑ The Date selection pane does not support Julian dates. However, if you are using Julian dates, the Date controls still open.
- ❑ Dates containing only a day format (D, I2D, A2D) are not supported from the Date selection pane. Instead, the data source provides a list of values.
- ❑ The Range check box is enabled on the Date selection pane when the date format contains one of the following formats:
  - ❑ Any smart date format—for example, YMD, MDY, YYMD, MDYY, Q, M.
  - ❑ A4YY

- I4YY
- I8YYMD
- A8YYMD
- I6YYM
- A6YYM

Visualizing Trends

How to:

Add a Column of Bar Graphs for a Numeric Measure

To make your reports more powerful, you can insert visual representations of selected data directly into the report output. These visual representations, which appear as a column of vertical or horizontal bar graphs adjacent to the numeric data, make relationships and trends among data more obvious.

You can apply data visualization graphs to selected measures from:

- Context menus in the report itself.

  This is the quickest way to apply data visualization bar graphs to numeric measures.
- The Measures control in the Selections pane.
- Check boxes in the Measures box on the Control Panel.

For details about data visualization graphs, see *Visualizing Trends in Reports*.

Procedure: How to Add a Column of Bar Graphs for a Numeric Measure

The quickest way to apply data visualization graphics is from the report itself:

1. Right-click the title of a measure column.
2. Choose *Visualize* from the menu.

The report runs automatically, displaying a column of bar graphs following the selected measures column.

Tip: To remove the bar graphs, right-click the measure column title and choose *Remove Visualization* from the menu.

For other methods of applying bar graphs to columns, see *Visualizing Trends in Reports*.

Displaying Graphs and Reports

In this section:

Title Added to OLAP Graphs

How to:

Graph a Measure From the Selections Pane

Graph a Measure From the Control Panel

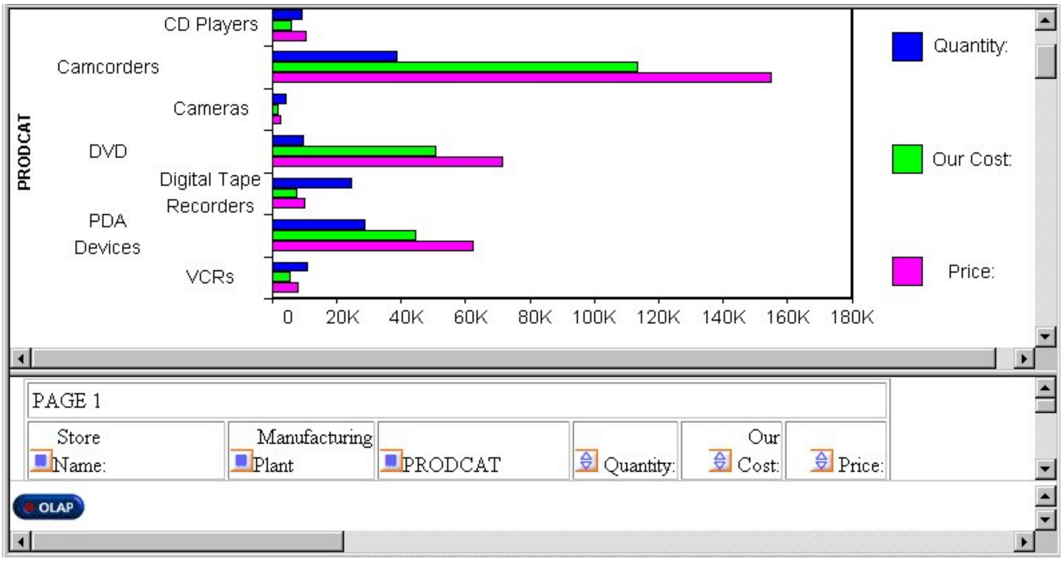
Reference:

Combining Graph Styles and Measure Styles in OLAP Graphs

When you graph a measure in an OLAP report, you select the specific data elements to include and view the tabular report and a graphical representation of the identical information simultaneously in a split window. The graph appears in a frame in the top half of the window to facilitate comparison.

To be graphable, the data in the report must include at least one numeric measure and one sort field (By or Across). The Graph control is activated in the Selections pane or the Control Panel when these basic requirements are met.

As shown in the following image, it includes three sort fields (PRODCAT, Store Name, and Manufacturing Plant) and three numeric measures (Quantity, Our Cost, and Price), displayed as horizontal bar charts for quick comparison.



You can request a graph from an OLAP report, from the Selections pane, or from the Control Panel:

- **From an OLAP report,** you can create a vertical bar chart to represent the data in a selected measure.
- **From the Selections pane or the Control Panel,** you can create seven different types of graphs and apply them to one or more measures:
  - Vertical Bar (This is the default graph type.)
  - Vertical Line
  - Vertical Area
  - Horizontal Bar
  - Horizontal Line
  - Horizontal Area
  - Pie

If you choose to graph more than one measure, you can employ different graph types to suit the data in each column, with the following restrictions:

- When you select Vertical or Horizontal Bar, Line, or Area as the controlling graph style for a measure, you can apply any combination of these styles to other measures. For example, the first measure can appear as bars, the second measure as lines, and the third measure as areas. All measures must have the same orientation (vertical or horizontal).
- When you choose Pie as the controlling graph style, you can use only pie charts for other measures.

For details about supported combinations, see *Combining Graph Styles and Measure Styles in OLAP Graphs* on page 53.

Note: If drill down capability has been enabled for the dimensions in a report, the same functionality is automatically enabled for graphs. You can, therefore, drill down from one graphical representation of your data to another.

Title Added to OLAP Graphs

Multiple graphs are generated when using the Graph feature in OLAP with multiple sort fields. A title has been added to each graph where the title is equal to the value of the outer sort field.

Reference: Combining Graph Styles and Measure Styles in OLAP Graphs

The following table lists the available style combinations in the second column for each graph style in the first column.

Controlling Graph Style	**Potential Measure Styles**
Vertical Bar (default)	Vertical Bar (default)
	Vertical Line
	Vertical Area
Vertical Line	Vertical Line (default)
	Vertical Bar
	Vertical Area
Vertical Area	Vertical Area (default)
	Vertical Bar
	Vertical Line
Horizontal Bar	Horizontal Bar (default)
	Horizontal Line
	Horizontal Area
Horizontal Line	Horizontal Line (default)
	Horizontal Bar
	Horizontal Area
Horizontal Area	Horizontal Area (default)
	Horizontal Line
	Horizontal Area
Pie	Pie

Procedure: How to Graph a Measure From the Selections Pane

1. Click the down arrow to the left of the Graph control to open a pane that contains all the numeric measures in the current report.

   There is a check box to the left of each measure and a graph button to the right of each measure. All check boxes are unchecked by default and all graph buttons are grayed (inactive) by default.

2. Select a check box associated with a measure.

   The graph button to the right of the measure becomes active. The default graph style is Vertical bar.

3. Toggle through the seven graph style icons until you reach the one you want to apply to the selected measure.
4. Repeat steps 2 and 3 for any other measures you want to graph.

   For a list of graph types that can be defined, see *Combining Graph Styles and Measure Styles in OLAP Graphs* on page 53.

5. Click the *Run* button on the band below the Selections pane.

   The graph opens in a separate frame above the report and Selections pane.

Procedure: How to Graph a Measure From the Control Panel

1. Open the Control Panel.
2. Select the *Show Graph* check box located below the Measures Properties box.

   Note that the contents of the Drill Down and Drill Across boxes determine the X-axis fields. When there are multiple drill (X-axis) fields, multiple graphs appear vertically stacked in the same frame. The measures appear as Y-axis fields on the graphs you display.

3. Click the *Graph* icon adjacent to the Show Graph check box.

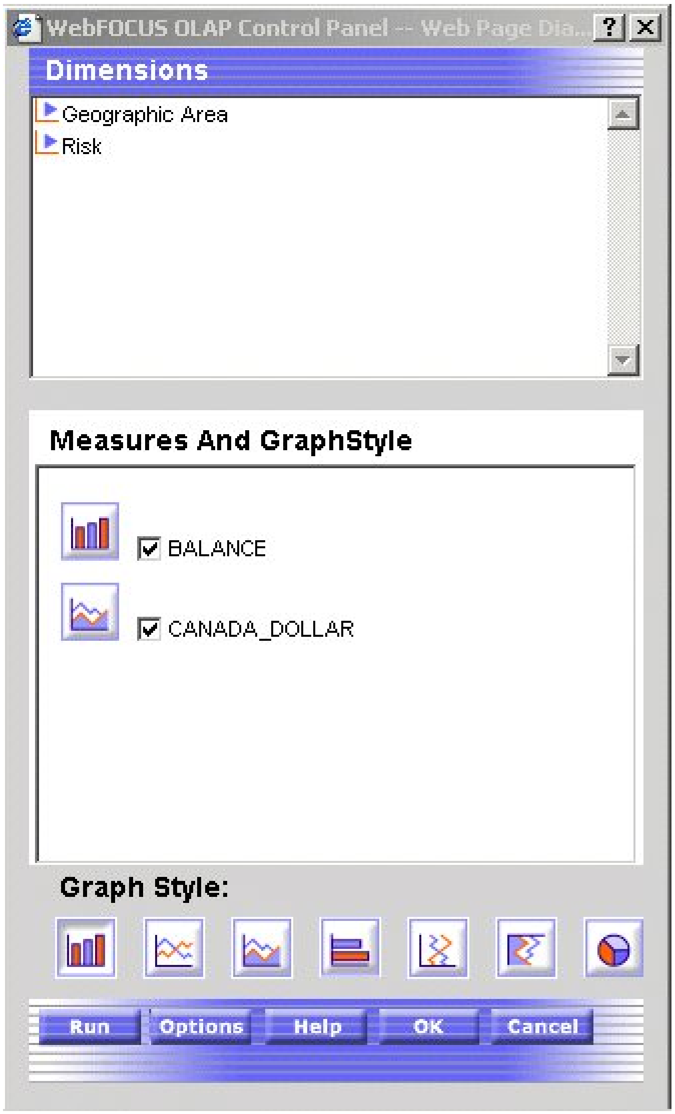
   The Measures and GraphStyle pane opens.

   Check boxes associated with the available measures are unchecked by default.

4. Click one of the seven icons at the bottom of the window to set a controlling graph style.
5. Select the check box(es) for the measure(s) you wish to graph.

   The graph icon corresponding to the controlling graph style appears next to each selected measure.

6. Click the icon next to a measure to choose a different graph style from the supported combinations as shown in the following image.



☑ BALANCE

☑ CANADA_DOLLAR

7. Click *OK* to return to the main Control Panel window with all the graph settings retained.

8. Click *Run* to display the graph(s) and the tabular report in a split window.

Note:

- If you select the Show Graph check box and click Run without selecting a controlling graph style, the default style (Vertical Bar) is applied.
- If you click Run without selecting the Show Graph check box, a tabular report appears without a graph.
- If you select at least one measure in the Measures and GraphStyle pane without selecting the Show Graph check box, when you click OK the system automatically selects the Show Graph check box. The tabular report appears with a graph.
- You cannot choose to graph alphanumeric or date fields. If there are no numeric measures, the Show Graph check box and the Graph button are disabled (grayed out).

Controlling the Display of Measures in a Report

In this section:

While you cannot add new measures to an OLAP report without returning to the original report request, you can adjust the display of the measures in the report in several ways. You can:

- Stack measures in rows.
- Change the order of measure columns.
- Hide and expose measures.
- Add a column of data visualization bar graphs following any numeric measure.

Stacking Measures

How to:

Display Stacked Measures

When you have more than one measure in an OLAP report, you can stack the measures in separate rows within the same column to reduce the width of the report.

You cannot apply data visualization bar graphs to stacked measures.

Procedure: How to Display Stacked Measures

1. Open the Control Panel.
2. Select the *Stack Measures* check box to display measures in separate rows under one column.
3. Click *Run* to execute your report.

Tip: To restore the standard display, deselect the *Stack Measures* check box and rerun the report.

Changing the Order of Measure Columns

How to:

Reposition Measure Columns in an OLAP Report

You can change the order in which measure columns are presented in the report.

Procedure: How to Reposition Measure Columns in an OLAP Report

To reposition a numeric column, drag and drop the field into a new column position.

The cursor changes to a plus (+) sign to indicate acceptable places into which you can drop the field. (Unacceptable positions are indicated by a circle with a slash cross the center.)

Hiding and Displaying Measures

How to:

Hide or Expose a Measure From the Report

Hide or Display a Measure From the Selections Pane

Display or Hide a Measure From the Control Panel

You can hide and expose measures from an OLAP report, the Selections pane, or the Control Panel.

Procedure: How to Hide or Expose a Measure From the Report

To hide a measure column, right-click the column title and choose *Hide* from the menu. The column is automatically removed from the display.

To expose a hidden measure column, right-click a displayed measure and choose *Unhide* from the menu. A secondary menu lists any hidden measures.

Choose the one you want to re-expose in the report.

Tip: If you want to add a new measure to the report, you must return to the original request and add the field there.

Procedure: How to Hide or Display a Measure From the Selections Pane

1. Click the down arrow to the left of the Measures control to display a list of the measures in the report.
2. Click the check box next to a measure to display or hide it. The check box toggles through three positions.
   - **To hide the measure,** click the check box until it is blank.
   - **To expose a hidden measure,** click the check box until you see a check mark.

Tip: You can use the same check box to display a column of data visualization bar graphs for numeric measures. This setting is represented as a graph in the check box. For details, see *Visualizing Trends* on page 50.

Procedure: How to Display or Hide a Measure From the Control Panel

1. Open the Control Panel.
2. In the Measures box, click the check box next to a measure to display or hide it. The check box toggles through three positions.

- **To hide the measure,** click the check box until it is blank.
- **To expose a hidden measure,** click the check box until you see a check mark.

Tip: You can use the same check box to display a column of data visualization bar graphs for numeric measures. This setting is represented as a graph in the check box. For details, see *Visualizing Trends* on page 50.

3. Click *Run* to execute your report.

Adding and Removing Dimensions

How to:

Add a Dimension Element From the Control Panel

Delete a Dimension Element From the Report

Delete a Dimension Element From the Control Panel

Since all of the values in a dimensions hierarchy are available in an OLAP report from Developer Workbench, you can add dimensions to the OLAP report at any time, without returning to the original report request. You can add dimensions from:

- An OLAP report.
- The Control Panel.

Procedure: How to Add a Dimension Element From the Control Panel

1. Open the Control Panel.

2. Select a report layout box (Drill Down or Drill Across) to indicate how you want the new sort dimension to be used in the report.

3. Expand a dimension in the Dimensions box at the top of the window, then click the dimension element you want to add to the designated layout box. The new dimension is added to the bottom of the list.

4. If you wish to change the position of the new sort field, click the up arrow to reposition it.

5. Click *Run* to execute your report with the new settings.

Procedure: How to Delete a Dimension Element From the Report

Right-click the dimension column you wish to remove and choose *Delete* from the menu.

The report runs automatically.

Procedure: How to Delete a Dimension Element From the Control Panel

1. Select the element in the Drill Down or Drill Across box. The buttons above the box become active.
2. Click *Remove* ☒. The element is deleted from the Drill Down or Drill Across box.
3. Click *Run* to see the new report.

Saving OLAP Reports

In this section:

Saving Options

Uniform Field Name Referencing in OLAP

DB2 Web Query includes several saving features for OLAP reports:

- The dialog for saving reports allows them to be placed in subfolders of the user's choice under Reports; subfolders can also be created from the OLAP tool.
- Users can save their reports with the OLAP functionality removed, so the OLAP tool can serve as a simple report writer tool.
- Field name referencing is uniform throughout the OLAP product. For example, the AS or TITLE phrases will appear in reports generated using the OLAP Selections pane or the OLAP Control Panel (OCP).

Note: When saving OLAP reports, you must refresh the Domain to see the newly saved reports. Otherwise, the new reports will not be listed in the domain tree.

Saving Options

How to:

Save a Report in OLAP

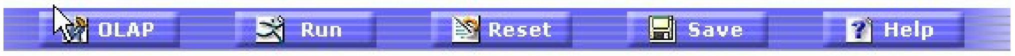
Save an OLAP-enabled Report as a Plain Report Without OLAP Functionality

If you are working in the OLAP Selections pane, you can save your report by using the Save button on the toolbar. You can also access other saving options in the OCP by selecting the Options button.

Procedure: How to Save a Report in OLAP

1. Click the *Save* button in the OLAP Selections Pane.

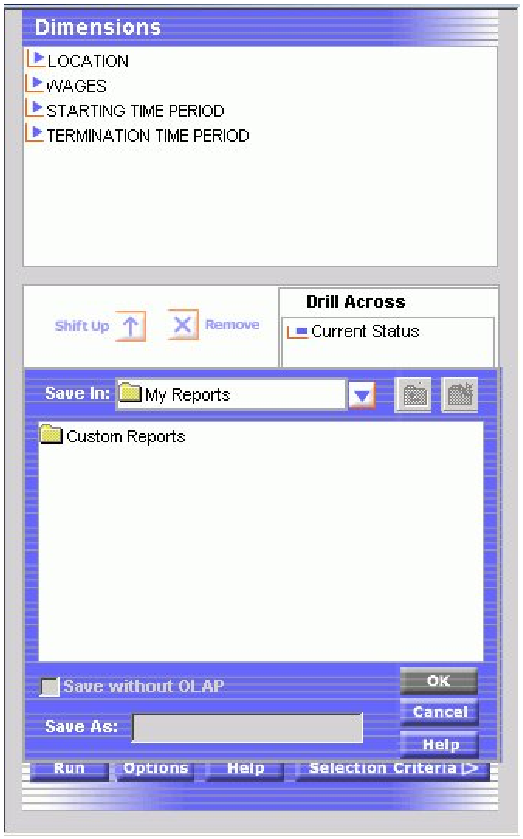
The following image shows the OLAP Selections Pane containing five buttons, OLAP, Run, Reset, Save, and Help.



or

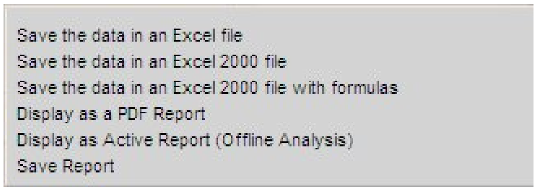
Click the *Options* button in the OLAP Control Panel (OCP).

The following image shows the OLAP Control Panel where the Options button appears at the bottom of the panel.

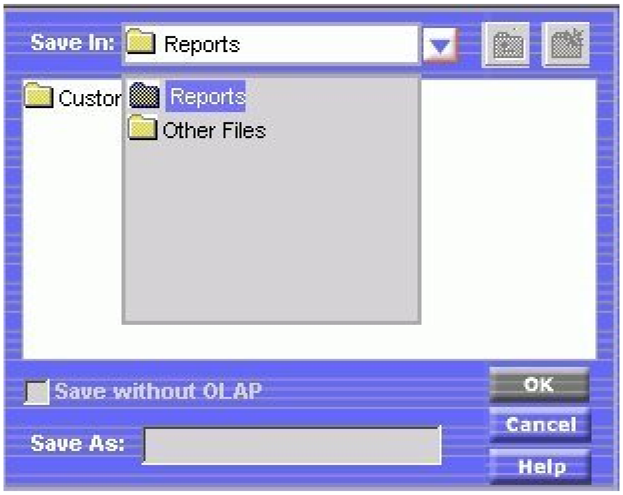


You are now presented with several saving options.

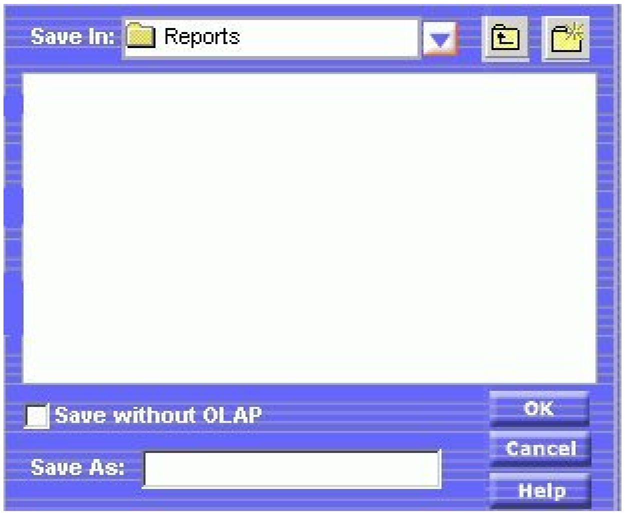
The following image shows the Options menu containing six options, Save the data in an Excel file, Save the data in an Excel 2000 file, Save the data in an Excel 2000 file with formulas, Display as a PDF Report, Display as active report (Offline Analysis), and Save Report.


Save the data in an Excel file
Save the data in an Excel 2000 file
Save the data in an Excel 2000 file with formulas
Display as a PDF Report
Display as Active Report (Offline Analysis)
Save Report

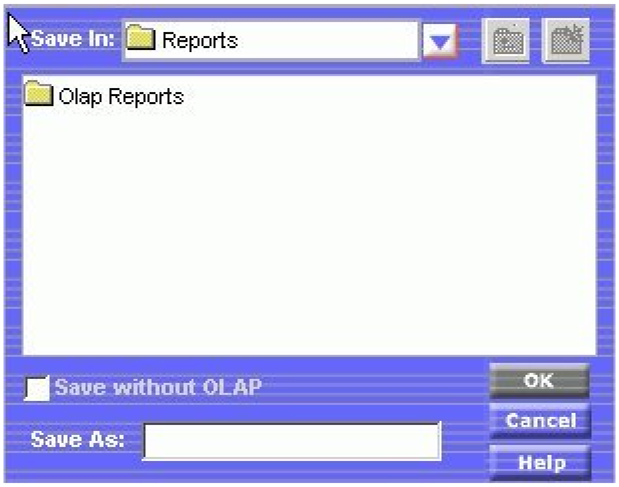
2. Click *Save Report* to open a new dialog box as shown in the following image.



3. Click *Reports* to enable the New Folder icon in the upper right corner as shown in the following image.



4. A pop-up prompts you to create a new folder, for example, a subfolder called OLAP Reports as shown in the following image.



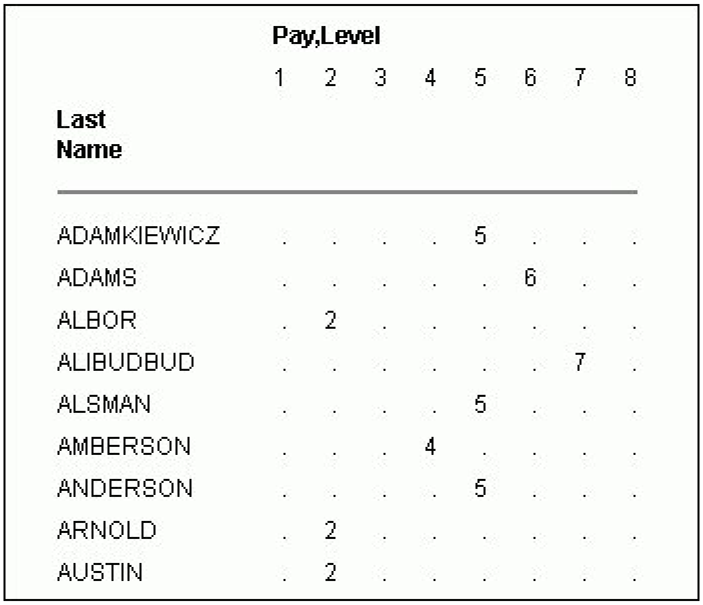
5. Name your report in the Save As box and click *OK* to return to your report.

Procedure: How to Save an OLAP-enabled Report as a Plain Report Without OLAP Functionality

Users can save their reports with the OLAP functionality removed, so the OLAP tool can be used as a simple report writer tool.

1. Create a subfolder by following these procedures: *How to Save a Report in OLAP* on page 60.
2. After a subfolder is selected, check the *Save without OLAP* box.
3. Enter a new name for your report in the *Save As* box and click *OK*.

DB2 Web Query generates a new report without OLAP functionality. The following image shows a normal report of employees' pay levels which is sorted by employee's last name.



	Pay,Level							
Last Name	1	2	3	4	5	6	7	8
ADAMKIEWICZ	.	.	.	.	5	.	.	.
ADAMS	.	.	.	.	.	6	.	.
ALBOR	.	2	.	.	.	.	.	.
ALIBUDBUD	.	.	.	.	.	.	7	.
ALSMAN	.	.	.	.	5	.	.	.
AMBERSON	.	.	.	4	.	.	.	.
ANDERSON	.	.	.	.	5	.	.	.
ARNOLD	.	2	.	.	.	.	.	.
AUSTIN	.	2	.	.	.	.	.	.

Uniform Field Name Referencing in OLAP

However the developer designs the report with regard to field referencing will carry through to both the OCP and the OLAP Selections pane. Field referencing does not differ between the report and the OCP and OLAP Selections pane. Field references by AS, TITLE, or Field Name, are uniform in the report output and OLAP controls.

Saving and Displaying OLAP Reports and Graphs in Other Formats

How to:

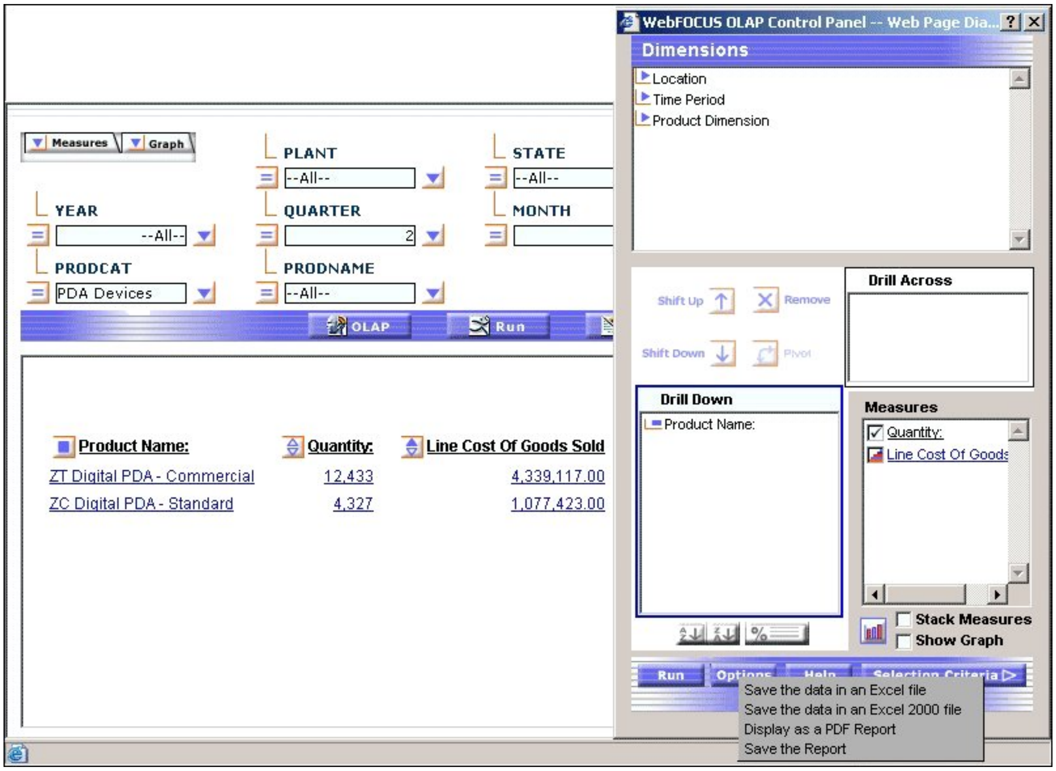
Display an OLAP Report and Graph in PDF Format

Save an OLAP Report and Graph as an Excel File

Save an OLAP Report and Graph in the Reports Folder

OLAP reports and graphs appear in your browser in HTML format. You can display the report and corresponding graph in PDF and Excel formats.

The following image shows a pop up menu listing the available formats to save and/or display.



Product Name:	Quantity:	Line Cost Of Goods Sold
ZT Digital PDA - Commercial	12,433	4,339,117.00
ZC Digital PDA - Standard	4,327	1,077,423.00

- ❏ PDF (Adobe Acrobat's Portable Document Format) is useful when you want a report or graph to maintain its presentation and layout regardless of a user's browser or printer type.

  When you choose PDF format, the report appears in Adobe Acrobat Reader; the graph continues to appear above it in a browser window. If you print from Acrobat, only the report will be printed.

- ❏ Excel is useful when you want to convert a large database to a spreadsheet or save a report and graph in a commonly used Office tool. Two Excel formats are available:

  - ❏ Excel 2000 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.

    When you choose Excel 2000, the report and graph are displayed in the same tool where you can manipulate the data using Excel options. From Excel you can print both the report and the graph.

    When you save in Excel 2000 format, only explicit drill-downs (based on parameters passed from the base report to the drill-down report) continue to work. Automatic drill downs on Dimensions and Measures are not supported in Excel.

  - ❏ Excel is a binary display format with limited formatting support. The computer on which the report is being displayed must have Microsoft Excel installed.

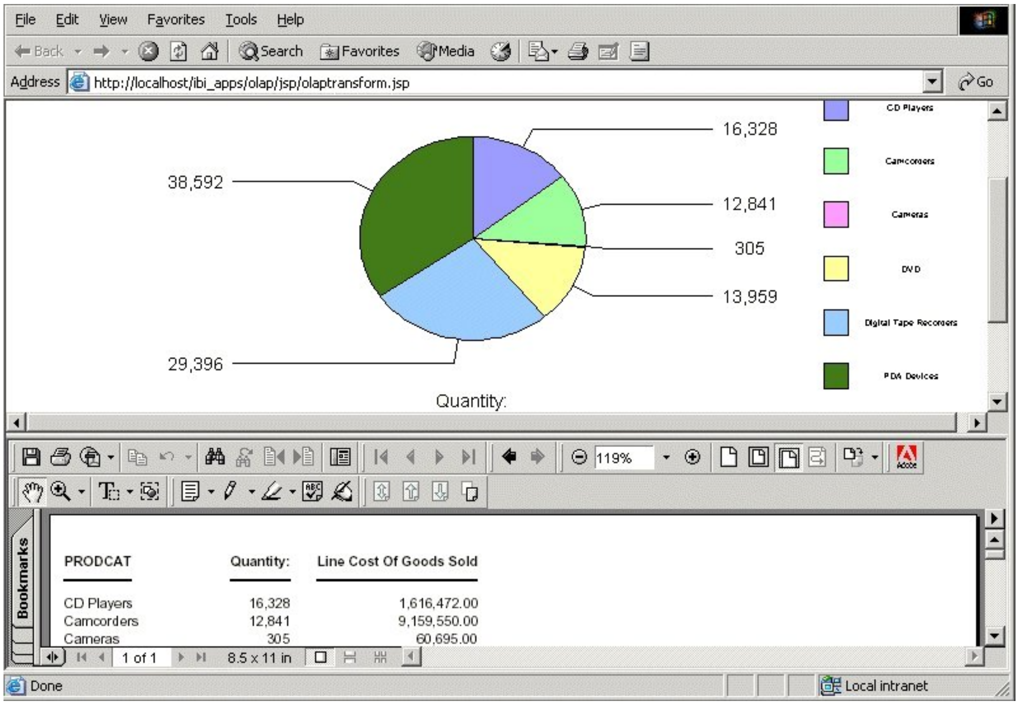
    Drill-downs of any kind are not supported.

A user can save the HTML output in the Reports folder.

Procedure: How to Display an OLAP Report and Graph in PDF Format

1. Open the Control Panel.
2. Click the *Options* button at the bottom of the window.
3. Select *Display as a PDF Report*.

The graph appears in the browser above the report, while a second browser opens and launches the report output in Adobe Acrobat as shown in the following image.



PRODCAT	Quantity:	Line Cost Of Goods Sold
CD Players	16,328	1,616,472.00
Camcorders	12,841	9,159,550.00
Cameras	305	60,695.00

Tip: If you wish, you can save and print the PDF report from Adobe Acrobat.

Procedure: How to Save an OLAP Report and Graph as an Excel File

1. Open the Control Panel.
2. Click the *Options* button at the bottom of the window.
3. Select *Save the data in an Excel file* or *Save the data in an Excel 2000 file*.
4. Follow the instructions to export the data.

Procedure: How to Save an OLAP Report and Graph in the Reports Folder

1. Open the Control Panel.
2. Click the *Options* button at the bottom of the window.
3. Select *Save as*.

   A secondary window opens.

4. Enter a descriptive name and click *OK* to save the graph(s) and the tabular report.

Note: There is no limit to the number of characters in a graph legend's label, but long labels may appear truncated.

2 Describing and Accessing Data Overview

DB2 Web Query and Developer Workbench provide a set of graphical tools that you can use to describe and access many types of data sources, including:

- Relational, such as DB2 and Microsoft SQL Server
- Procedures (Query/400)
- Multiple member file format (Multiple DB Heritage File)
- ERP (JDE World and JDE EnterpriseOne)

These graphical tools are designed to:

- Translate the schema of the data source into metadata that DB2 Web Query can read and report against.
- Developer Workbench is an optional product that enables you to customize and enhance the generated metadata without having to know the details of the underlying DB2 Web Query data description language.

Topics:

A Note About Data Source Terminology

Different types of data sources make use of similar concepts, but refer to them differently. For example, the smallest meaningful element of data is called a *field* by many hierarchical database management systems and indexed data access methods, but called a *column* by relational database management systems.

There are other cases in which a common concept is identified by a number of different terms. For simplicity, we have standardized on a single set of terms. For example, we usually refer to the smallest meaningful element of data as a *field*, regardless of the type of data source. However, when required for clarity, we use the term specific to a given data source. Each time we introduce a new standard term, we define it and compare it to equivalent terms used with different types of data sources.

How Applications Interpret Data

When your application accesses a data source, the application needs to know how to interpret the data that it finds. To accomplish this, the application reads a *synonym*, which is a term for the generated *metadata* associated with the particular data source.

Your application needs to know:

- The overall structure of the data. For example, is the data relational, hierarchical, multidimensional, or sequential? Depending upon the structure, how is it arranged or indexed?
- The specific data elements. For example, what fields are stored in the data source, and what is the data type of each field (character, date, integer, or some other type)?

The synonym provides an alias for the data source that tells the server how tables are described and where to find them.

The primary component of a synonym is a Master File. The Master File describes the structure of a data source and its fields. For example, it includes information such as field names and data types.

For some types of data sources, an Access File supplements the Master File. An Access File includes additional information that completes the description of the data source. For example, it includes the full data source name and location. The nature of the information in the Access File is specific to each data source. You need one Master File and for some types of data sources, one Access File to describe a data source.

CreatingSynonyms with a Data Adapter

How to:

Configure an Adapter Without the Wizard

Reference:

How an Adapter Works

You can generate synonyms using either of the Metadata menu options in DB2 Web Query. This enables you to explore DBMS catalogs and select the objects for which you wish to create synonyms. The tool prompts for the information it needs to create a synonym for a particular data source and stores the generated synonym on the server.

In order to generate a synonym, you must be authorized to use the data against which you plan to report and you must have configured an adapter to access that type of data. When you begin to create a synonym, Developer Workbench opens the Select adapter to configure or Select connection to create synonym window. The option to create the synonym becomes available only after the adapter is successfully configured.

Adapters are available for many data sources. Every adapter is specifically designed for the data source that it accesses, and, as a result, is able to translate between SQL or DB2 Web Query and the data management language (DML) of the data source. Adapters provide solutions to product variations, including product differences in syntax, functionality, schema, data types, catalogs, data representations, message processing, and answer set retrieval. It is the adapter that manages the synonym creation process. For related information, see *How an Adapter Works* on page 71.

Reference: How an Adapter Works

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

- Translates the request to the applicable DML.
- Attaches to the targeted data source, using standard attachment calls.
- Passes the request to the data source for processing.
- Returns results or error conditions to the client application for further processing.

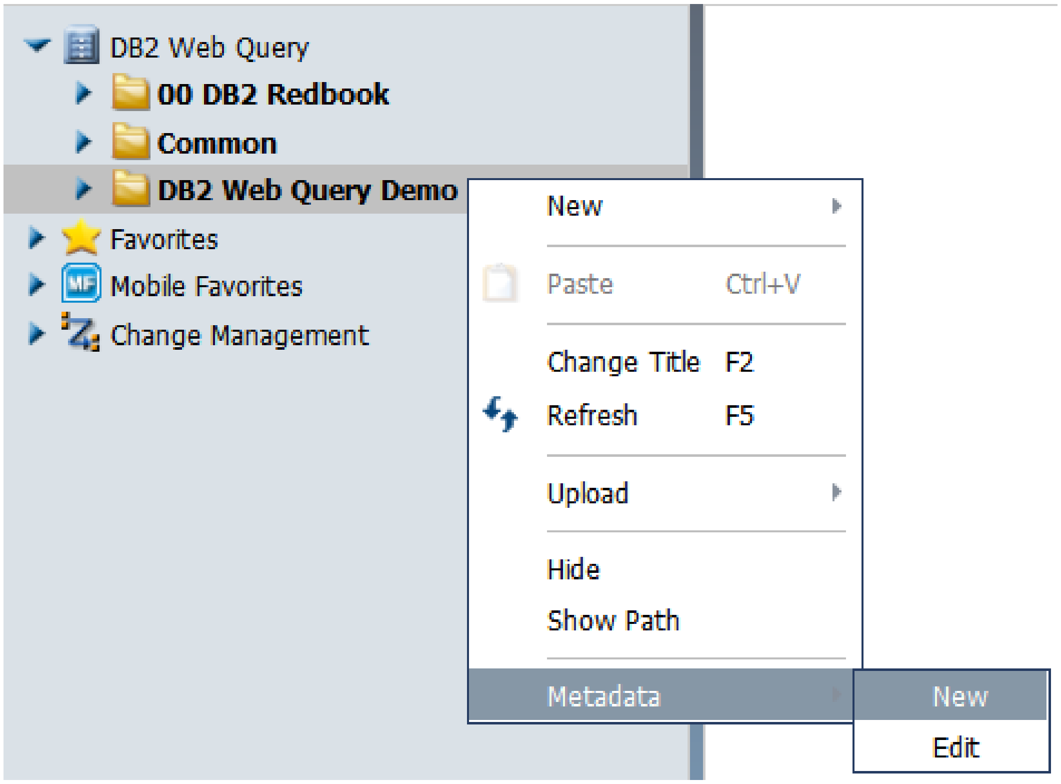
Procedure: How to Configure an Adapter Without the Wizard

1. Log on to DB2 Web Query as a Web Query administrator (a user ID that is a member of the WebQueryAdministrator Group) or a user who has DBA privileges (a user ID that is a member of any top level folder-dba group).

   These type of users can configure and manage the adapter configuration, other users are not permitted to manage and configure adapters.

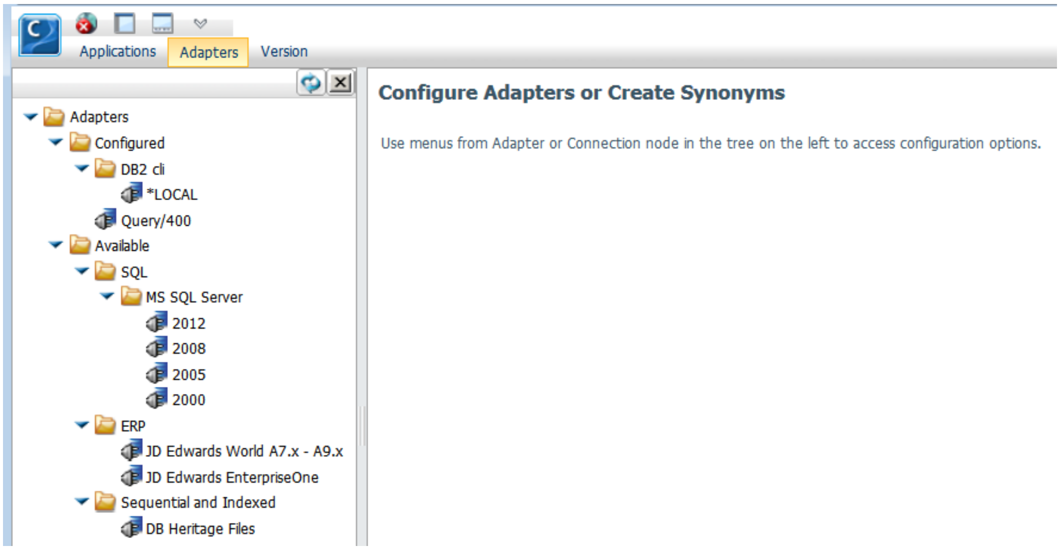
2. Right-click any folder and select *Metadata Edit*.

   The following image illustrates the steps to open the Configuring Adapter window in the Reporting Server.



3. Expand the *Available* folder for a list of available adapters.

The following image shows the expanded Available folder of the Reporting Server.



Configure Adapters or Create Synonyms

Use menus from Adapter or Connection node in the tree on the left to access configuration options.

4. Select the adapter you wish to configure and provide the connection parameter values.

5. Refer to the Reporting Server Help for information on configuring the selected adapter.

What You Can Do With a Synonym

Once you have generated a synonym, you can report against the synonym using all of the DB2 Web Query reporting tools. In many instances, the configured adapter and the generated synonym are all you need to access your data and create reports and graphs.

However, you may wish to enhance the synonym in order to implement particular capabilities that are supported in the DB2 Web Query data description language. To do this, it requires the optional product Developer Workbench which contains the Synonym Editor.

When you use the Synonym Editor, there is no need to know the data description language. The graphical tool displays all viewable and editable attributes of the synonym components. If you make changes to the generated synonym, the Synonym Editor validates your entries and displays messages if they violate the underlying syntax of the data description language.

Ways to Enhance a Synonym

The following are some of the attributes you might want to add to the synonym to enhance your data access and reporting capabilities. You can:

❏ Create a cluster join view by linking available synonyms to create a multisegment (multitable) file for reporting.

- Add dimensions for OLAP analysis.
- Add virtual columns (DEFINE fields) and columns for aggregated values (COMPUTE fields).
- Add filters to specify data selection criteria.
- Add group definitions for data sources that support groups.
- Add meaningful titles and descriptions, including multilanguage variations.
- Change the format of fields (for example, the size of an alphanumeric field or the format of a date field).
- Create business views of the metadata in order to limit the fields available to any retrieval request that references the business view and to group fields together based on their roles in an application.
- Define parent and child hierarchies for cube data sources.

The Synonym Editor is described in more detail in *Using the Synonym Editor* on page 75.

How an Application Uses a Synonym

Synonyms are stored separately from the associated data source. Your application uses a synonym to interpret the data source in the following way:

1. It identifies, locates, and reads the Master File for the data source named in a request.
2. It locates and reads the Access File for the data source named in the request, if that type of data source requires an Access File.
3. It locates and reads the data source(s).

The data source contents are interpreted based on the information in the synonym.

IBM

3 Using the Synonym Editor

A synonym consists of a set of attributes that describe a data source. It provides the metadata for a data source, which enables an adapter to access and interpret the corresponding data. The Synonym Editor provides a graphical interface that enables you to work with synonyms and perform tasks, such as create, view, and modify.

Note: A browser based version of the Synonym Editor has been added with DB2 Web Query V2R1M0. Please see the Summary of New Features document for more details on the browser based version. This chapter describes the version that is accessible through the DB2 Web Query Developer Workbench.

Topics:

Synonym Editor Layout

Reference:

Synonym Editor Main Attributes

Synonym Editor Toolbar

Synonym Editor Options Settings

Synonym Editor Format Options Settings

Synonym Editor Column Management Settings

Synonym Editor Traces Options Settings

Synonym Editor Run Options Settings

Synonym Editor - Segment Context Menu

Synonym Editor - Column/Field Context Menu

Synonym Editor - Field View Tab

Synonym Editor - Segment View Tab

Synonym Editor - List View Tab

Synonym Editor - Modeling View Tab

Synonym Editor - Text View Tab

Synonym Editor - Access File Text View Tab

The Synonym Editor is available from DB2 Web Query Developer Workbench. It provides the following tabs that enable you to view and manage synonyms: Field View, Segment View, List View, Modeling View, Text View, and Access File Text View.

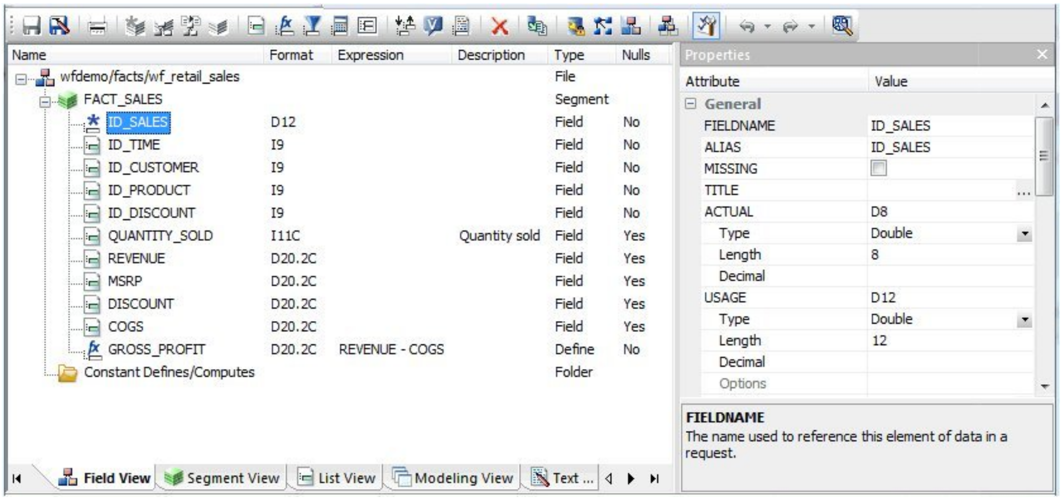
Reference: Synonym Editor Main Attributes

The Field View tab and Segment View tab show a hierarchy of segments and columns on the left with the attributes and values of the selected item in the Properties window, which displays by default on the right side of the tool.

Note: The attributes available depend on the type of synonym.

The Properties pane and the other available panes (Business View, Dimension Builder, or DBA) can be resized, positioned in different areas, or hidden by the available toolbar icons.

The following image is an example of an SQL data source with a key column selected.



Name	Format	Expression	Description	Type	Nulls
wfdemo/facts/wf_retail_sales				File	
FACT_SALES				Segment	
ID_SALES	D12			Field	No
ID_TIME	I9			Field	No
ID_CUSTOMER	I9			Field	No
ID_PRODUCT	I9			Field	No
ID_DISCOUNT	I9			Field	No
QUANTITY_SOLD	I11C		Quantity sold	Field	Yes
REVENUE	D20.2C			Field	Yes
MSRP	D20.2C			Field	Yes
DISCOUNT	D20.2C			Field	Yes
COGS	D20.2C			Field	Yes
GROSS_PROFIT	D20.2C	REVENUE - COGS		Define	No
Constant Defines/Computes				Folder	

Properties

Attribute	Value
General	
FIELDNAME	ID_SALES
ALIAS	ID_SALES
MISSING	☐
TITLE	
ACTUAL	D8
Type	Double
Length	8
Decimal	
USAGE	D12
Type	Double
Length	12
Decimal	
Options	

FIELDNAME
The name used to reference this element of data in a request.

Information about the selected attribute is displayed at the bottom of the Properties pane. In this case, an explanation of the FIELDNAME attribute appears.

The following objects may appear in the tabs.

Object	Name	Function
▲	Synonym.	General icon used to indicate synonyms, visible on the left of the Field View tab. Provides information about the file name used and the application in which it resides.
🟩	Segment.	Indicates a segment in a synonym. The root or parent segment appears first in the tree.
▨	Virtual Segment (Cross-Referenced).	Indicates a virtual or cross-referenced segment. The icon is dimmed.
*	Key Column.	Indicates a key column.
▤	Column.	Indicates a general column.

Object	Name	Function
	Virtual Column (DEFINE).	Indicates a virtual or defined column. For more information about virtual columns, see *Adding Virtual Columns (DEFINE) in a Synonym* on page 150.
	Index Column.	Indicates that the native DBMS has an index for quick retrieval of values for this field.
	Filter.	Indicates a Master File filter.
	Compute Field.	Indicates a computed field.
	Group.	Indicates a group. This option is enabled when you select *Options* from the Tools menu. In the Options dialog box, click the check box for *Support extended options*.
	Sort Object.	Indicates a sort object. This option is enabled when you select *Options* from the Tools menu. In the Options dialog box, click the check box for *Support extended options*.
	Style.	Indicates a style object. This option is enabled when you select *Options* from the Tools menu. In the Options dialog box, click the check box for *Support extended options*.

Reference: Synonym Editor Toolbar

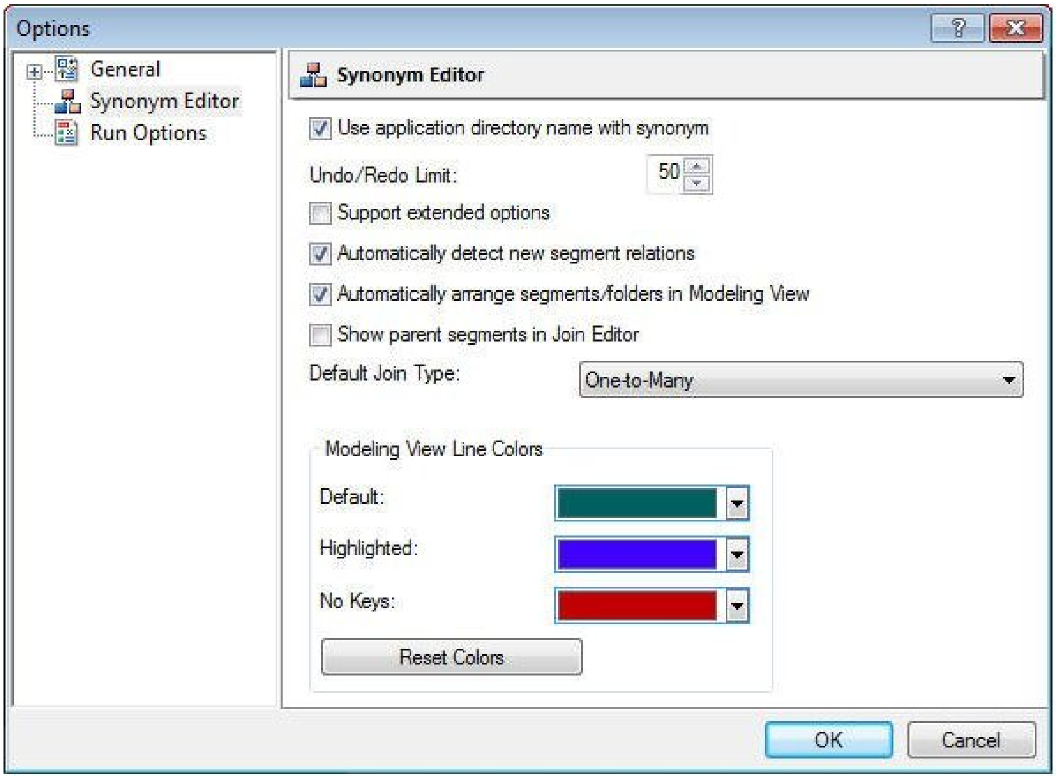
The Synonym Editor toolbar contains buttons that provide quick access to commonly performed functions. The behavior of the button is determined by the selected object. Therefore, certain toolbar buttons may be inactive.

Button	Description
💾	Saves the edits made to the synonym.
	Saves the edits made to the synonym to a new file (Save As).
🖨	Enables you to print the current view.
	References segment from an existing synonym.
	Enables you to add segments from an existing synonym.
	Enables you to add segments through the Segment via Metadata Import option using the Create Synonym tool. This tool creates a synonym and includes it as a segment in the synonym from which the tool was launched.
	Enables you to manually add segments, assigning values to segment attribute fields in the Synonym Editor. Use this approach only if you are coding a new Master File, as you would for a DB2 Web Query data source.
	Adds a column field.
	Adds a virtual column (DEFINE) field. For more information about virtual columns, see *Adding Virtual Columns (DEFINE) in a Synonym* on page 150.
	Adds a Master File filter. For more information about filters, see *Creating Filters in a Synonym* on page 154.
	Adds a COMPUTE field. For more information about computed fields, see *Adding Computed Fields (COMPUTE) in a Synonym* on page 158.

Button	Description
	Adds a group. For more information about groups, see *Adding Group Fields in a Synonym* on page 173.
	Inserts a variable in the segment.
	Deletes the selected item.
	Enables you to view and refresh sample data for the object.
	Enables DBA data source access.
	Enables a parent and child hierarchy in Dimension Builder.
	Enables you to create a Business View and a custom Master File that can use selected fields from the original synonym. In addition, you can customize field names, titles, and descriptions. For more information about Business Views, see *Creating Business Views* on page 143.
	Creates a default tree structure in the Business View and Dimension Builder panes based on existing segments and fields in the Master File.
	Describes the attribute properties.
	Enables you to undo or redo your actions.
	Provides a window with a compressed view of the synonym. Only available from the Modeling View tab.

Reference: Synonym Editor Options Settings

When you click *Tools* from the menu bar and select *Options*, the Options dialog box appears, as shown in the following image. It enables you to set preferences and customize the look of Synonym Editor.



[✓] Use application directory name with synonym

Undo/Redo Limit: 50

[ ] Support extended options

[✓] Automatically detect new segment relations

[✓] Automatically arrange segments/folders in Modeling View

[ ] Show parent segments in Join Editor

Default Join Type: One-to-Many

Modeling View Line Colors

Default: 🟩

Highlighted: 🟦

No Keys: 🟥

The Synonym Editor settings page has the following fields and options:

Use application directory name with synonym

If this check box is selected, an application directory name is used when you select a synonym name for both referencing an existing synonym and a transformation with db_lookup.

Undo/Redo Limit

Specifies the maximum number of undo/redo operations allowed in the Synonym Editor.

Support extended options

If this check box is selected, both the Sort objects and Styles folders appear when you edit a synonym.

Automatically detect new segment relations

When you create a new synonym in the modeling view of the Synonym Editor with the Automatically detect new segment relations option selected and select or drag tables into the work area (modeling view). The relationships (joins) will automatically be created for you based on the foreign key information in the access file.

Automatically arrange segments/folders in Modeling View

If you select or drag multiple or individual tables while in Modeling View, they will be automatically arranged for you.

Show parent segments in Join Editor

If this check box is selected, it controls whether or not columns in parent segments are displayed in the Join Editor for a cluster join or a synonym that references or includes, other synonyms.

Default Join Type

Sets the default Join type.

- ❑ **One-to-Many.** This join indicates a multiple instance (one-to-many) type of join. At run time, each host record can have many matching records in the cross-referenced file. Join All is the default option.
- ❑ **One-to-One.** This join indicates a single instance (one-to-one) type of join. At run time, each host record has, at most, one matching record in the cross-referenced file.

Modeling View Line Colors list box

Allows you to set colors for connector lines in the modeling view.

Default. Changes the color lines in the modeling view.

Highlighted. Changes the color of the highlighted lines in the modeling view.

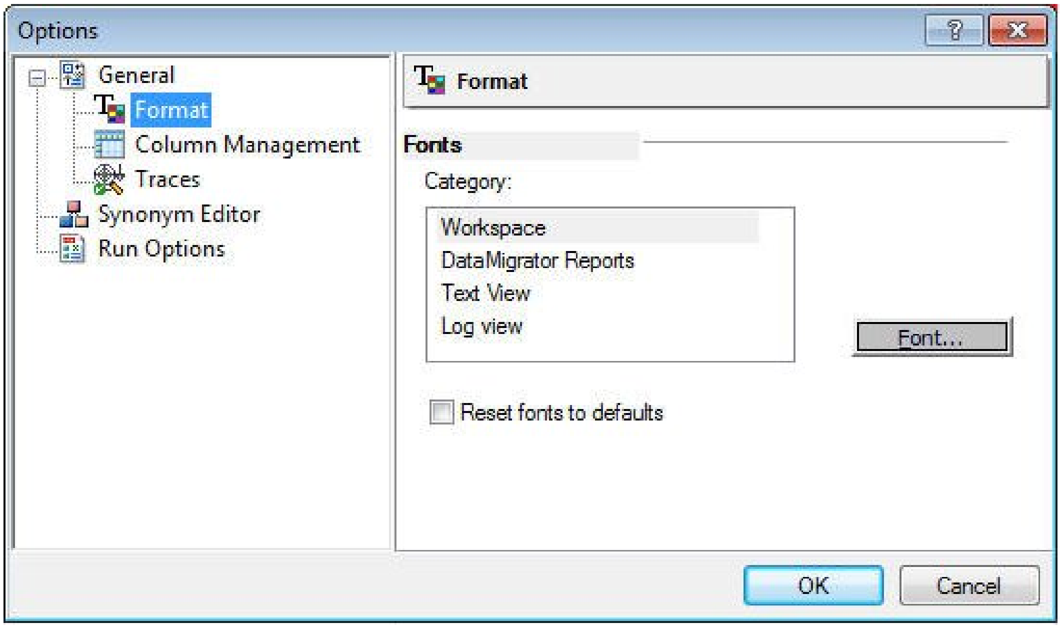
No keys. Changes the color of lines with no keys in the modeling view.

Reset Colors

Restores the default colors.

Reference: Synonym Editor Format Options Settings

The Format pane is available from the Options dialog box. To open the Options dialog box, select *Options* from the Tools menu. Expand the *General* node, and select *Format*, as shown in the following image.



The Fonts section has the following fields and options.

Category

Allows you to set the font for text in the process flow workspace, reports, text views, and log views.

Font

Launches a dialog box for specifying font settings.

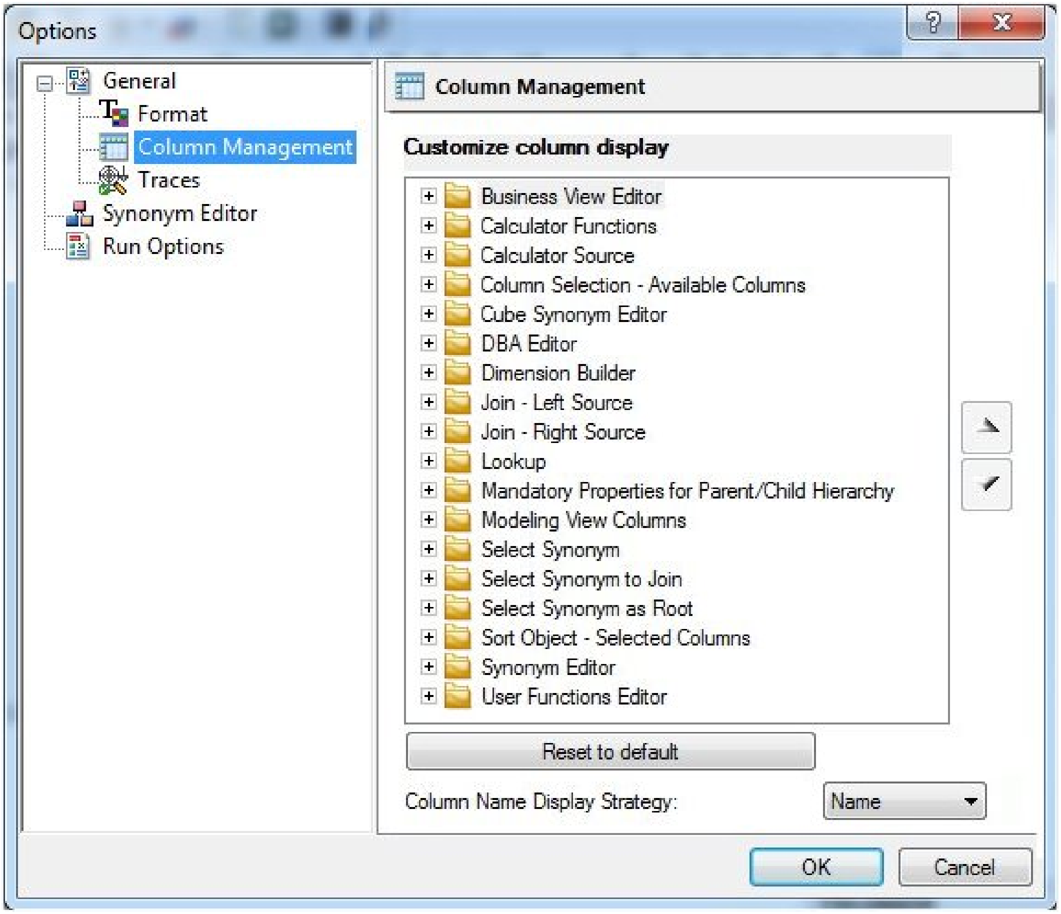
Reset font to defaults

Restores the default fonts.

Reference: Synonym Editor Column Management Settings

Column Management user preferences enable you to choose which columns to display on the grids for transformations, column selection, joins, and sorts. You can also set the column display order. The settings apply to all column-related dialog boxes.

The Column Management pane is available from the Options dialog box. To open the Options dialog box, select *Options* from the Tools menu. Expand the *General* node, and select *Column Management*, as shown in the following image.



Column Name Display Strategy: Name

The Column Management pane has the following fields and options:

Customize column display

Lists the column-related dialog boxes that can be customized. Expanding a folder will display check boxes that can be used to add columns to each dialog box.

Reset to default

Restores the default values.

Column Name Display Strategy

Controls the information that appears in trees and grids. The available options are Name, Title, Description, and Alias. If no Title, Description, or Alias exists, the display will default to the Name.

Note: It is recommended that you use Title when working with Business Views.

Expand any of the available Customize column display options to see default settings. Not all columns are on every grid. The following columns can be added:

Alias

Assigns an alternative name for a column or the real column name for a DBMS synonym.

Application

Indicates the application where the synonym resides.

Belongs To Segment

Shows the parent segment.

Connection

Indicates the adapter connection name used.

Data Origin

Indicates the date the synonym was created.

Datetime Modified

Indicates when the synonym was last modified.

Description

Is a description or comments about the column.

Expression

Is the expression for the column.

Extension

Indicates the suffix (data source type) of the synonym.

Field Type

Indicates that a column is an index (I) or is read-only.

Format

Is the type and length of a column data as stored.

Function

Indicates the name of the function.

Has Foreign Keys

Indicates that the synonym includes foreign keys.

Index

Indicates an index column.

Join Conditions

Indicates the conditions for the join.

Join Parent

Indicates the parent of the join.

Join Strategy

Indicates strategy for the join.

Keys

Indicates the keys in the synonym.

Length

Is the column length.

Nulls

Indicates whether or not the column can contain null data.

Number of Segments

Indicates the number of segments in the synonym.

Order

Indicates the order of the column in the segment.

Prefix

Indicates a prefix for the column.

Primary Key Tables

Indicates the primary key for the synonym.

Property

Indicates whether the column is an attribute or a measure.

Real Table Name

Indicates the actual name of the table or the physical file name.

Reference

Indicates a reference for the column to an index column.

Scale

Is the maximum number of digits to the right of the decimal.

SCD Type

Used for processing slowly changing dimensions.

Segment

Indicates the parent segment.

Size

Indicates the size of the synonym.

Source

Indicates the source of the synonym.

SQL Conversion Notes

Indicates how SQL is converted.

Table

Is the synonym that contains the column.

Title

Supplies a title to replace the column name normally used in reports.

Type

Is the type of object in an application directory.

The following columns are available from each column display option. The default columns are shown in bold.

- Business View Editor.

  Belongs to Segment, Format, Expression, Description, Nulls, Segment, Type, Length, Scale, Alias, Title, SCD Type, Field Type, Index, Order, Property, Reference.

- Calculator Functions.

  Format, Description, SQL Conversion Notes.

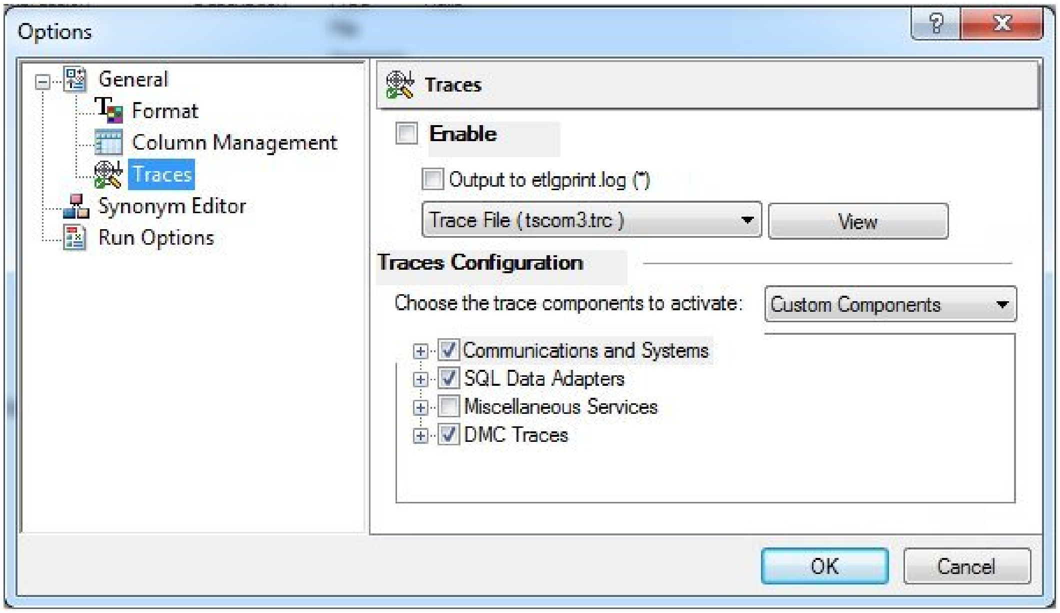
- Calculator Source.

Table, Format, Description, Nulls, Prefix, Segment, Type, Expression, Length, Scale, Alias, Title, SCD Type, Field Type, Index, Order, Property, Reference.

You can change the display order of the columns by moving them up or down using the arrows.

Reference: Synonym Editor Traces Options Settings

Trace option settings enable you to configure tracing for the server, as shown in the following image.



[ ] Enable

[ ] Output to etlgprint.log (*)

Trace File (tscom3.trc)

Choose the trace components to activate: Custom Components

[✓] Communications and Systems

[✓] SQL Data Adapters

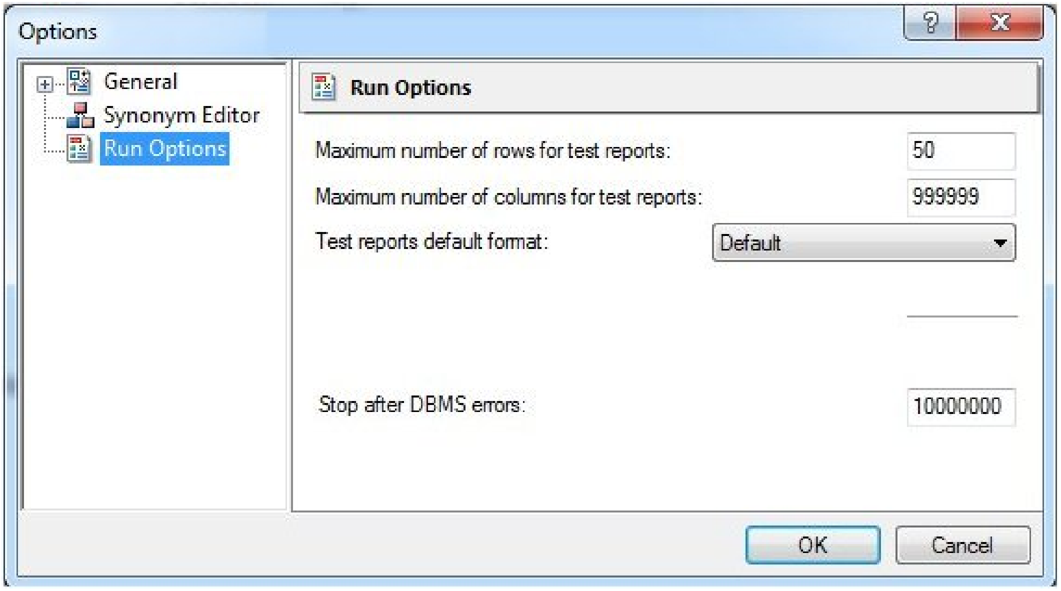
[ ] Miscellaneous Services

[✓] DMC Traces

You can enable tracing for all components, default components, or a set of custom components. Trace output can be directed to the Web Console log (etlgprint.log), or to the tscom3.trc or connection.trc trace files.

Reference: Synonym Editor Run Options Settings

Run Options user preferences include determining the number of rows and columns to retrieve, as well as the default format when sampling data.



Options

General
Synonym Editor
Run Options

Run Options

Maximum number of rows for test reports:	50
Maximum number of columns for test reports:	999999
Test reports default format:	Default
Stop after DBMS errors:	10000000

The Run Options settings window has the following fields and options:

Maximum number of rows for test reports

Sets the number of rows retrieved to produce sample data when testing transformations or SQL. The default is 50.

Maximum number of columns for test reports

Sets the number of columns retrieved to produce sample data when testing transformations or SQL. The default is to retrieve all columns (with a highest value setting of 999999).

Test reports default format

Sets the format of reports for retrieving sample data when testing transformations or SQL. The default is Default. There are seven report formats available:

- **Default.** Formats numeric and date columns based on edit options in the synonym.
- **HTML.** Produces the report in HTML format.
- **HTML - Plain Text.** Produces the report in plain text format.
- **active report.** Produces an HTML active report designed for offline analysis.

- **Excel.** Produces the report in Excel format.
- **PDF.** Produces the report in PDF format.
- **Unformatted.** Does not apply formatting to numeric and date columns.

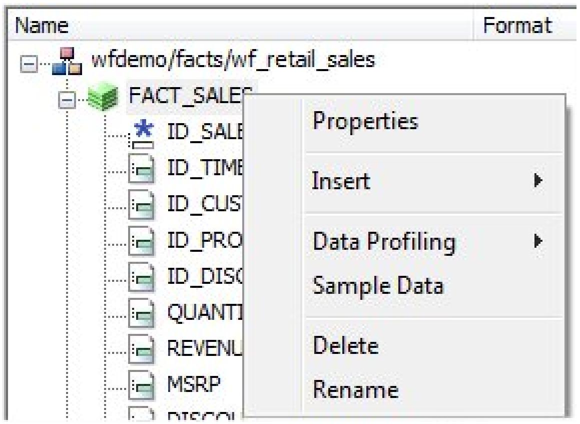
Stop after DBMS error

Sets the number of DBMS-related errors allowed before the server stops running the procedure.

Reference: Synonym Editor - Segment Context Menu

When you right-click a segment in the Synonym Editor, the following context menu appears. This menu is available from the Field View and Segment View tabs.

The following image is an example of an SQL data source with the root segment selected.



The following options are available:

Properties

Opens a pane showing the properties of the selected segment.

Insert

Enables you to insert one of the following:

Reference to Existing Synonym

Adds a reference pointer from the current synonym to an existing synonym. The selection list will show candidate synonyms. If you subsequently make changes to the source synonym, reopening the current synonym will reflect those changes.

Reference to Existing Synonym with Snowflake

Adds a reference pointer from the current synonym to an existing synonym. The selection list will show candidate synonyms as well as any synonyms that they reference.

Copy of Existing Synonym

Adds a static copy of an existing synonym to the current synonym. If you subsequently make changes to the source synonym, the current synonym will not reflect those changes.

Segment via Metadata Import

Enables you to create and add a new synonym to the current synonym through the Create Synonym tool.

Segment Manually

Inserts a synonym that must be coded manually.

Field

Inserts a general column to the segment.

Define

Inserts a virtual or defined column to the segment.

Filter

Inserts a filter to the segment.

Compute

Inserts a calculated value to the file.

Group

Inserts a group column to the segment.

Variable

Inserts a variable name that can be used for a Synonym/Access File parameter.

Data Profiling

Provides the characteristics of the data for a segment.

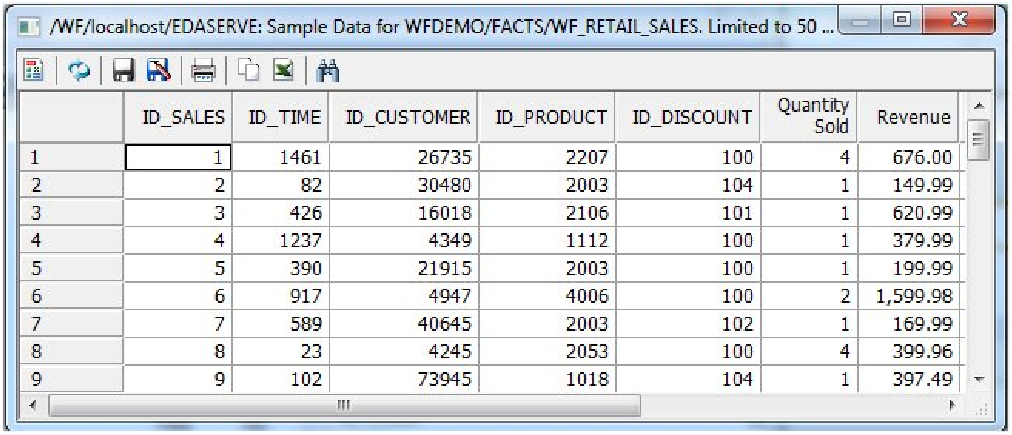
Data Profiling is available from the right-click context menu for all columns in the Master File hierarchy tree.

Sample Data

Displays sample data in the workspace for the selected segment or synonym.

Sample data is available throughout the Synonym Editor toolbar.

The following image is an example of the sample data that appears for a segment.



/WF/localhost/EDASERVE: Sample Data for WFDEMO/FACTS/WF_RETAIL_SALES. Limited to 50 ...

	ID_SALES	ID_TIME	ID_CUSTOMER	ID_PRODUCT	ID_DISCOUNT	Quantity Sold	Revenue
1	1	1461	26735	2207	100	4	676.00
2	2	82	30480	2003	104	1	149.99
3	3	426	16018	2106	101	1	620.99
4	4	1237	4349	1112	100	1	379.99
5	5	390	21915	2003	100	1	199.99
6	6	917	4947	4006	100	2	1,599.98
7	7	589	40645	2003	102	1	169.99
8	8	23	4245	2053	100	4	399.96
9	9	102	73945	1018	104	1	397.49

Note: Sample data is a great way to test the synonym for field data and to view the type of records returned. This can assist when performing Joins, testing connectivity to data sources, and so on.

Sample Data with parent key

Displays sample data for the selected segment joined to the parent segment. This option is only available when a child segment is selected in a multisegment synonym.

Delete

Deletes the segment.

Rename

Enables you to rename the segment.

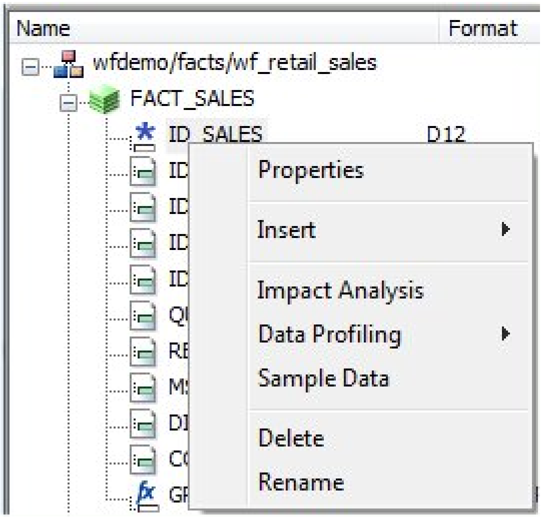
Join Properties

Opens the Join Editor so that you can specify how the selected segment is joined to the parent segment. This option is only available when a child segment is selected in a multisegment synonym.

Reference: Synonym Editor - Column/Field Context Menu

When you right-click a column in the Synonym Editor, the following context menu appears. This menu is available from the Field View, Segment View, and List View tabs.

The following image is an example of an SQL data source with a column selected.



Note: The options that are available will depend on the data source.

The following options are available:

Properties

Opens a pane showing the properties of the selected column.

Insert

Enables you to insert one of the following:

Field

Inserts a general column to the synonym.

Define

Inserts a virtual or defined column to the synonym.

Filter

Inserts a filter to the segment.

Compute

Inserts a calculated value to the file.

Group

Inserts a group column to the synonym.

Variable

Inserts a variable name that can be used for a Synonym/Access File parameter.

Impact Analysis

Displays an Impact Analysis report for the particular column in the workspace. An Impact Analysis report identifies the procedures that access a Master File or field within a Master File. For detailed information on Impact Analysis, see *Analyzing Metadata and Procedures* on page 189.

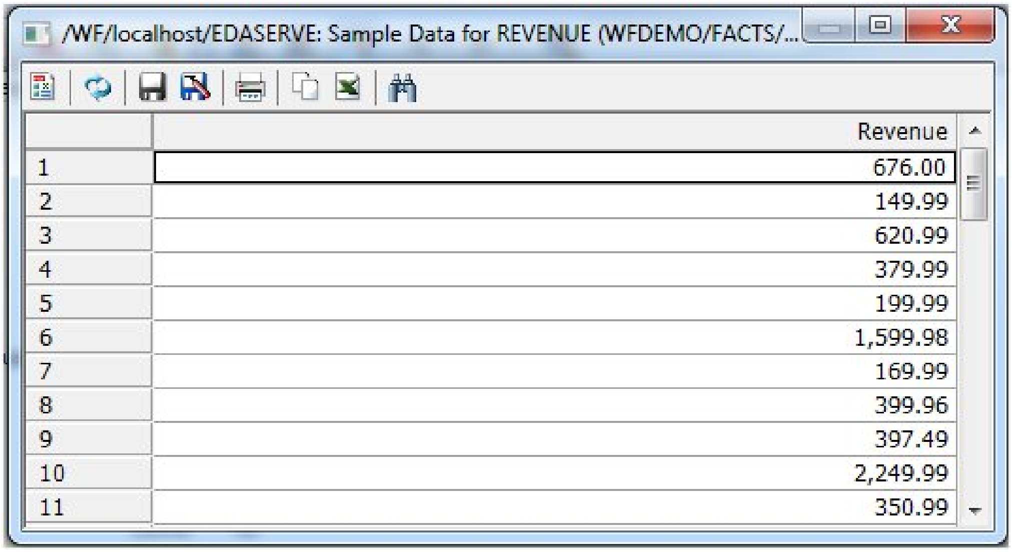
Data Profiling

Provides the characteristics of the data for a column. Data Profiling is available from the right-click context menu for all columns in the Master File hierarchy tree.

Sample Data

Displays sample data in the workspace.

The following image is an example of the sample data that appears for a column.


/WF/localhost/EDASERVE: Sample Data for REVENUE (WFDEMO/FACTS/...

	Revenue
1	676.00
2	149.99
3	620.99
4	379.99
5	199.99
6	1,599.98
7	169.99
8	399.96
9	397.49
10	2,249.99
11	350.99

Note: Sample data is a great way to test the synonym for field data and to view the type of records returned. This can assist when performing Joins, testing connectivity to data sources, and so on.

Decompose Date

Decomposes date fields into virtual columns representing Year, Quarter, Month, and Day fields.

Note: Decompose Date is only visible for date fields.

Delete

Deletes the column.

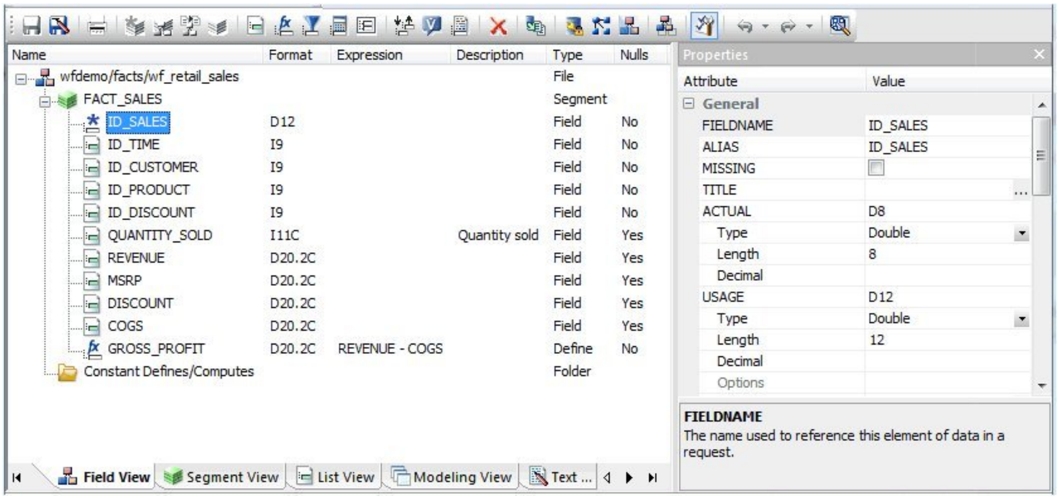
Rename

Allows you to rename the column.

Reference: Synonym Editor - Field View Tab

The Field View tab shows a table with the available fields and segments.

The following image is an example of a Microsoft SQL Server data source in the Field View tab.



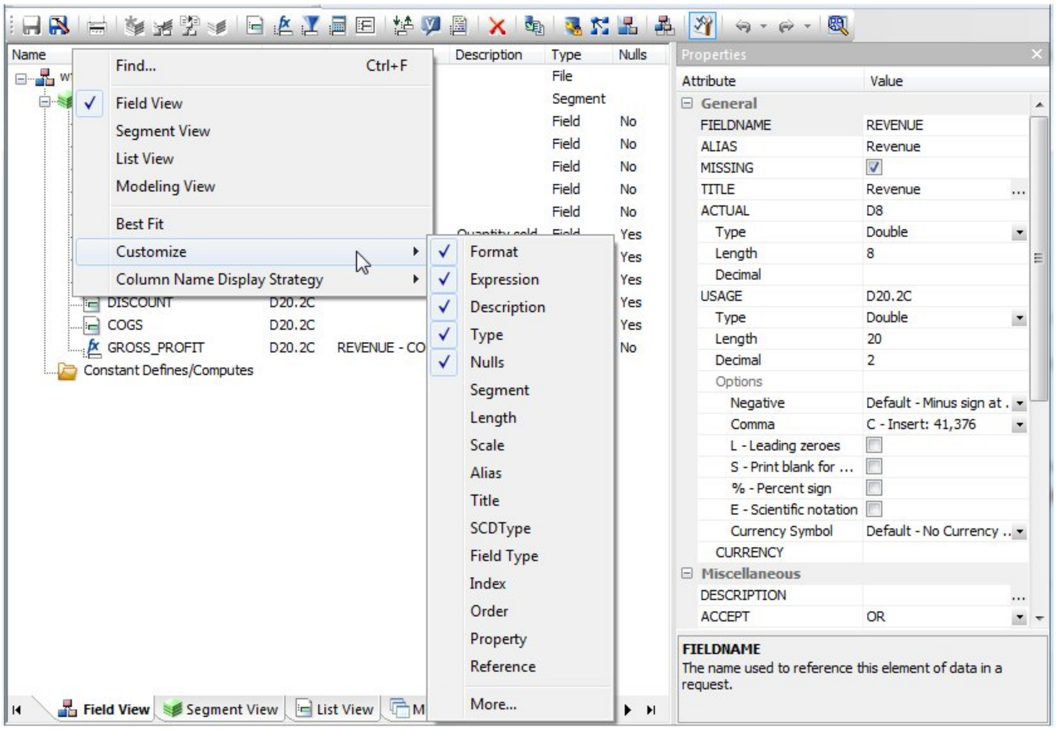
Name	Format	Expression	Description	Type	Nulls
wfdemo/facts/wf_retail_sales				File	
FACT_SALES				Segment	
ID_SALES	D12			Field	No
ID_TIME	I9			Field	No
ID_CUSTOMER	I9			Field	No
ID_PRODUCT	I9			Field	No
ID_DISCOUNT	I9			Field	No
QUANTITY_SOLD	I11C		Quantity sold	Field	Yes
REVENUE	D20.2C			Field	Yes
MSRP	D20.2C			Field	Yes
DISCOUNT	D20.2C			Field	Yes
COGS	D20.2C			Field	Yes
GROSS_PROFIT	D20.2C	REVENUE - COGS		Define	No
Constant Defines/Computes				Folder	

Properties

Attribute	Value
General	
FIELDNAME	ID_SALES
ALIAS	ID_SALES
MISSING	☐
TITLE	...
ACTUAL	D8
Type	Double
Length	8
Decimal	
USAGE	D12
Type	Double
Length	12
Decimal	
Options	

FIELDNAME
The name used to reference this element of data in a request.

The row below the toolbar menu, which includes the Name, Format, Expression, Description, and Nulls, enables users to right-click and drag to move a column to the right or left, changing the column order in the display grid. You cannot move the Name column. Users can also customize visible columns by right-clicking the row and selecting *Customize*, as shown in the following image.



Name	Format	Expression	Type	Nulls
			File	
			Segment	
			Field	No
			Field	No
			Field	No
			Field	No
			Field	No
			Field	Yes
				Yes
				Yes
DISCOUNT	D20.2C			Yes
COGS	D20.2C			Yes
GROSS_PROFIT	D20.2C	REVENUE - CO		No
Constant Defines/Computes				

Attribute	Value
General	
FIELDNAME	REVENUE
ALIAS	Revenue
MISSING	✓
TITLE	Revenue
ACTUAL	D8
Type	Double
Length	8
Decimal	
USAGE	D20.2C
Type	Double
Length	20
Decimal	2
Options	
Negative	Default - Minus sign at .
Comma	C - Insert: 41,376
L - Leading zeroes	☐
S - Print blank for ...	☐
% - Percent sign	☐
E - Scientific notation	☐
Currency Symbol	Default - No Currency ..
CURRENCY	
Miscellaneous	
DESCRIPTION	
ACCEPT	OR

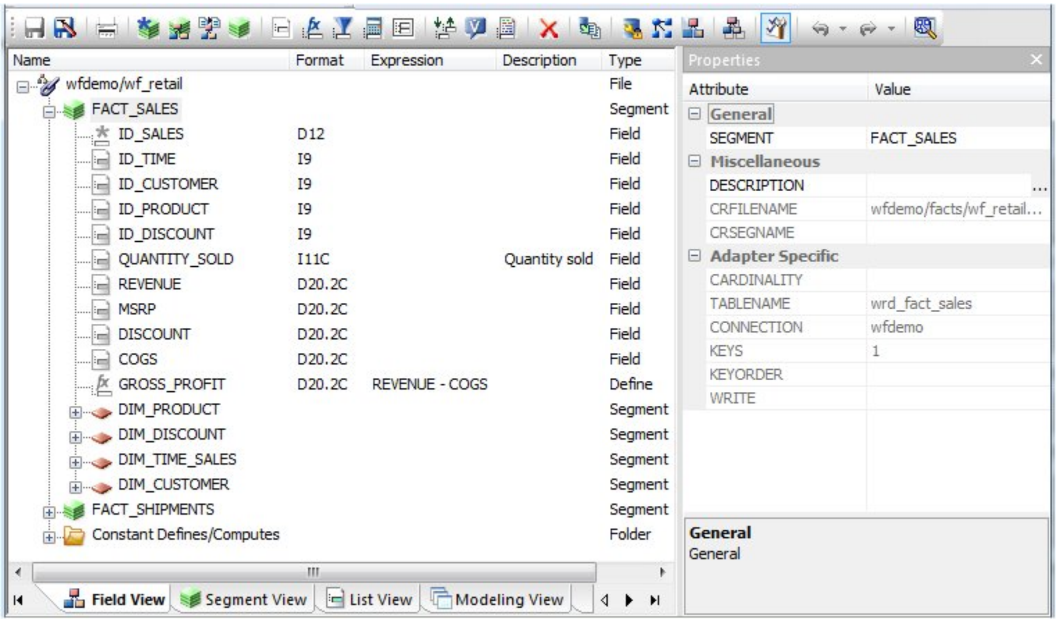
FIELDNAME
The name used to reference this element of data in a request.

The same menu displays when you right-click in an empty space. From this menu, a user is also given an option that enables them to switch views. More, the last option, opens the Column Management section of the Options dialog box. For more information, see *Synonym Editor Column Management Settings* on page 83.

Reference: Synonym Editor - Segment View Tab

The Segment View tab shows the segments that the synonym contains.

The following image is an example of a Microsoft SQL Server data source in the Segment View tab.



Name	Format	Expression	Description	Type
wfdemo/wf_retail				File
FACT_SALES				Segment
ID_SALES	D12			Field
ID_TIME	I9			Field
ID_CUSTOMER	I9			Field
ID_PRODUCT	I9			Field
ID_DISCOUNT	I9			Field
QUANTITY_SOLD	I11C		Quantity sold	Field
REVENUE	D20.2C			Field
MSRP	D20.2C			Field
DISCOUNT	D20.2C			Field
COGS	D20.2C			Field
GROSS_PROFIT	D20.2C	REVENUE - COGS		Define
DIM_PRODUCT				Segment
DIM_DISCOUNT				Segment
DIM_TIME_SALES				Segment
DIM_CUSTOMER				Segment
FACT_SHIPMENTS				Segment
Constant Defines/Computes				Folder

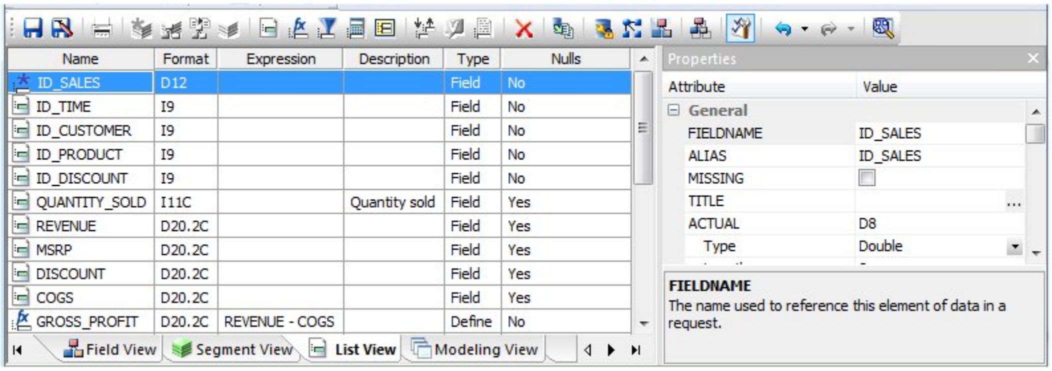
Properties

Attribute	Value
General	
SEGMENT	FACT_SALES
Miscellaneous	
DESCRIPTION	
CRFILENAME	wfdemo/facts/wf_retail...
CRSEGNAME	
Adapter Specific	
CARDINALITY	
TABLENAME	wrd_fact_sales
CONNECTION	wfdemo
KEYS	1
KEYORDER	
WRITE	

General
General

Reference: Synonym Editor - List View Tab

The List View tab shows a list of objects on the left, with the attributes and values of the selected item on the right, as shown in the following image.



Name	Format	Expression	Description	Type	Nulls
ID_SALES	D12			Field	No
ID_TIME	I9			Field	No
ID_CUSTOMER	I9			Field	No
ID_PRODUCT	I9			Field	No
ID_DISCOUNT	I9			Field	No
QUANTITY_SOLD	I11C		Quantity sold	Field	Yes
REVENUE	D20.2C			Field	Yes
MSRP	D20.2C			Field	Yes
DISCOUNT	D20.2C			Field	Yes
COGS	D20.2C			Field	Yes
GROSS_PROFIT	D20.2C	REVENUE - COGS		Define	No

Properties

Attribute	Value
General	
FIELDNAME	ID_SALES
ALIAS	ID_SALES
MISSING	☐
TITLE	
ACTUAL	D8
Type	Double

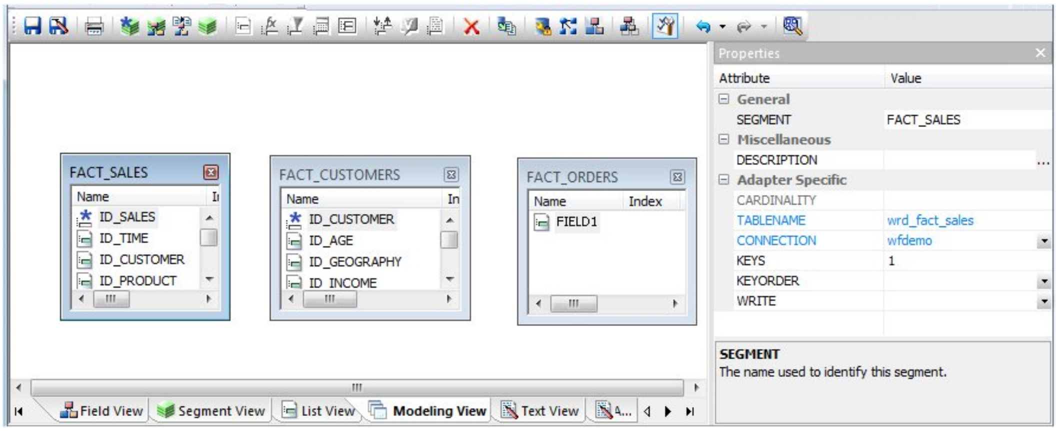
FIELDNAME
The name used to reference this element of data in a request.

Note: When you right-click a column heading in the List View, a context menu provides options to sort the display based on ascending or descending column values. The display can be sorted by any column.

Reference: Synonym Editor - Modeling View Tab

The Modeling View tab shows a graphical representation of the synonym. Use the Modeling View to, create cluster joins, view join properties, and add or edit segments.

The following image is an example of an SQL data source in the Modeling View.



Attribute	Value
General	
SEGMENT	FACT_SALES
Miscellaneous	
DESCRIPTION	
Adapter Specific	
CARDINALITY	
TABLENAME	wrd_fact_sales
CONNECTION	wfdemo
KEYS	1
KEYORDER	
WRITE	

SEGMENT
The name used to identify this segment.

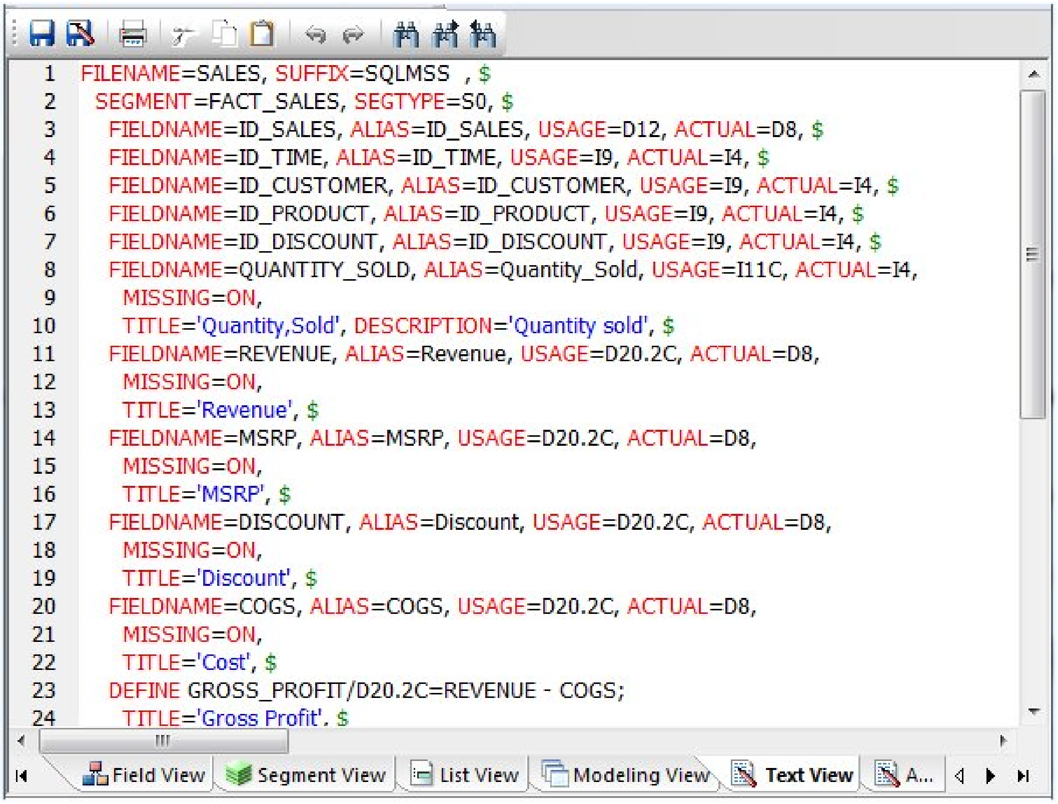
In the Modeling View, when you right-click in an empty space, you are presented with tasks permitted in this view, for example, Insert a Segment and Auto Arrange available synonyms. For more information about the Modeling View, see *Enhancing Synonyms Using the Modeling View* on page 112.

Reference: Synonym Editor - Text View Tab

This view opens the synonym in the editor.

You can print the Master File code by selecting *Print* from the File menu, perform search operations, and even make changes.

The following image is an example of a Microsoft SQL Server data source in the Text View tab.



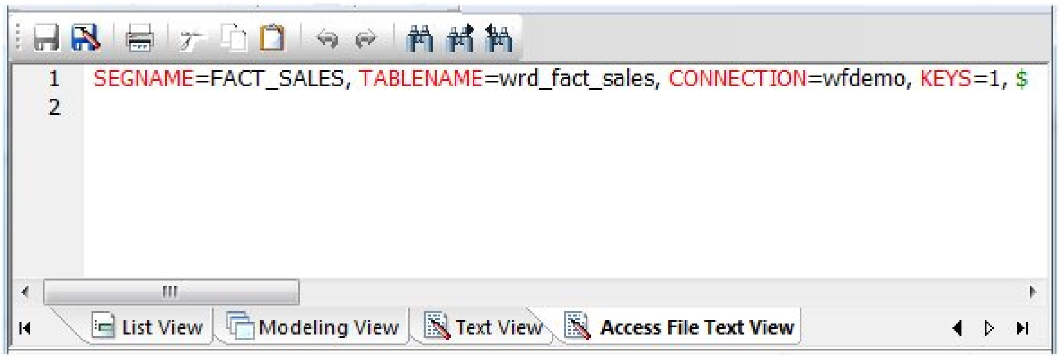
```
FILENAME=SALES, SUFFIX=SQLMSS  , $
 SEGMENT=FACT_SALES, SEGTYPE=S0, $
  FIELDNAME=ID_SALES, ALIAS=ID_SALES, USAGE=D12, ACTUAL=D8, $
  FIELDNAME=ID_TIME, ALIAS=ID_TIME, USAGE=I9, ACTUAL=I4, $
  FIELDNAME=ID_CUSTOMER, ALIAS=ID_CUSTOMER, USAGE=I9, ACTUAL=I4, $
  FIELDNAME=ID_PRODUCT, ALIAS=ID_PRODUCT, USAGE=I9, ACTUAL=I4, $
  FIELDNAME=ID_DISCOUNT, ALIAS=ID_DISCOUNT, USAGE=I9, ACTUAL=I4, $
  FIELDNAME=QUANTITY_SOLD, ALIAS=Quantity_Sold, USAGE=I11C, ACTUAL=I4,
   MISSING=ON,
   TITLE='Quantity,Sold', DESCRIPTION='Quantity sold', $
  FIELDNAME=REVENUE, ALIAS=Revenue, USAGE=D20.2C, ACTUAL=D8,
   MISSING=ON,
   TITLE='Revenue', $
  FIELDNAME=MSRP, ALIAS=MSRP, USAGE=D20.2C, ACTUAL=D8,
   MISSING=ON,
   TITLE='MSRP', $
  FIELDNAME=DISCOUNT, ALIAS=Discount, USAGE=D20.2C, ACTUAL=D8,
   MISSING=ON,
   TITLE='Discount', $
  FIELDNAME=COGS, ALIAS=COGS, USAGE=D20.2C, ACTUAL=D8,
   MISSING=ON,
   TITLE='Cost', $
  DEFINE GROSS_PROFIT/D20.2C=REVENUE - COGS;
   TITLE='Gross Profit', $
```

Reference: Synonym Editor - Access File Text View Tab

The Access File Text View tab shows the description of the Access File for a synonym, which is used to access the database.

Note: You can print the Master File code by selecting *Print* from the File menu, performing search operations, and making changes, if necessary.

The following image is an example of an SQL data source in the Access File Text View tab.



```
SEGNAME=FACT_SALES, TABLENAME=wrd_fact_sales, CONNECTION=wfdemo, KEYS=1, $
```

Viewing and Editing Synonym Attributes

How to:

View and Edit Synonym Attributes

Reference:

File Attributes Summary

Segment Attributes Summary

Column/Field Attribute Summary

The Synonym Editor enables you to view and edit the attributes of a synonym.

Procedure: How to View and Edit Synonym Attributes

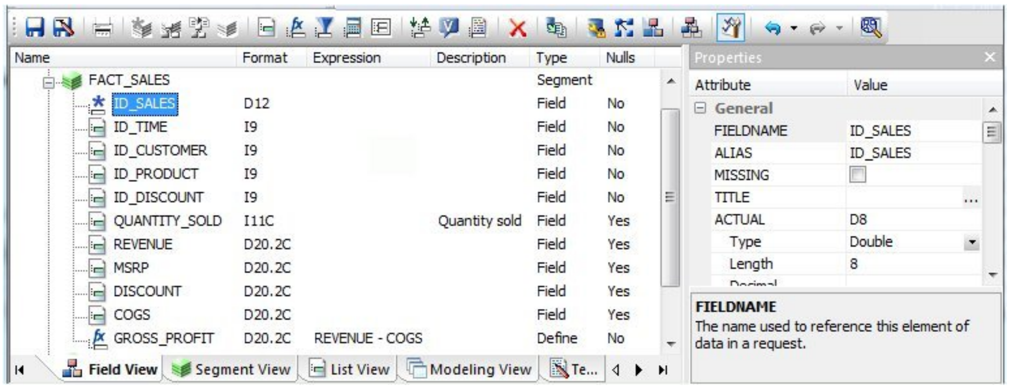
To view and edit synonym attributes:

1. From the Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   By default, the Synonym Editor opens to the last view used, the Field View tab shows a hierarchy of segments and columns on the left, with the attributes and values of the selected item on the right.

Note: The available attributes depend on the type of synonym.

The following image is an example of an SQL data source with a key column selected.



Name	Format	Expression	Description	Type	Nulls
FACT_SALES				Segment	
ID_SALES	D12			Field	No
ID_TIME	I9			Field	No
ID_CUSTOMER	I9			Field	No
ID_PRODUCT	I9			Field	No
ID_DISCOUNT	I9			Field	No
QUANTITY_SOLD	I11C		Quantity sold	Field	Yes
REVENUE	D20.2C			Field	Yes
MSRP	D20.2C			Field	Yes
DISCOUNT	D20.2C			Field	Yes
COGS	D20.2C			Field	Yes
GROSS_PROFIT	D20.2C	REVENUE - COGS		Define	No

Attribute	Value
General	
FIELDNAME	ID_SALES
ALIAS	ID_SALES
MISSING	☐
TITLE	
ACTUAL	D8
Type	Double
Length	8

FIELDNAME
The name used to reference this element of data in a request.

Note: The values for Format, Expression, Description, and Nulls are viewable in the hierarchy with the columns. To edit these values, use the corresponding attribute fields on the right-hand side of the Synonym Editor.

2. You can change the attribute values by typing in new values or by using the drop-down menus and check boxes.

   Note: The Synonym Editor does not let you make any changes that would render the Master File unusable. Therefore, you cannot edit any value field that is highlighted gray. In addition, if a change does not have proper syntax or format applied, the field may appear in red text. Messages and warnings appear if you try to save a file that contains an error.

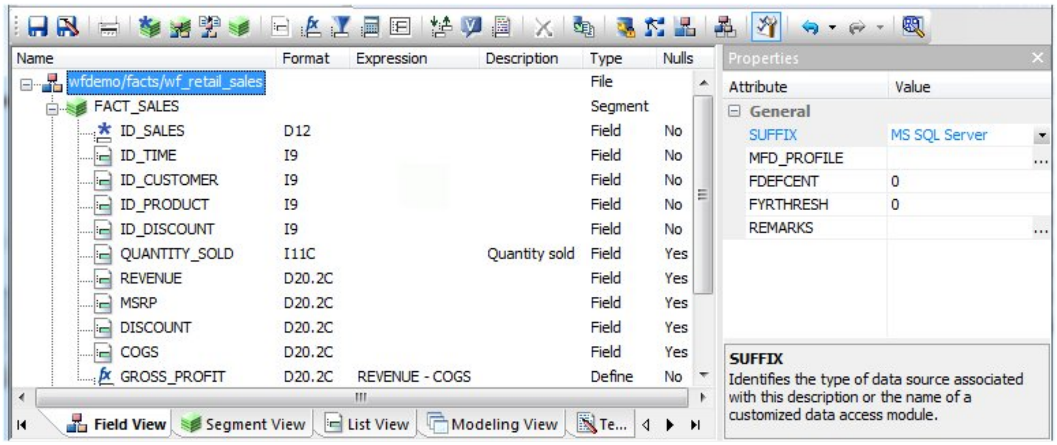
3. Save changes by clicking *Save* from the File menu.

4. To close the Synonym Editor, select *Close* from the File menu or click the control button in the upper-right corner.

Note: If you close the Synonym Editor without saving your changes, you are prompted to save.

Reference: File Attributes Summary

The following image is an example of an SQL data source with the synonym file name selected.



Name	Format	Expression	Description	Type	Nulls
wfdemo/facts/wf_retail_sales				File	
FACT_SALES				Segment	
ID_SALES	D12			Field	No
ID_TIME	I9			Field	No
ID_CUSTOMER	I9			Field	No
ID_PRODUCT	I9			Field	No
ID_DISCOUNT	I9			Field	No
QUANTITY_SOLD	I11C		Quantity sold	Field	Yes
REVENUE	D20.2C			Field	Yes
MSRP	D20.2C			Field	Yes
DISCOUNT	D20.2C			Field	Yes
COGS	D20.2C			Field	Yes
GROSS_PROFIT	D20.2C	REVENUE - COGS		Define	No

Attribute	Value
General	
SUFFIX	MS SQL Server
MFD_PROFILE	
FDEFCENT	0
FYRTHRESH	0
REMARKS	

SUFFIX
Identifies the type of data source associated with this description or the name of a customized data access module.

Note: Information about the attribute that has focus is displayed at the bottom of the attribute list. In this case, an explanation of the SUFFIX attribute appears.

Synonyms can have the following file attributes:

General

SUFFIX

Identifies the type of synonym or data source.

MFD_PROFILE

The name of the FOCEXEC that will be executed before a request containing MFD.

FDEFCENT

Defines the default century value, specifying a century number for handling cross-century dates.

Note: Use the default setting (0) unless you wish to retrieve data from an earlier century. For example, 19xx.

FYRTHRESH

Defines the base years, to represent the lowest year to which the century value applies (FDEFCENT).

Note: Use the default setting (0) unless you wish to retrieve data from an earlier century. For example, 19xx.

REMARKS

Enables you to include descriptive information at the file level and specify multiple language descriptions for the synonym. Remarks are displayed along with the file name during reporting.

For more information about multilingual descriptions, see *Setting Up Multilingual Titles and Descriptions* on page 109.

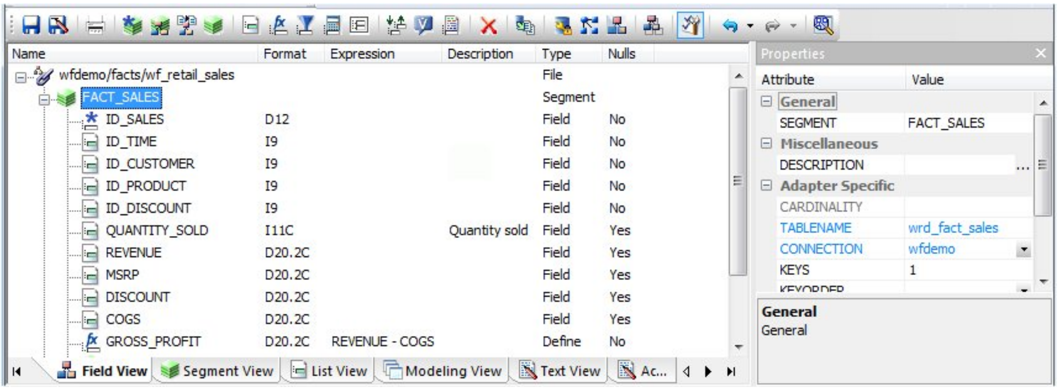
DATASET

Identifies the physical location of the data source to be used in the file name, including the extension and the location of the data file.

Note: The available attributes depend on the type of synonym.

Reference: Segment Attributes Summary

The following image is an example of an SQL data source with a segment selected.



Name	Format	Expression	Description	Type	Nulls
wfdemo/facts/wf_retail_sales				File	
FACT_SALES				Segment	
ID_SALES	D12			Field	No
ID_TIME	I9			Field	No
ID_CUSTOMER	I9			Field	No
ID_PRODUCT	I9			Field	No
ID_DISCOUNT	I9			Field	No
QUANTITY_SOLD	I11C		Quantity sold	Field	Yes
REVENUE	D20.2C			Field	Yes
MSRP	D20.2C			Field	Yes
DISCOUNT	D20.2C			Field	Yes
COGS	D20.2C			Field	Yes
GROSS_PROFIT	D20.2C	REVENUE - COGS		Define	No

Attribute	Value
General	
SEGMENT	FACT_SALES
Miscellaneous	
DESCRIPTION	
Adapter Specific	
CARDINALITY	
TABLENAME	wrd_fact_sales
CONNECTION	wfdemo
KEYS	1
KEYORDER	

General
General

Note: Information about the attribute in focus is displayed at the bottom of the attribute list. In this case, an explanation of the SEGMENT attribute appears.

Segments in a synonym can have the following attributes:

General

SEGMENT

Is the name of the segment.

SEGTYPE

Specifies the type or relationship that a segment has to its parent and indicates which of the segment fields are key fields, and in what order they are sorted.

Type. Identifies the segment type and sorting options from the Type drop-down list.

Keys. Records are sorted in a data source by key fields. Enter the number of key fields that you want to use for sorting. For example, no two employees can have the same employee ID number, so you can use that field as the key. A segment instance can have more than one field that makes up the key. That is, two or more field values may be used to distinguish records.

SEGSUF

SEGSUF is used when part of the data source being described by the Master File is of a different data source type than that declared for the entire structure.

Note: SEGSUF is the data source type of a segment and any descendants it might have, where that type differs from the SUFFIX value.

Miscellaneous

DESCRIPTION

Contains a description or comments about the segment.

For more information about multilingual descriptions, see *Setting Up Multilingual Titles and Descriptions* on page 109.

CRFILENAME

Is the name of the cross-referenced data source.

CRSEGNAME

Is the name of the cross-referenced segment.

CRKEY

Identifies the common join field for the cross-referenced segment.

Adapter Specific

Note: Adapter Specific fields are shown if an Access File component has been generated with the synonym.

CARDINALITY

Defines the number of records that were found in the original data source when the synonym was created.

TABLENAME

Identifies the table or view. It may contain the owner ID, as well as the table name. For some synonyms, it must also contain the data source name.

CONNECTION

Indicates the adapter connection name used.

KEYS

Identifies how many columns constitute the primary key.

KEYORDER

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

WRITE

Specifies whether write operations are allowed against the table.

DBSPACE

Identifies the storage area in which the table resides.

PERSISTENCE

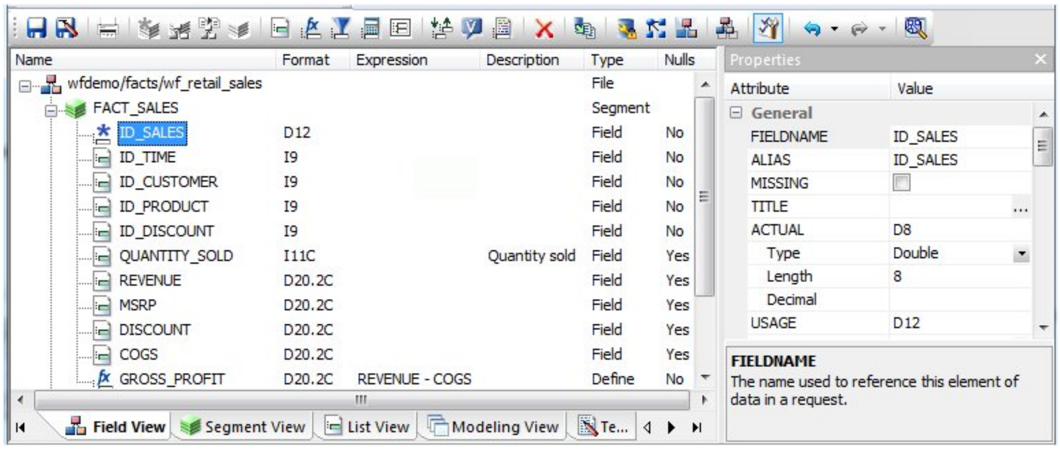
Specifies the type of table persistence and related table properties. This is optional for database management systems that support volatile tables, and required otherwise.

Note: The attributes available depend on the type of synonym.

Reference: Column/Field Attribute Summary

The following image is an example of an SQL data source with a key column selected.

Note: The attributes available depend on the type of synonym.



Name	Format	Expression	Description	Type	Nulls
wfdemo/facts/wf_retail_sales				File	
FACT_SALES				Segment	
ID_SALES	D12			Field	No
ID_TIME	I9			Field	No
ID_CUSTOMER	I9			Field	No
ID_PRODUCT	I9			Field	No
ID_DISCOUNT	I9			Field	No
QUANTITY_SOLD	I11C		Quantity sold	Field	Yes
REVENUE	D20.2C			Field	Yes
MSRP	D20.2C			Field	Yes
DISCOUNT	D20.2C			Field	Yes
COGS	D20.2C			Field	Yes
GROSS_PROFIT	D20.2C	REVENUE - COGS		Define	No

Properties

Attribute	Value
General	
FIELDNAME	ID_SALES
ALIAS	ID_SALES
MISSING	☐
TITLE	
ACTUAL	D8
Type	Double
Length	8
Decimal	
USAGE	D12

FIELDNAME
The name used to reference this element of data in a request.

Note: Information about the attribute in focus is displayed at the bottom of the attribute list. In this case, an explanation of the FIELDNAME attribute appears.

Columns in a synonym can have the following attributes:

General

FIELDNAME

Is the name of the column.

ALIAS

Assigns an alternative name for a column or the real column name for a DBMS synonym.

MISSING

Controls how null data is handled, that is, if no transaction value is supplied.

TITLE

Supplies a title to replace the column name that is normally used in reports and enables you to specify multiple language titles for the column or field.

For more information about multilingual titles, see *Setting Up Multilingual Titles and Descriptions* on page 109.

ACTUAL

Describes the type and length of data as it is actually stored in the data source.

USAGE

Describes the data type and format for the column for usage or display.

Note: Additional attributes, DEFCENT and YRTHRESH, are available if the Usage field is set to Date, Time, or DateTime (Timestamp) format. Use these attributes to enter the century and year threshold values for the column or field.

- **Type.** Enables you to set a value for the field as Alpha fixed, Alpha variable, Text, Integer, Float, Double, Decimal Packed, Date, Time, DateTime (Timestamp), and Binary Large Object.
- **Length.** Enables you to set the character length.
- **Options.** Enables you to set how negative integers appear, set the appearance or suppression of commas, set leading zeroes, print blank for zeroes, set percentage signs, and select currency symbols.

Miscellaneous

DESCRIPTION

Contains a description or comments about the column or field.

For more information about multilingual descriptions, see *Setting Up Multilingual Titles and Descriptions* on page 109.

ACCEPT

Specifies criteria for validating data.

OR. Enables you to specify an acceptable value.

FROM-TO. Enables you to specify a range of acceptable value fields.

FIND. Enables you to supply file and field names to instruct DB2 Web Query where to search for a data source, and for a list of acceptable values. You supply the field name of the data field for which the validation criteria are being assigned, the file name of the target DB2 Web Query data source where the field can be found, and the field name of the target data field that contains the validation criteria.

Note: FIND is only available for DB2 Web Query data sources and does not apply to OLAP-enabled Master Files.

WITHIN

Contains the name of a field to be included in a dimension.

These WITHIN statements are added to the synonym through the Dimension Builder to OLAP-enable relational tables. This enables you to perform OLAP analysis using the OLAP Control Panel.

Property

Sets the Property for the field.

Reference

Enables you to reference another data source.

FIELDTYPE

Identifies an indexed column. You can check the Index check box to index the FIELDTYPE.

Note: FIELDTYPE=R indicates a read-only column. This setting is useful for columns that are automatically assigned a value by the RDBMS.

ACCESS_PROPERTY

Specifies access options for the column data.

- **INTERNAL.** Defines a column that does not appear in sample data or in the list of available columns. Restricts the field from showing in any of the Field Lists in the reporting tools.
- **NEED_VALUE.** Defines a column that requires a value to access the data.

❏ **Select By.** Defines a column by value, range, or multivalues.

AUTHRESP

Defines a column that describes the result of an authentication operation. Correct response values must be provided in the ACCEPT attribute (using the OR predicate if more than one value is acceptable).

AUTHTOKEN

Defines a column that contains a response token to be passed as an input value to the operation to be executed.

HELPMESSAGE

Appends a help message to a column.

SCD Type

Sets slowly changing dimensions. This option is only available for existing relational targets and is only used by a flow when SCD is enabled in the target object.

A **surrogate key** is the KEY column in the table and has an SCD type of **blank**. Other columns with a blank SCD type have no SCD processing done to them. In a synonym, this column will always appear first, even if it is not the first column in the table.

Note: The surrogate key must be an integer. If the column is identified in the synonym as Readonly and Autoincrement, then the database Autoincrement or Identity processing is used to assign surrogate key values. If it is not identified, then DataMigrator processing assigns values for this column, incrementing for each new row added.

Logical Key Field. Is the source database key.

Activation Flag. Indicates if the row is current.

Begin Date/End Date. Indicates date range for the row values. A null end date indicates the row is current. If you use Begin and End Date, the End Date column must be created as nullable.

Type I. (overwriting history) designates columns whose database values are overwritten with new values.

Type II. (preserving history) designates columns whose database rows are flagged as inactive or assigned an end date. New rows are inserted with the new values.

blank. (non-key permanent columns) indicates that database values are not changed.

Setting Up Multilingual Titles and Descriptions

How to:

Set Up Multilingual Titles and Descriptions

You can open a synonym in the Synonym Editor and provide text for the title, caption, and description in multiple languages. These descriptions appear in the specified language in reports generated against the synonym.

The Multilingual Titles dialog box is available from the Remarks, Title, and Description attribute value fields in the Synonym Editor.

Note: The attributes available depend on the type of synonym.

Procedure: How to Set Up Multilingual Titles and Descriptions

1. From the Data Servers area, double-click the Master File or select *Edit in Synonym Editor* from the File menu.

   The Master File opens to the Field View tab in the Synonym Editor.

2. To add multilingual text:

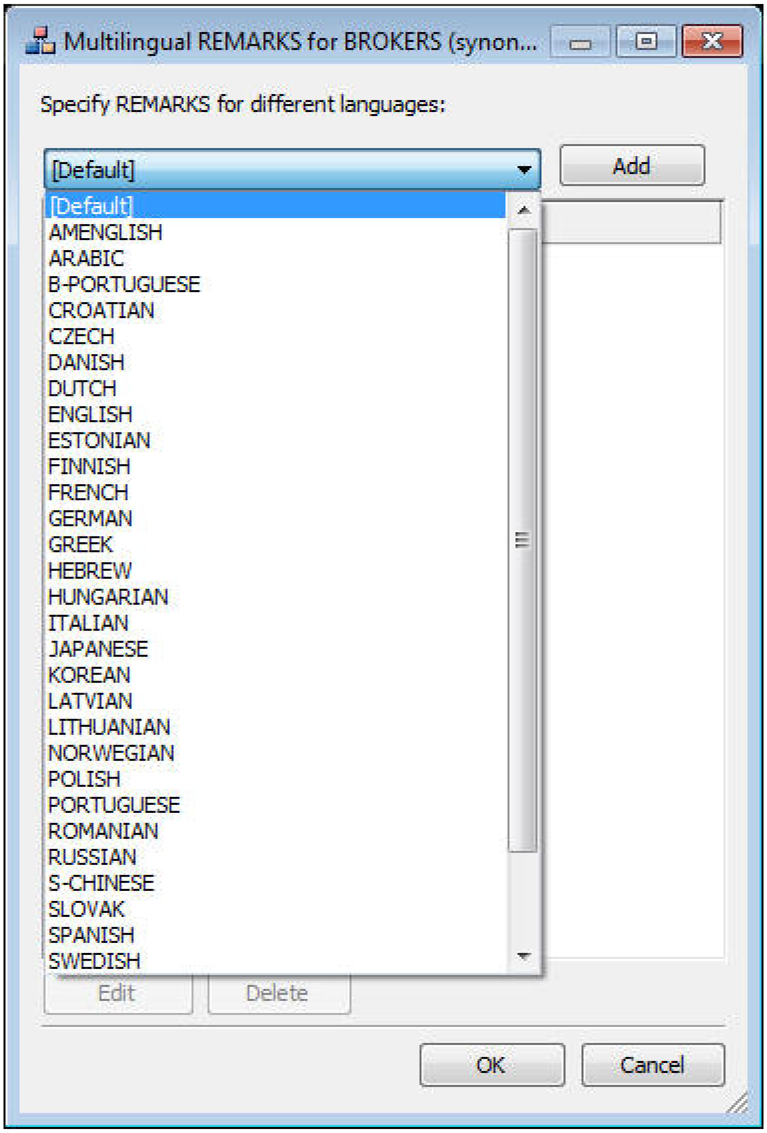
   - For *Title*, click a column from the Master File hierarchy of columns on the left.
   - For *Remarks*, click the root level of the Master File (application/filename) on the left.
   - For *Descriptions*, click a column, segment, or custom field from the Master File hierarchy of fields on the left.

   The corresponding attributes and values appear on the right.

3. Click the browse (...) button at the end of the value field for either Remarks, Title, or Description.

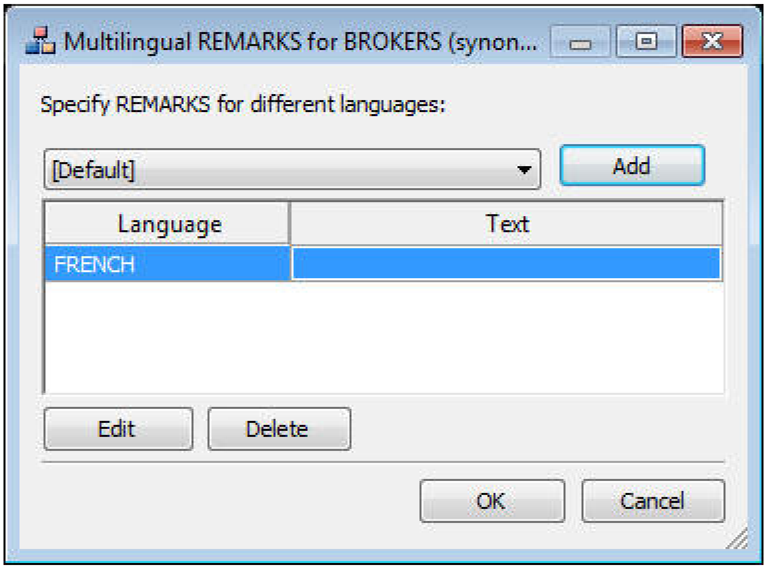
   The Multilingual dialog box opens.

4. From the Specify Titles/Descriptions for different languages drop-down list, choose the language in which you want the titles or remarks (descriptions) to be displayed.



5. Click *Add*.

The selected language is added below the default language (which is determined by your code page selection).


Multilingual REMARKS for BROKERS (synon...
Specify REMARKS for different languages:
[Default]
Add
Edit
Delete
OK
Cancel

Language	Text
FRENCH	

6. Type a description or title in the Text field.

7. You may add, edit, or delete additional titles or descriptions.

   To add an additional language:

   a. Select another language from the drop-down list.
   b. Click *Add*.
   c. Type a title or description for the field.

   To edit an existing specified title or description:

   a. Select the title or description and click *Edit*.

      You may also double-click the title.

   b. Manually type a title or description name.

   To delete a specified title:

   a. Select the title or description and language to be deleted.
   b. Click *Delete*.

8. Click *OK* to close the Multilingual Titles dialog box.
9. Click *Save* from the File menu to save the synonym.
10. To close the synonym, select *Close* from the File menu or click the control button in the upper-right corner.

Enhancing Synonyms Using the Modeling View

How to:

Enhance Synonyms Using the Modeling View

Edit Synonyms Using the Modeling View

Reference:

Modeling View Context Menu

Join Editor Dialog Box

The Synonym Editor Modeling View tab provides a visual presentation for the synonym for which positioning is preserved and stored in the Access File. Use the Modeling View to define dimensions for OLAP analysis, view join properties, create cluster joins, and add or edit segments. For more information about using the Dimension Builder in the Modeling View, see *Defining Dimensions for OLAP Analysis* on page 139.

Note: The Modeling View is not available for Cube data sources.

Procedure: How to Enhance Synonyms Using the Modeling View

Use the Modeling View to enhance a synonym by adding a segment.

1. From the Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

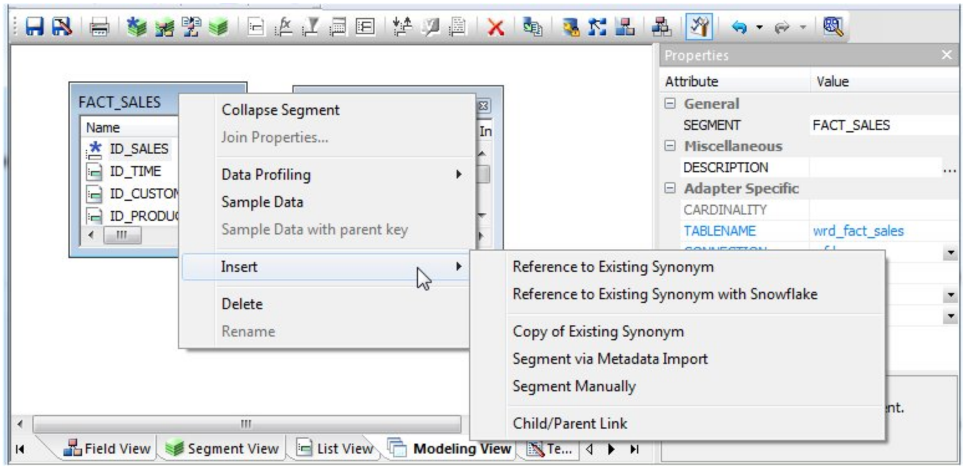
   The Synonym Editor opens.

2. Click the *Modeling View* tab.

   The Synonym Editor Modeling View tab opens in the workspace.

3. Right-click a segment in the workspace.

   The following context menu appears.



4. Insert segments through one of the methods listed:

   ❑ Insert Reference to Existing Synonym adds an existing synonym as a segment to the current synonym.

   ❑ Insert Reference to Existing Synonym with Snowflake adds an existing synonym as a segment to the current synonym. The selection list will show candidate synonyms as well as any synonyms that they reference.

   ❑ Insert Copy of Existing Synonym enables you to add a table as a segment from an existing synonym.

   ❑ Insert Segment via Metadata Import enables you to add segments by using the Create Synonym tool. This tool creates a synonym and includes it as a segment in the synonym from which the tool was launched.

❑ Insert Segment Manually enables you to add segments manually to the current synonym.

Note: You would use this approach only if you are coding a new Master File, as you would for a DB2 Web Query data source.

Procedure: How to Edit Synonyms Using the Modeling View

To edit synonyms using Modeling View:

1. From the Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

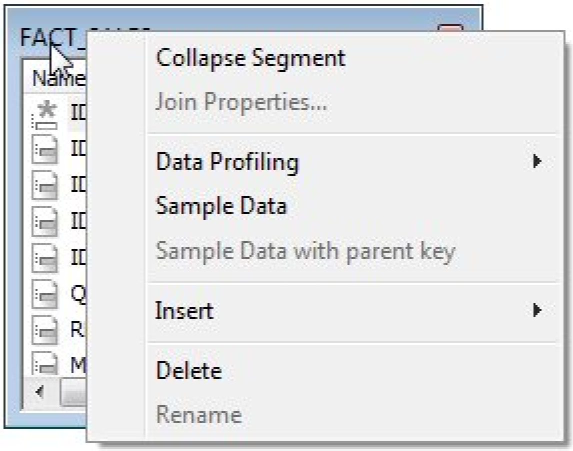
   The Synonym Editor opens.

2. Click the *Modeling View* tab.

3. Right-click a segment icon.

   A context menu appears, providing options for adding segments.

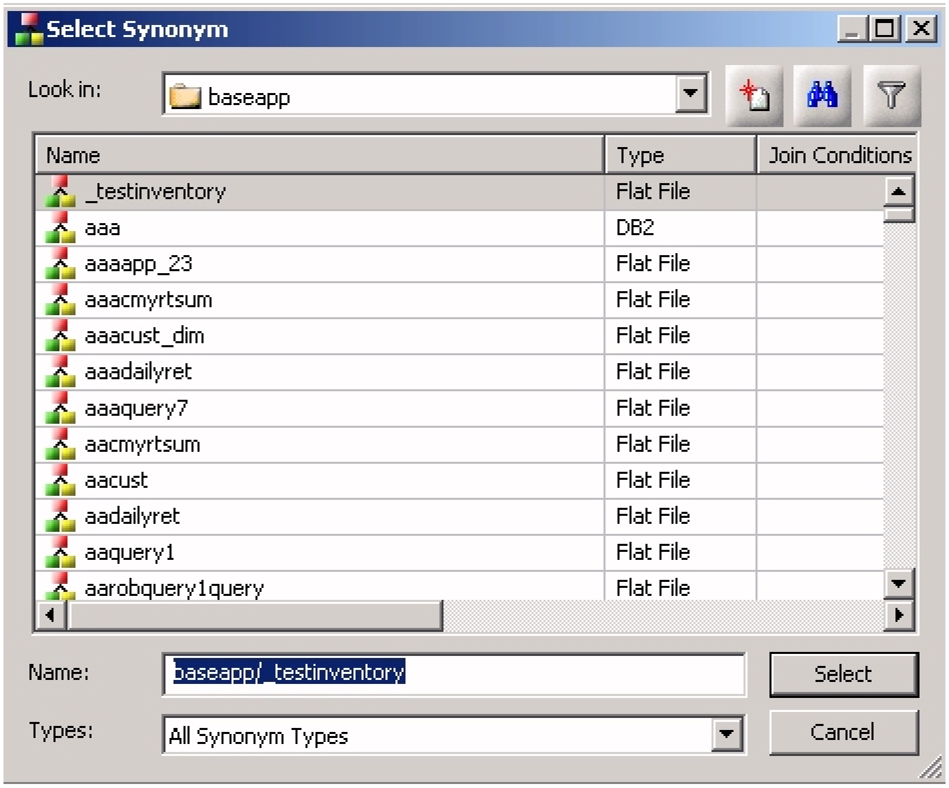
   Note: Options for adding segments are also available from the icons above the workspace.



To insert a segment from an existing synonym:

a. Select *Insert*, then *Reference to Existing Synonym* from the context menu.

The Select Synonym dialog box opens.



Name	Type	Join Conditions
_testinventory	Flat File	
aaa	DB2	
aaaapp_23	Flat File	
aaacmyrtsum	Flat File	
aaacust_dim	Flat File	
aaadailyret	Flat File	
aaaquery7	Flat File	
aacmyrtsum	Flat File	
aacust	Flat File	
aadailyret	Flat File	
aaquery1	Flat File	
aarobquery1query	Flat File	

Name: baseapp/_testinventory

Types: All Synonym Types

Columns can be customized and sorted. Synonyms displayed in this dialog box are based on the Reporting Servers APP Path configuration.

b. Click a synonym and click *Select*.

The segment is added to the synonym.

To insert a segment from an existing synonym with Snowflake:

a. Select *Insert*, then *Reference to Existing Synonym with Snowflake* from the context menu.

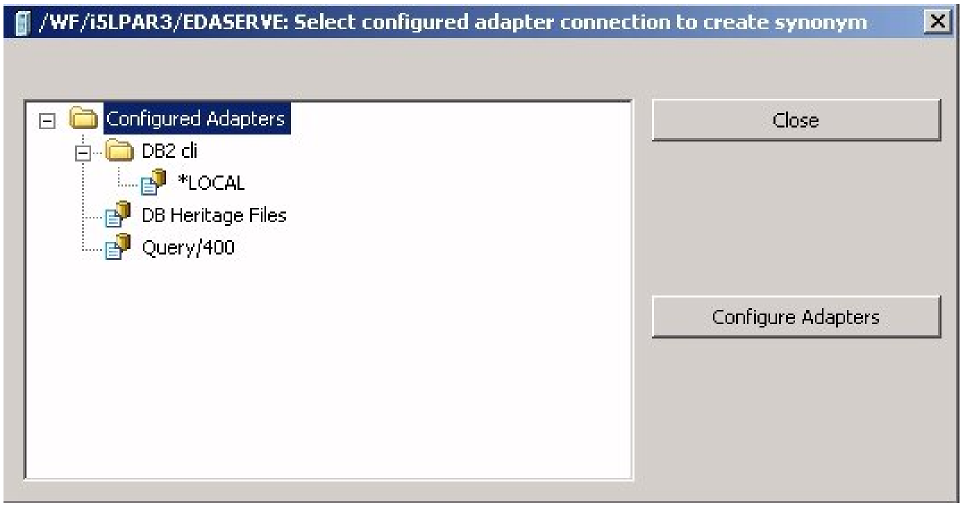
The procedure is the same as Reference to Existing Synonym, except the selection list will show both candidate synonyms and any synonyms that they reference.

To insert a segment via Metadata Import:

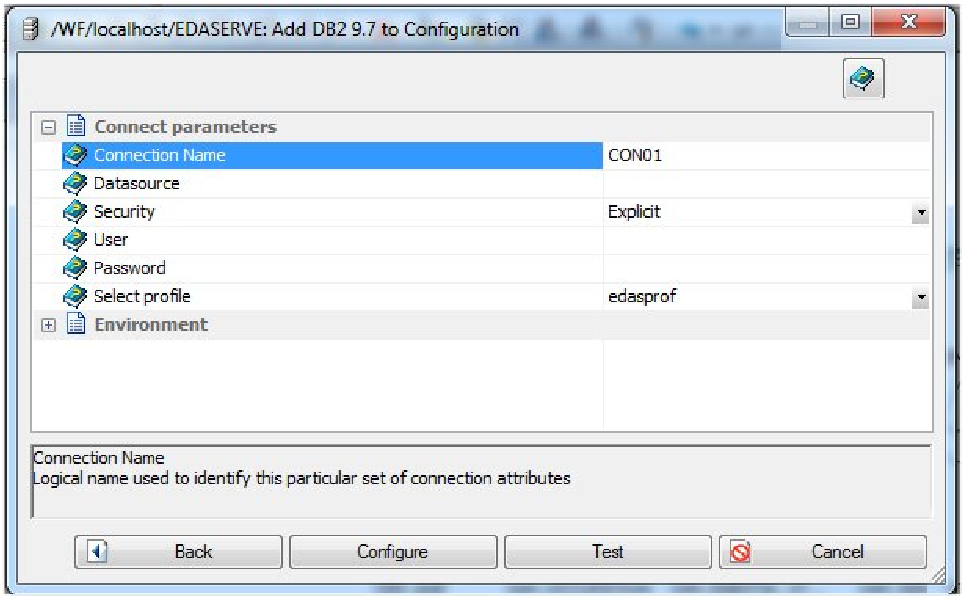
a. Select *Insert*, then *Segment via Metadata Import* from the context menu.

This method enables you to launch the Create Synonym tool, create a synonym, and incorporate it as a new segment in the synonym from which you initiate the import.

b. When this option is selected, you are first presented with the Select Adapter dialog box where you can select a configured adapter connection to continue or configure a new adapter, if necessary, as the following example demonstrates.



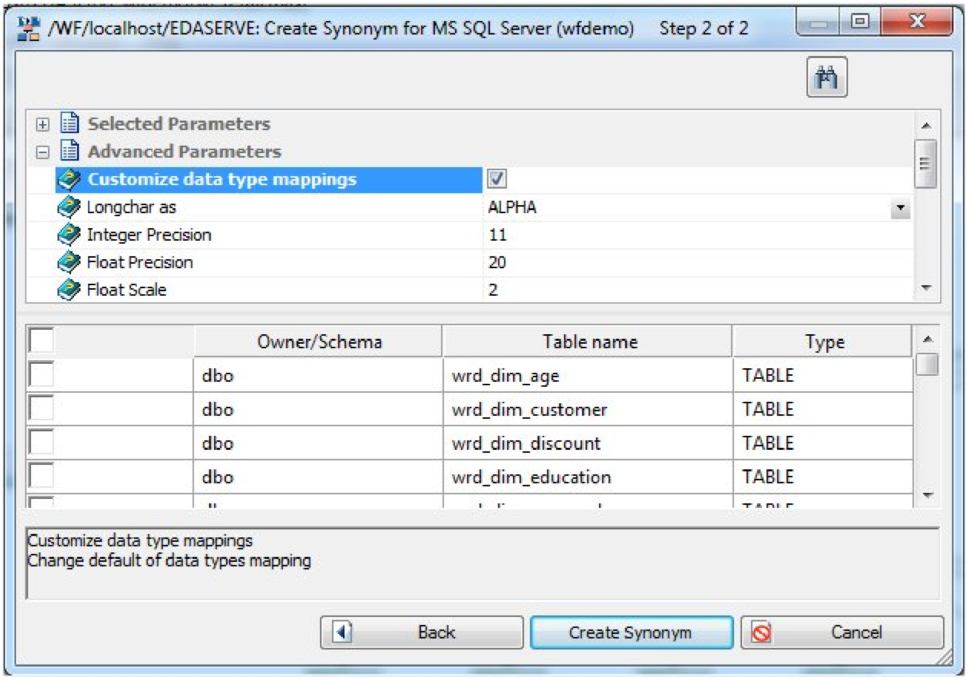
The following image is an example of a screen where you provide information for the connection parameters.


/WF/localhost/EDASERVE: Add DB2 9.7 to Configuration

Connect parameters	
Connection Name	CON01
Datasource	
Security	Explicit
User	
Password	
Select profile	edasprof
Environment	

Connection Name
Logical name used to identify this particular set of connection attributes

The following image is an example of a screen that enables you to select tables to create synonyms.


/WF/localhost/EDASERVE: Create Synonym for MS SQL Server (wfdemo) Step 2 of 2

Selected Parameters
Advanced Parameters

Customize data type mappings	✓
Longchar as	ALPHA
Integer Precision	11
Float Precision	20
Float Scale	2

	Owner/Schema	Table name	Type
	dbo	wrd_dim_age	TABLE
	dbo	wrd_dim_customer	TABLE
	dbo	wrd_dim_discount	TABLE
	dbo	wrd_dim_education	TABLE

Customize data type mappings
Change default of data types mapping

The segment is added to the synonym.

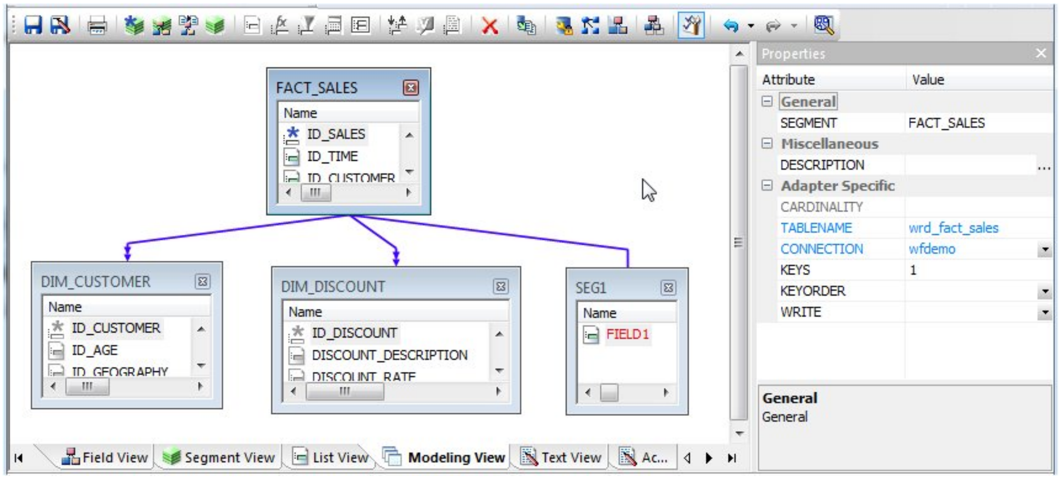
To insert a segment manually:

a. Select *Insert*, then *Segment Manually*.

The segment is added to the Modeling View and a default field is created, using a default segment name.

b. Attributes for the new segment or segments can be modified in the Modeling View tab through the Properties window or from the other views.

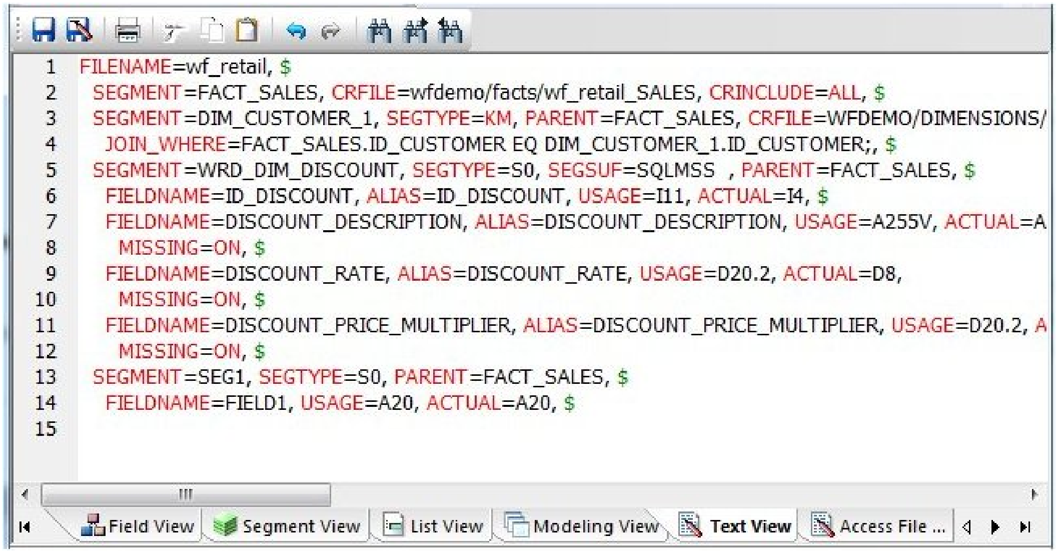
The following image shows a synonym that has had several segments added in the Modeling View tab.



Attribute	Value
General	
SEGMENT	FACT_SALES
Miscellaneous	
DESCRIPTION	
Adapter Specific	
CARDINALITY	
TABLENAME	wrd_fact_sales
CONNECTION	wfdemo
KEYS	1
KEYORDER	
WRITE	

General
General

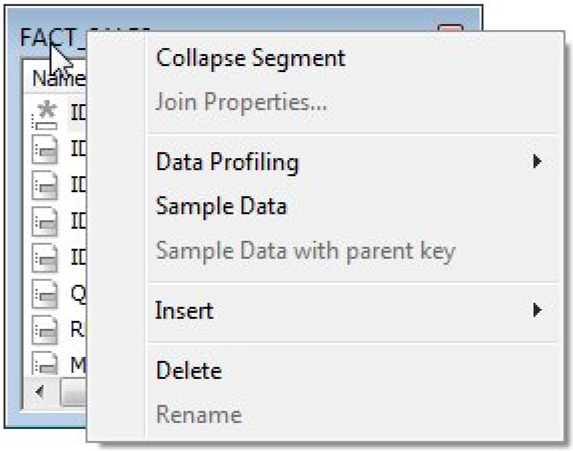
You can select the Synonym Text View or Access File Text View tab to see the resulting segment attributes.



```
FILENAME=wf_retail, $
  SEGMENT=FACT_SALES, CRFILE=wfdemo/facts/wf_retail_SALES, CRINCLUDE=ALL, $
  SEGMENT=DIM_CUSTOMER_1, SEGTYPE=KM, PARENT=FACT_SALES, CRFILE=WFDEMO/DIMENSIONS/
    JOIN_WHERE=FACT_SALES.ID_CUSTOMER EQ DIM_CUSTOMER_1.ID_CUSTOMER;, $
  SEGMENT=WRD_DIM_DISCOUNT, SEGTYPE=S0, SEGSUF=SQLMSS  , PARENT=FACT_SALES, $
    FIELDNAME=ID_DISCOUNT, ALIAS=ID_DISCOUNT, USAGE=I11, ACTUAL=I4, $
    FIELDNAME=DISCOUNT_DESCRIPTION, ALIAS=DISCOUNT_DESCRIPTION, USAGE=A255V, ACTUAL=A
      MISSING=ON, $
    FIELDNAME=DISCOUNT_RATE, ALIAS=DISCOUNT_RATE, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON, $
    FIELDNAME=DISCOUNT_PRICE_MULTIPLIER, ALIAS=DISCOUNT_PRICE_MULTIPLIER, USAGE=D20.2, A
      MISSING=ON, $
  SEGMENT=SEG1, SEGTYPE=S0, PARENT=FACT_SALES, $
    FIELDNAME=FIELD1, USAGE=A20, ACTUAL=A20, $
```

Reference: Modeling View Context Menu

When you right-click a synonym or segment in the Synonym Editor Modeling View tab, the following context menu appears.



The context menu has the following options:

Collapse Segment

Changes the view from a file icon to a table view, which enables you to see columns, sample data, and sample data parent keys.

Tip: You may also double-click a file icon to open the table view. Double-click the table to close the table view, or click the *X* button from the toggle toolbar to close.

Join Properties

Provides access to the Join Properties window.

Data Profiling

Provides the characteristics of the data for a segment.

Sample Data

Displays sample data in the workspace.

Sample Data with parent key

Displays sample data with parent key in the workspace.

Insert

- **Reference to Existing Synonym.** Enables you to reference an existing synonym as a segment to the current synonym.

- **Reference to Existing Synonym with Snowflake.** Enables you to reference an existing synonym as a segment to the current synonym. The selection list will show candidate synonyms as well as any synonyms that they reference.
- **Copy of Existing Synonym.** Copies an existing synonym to the current synonym.
- **Segment via Metadata Import.** Enables you to create a new synonym through the Create Synonym tool and add it to the current synonym.
- **Segment Manually.** Inserts a segment to the current synonym, that must by coded manually.

Delete

Deletes the segment.

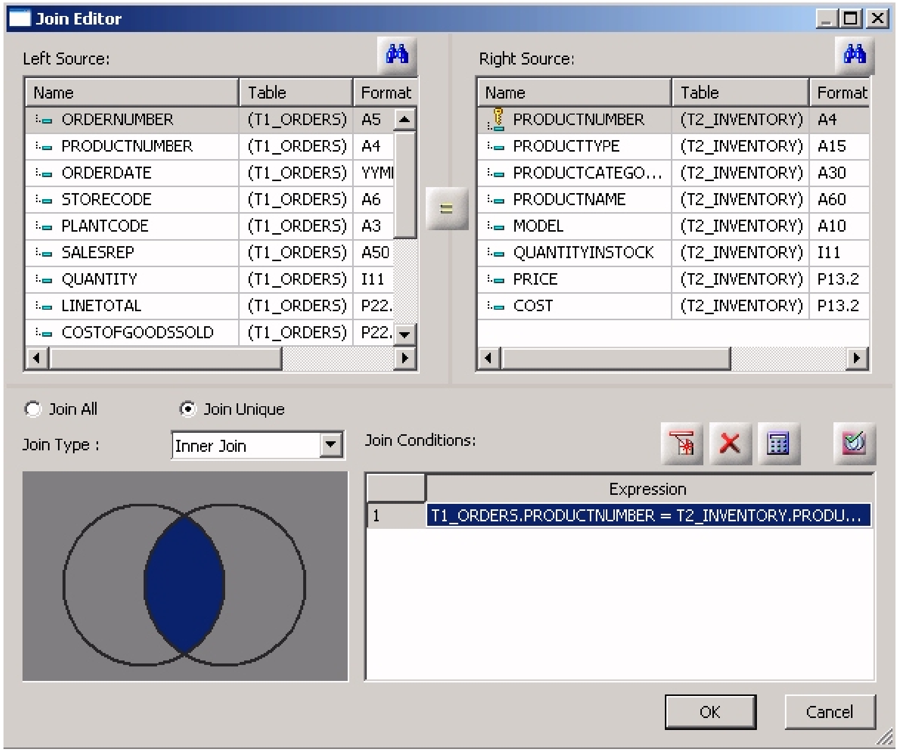
Rename

Enables you to rename the segment.

Reference: Join Editor Dialog Box

When you select Join Properties from a segment in the Synonym Editor Modeling View tab, the Join Editor dialog box appears.

The Join Editor dialog box contains Left and Right Source columns, Join Type, and Join Condition options, as well as One-to-Many and One-to-One choices. Use the left and right source columns to create join maps and view sample data.



Left Source:

Name	Table	Format
ORDERNUMBER	(T1_ORDERS)	A5
PRODUCTNUMBER	(T1_ORDERS)	A4
ORDERDATE	(T1_ORDERS)	YYMI
STORECODE	(T1_ORDERS)	A6
PLANTCODE	(T1_ORDERS)	A3
SALESREP	(T1_ORDERS)	A50
QUANTITY	(T1_ORDERS)	I11
LINETOTAL	(T1_ORDERS)	P22.
COSTOFGOODSSOLD	(T1_ORDERS)	P22.

Right Source:

Name	Table	Format
PRODUCTNUMBER	(T2_INVENTORY)	A4
PRODUCTTYPE	(T2_INVENTORY)	A15
PRODUCTCATEGO...	(T2_INVENTORY)	A30
PRODUCTNAME	(T2_INVENTORY)	A60
MODEL	(T2_INVENTORY)	A10
QUANTITYINSTOCK	(T2_INVENTORY)	I11
PRICE	(T2_INVENTORY)	P13.2
COST	(T2_INVENTORY)	P13.2

Join All [✓ Join Unique]

Join Type : Inner Join

Join Conditions:

	Expression
1	T1_ORDERS.PRODUCTNUMBER = T2_INVENTORY.PRODU...

Viewing Data Profiling Characteristics

In this section:

Data Profiling a Synonym or Segment

Data Profiling Columns

Data Profiling provides data characteristics for the columns in a synonym. You can display the characteristics for all the columns in a synonym or segment, or for an individual column.

For alphanumeric columns, Data Profiling provides the segment, format, count of distinct values, total count, patterns count, maximum, minimum, and average length, minimum and maximum values, and number of nulls. Patterns count shows the number of patterns found in each alphanumeric column.

For numeric columns, Data Profiling provides the segment, format, count of distinct values, total count, maximum, minimum, and average values, and number of nulls.

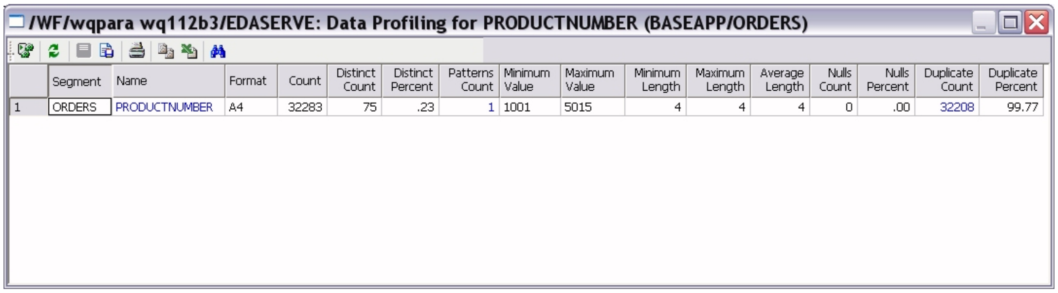
Data Profiling for an individual column provides access to Statistics, Patterns, Values, and Outliers reports.

Data Profiling a Synonym or Segment

How to:

View Data Profiling for a Synonym or Segment

Data Profiling provides information on all the columns in a synonym or segment. You can also drill down to the Values or Patterns reports for an individual column from a synonym or segment Data Profiling report.


/WF/wqpara wq112b3/EDASERVE: Data Profiling for PRODUCTNUMBER (BASEAPP/ORDERS)

	Segment	Name	Format	Count	Distinct Count	Distinct Percent	Patterns Count	Minimum Value	Maximum Value	Minimum Length	Maximum Length	Average Length	Nulls Count	Nulls Percent	Duplicate Count	Duplicate Percent
1	ORDERS	PRODUCTNUMBER	A4	32283	75	.23	1	1001	5015	4	4	4	0	.00	32208	99.77

Procedure: How to View Data Profiling for a Synonym or Segment

To view the Data Profiling information for a synonym or segment:

1. From the Projects or Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   By default, the Synonym Editor opens to the last view used.

2. Right-click the synonym or segment name, select *Data Profiling*, and then *Statistics*.

   The Data Profiling information displays in the workspace. The last four columns are shown below the rest of the information for illustrative purposes only. The actual report runs across the workspace.

You may use the Data Profiling Results toolbar to view server messages, print the report, copy data as text, and export the report.

3. Optionally, you can click a column name or patterns count (for alphanumeric columns) to drill down to the Values or Patterns reports, respectively.

   This is a partial Values report produced by clicking a column name.

   For pattern analysis, a *9* represents a digit, an *A* represents any uppercase letter, and an *a* represents any lowercase letter. All printable special characters are represented by themselves and unprintable characters are represented by an *X*.

Data Profiling Columns

How to:

View Data Profile Statistics

View Data Profile Patterns

View Data Profile Values

View the Data Profile Values Graph

View the Data Profile Values Pie Graph

View Data Profile Duplicate Values

View Data Profile Outliers

Data Profiling can be done for all the columns in a synonym or segment (press the Shift or CTRL key while selecting multiple columns), or for an individual column. Data Profiling for an individual column provides access to the following reports:

- **Statistics.** Shows the same information as a Data Profile report for a synonym or segment.

  For alphanumeric columns, the Statistics report provides the segment, format, count of distinct values, total count, patterns count, maximum, minimum, and average length, minimum and maximum values, and number of nulls.

  For numeric columns, the Statistics report provides the segment, format, count of distinct values, total count, maximum, minimum, and average values, and number of nulls.

- **Patterns.** Only available for alphanumeric columns, shows patterns of letters, digits, and special characters, as well as counts and their percents.

- **Values.** Shows unique values and their percents.

- **Values Graph.** Displays a graph for alphanumeric field types.

- **Values Pie Graph.** Displays a pie graph for alphanumeric field types.
- **Duplicate Values.** Shows identical values and their percents.
- **Outliers.** Shows the ten highest and lowest distinct values and their counts.

These reports are available by right-clicking a column in the Synonym Editor and selecting *Data Profiling*.

Procedure: How to View Data Profile Statistics

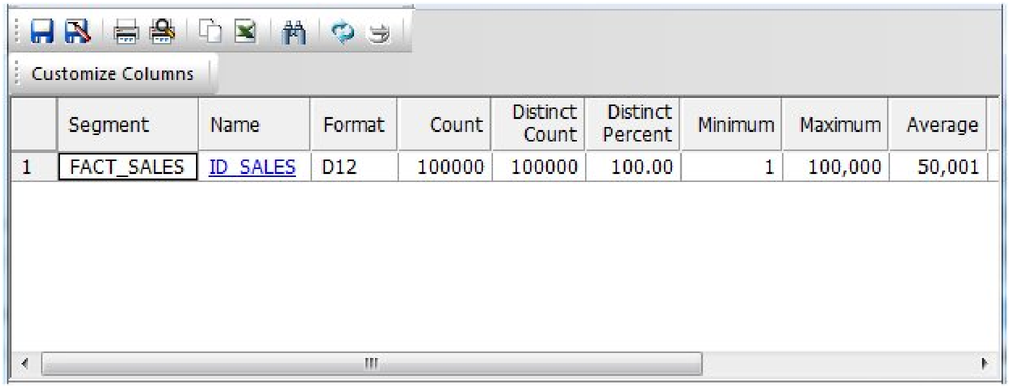
To view the Statistical Data Profiling information for a single column:

1. From the Projects or Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   By default, the Synonym Editor opens to the last view used.

2. Right-click a column, select *Data Profiling*, and then *Statistics*.

   The Statistical Data Profiling information opens in the workspace, as shown in the following image.



	Segment	Name	Format	Count	Distinct Count	Distinct Percent	Minimum	Maximum	Average
1	FACT_SALES	ID_SALES	D12	100000	100000	100.00	1	100,000	50,001

3. Optionally, you can click a column name or patterns count (for alphanumeric columns) to drill down to the Values or Patterns reports, respectively.

Procedure: How to View Data Profile Patterns

Data Profile Patterns shows patterns of letters, digits, and special characters, as well as counts. This is only available for alphanumeric columns.

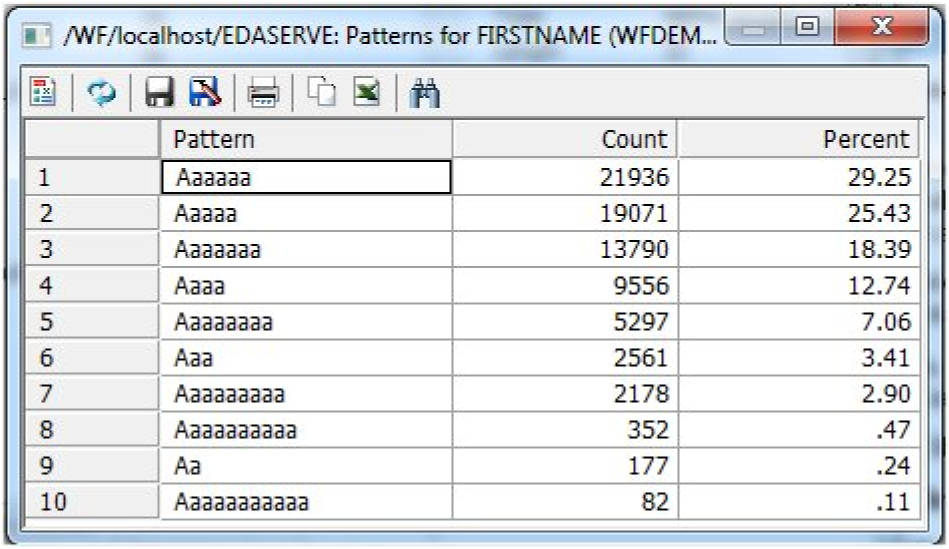
To view the Patterns Data Profiling information for a single column:

1. From the Projects or Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

By default, the Synonym Editor opens to the last view used.

2. Right-click a column, select *Data Profiling*, and then *Patterns*.

   The Patterns Data Profiling information displays, as shown in the following image.



/WF/localhost/EDASERVE: Patterns for FIRSTNAME (WFDEM...

	Pattern	Count	Percent
1	Aaaaaa	21936	29.25
2	Aaaaa	19071	25.43
3	Aaaaaaa	13790	18.39
4	Aaaa	9556	12.74
5	Aaaaaaaa	5297	7.06
6	Aaa	2561	3.41
7	Aaaaaaaaa	2178	2.90
8	Aaaaaaaaaa	352	.47
9	Aa	177	.24
10	Aaaaaaaaaaa	82	.11

For pattern analysis, a *9* represents a digit, an *A* represents any uppercase letter, and an *a* represents any lowercase letter. All printable special characters are represented by themselves and unprintable characters are represented by an *X*.

Procedure: How to View Data Profile Values

Data Profile Values shows unique values.

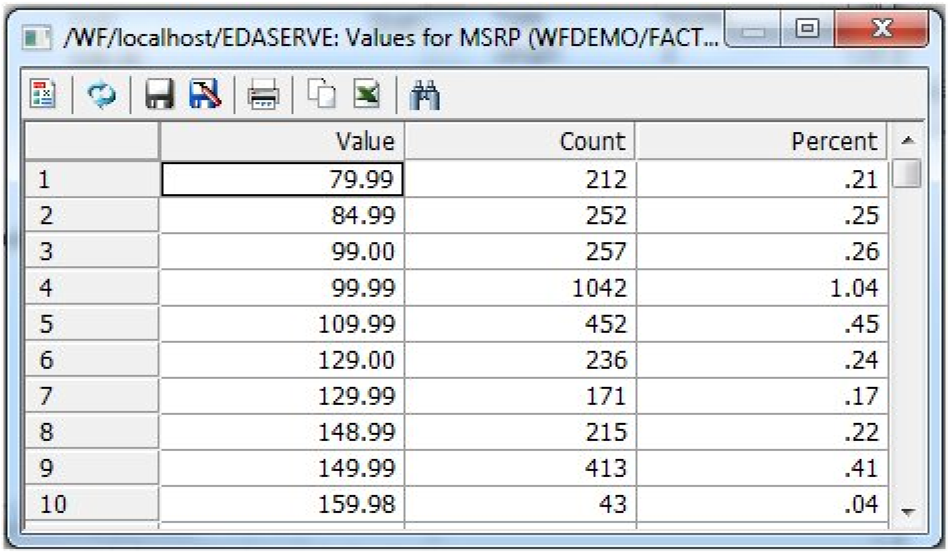
To view the Values Data Profiling information for a single column:

1. From the Projects or Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   By default, the Synonym Editor opens to the last view used.

2. Right-click a column, select *Data Profiling*, and then *Values*.

The Values Data Profiling information displays, as shown in the following image.


/WF/localhost/EDASERVE: Values for MSRP (WFDEMO/FACT...

	Value	Count	Percent
1	79.99	212	.21
2	84.99	252	.25
3	99.00	257	.26
4	99.99	1042	1.04
5	109.99	452	.45
6	129.00	236	.24
7	129.99	171	.17
8	148.99	215	.22
9	149.99	413	.41
10	159.98	43	.04

Procedure: How to View the Data Profile Values Graph

The Data Profile Values Graph displays values as a bar graph.

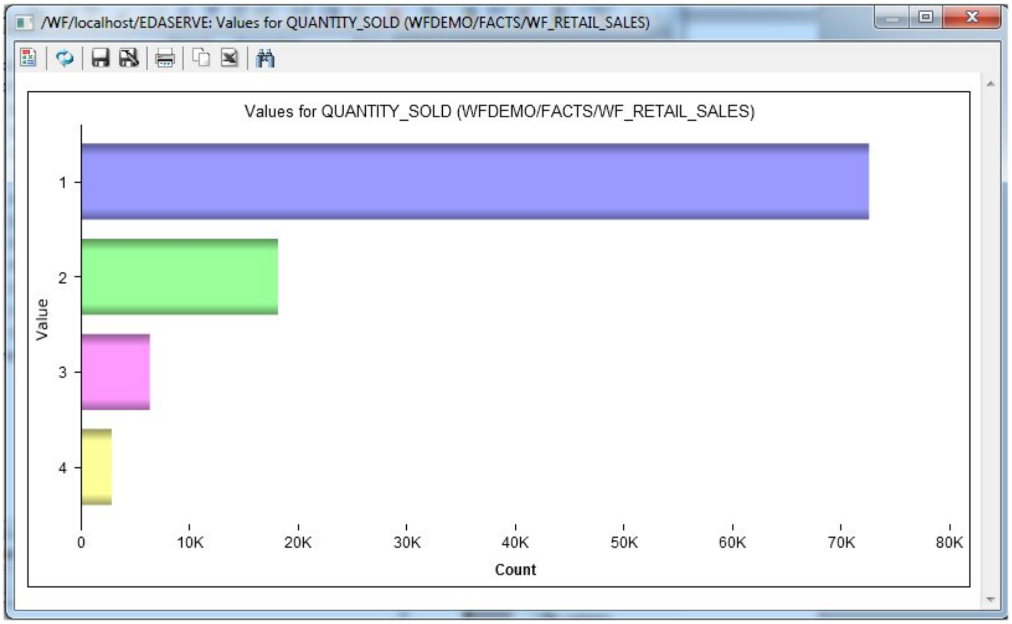
To view the Values Graph Data Profiling information for a single column:

1. From the Projects or Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   By default, the Synonym Editor opens to the last view used.

2. Right-click a column, select *Data Profiling*, and then *Values Graph*.

The Values Graph Data Profiling information displays, as shown in the following image.


/WF/localhost/EDASERVE: Values for QUANTITY_SOLD (WFDEMO/FACTS/WF_RETAIL_SALES)
Values for QUANTITY_SOLD (WFDEMO/FACTS/WF_RETAIL_SALES)
Value
1
2
3
4
0
10K
20K
30K
40K
50K
60K
70K
80K
Count

Procedure: How to View the Data Profile Values Pie Graph

The Data Profile Values Pie Graph displays values as a pie graph.

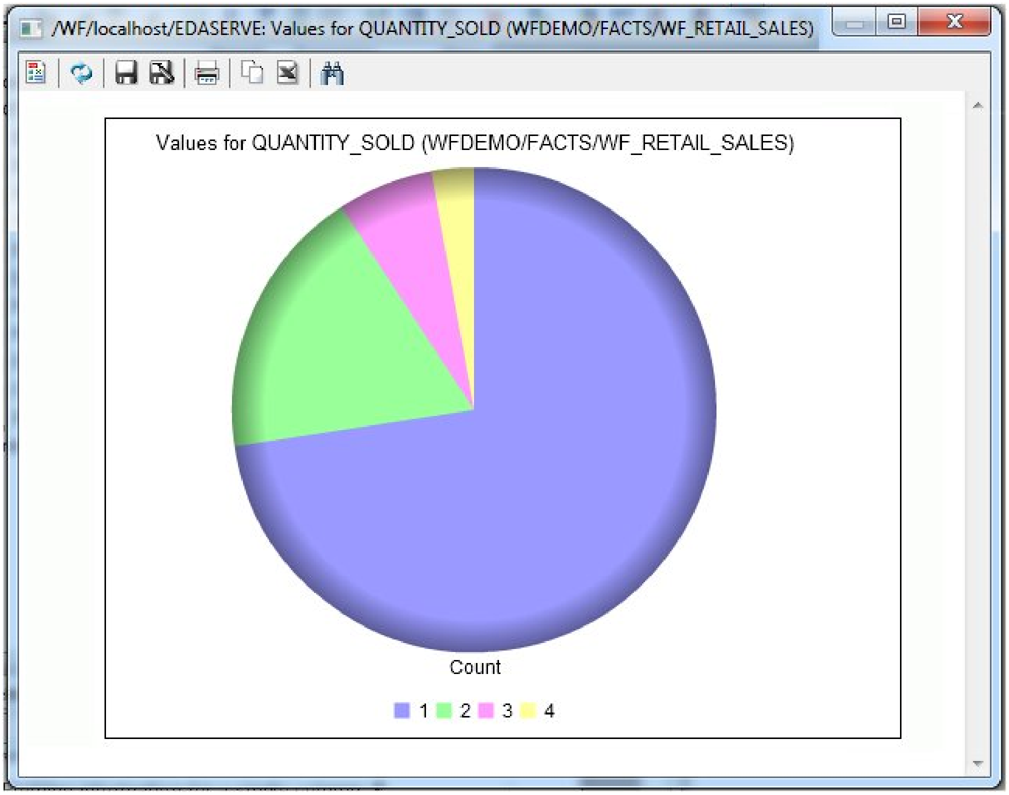
To view the Values Pie Graph Data Profiling information for a single column:

1. From the Projects or Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   By default, the Synonym Editor opens to the last view used.

2. Right-click a column, select *Data Profiling*, and then *Values Pie Graph*.

The Values Pie Graph Data Profiling information displays, as shown in the following image.


/WF/localhost/EDASERVE: Values for QUANTITY_SOLD (WFDEMO/FACTS/WF_RETAIL_SALES)
Values for QUANTITY_SOLD (WFDEMO/FACTS/WF_RETAIL_SALES)
Count
1
2
3
4

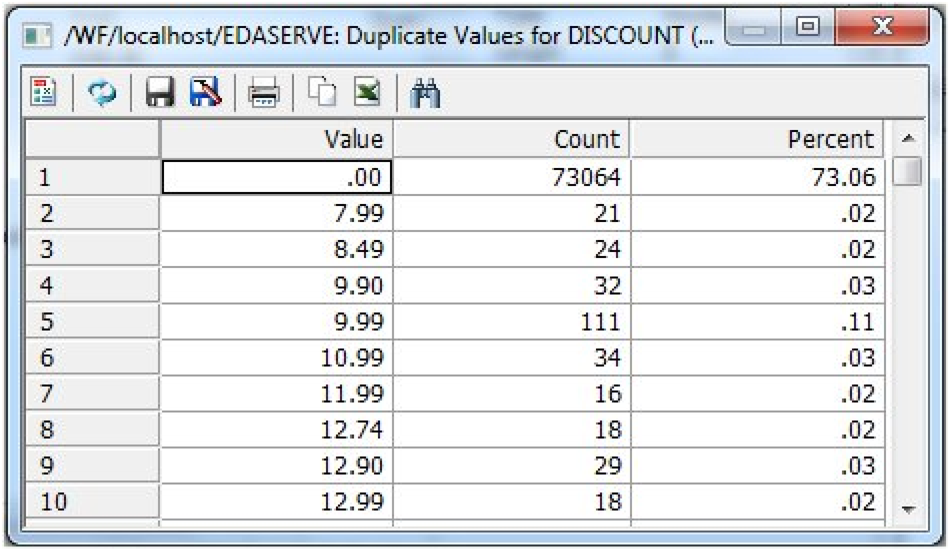
Procedure: How to View Data Profile Duplicate Values

Data Profile Duplicate Values shows identical values.

To view the Duplicate Values Data Profiling information for a single column:

1. From the Projects or Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.
2. Right-click a column, select *Data Profiling*, and then *Duplicate Values*.

The Duplicate Values Data Profiling information displays, as shown in the following image.



	Value	Count	Percent
1	.00	73064	73.06
2	7.99	21	.02
3	8.49	24	.02
4	9.90	32	.03
5	9.99	111	.11
6	10.99	34	.03
7	11.99	16	.02
8	12.74	18	.02
9	12.90	29	.03
10	12.99	18	.02

Procedure: How to View Data Profile Outliers

Data Profile Outliers shows the ten highest and ten lowest distinct values.

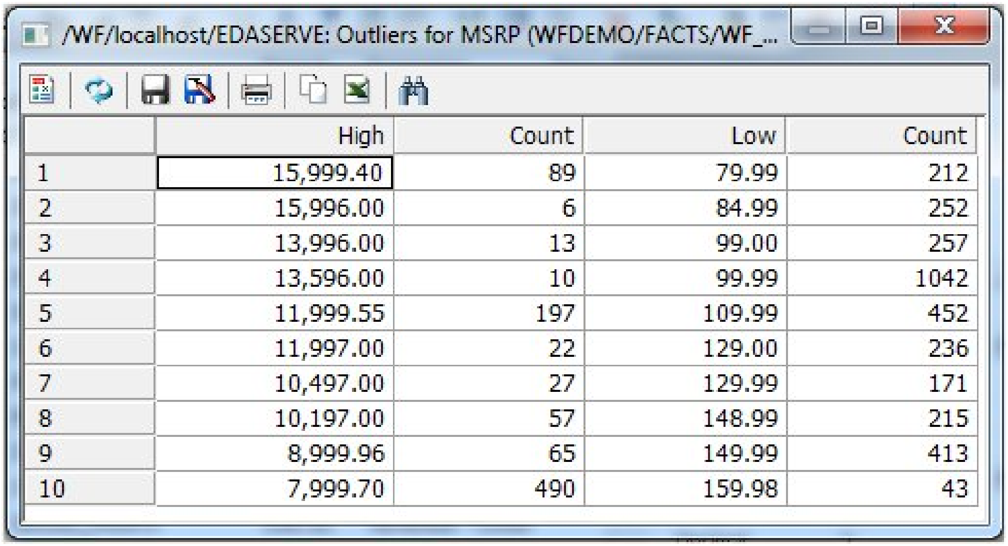
To view the Outliers Data Profiling information for a single column:

1. From the Projects or Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   By default, the Synonym Editor opens to the last view used.

2. Right-click a column, select *Data Profiling*, and then *Outliers*.

The Outliers Data Profiling information displays, as shown in the following image.


/WF/localhost/EDASERVE: Outliers for MSRP (WFDEMO/FACTS/WF_...

	High	Count	Low	Count
1	15,999.40	89	79.99	212
2	15,996.00	6	84.99	252
3	13,996.00	13	99.00	257
4	13,596.00	10	99.99	1042
5	11,999.55	197	109.99	452
6	11,997.00	22	129.00	236
7	10,497.00	27	129.99	171
8	10,197.00	57	148.99	215
9	8,999.96	65	149.99	413
10	7,999.70	490	159.98	43

Note: Outliers produce a maximum of the ten highest and ten lowest distinct values, if they exist.

Creating Cluster Joins

How to:

Create a Cluster Join by Enhancing Existing Synonyms

Create a Cluster Join Using a New Synonym

Cluster joins enable you to create a new file structure by linking existing synonyms of two or more relational tables using the same or mixed data sources. For example, you may join a DB2 table and an Oracle table, and so on. Use cluster joins to create new views in the metadata by linking together physical tables and easily report against the new view or structure. You can create cluster joins by using the Modeling View of the Synonym Editor.

The Master File that is created combines the fields of the joined tables within a single file. The Access File from the combined file contains information about the actual location of the data sources and the Join information. It also shows how the tables are linked.

The total number of tables that you can add to the tool is 64 (using 63 joins), which results in a new Master File that has a maximum of 64 segments.

The Cluster Join tool enables you to create a Star Schema which consists of a fact table referencing a number of dimension tables. Optionally, you can also create a view that has more than one fact table.

Procedure: How to Create a Cluster Join by Enhancing Existing Synonyms

Use the Modeling View to enhance an existing synonym by adding a segment.

1. From the Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

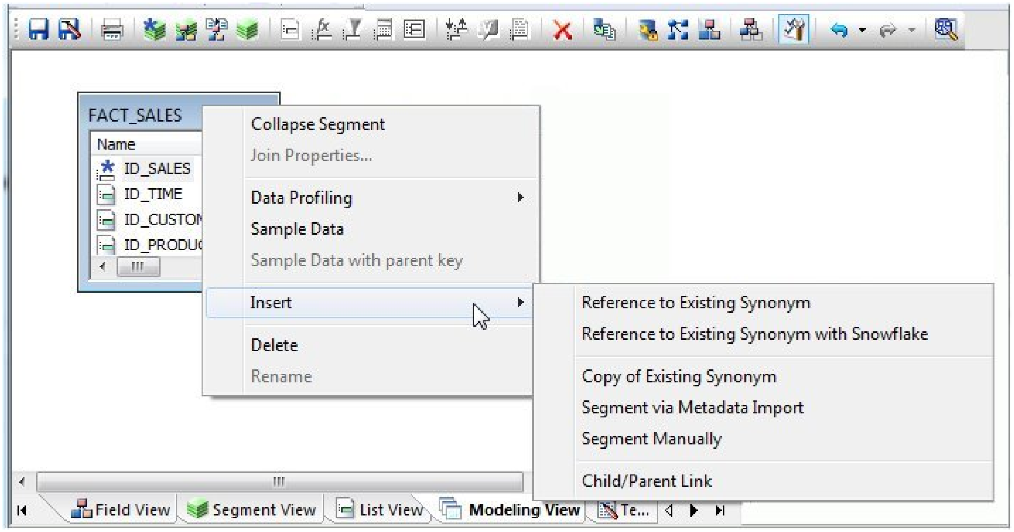
The Synonym Editor opens.

2. Click the *Modeling View* tab.

The Synonym Editor Modeling View tab opens in the workspace.

3. Right-click a segment in the workspace and select *Insert*.

The following context menu appears, as shown in the following image.

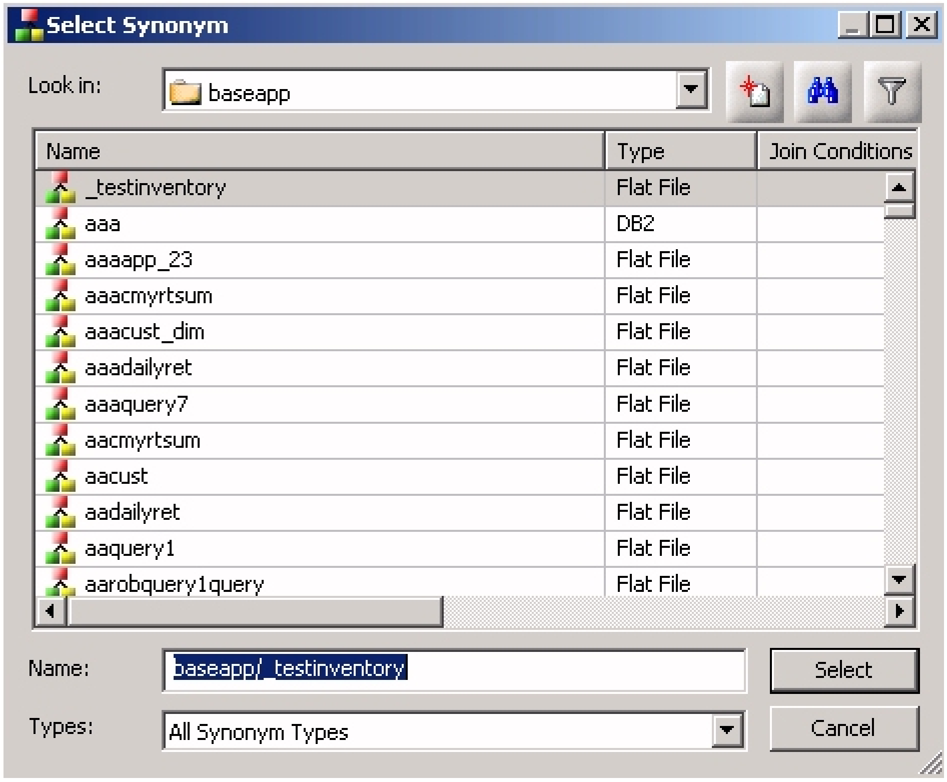


4. Insert tables (segments) through one of the methods listed:

To insert a segment from an existing synonym:

a. Select *Insert*, then *Reference to Existing Synonym*.

The Select Synonym dialog box opens, as shown in the following image.



Name	Type	Join Conditions
_testinventory	Flat File	
aaa	DB2	
aaaapp_23	Flat File	
aaacmyrtsum	Flat File	
aaacust_dim	Flat File	
aaadailyret	Flat File	
aaaquery7	Flat File	
aacmyrtsum	Flat File	
aacust	Flat File	
aadailyret	Flat File	
aaquery1	Flat File	
aarobquery1query	Flat File	

Name: baseapp/_testinventory

Types: All Synonym Types

b. Select a synonym to be inserted and click *Select*.

Note: Use this method if you are creating a cluster join with an existing table or synonym.

Tip: Click *Save As* from the Modeling View File menu if you do not want to modify the original synonym.

To insert a segment from an existing synonym with Snowflake:

Select *Insert*, then *Reference to Existing Synonym with Snowflake*.

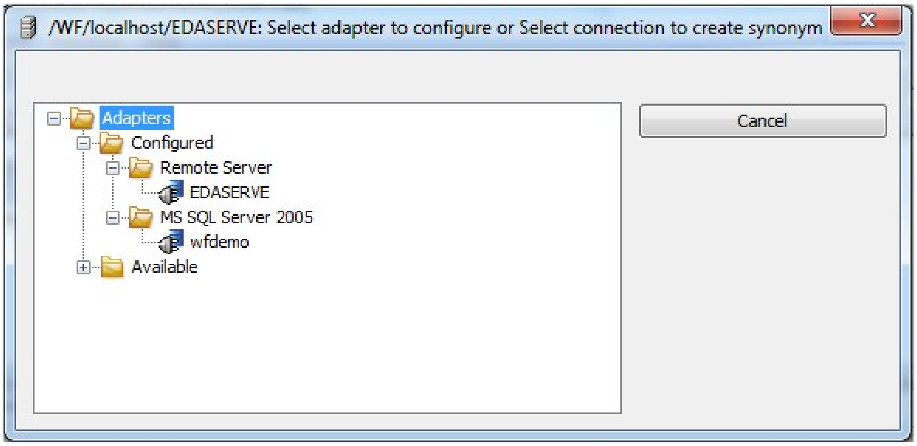
The procedure is the same as Reference to Existing Synonym, except the selection list will show both candidate synonyms and any synonyms that they reference.

To insert a segment via Metadata Import:

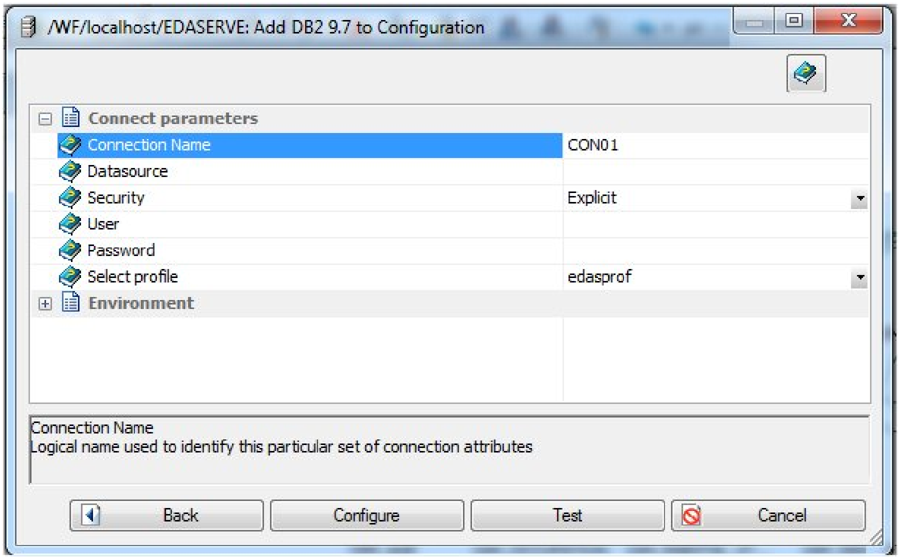
a. Select *Insert*, then *Segment via Metadata Import*. This enables you to add segments by using the Create Synonym tool. This tool creates a synonym and includes it as a segment in the synonym from which the tool was launched.

Note: Use this method if you are creating a cluster join and need to use a synonym that does not exist. This option enables you to create the synonym and continue to create the cluster join.

b. When this option is selected, you are first presented with the Adapter dialog box where you can select a configured adapter connection to continue or configure a new adapter, if necessary. The Adapter dialog box is shown in the following image.


/WF/localhost/EDASERVE: Select adapter to configure or Select connection to create synonym
Adapters
Configured
Remote Server
EDASERVE
MS SQL Server 2005
wfdemo
Available
Cancel

The following image is an example of a screen that appears where you provide information for the connection parameters.


/WF/localhost/EDASERVE: Add DB2 9.7 to Configuration

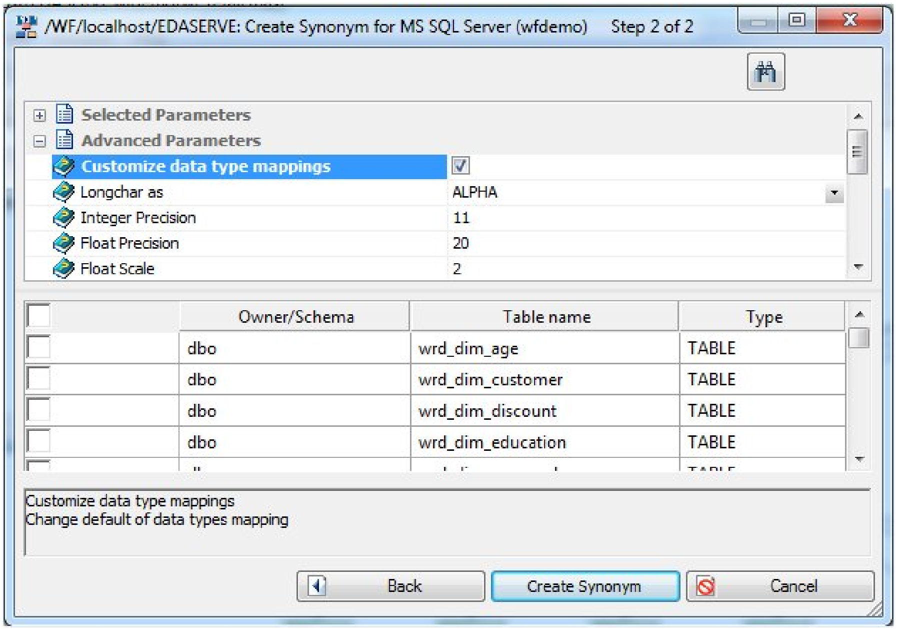
Connect parameters

Connection Name	CON01
Datasource	
Security	Explicit
User	
Password	
Select profile	edasprof

Environment

Connection Name
Logical name used to identify this particular set of connection attributes

The following image is an example of a screen that enables you to select tables to create synonyms.



/WF/localhost/EDASERVE: Create Synonym for MS SQL Server (wfdemo) Step 2 of 2

Selected Parameters

Advanced Parameters

Customize data type mappings	✓
Longchar as	ALPHA
Integer Precision	11
Float Precision	20
Float Scale	2

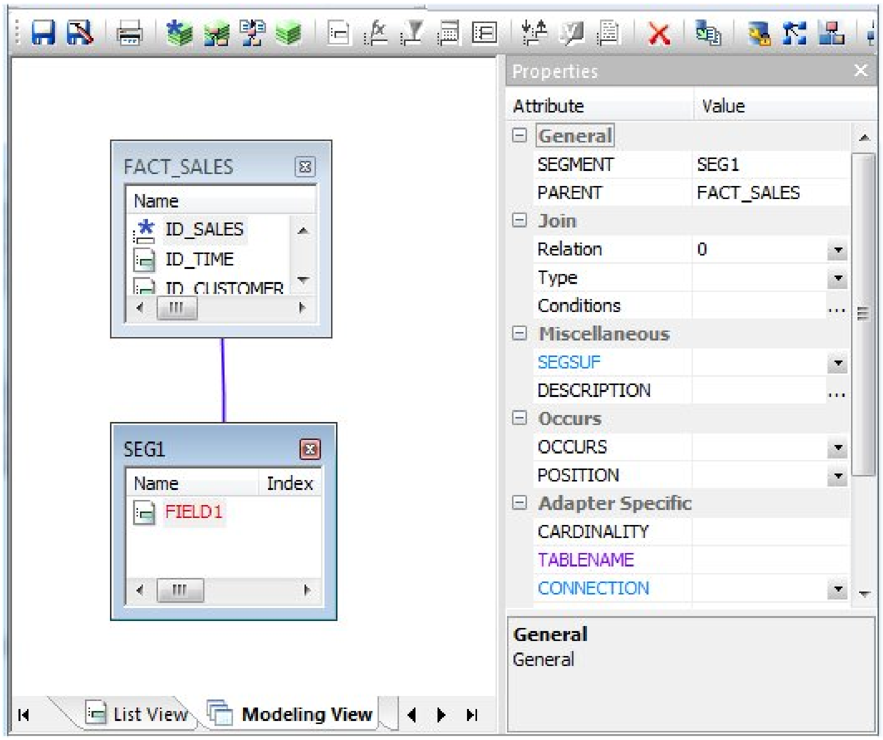
	Owner/Schema	Table name	Type
	dbo	wrd_dim_age	TABLE
	dbo	wrd_dim_customer	TABLE
	dbo	wrd_dim_discount	TABLE
	dbo	wrd_dim_education	TABLE

Customize data type mappings
Change default of data types mapping

The selected synonyms will be created and added to the Modeling View.

To insert a segment manually:

Select *Insert*, then *Segment Manually*. This enables you to assign values to segment attribute fields in the Synonym Editor.



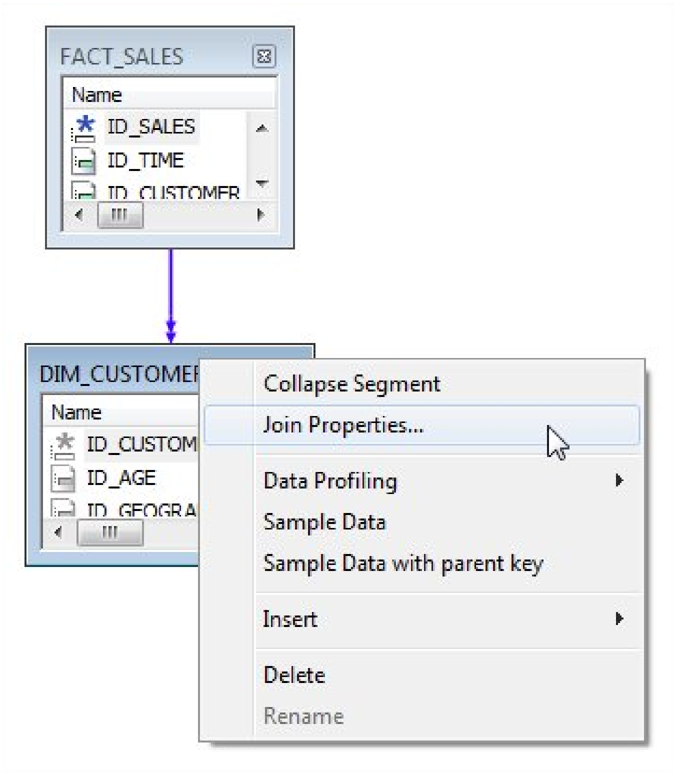
Attribute	Value
General	
SEGMENT	SEG1
PARENT	FACT_SALES
Join	
Relation	0
Type	
Conditions	
Miscellaneous	
SEGSUF	
DESCRIPTION	
Occurs	
OCCURS	
POSITION	
Adapter Specific	
CARDINALITY	
TABLENAME	
CONNECTION	

General
General

Note: Use this method if you are coding a new Master File, as you would for a Web Query data source.

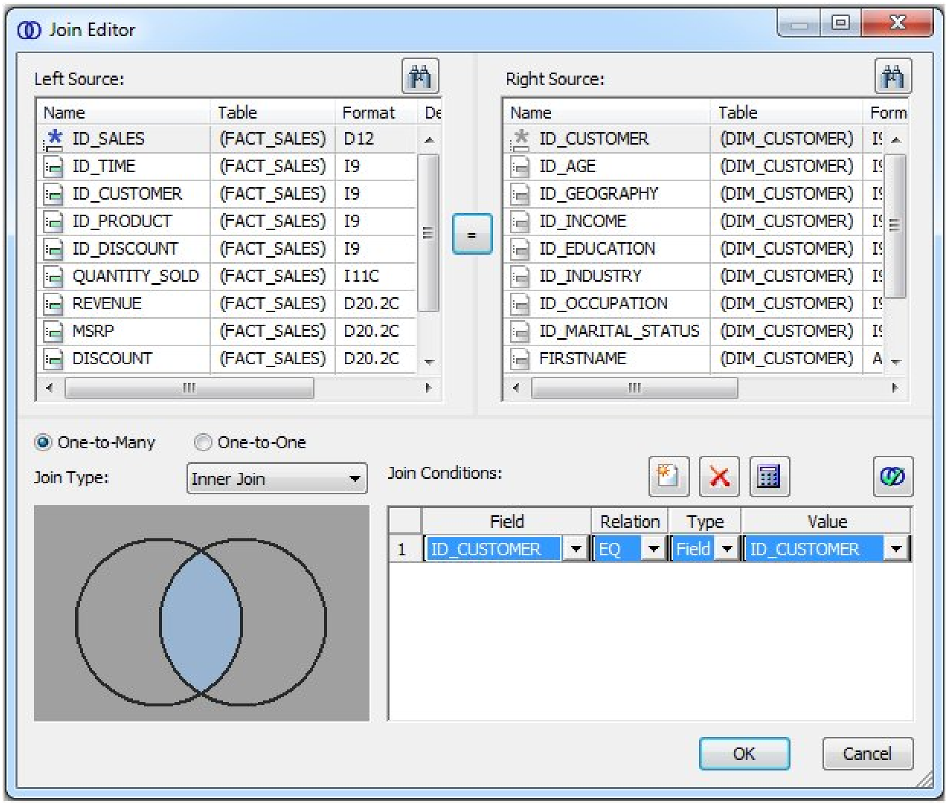
The segment is added in the Modeling View.

5. Right-click a segment and select *Join Properties*, as shown in the following image.



Note: This option is only available when using relational tables.

The Join Editor dialog box opens, as shown in the following image.



Left Source:

Name	Table	Format	De
ID_SALES	(FACT_SALES)	D12	
ID_TIME	(FACT_SALES)	I9	
ID_CUSTOMER	(FACT_SALES)	I9	
ID_PRODUCT	(FACT_SALES)	I9	
ID_DISCOUNT	(FACT_SALES)	I9	
QUANTITY_SOLD	(FACT_SALES)	I11C	
REVENUE	(FACT_SALES)	D20.2C	
MSRP	(FACT_SALES)	D20.2C	
DISCOUNT	(FACT_SALES)	D20.2C	

Right Source:

Name	Table	Form
ID_CUSTOMER	(DIM_CUSTOMER)	I!
ID_AGE	(DIM_CUSTOMER)	I!
ID_GEOGRAPHY	(DIM_CUSTOMER)	I!
ID_INCOME	(DIM_CUSTOMER)	I!
ID_EDUCATION	(DIM_CUSTOMER)	I!
ID_INDUSTRY	(DIM_CUSTOMER)	I!
ID_OCCUPATION	(DIM_CUSTOMER)	I!
ID_MARITAL_STATUS	(DIM_CUSTOMER)	I!
FIRSTNAME	(DIM_CUSTOMER)	A

[✓ One-to-Many] One-to-One

Join Type: Inner Join

Join Conditions:

	Field	Relation	Type	Value
1	ID_CUSTOMER	EQ	Field	ID_CUSTOMER

6. Select a radio button for *One-to-Many* (Join All) or *One-to-One* (Join Unique).

- **One-to-Many.** Indicates a multiple-instance join. At run time, each host record can have many matching records in the cross-referenced file.
- **One-to-One.** Indicates a single-instance join. At run time, each host record has, at most, one matching record in the cross-referenced file.

7. Select a *Join Type* from the drop down menu. The choices are Inner, Left Outer, or Cross Join.

The Join Condition field automatically creates a Join if identical fields exist in both segments.

Procedure: How to Create a Cluster Join Using a New Synonym

Another way to create a cluster join is to start with an empty synonym:

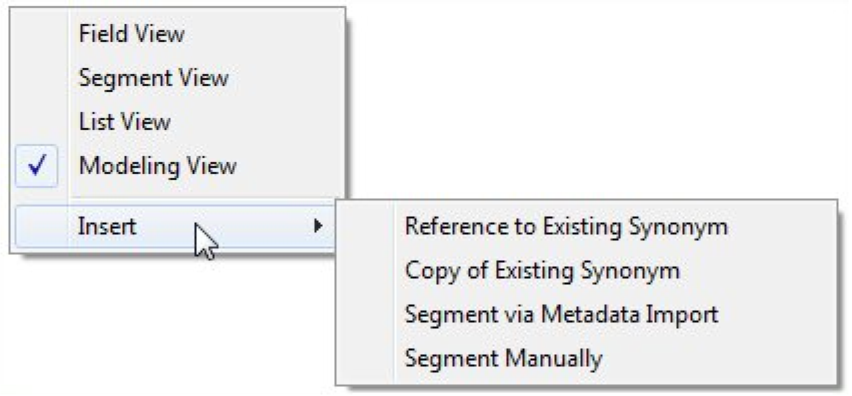
1. Right-click a Master Files folder, select *New*, then *Synonym via Synonym Editor*.

The New Master File dialog box opens.

2. Enter a unique file name in the File name field.
3. Click *Create*.

   The Synonym Editor opens.

4. Click the *Modeling View* tab.
5. Right-click in the workspace and select from one of the available options to start building the new view.



Defining Dimensions for OLAP Analysis

In this section:

Using the Dimension Builder in the Synonym Editor Modeling View

Synonyms can be modified to support Online Analytical Processing (OLAP). The Synonym Editor provides tools to create OLAP hierarchies and dimensions. OLAP enables you to drill down or roll up on hierarchical data, pivot fields from columns to rows (or vice versa), and slice-and-dice information by filtering or querying data sources based on specified criteria thresholds.

You OLAP-enable the Master File by using the Synonym Editor to create dimension(s) at the field level and associate fields with each dimension.

Note: OLAP is a reporting facility. It is not relevant to data maintenance projects.

Using the Dimension Builder in the Synonym Editor Modeling View

How to:

Add a Parent/Child Hierarchy

Delete a Dimension

Add Levels to the Hierarchy

Reference:

Dimension Builder Toolbar

The Dimension Builder enables you to create logical views based on enterprise data (relational or legacy data sources) for multidimensional analysis without manually editing metadata. The Dimension Builder works with relational and non-relational data sources. You can enable the Dimension Builder by selecting *Dimension Builder* from the toolbar or the Tools menu.

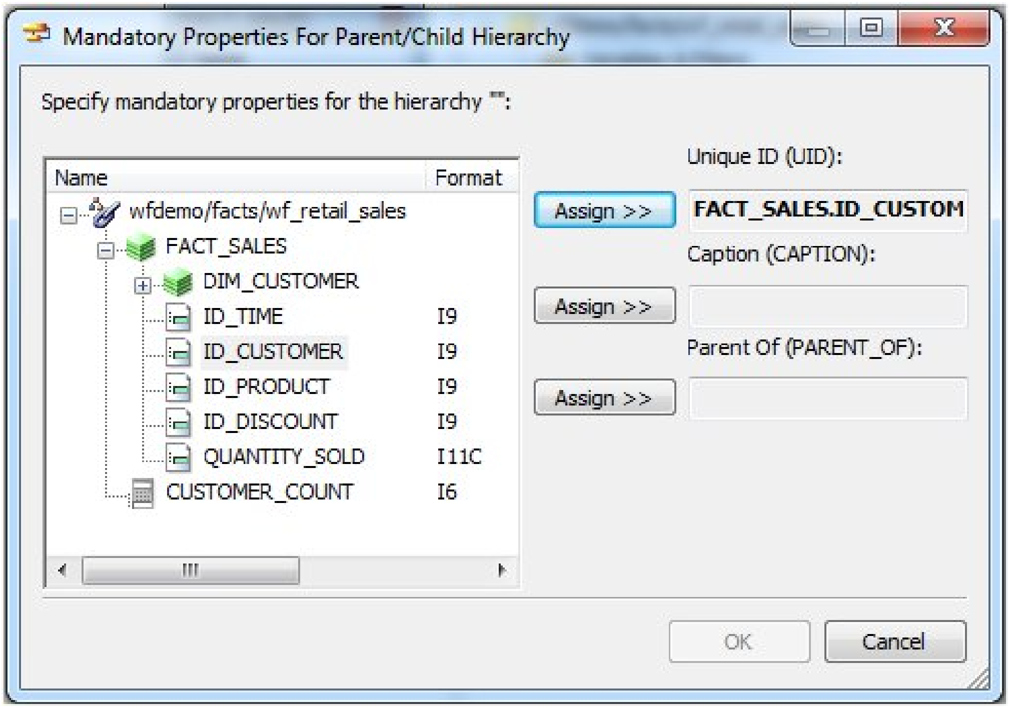
Procedure: How to Add a Parent/Child Hierarchy

1. From the Synonym Editor, click the *Modeling View* tab.
2. Click the *Dimension Builder* icon on the toolbar.

   The Dimension Builder opens.
3. Click the *Parent/Child Hierarchy* icon from the Dimension Builder toolbar.

   The Mandatory Properties for Parent/Child Hierarchy dialog box opens.

4. Select a field from the Field Tree and click *Assign* to assign a Unique ID for the hierarchy.



Mandatory Properties For Parent/Child Hierarchy

Specify mandatory properties for the hierarchy "":

Name	Format
wfdemo/facts/wf_retail_sales	
FACT_SALES	
DIM_CUSTOMER	
ID_TIME	I9
ID_CUSTOMER	I9
ID_PRODUCT	I9
ID_DISCOUNT	I9
QUANTITY_SOLD	I11C
CUSTOMER_COUNT	I6

Unique ID (UID): FACT_SALES.ID_CUSTOM

Caption (CAPTION):

Parent Of (PARENT_OF):

5. Repeat step 4 for the Caption and Parent hierarchy properties.
6. Click *OK* to close the Mandatory Properties for Parent/Child Hierarchy dialog box.

   The Mandatory Properties are added to the Dimension Builder.

7. Use the right-click menu to rename the dimension, view properties, or view sample data for the dimension.
8. Click *Save* from the File menu to save the dimension.

Procedure: How to Delete a Dimension

To delete a dimension, right-click a dimension and select *Delete*, or click the *Delete* button from the Dimension Builder toolbar.

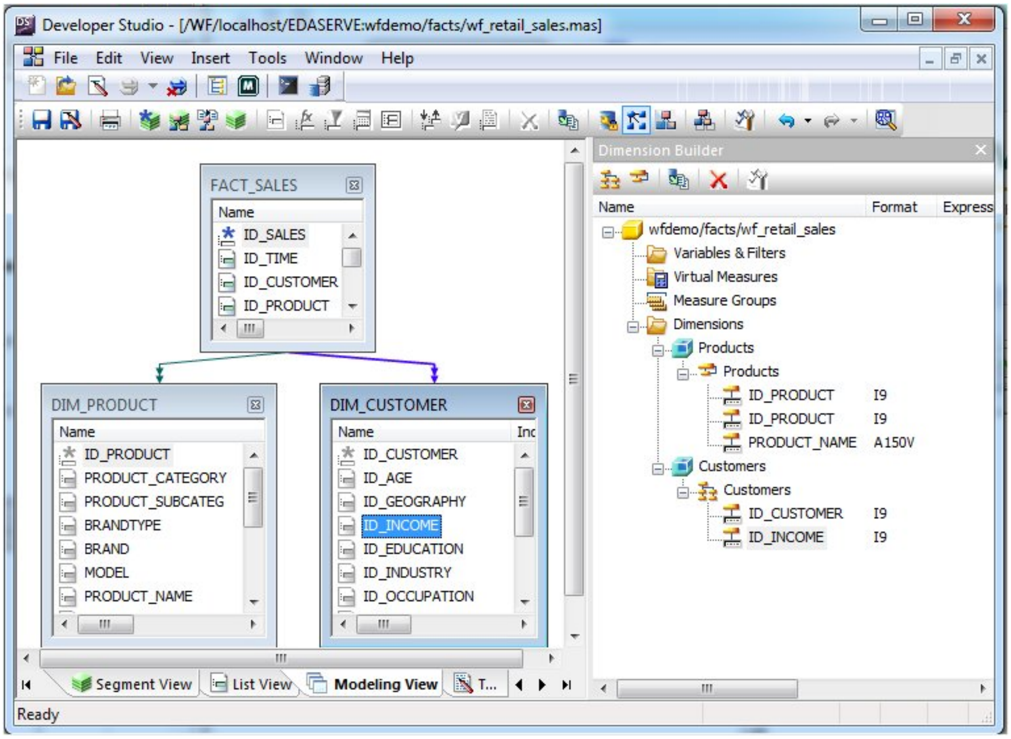
Procedure: How to Add Levels to the Hierarchy

This process enables you to edit an existing Master File, add tables and create and modify dimensions.

1. From the Synonym Editor, click the *Modeling View* tab.
2. Select the *Levels Hierarchy* button from the Dimension Builder toolbar.

   A level is added to the hierarchy. Use the right-click menu to rename the dimension or keep the default name.

3. Drag the selected fields from a segment into the Dimension Builder hierarchy folder.



4. Click *Save* from the File menu to save the dimension.

   The dimension is saved and stored in the Master File.

Reference: Dimension Builder Toolbar

You can access the following commands from the Dimension Builder toolbar.

Button	Definition
🗂	Adds a level hierarchy.

Button	Definition
	Enables you to create a parent/child hierarchy and assign mandatory properties for the hierarchy.
	Deletes the selected item.
	Enables you to view and refresh sample data for the selected field.
	Enables you to edit a hierarchy.

Creating Business Views

How to:

Create a Business View Using the Synonym Editor

Alternatively Create a Business View Using the Synonym Editor

Reference:

Usage Notes for Business Views

Using a Business View Master File

By defining a Business View of a Master File, you are creating an alternative view of the Master File and can limit the fields available or create a subset of fields from the original Master File. Fields can be grouped into meaningful folders. Field names, titles, and descriptions can be customized for each Business View.

Fields in a Business View are organized into folders. Each folder contains a group of fields. The fields in a folder can come from different segments in the original Master File. The Business View may contain existing fields and can include existing custom fields for DEFINE, COMPUTE, and Filters. Custom fields are associated with a specific segment in the original Master File and are subject to the same rules as real fields. A report can reference fields from multiple folders if they all lie along a single path in the original Master File.

When opening a Master File in the Synonym Editor and clicking the Business View button on the toolbar, a Business View pane opens inside the Synonym Editor where Business View attributes can be added. Adding Business View attributes at this point would update the Master File by inserting the Business View attributes at the end of the file, and once the file is saved, the Business View becomes the active view when the file is used for reporting.

If you do not wish to turn your Master File into a Business View, it is recommended that you save the Business View using a different name or preferably start by creating a new Master File by selecting *New*, then *Synonym via Synonym Editor*. Then, insert a reference to an existing synonym and continue to build the Business View. In this case, the Business View points to the cross-referenced Master File, and all of the actual fields and security information comes from the referenced file when the Business View is used in WebFOCUS tools (such as Joins and Defines) and in reports.

Note: Impact Analysis searches Business Views in addition to DB2 Web Query procedures. This enables you to see if changes in the original Master File will impact fields used in the Business View. For detailed information on Impact Analysis, see *Analyzing Metadata and Procedures* on page 189.

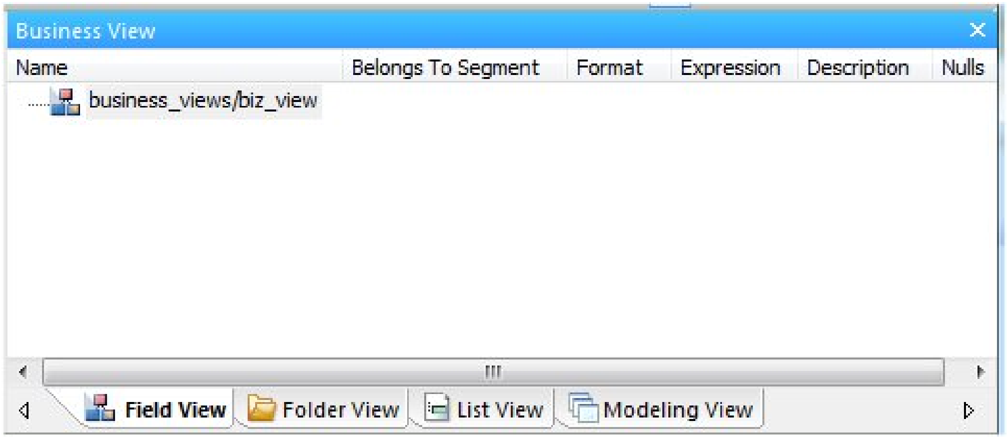
You may create a Business View for an existing Master File by using the Synonym Editor.

Procedure: How to Create a Business View Using the Synonym Editor

1. From the Data Servers area, navigate to the Master Files folder where you want to create the Business View.
2. Right-click the Master Files folder and select *New*, then *Synonym via Synonym Editor*.
3. Provide a unique name for the new file and click *Open* or *Create* if in the Data Servers area.
4. While in the Field View tab of the Synonym Editor, right-click the file name in the upper-left corner, select *Insert*, and then *Reference to Existing Synonym*.
5. From the Insert Reference to Existing Synonym dialog box, select the synonym for which you want to create a Business View.

   The referenced file is added to the new synonym and its fields are visible in the left frame.
6. Select *Business View* from the Tools menu to open the Business View pane. You can also click the Business View icon from the Synonym Editor.

Note: A Business View Master File may contain only one root folder.



Tip: Select *Properties* from the Tools menu to open the Properties pane and view additional information for items selected in the Business View pane. Use the Properties pane to change titles, descriptions, or field names. Items that cannot be edited are grayed out.

7. In the Business View pane, right-click the file name and select *Create Default Business View* or *New Folder*.

   The *Create Default Business View* option duplicates the segments and fields that are available in the Master File. You can reorganize the view as necessary. The *New Folder* option creates a root folder to which you can add fields from the Master File on the left pane by selecting them and then dragging them inside the folder. You can create additional folders to create the structure you want.

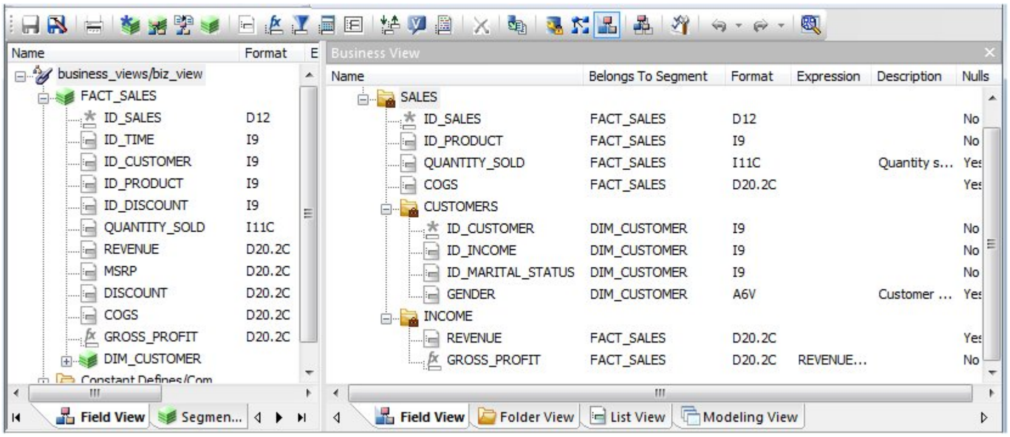
8. To add additional folders for the Business View, right-click the Business View root folder that was created in the Business View pane and select *New Folder*.

   Note: Multiple subfolders can be created and folders may be empty for organizational purposes.

9. Select fields from the Master File on the left and drag them to the appropriate folder in the Business View pane. Press the Shift or Ctrl key while selecting multiple fields.

   Note: If needed, fields may be duplicated by placing them in multiple folders, but any given folder may contain a field only once.

   The selected fields appear in the Business View pane.



Name	Format	E
business_views/biz_view		
FACT_SALES		
ID_SALES	D12	
ID_TIME	I9	
ID_CUSTOMER	I9	
ID_PRODUCT	I9	
ID_DISCOUNT	I9	
QUANTITY_SOLD	I11C	
REVENUE	D20.2C	
MSRP	D20.2C	
DISCOUNT	D20.2C	
COGS	D20.2C	
GROSS_PROFIT	D20.2C	
DIM_CUSTOMER		
Constant Defines/Com		

Business View

Name	Belongs To Segment	Format	Expression	Description	Nulls
SALES					
ID_SALES	FACT_SALES	D12			No
ID_PRODUCT	FACT_SALES	I9			No
QUANTITY_SOLD	FACT_SALES	I11C		Quantity s...	Yes
COGS	FACT_SALES	D20.2C			Yes
CUSTOMERS					
ID_CUSTOMER	DIM_CUSTOMER	I9			No
ID_INCOME	DIM_CUSTOMER	I9			No
ID_MARITAL_STATUS	DIM_CUSTOMER	I9			No
GENDER	DIM_CUSTOMER	A6V		Customer ...	Yes
INCOME					
REVENUE	FACT_SALES	D20.2C			Yes
GROSS_PROFIT	FACT_SALES	D20.2C	REVENUE...		No

10. Select the *Save* or *Save As* icon to save the Business View as a Business View Master File.

    Note: The Business View Master File may be saved in a different application than the main files.

Procedure: How to Alternatively Create a Business View Using the Synonym Editor

It is recommended that you create a Business View using the Synonym Editor as outlined in *How to Create a Business View Using the Synonym Editor* on page 144. However, the following is an alternative way to create a Business View using the Synonym Editor.

1. From the Data Servers area, navigate to the Master Files folder where you wish to create the Business View and highlight the Master File to be altered.

   Note: You may create a Business View anywhere that you can select a Master File.

2. Double-click the Master File or select *Edit in Synonym Editor* from the File menu.

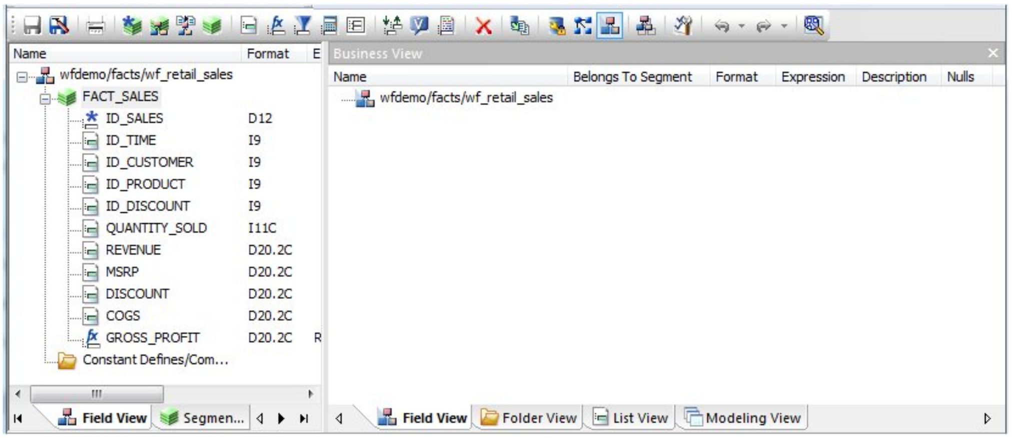
   The Master File opens to the Field View tab in the Synonym Editor.

   Note: When opening a Master File in the Synonym Editor, the tool opens to the last tab selected when the Synonym Editor was last accessed.

3. Select *Business View* from the Tools menu. You can also click the Business View icon from the Synonym Editor.

   The Business View pane opens.

   Note: A Business View Master File may contain only one root folder.



Name	Format	E
wfdemo/facts/wf_retail_sales		
FACT_SALES		
ID_SALES	D12	
ID_TIME	I9	
ID_CUSTOMER	I9	
ID_PRODUCT	I9	
ID_DISCOUNT	I9	
QUANTITY_SOLD	I11C	
REVENUE	D20.2C	
MSRP	D20.2C	
DISCOUNT	D20.2C	
COGS	D20.2C	
GROSS_PROFIT	D20.2C	R
Constant Defines/Com...		

Name	Belongs To Segment	Format	Expression	Description	Nulls
wfdemo/facts/wf_retail_sales					

   Tip: Select *Properties* from the Tools menu to open the Properties pane and view additional information for items selected in the Business View pane. Use the Properties pane to change titles, descriptions, or field names. Items that cannot be edited are grayed out.

4. In the Business View pane, right-click the file name and select *Create Default Business View* or *New Folder*.

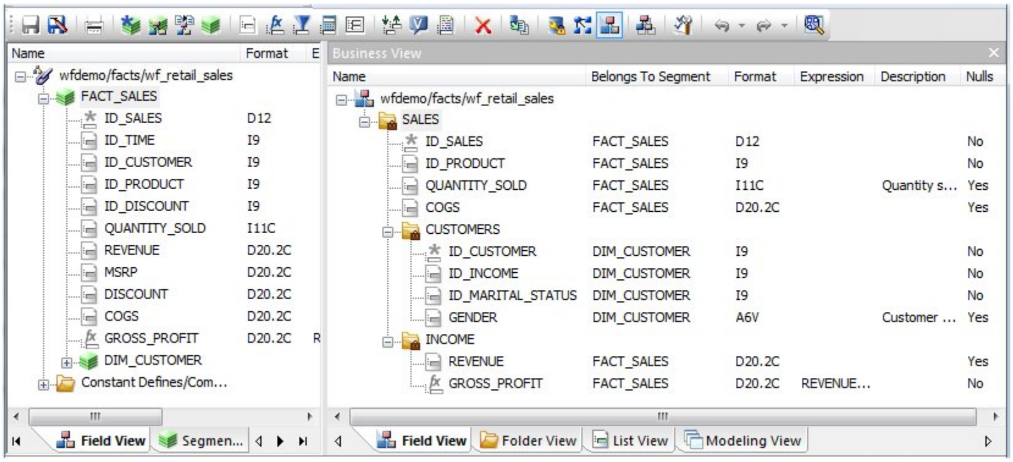
   The *Create Default Business View* option duplicates the segments and fields that are available in the Master File, and you can reorganize the view as necessary. The *New Folder* option creates a root folder to which you can add fields from the Master File on the left pane by selecting them and then dragging and dropping them inside the folder. You can create additional folders to create the structure you want.

5. To add additional folders for the Business View, right-click the Business View root folder that was created in the Business View pane and select *New Folder*.

   Note: Multiple subfolders can be created and folders may be empty for organizational purposes.

6. Select fields from the Master File on the left and drag and drop them to the appropriate folder in the Business View pane. Press the Shift or Ctrl key while selecting multiple fields.

   The selected fields appear in the Business View Tree tab.



Name	Format	E
wfdemo/facts/wf_retail_sales		
FACT_SALES		
ID_SALES	D12	
ID_TIME	I9	
ID_CUSTOMER	I9	
ID_PRODUCT	I9	
ID_DISCOUNT	I9	
QUANTITY_SOLD	I11C	
REVENUE	D20.2C	
MSRP	D20.2C	
DISCOUNT	D20.2C	
COGS	D20.2C	
GROSS_PROFIT	D20.2C	R
DIM_CUSTOMER		
Constant Defines/Com...		

Business View

Name	Belongs To Segment	Format	Expression	Description	Nulls
wfdemo/facts/wf_retail_sales					
SALES					
ID_SALES	FACT_SALES	D12			No
ID_PRODUCT	FACT_SALES	I9			No
QUANTITY_SOLD	FACT_SALES	I11C		Quantity s...	Yes
COGS	FACT_SALES	D20.2C			Yes
CUSTOMERS					
ID_CUSTOMER	DIM_CUSTOMER	I9			No
ID_INCOME	DIM_CUSTOMER	I9			No
ID_MARITAL_STATUS	DIM_CUSTOMER	I9			No
GENDER	DIM_CUSTOMER	A6V		Customer ...	Yes
INCOME					
REVENUE	FACT_SALES	D20.2C			Yes
GROSS_PROFIT	FACT_SALES	D20.2C	REVENUE...		No

   Note: If needed, fields may be duplicated by placing them in multiple folders, but any given folder may contain a field only once.

7. Select the *Save* or *Save As* icon to save the Business View as a Business View Master File.

   If you use the Save option, the Business View will be saved inside the current Master File and the Business View will be visible when the Master File is opened in reporting tools.

   Note: The Business View Master File may be saved in a different application than the main files.

8. Select *Close* from the File menu to close to the Synonym Editor.

Reference: Usage Notes for Business Views

- ❑ When creating a Business View using a referenced Master File:
  - ❑ The detailed information about fields, such as USAGE and ACTUAL formats or indexes remain in the referenced Master File.
  - ❑ All information about Cluster Master Files remain in the referenced Master File.
  - ❑ DBA attributes specified in the referenced Master File are respected by the Business Views

- When a Master File contains more than one field with the same name, as can occur when files are joined, the BELONGS_TO_SEGMENT attribute identifies which instance of the field name is being referenced in the Business View.
- Folders can be empty for organizational purposes. For example, Region can have empty folders called North, South, East, and West.
- You can issue an SQL SELECT command against a Business View. However, a Direct SQL Passthru request is not supported against a Business View.
- Business Views support alternate file views and fully qualified field names.
- The SEG. operator against a Business View folder displays all of the fields in that folder, not all of the fields in the real segment.
- Requests against a Business View cannot reference any fields or segments not in the Business View.
- All HOLD formats are supported against a Business View.
- All adapterssupport retrieval requests against a Business View.
- Business Views are not supported with data source maintenance commands, such as Maintain or REBUILD.
- The referenced Master File or the Master File currently being used to create the Business View, may contain Defines, Computes, Filters, and other fields for use in the Business View.

Reference: Using a Business View Master File

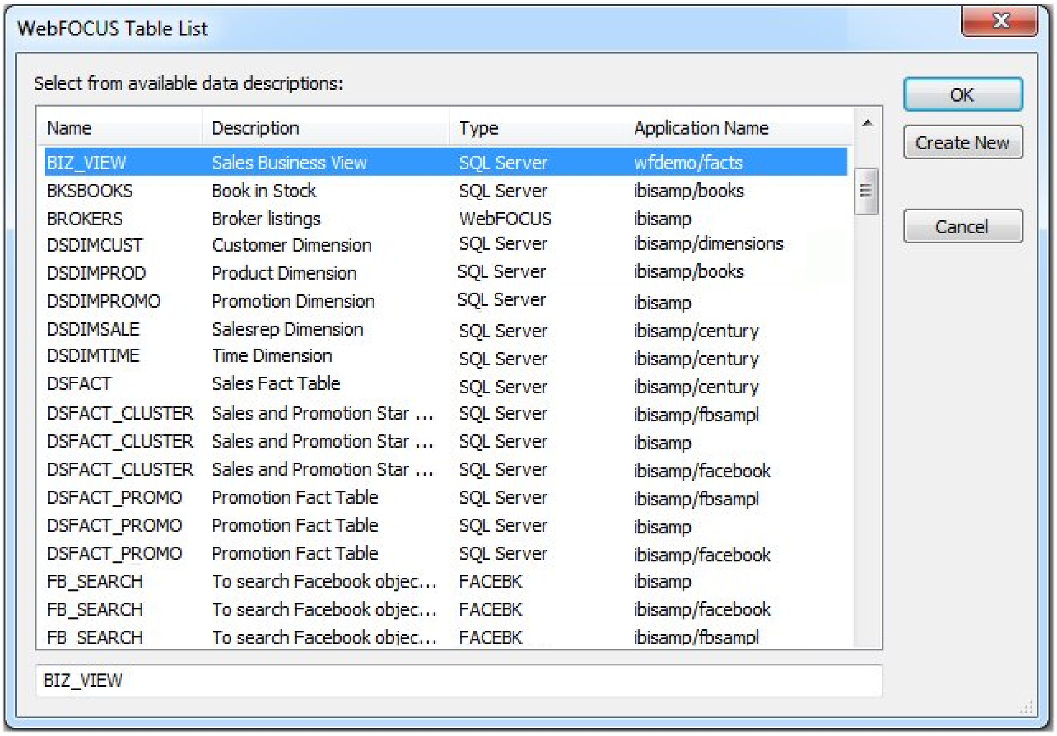
When you use the Business View Master File with DB2 Web Query tools and reports, the field formats, descriptions, and titles will be retrieved from the original Master File, unless they are customized and a title and description is available through the Business View.

You may access the Business View Master Files from all development areas of Developer Workbench.

In the Data Servers or Repository area:

The Table List shows all available Master Files, including the Business Views that are available. The Description column shows a description from the Master File or the Business View file.

The following image is an example of a Business View Master File in the WebFOCUS Table List, that appears when creating a report in the Data Servers area.


WebFOCUS Table List

Select from available data descriptions:

Name	Description	Type	Application Name
BIZ_VIEW	Sales Business View	SQL Server	wfdemo/facts
BKSBOOKS	Book in Stock	SQL Server	ibisamp/books
BROKERS	Broker listings	WebFOCUS	ibisamp
DSDIMCUST	Customer Dimension	SQL Server	ibisamp/dimensions
DSDIMPROD	Product Dimension	SQL Server	ibisamp/books
DSDIMPROMO	Promotion Dimension	SQL Server	ibisamp
DSDIMSALE	Salesrep Dimension	SQL Server	ibisamp/century
DSDIMTIME	Time Dimension	SQL Server	ibisamp/century
DSFACT	Sales Fact Table	SQL Server	ibisamp/century
DSFACT_CLUSTER	Sales and Promotion Star ...	SQL Server	ibisamp/fbsampl
DSFACT_CLUSTER	Sales and Promotion Star ...	SQL Server	ibisamp
DSFACT_CLUSTER	Sales and Promotion Star ...	SQL Server	ibisamp/facebook
DSFACT_PROMO	Promotion Fact Table	SQL Server	ibisamp/fbsampl
DSFACT_PROMO	Promotion Fact Table	SQL Server	ibisamp
DSFACT_PROMO	Promotion Fact Table	SQL Server	ibisamp/facebook
FB_SEARCH	To search Facebook objec...	FACEBK	ibisamp
FB_SEARCH	To search Facebook objec...	FACEBK	ibisamp/facebook
FB_SEARCH	To search Facebook objec...	FACEBK	ibisamp/fbsampl

BIZ_VIEW

This is also available in the Projects area, however, the Table List does not show remarks in the Projects area.

In InfoAssist:

When using a Business View Master File in InfoAssist, only fields from the Business View are shown in the Fields list.

The following image is an example of a Business View Master File in InfoAssist.

Adding Virtual Columns (DEFINE) in a Synonym

How to:

Create a Virtual Column in a Synonym

You may create a DEFINE field as a custom field in the Synonym Editor. A custom field can be used in a request as though it is a real data source field. Virtual columns (DEFINE fields) are available when the data source is used for reporting.

A virtual column can contain an expression, a constant, or a column name.

❑ If the virtual column is a complex expression, you can create the expression with the Virtual Column Calculator or just type it into the Expression field.

❑ If the virtual column is a simple expression, such as a constant value, you can type the value in the Expression field.

Virtual columns are designated by the following icon:

After creating a virtual column, you can test it by right-clicking the synonym and selecting *Sample Data*. Sample data appears in a separate dialog box.

Procedure: How to Create a Virtual Column in a Synonym

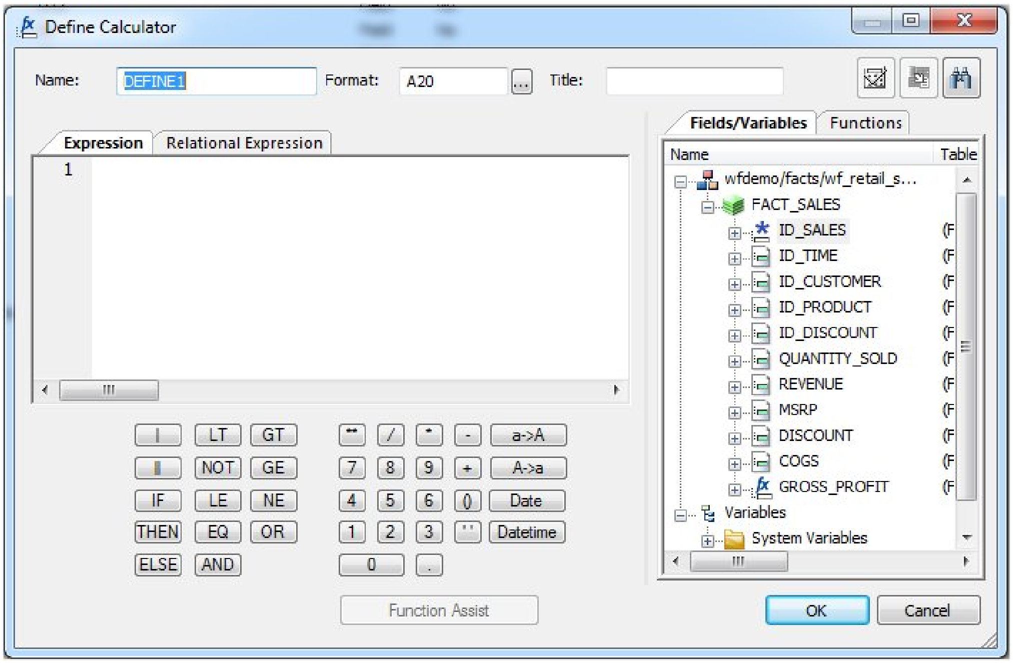
To create a virtual column in a synonym:

1. From the Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

The Synonym Editor opens.

2. Right-click a segment (or column), select *Insert*, and click *Define*.

The Define Calculator opens, as shown in the following image.



3. From the Define Calculator, type a name for the column in the Name input field, or use the default define name.

4. You may enter a descriptive title for the virtual column (DEFINE) in the Title input field.

   Tip: From the Synonym Editor, click the browse (...) button at the right of the TITLE and DESCRIPTION value fields to specify multiple language titles.

5. Use the Expression tab and the calculator buttons to build the expression for the virtual column (DEFINE).

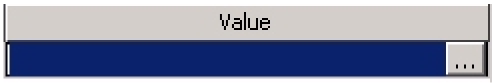
   or

   Use the Relational Expression tab to build the expression.

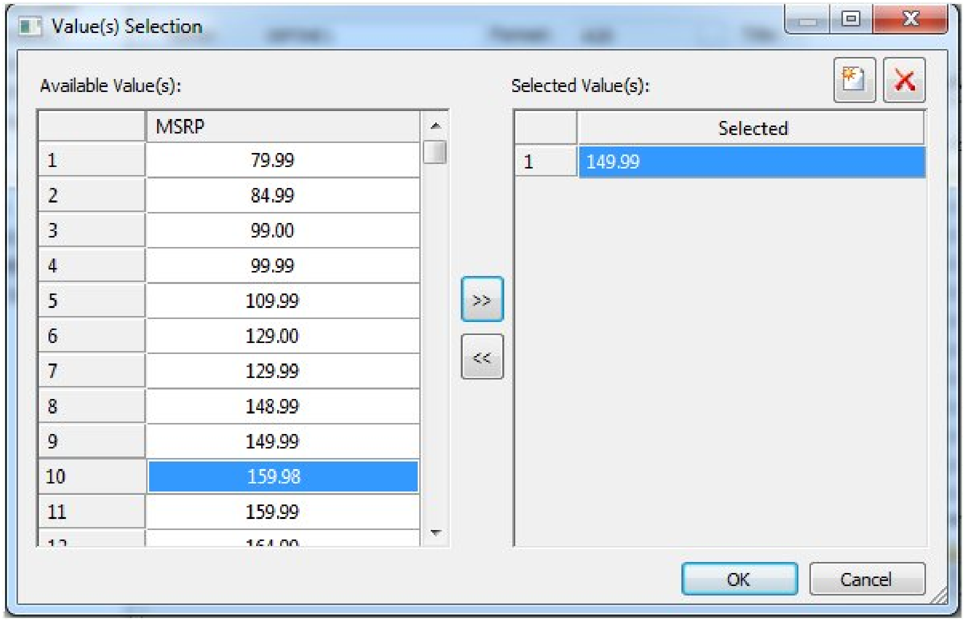
   a. From the Relational Expression tab, click the *Add New Row* button and use the drop-down menus to select the filter Column, Relation, and Type.

      Note: Parameters are not supported.

b. Click the browse (...) button at the right of the Value input field.



The Value Selection dialog box opens, as shown in the following image.



Available Value(s):

	MSRP
1	79.99
2	84.99
3	99.00
4	99.99
5	109.99
6	129.00
7	129.99
8	148.99
9	149.99
10	159.98
11	159.99

Selected Value(s):

	Selected
1	149.99

c. Select from the available values and use the arrows to add or remove values.

d. Click *OK* to close the Value Selection dialog box and return to the Relational Expression tab.

The selection is added to the value field.

e. To add another expression double-click a column or variable from the Columns/Variables tab on the right side of the calculator.

The row is added to the Relational Expression tab, where you can create the expression.

f. To delete an expression, select the number or field column of the expression and click the *Delete* button, or right-click it and select *Delete selected row(s)*. The expression is removed from the Relational Expression tab.

6. You may click the Check expression and Sample Data buttons, located on the top right of the calculator, to verify that the expression is valid and to view sample data for the filter.
7. Click *OK* to close the calculator and return to the Synonym Editor.

   Note: You can edit the Define, Title, or Expression directly from the Properties section of the Synonym Editor or you can click the browse (...) button located at the right of the EXPRESSION value field to relaunch the calculator.

8. If no columns from the synonym are used in the expression or have been defined, you can use the WITH option to identify the logical home of the defined calculation. You can also use the WITH option to move the logical home for the virtual column to a lower segment than it would otherwise be assigned (for example, to count instances in a lower segment).

   Tip: You can click and drag the DEFINE field and move it to a different segment in the Field View tab, which also changes the segment association.

9. Specify the Missing Data options for columns that allow null data. You can allow all missing data.
10. Click *Save* from the File menu to save the synonym.
11. To close the Synonym Editor, select *Close* from the File menu or click the control button in the upper-right corner.

For more information about expressions and virtual column (DEFINE) attributes, see *Defining Attributes and Creating Expressions for Custom Fields* on page 165.

Creating Filters in a Synonym

How to:

Create Filters in a Synonym

Filters are created in the Master File through the Synonym Editor and can be used in a Business View file or in reporting tools. You can also use filters to perform other data checking and validation, and sort data based on the conditions that you create.

Filters are created under a specific segment and, by default, they have association with the selected segment. Filters can also be created without segment association.

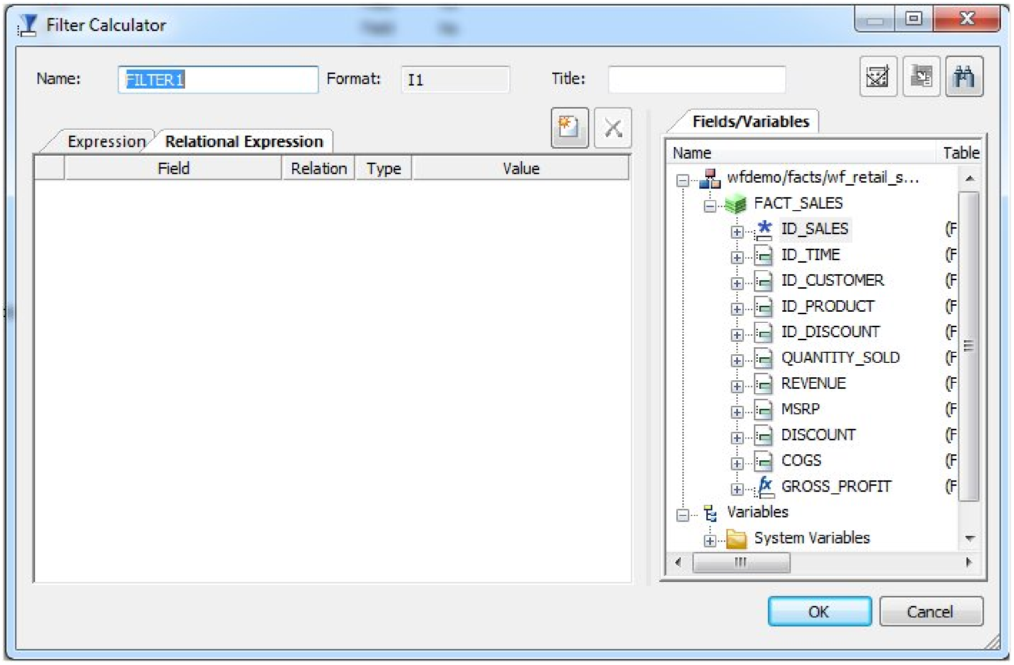
Procedure: How to Create Filters in a Synonym

1. From the Data Servers area, double-click a synonym from the Master Files folder, or right-click the synonym and select *Edit in Synonym Editor*.

   The Synonym Editor opens.

2. Right-click a segment or field, select *Insert*, and click *Filter*.

   The Filter Calculator opens, as shown in the following image.



3. From the Filter Calculator, type a name for the filter in the Name input field or use the default filter name.

   Note: It is recommended that filters have a descriptive name to help identify the filter action during reporting.

4. The Format field shows a default value of I1.

   Note: The Format field cannot be changed. Values for filters return 0 for false and 1 for true.

5. You may enter a descriptive title for the filter in the Title input field.

   Tip: From the Properties Pane in the Synonym Editor, click the browse (...) button at the right of the TITLE and DESCRIPTION value fields to specify multiple language titles.

6. Use the Expression tab and the calculator buttons to build the expression for the filter.

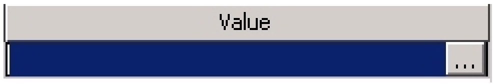
or

Use the Relational Expression tab to build the expression.

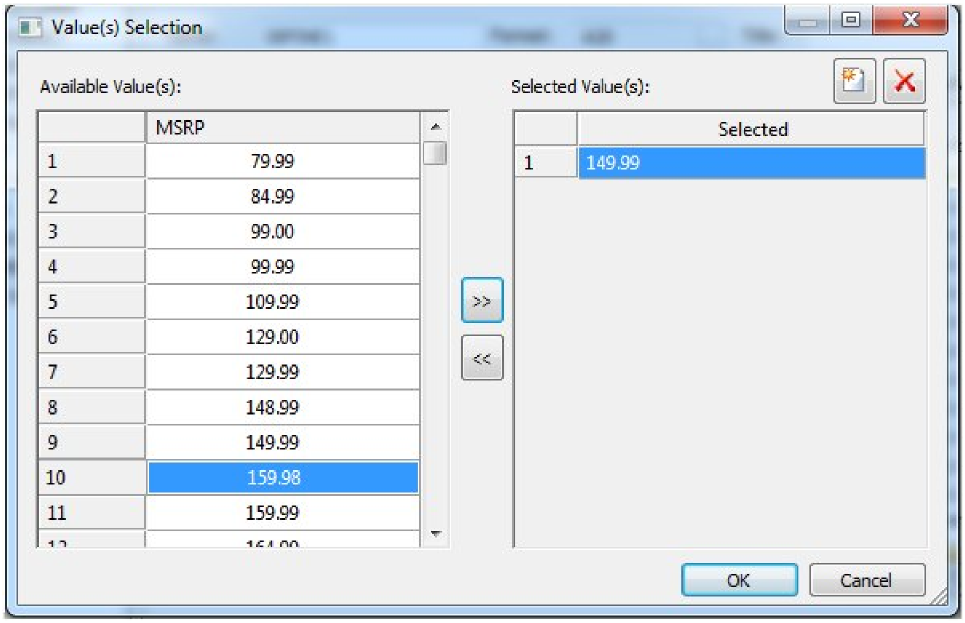
a. From the Relational Expression tab, click the *Add New Row* button and use the drop-down lists to select the filter Column, Relation, and Type.

Note: Parameters are not supported with Master File Filters.

b. Click the browse (...) button at the right of the Value input field.



The Value Selection dialog box opens, as shown in the following image.



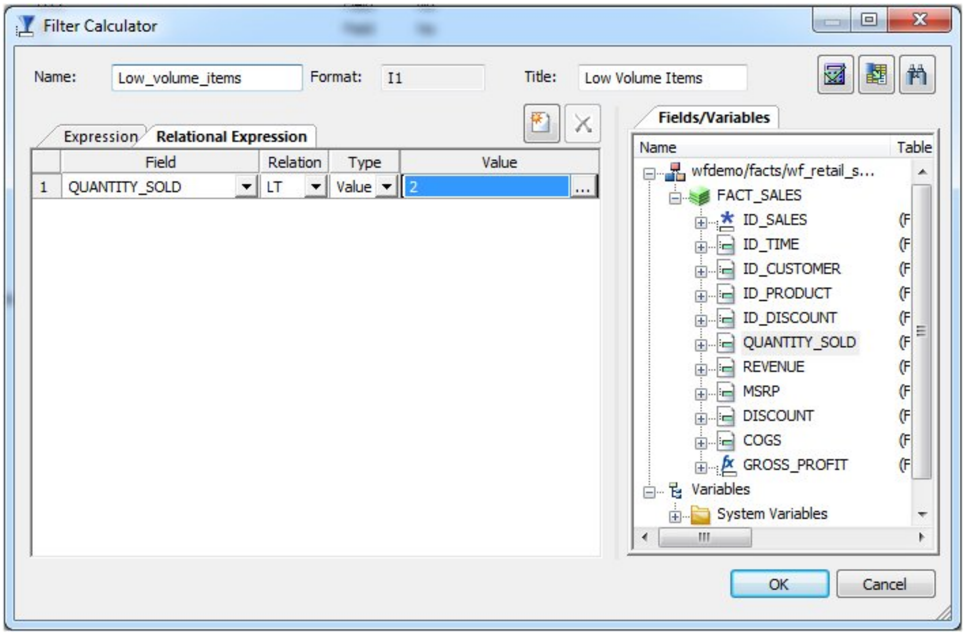
	MSRP
1	79.99
2	84.99
3	99.00
4	99.99
5	109.99
6	129.00
7	129.99
8	148.99
9	149.99
10	159.98
11	159.99

	Selected
1	149.99

c. Select from the available values and use the arrows to add or remove values.

d. Click *OK* to close the Value Selection dialog box and return to the Relational Expression tab.

The expression is added to the value field.



Filter Calculator

Name: Low_volume_items Format: I1 Title: Low Volume Items

	Field	Relation	Type	Value
1	QUANTITY_SOLD	LT	Value	2

e. To add another filter, double-click a column or variable from the Fields/Variables section on the right side of the Filter Calculator.

The filter is added to the Relational Expression tab where you can add the expression value.

f. To delete an expression, select the number or field column of the expression and click the *Delete* button, or right-click it and select *Delete selected row(s)*.

The expression is removed from the Relational Expression tab.

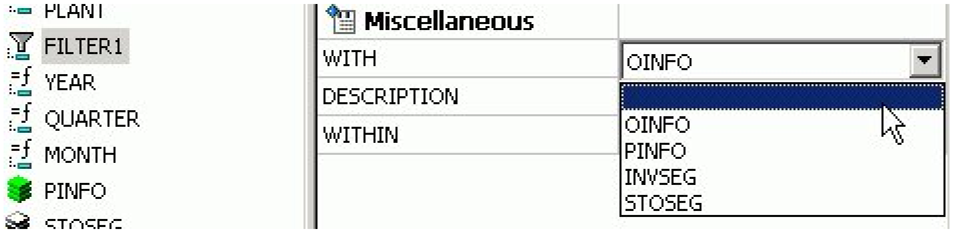
7. You may click the Check expression and Sample Data buttons, located on the top right of the Filter Calculator to verify that the expression is valid and to view sample data for the filter.

8. Click *OK* to close the Filter Calculator and return to the Synonym Editor.

Note: To edit the Filter, Title, or Expression, you may do so directly from the Properties Pane in the Synonym Editor or you may click the browse (...) button at the right of the EXPRESSION value field to relaunch the Filter Calculator.

9. To create a Filter without segment association, use the WITH drop-down list to select a blank segment.

   Filters are created under a specific segment and by default they have association with the selected segment. Filters can also be created without segment association.



   Note: If you are using a field that appears in multiple segments, the WITH segment associated should be the lowest level segment or it should be left empty to prevent errors.

   In addition, if no fields from the synonym are used in the expression or have not been computed, you can use the WITH option to identify the logical home of the filter calculation. You can also use the WITH option to move the logical home for the filter field to a lower segment than it would otherwise be assigned (for example, to count instances in a lower segment).

10. Click *Save* from the File menu to save the synonym.

    The filter is saved as part of the synonym.

11. To close the Synonym Editor, select *Close* from the File menu or click the control button in the upper-right corner.

Adding Computed Fields (COMPUTE) in a Synonym

How to:

Create a Computed Field in the Synonym Editor

You may create a Computed field as a custom field in the Synonym Editor. The procedure for adding a custom field is similar to the procedure used to add a Defined field. The Computed field is identified as the Master File Computed field and is differentiated from the Defined fields and the other Computed fields.

Procedure: How to Create a Computed Field in the Synonym Editor

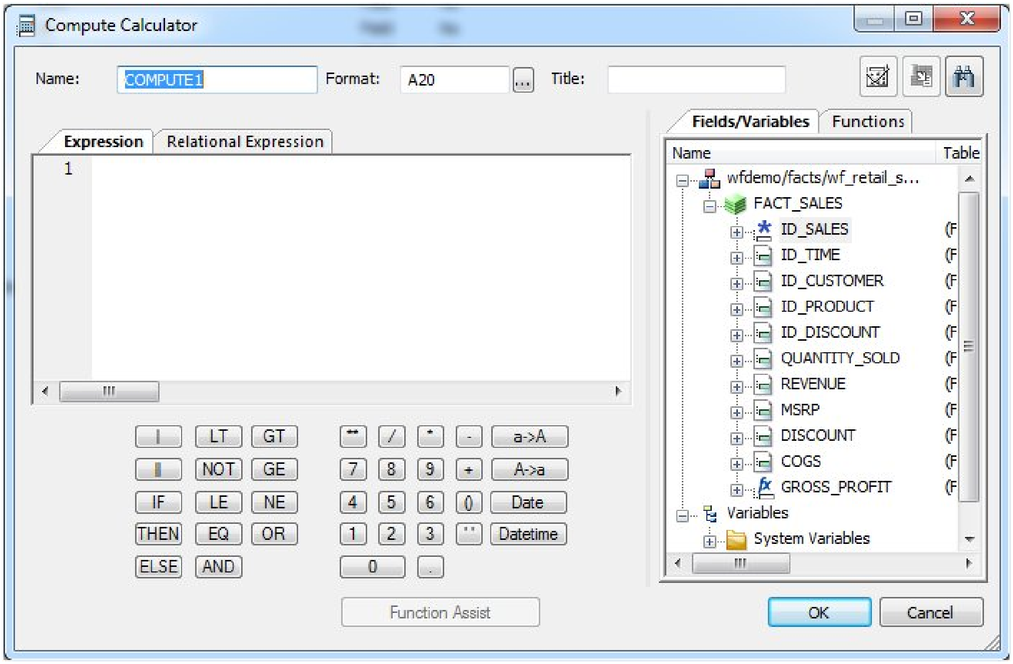
To create a computed column in a synonym

1. From the Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   The Synonym Editor opens.

2. Right-click a segment (or column), select *Insert*, and click *Compute*.

   The Compute Calculator opens.



3. From the Compute Calculator, type a name for the column in the Column input field or use the default compute name.

4. You may enter a descriptive title for the computed field (COMPUTE) in the Title input field.

   Tip: From the Synonym Editor, click the browse (...) button at the right of the TITLE and DESCRIPTION value fields to specify multiple language titles.

5. Use the Expression tab and the calculator buttons to build the expression for the computed field (COMPUTE).

   or

Use the Relational Expression tab to build the expression.

a. From the Relational Expression tab, click the *Add New Row* button and use the drop-down lists to select the filter Column, Relation, and Type.

   Note: Parameters are not supported.

b. Click the browse (...) button at the right of the Value input field.

c. Select from the available values and use the arrows to add or remove values.

d. Click *OK* to close the Value Selection dialog box and return to the Relational Expression tab.

   The expression is added to the value field.

e. To add another filter, double-click a column or variable from the Columns/Variables tab on the right side of the Compute Calculator.

   The filter is added to the Relational Expression tab, where you can add the expression value.

f. To delete an expression, select the number or field column of the expression and click the *Delete* button, or right-click it and select *Delete selected row(s)*. The expression is removed from the Relational Expression tab.

6. You may click the Check expression and Sample Data buttons, located on the top right of the Compute Calculator, to verify that the expression is valid and to view sample data for the filter.

7. Click *OK* to close the Compute Calculator and return to the Synonym Editor.

   Note: To edit the Compute, Title, or Expression, you may do so directly from the Properties section of the Synonym Editor or you may click the browse (...) button at the right of the EXPRESSION value field to relaunch the Compute Calculator.

8. Specify the Missing Data options for columns that allow null data. You can allow all missing data.

9. Click *Save* from the File menu to save the synonym.

10. To close the Synonym Editor, select *Close* from the File menu or click the control button in the upper-right corner.

For more information about expressions and COMPUTE attributes, see *Defining Attributes and Creating Expressions for Custom Fields* on page 165.

Storing the Number of Repetitions of a Repeating Field in a Virtual Field

How to:

Specify an OCCURS Segment Using a Virtual Field

Reference:

Usage Notes for Using a Virtual Field With OCCURS

The OCCURS attribute in a Master File describes repeating fields or groups of fields in a data source. The repeating group of fields is described as a descendent segment in the Master File, and the OCCURS attribute for that segment specifies how to determine the number of repetitions.

The number of repetitions does not have to be the same for every record instance. Sometimes, the number of repetitions can be derived from a field in the data source. In that case, you can create a virtual field in the Master File that indicates the number of repetitions for each record and use that virtual field as the value of the OCCURS attribute.

Syntax: How to Specify an OCCURS Segment Using a Virtual Field

```
SEGNAME = parent, SEGTYPE = segtype,$
    .
    .
    .
  DEFINE definefield/In = expression;
  SEGNAME = osegname, SEGTYPE=S0, PARENT = parent,
     OCCURS = definefield ,$
  FIELDNAME = rfield, ALIAS = ralias,
     USAGE = rufmt, ACTUAL = rafmt,$
    .
    .
    .
 [FIELDNAME = orderfield, ALIAS = ORDER,
     USAGE = In, ACTUAL = I4,$]
```

where:

parent

Is the name of the parent segment.

segtype

Is the SEGTYPE of the parent segment.

`definefield`

Is the virtual field that indicates the number of repetitions of the repeating field or group of fields. This field must be defined in a segment that is an ancestor of the segment containing the repeating fields.

`n`

Is the format of the virtual field that describes the number of repetitions. It must be an integer format.

`expression`

Is a valid expression that derives the number of repetitions for each record instance.

`osegname`

Is the name of the descendent OCCURS segment.

`rfield`

Is the name of a repeating field in the OCCURS segment.

`ralias`

Is the alias of a repeating field in the OCCURS segment.

`rufmt`

Is the display format for a repeating field in the OCCURS segment.

`rafmt`

Is the actual format for a repeating field in the OCCURS segment.

`orderfield`

Is the name of an internal counter field that you can specify as the last field in the OCCURS segment. The ORDER field associates a sequence number with each occurrence and is useful when the order of the repeating data is significant. For example, the values may represent monthly or quarterly data, but the record itself may not explicitly specify the month or quarter to which the data applies. The USAGE format must be integer and the ACTUAL format is I4.

Reference: Usage Notes for Using a Virtual Field With OCCURS

The virtual field used as the OCCURS value cannot be redefined inside or outside of the Master File.

Example: Using a Virtual Field With an OCCURS Segment

The following request against the EMPLOYEE data source creates a fixed-format sequential file with a repeating field. The request:

- Counts the number of FICA deductions for each employee.

- Creates a calculated field that contains the length of all FICA deduction fields for each employee.
- Creates a HOLD file in which each record contains the calculated length of the deduction fields for the employee, the identifying information for the employee, and all FICA deductions for the employee.

Note that the number of deductions will vary for each employee. The part of the record that contains the deductions will constitute the OCCURS segment. The number of repetitions will have to be derived from the length field created in the TABLE request.

The procedure to create the file with the repeating deduction field follows:

```
DEFINE FILE EMPLOYEE
  CTR/I5 WITH DED_AMT = IF EMP_ID NE LAST EMP_ID THEN 1 ELSE LAST CTR + 1;
END
TABLE FILE EMPLOYEE
  SUM CNT.DED_AMT NOPRINT EMP_ID LAST_NAME FIRST_NAME CURR_SAL
  COMPUTE DEDLEN/I5 = 12 * CNT.DED_AMT;
  BY EMP_ID NOPRINT
  SUM DED_AMT
  BY EMP_ID NOPRINT
  ACROSS CTR NOPRINT
  WHERE DED_CODE EQ 'FICA'
  ON TABLE SET HOLDLIST PRINTONLY
  ON TABLE HOLD AS OCCURS1 FORMAT ALPHA
END
```

The OCCURS1 file has one record per employee with a variable number of DED_AMT fields. The total length of the number of actual instances of DED_AMT is stored in the field named DEDLEN. The Master File generated by the HOLD command lists 10 DED_AMT fields:

```
FILENAME=OCCURS1 , SUFFIX=FIX     , $
  SEGMENT=OCCURS1, SEGTYPE=S0, $
    FIELDNAME=EMP_ID, ALIAS=E01, USAGE=A9, ACTUAL=A09, $
    FIELDNAME=LAST_NAME, ALIAS=E02, USAGE=A15, ACTUAL=A15, $
    FIELDNAME=FIRST_NAME, ALIAS=E03, USAGE=A10, ACTUAL=A10, $
    FIELDNAME=CURR_SAL, ALIAS=E04, USAGE=D12.2M, ACTUAL=A12, $
    FIELDNAME=DEDLEN, ALIAS=E05, USAGE=I5, ACTUAL=A05, $
    FIELDNAME=DED_AMT, ALIAS=E06, USAGE=D12.2M, ACTUAL=A12, $
    FIELDNAME=DED_AMT, ALIAS=E07, USAGE=D12.2M, ACTUAL=A12, $
    FIELDNAME=DED_AMT, ALIAS=E08, USAGE=D12.2M, ACTUAL=A12, $
    FIELDNAME=DED_AMT, ALIAS=E09, USAGE=D12.2M, ACTUAL=A12, $
    FIELDNAME=DED_AMT, ALIAS=E10, USAGE=D12.2M, ACTUAL=A12, $
    FIELDNAME=DED_AMT, ALIAS=E11, USAGE=D12.2M, ACTUAL=A12, $
    FIELDNAME=DED_AMT, ALIAS=E12, USAGE=D12.2M, ACTUAL=A12, $
    FIELDNAME=DED_AMT, ALIAS=E13, USAGE=D12.2M, ACTUAL=A12, $
    FIELDNAME=DED_AMT, ALIAS=E14, USAGE=D12.2M, ACTUAL=A12, $
    FIELDNAME=DED_AMT, ALIAS=E15, USAGE=D12.2M, ACTUAL=A12, $
```

You can edit the Master File to describe these repeating DED_AMT fields with an OCCURS segment. The DEFINE field named NUMOCC derives the number of occurrences from the DEDLEN field. The ORDER field is not actually in the file. It is an internal counter populated by DB2 Web Query:

```
FILENAME=OCCURS1 , SUFFIX=FIX     , $
  SEGMENT=OCCURS1, SEGTYPE=S0, $
    FIELDNAME=EMP_ID, ALIAS=E01, USAGE=A9, ACTUAL=A09, $
    FIELDNAME=LAST_NAME, ALIAS=E02, USAGE=A15, ACTUAL=A15, $
    FIELDNAME=FIRST_NAME, ALIAS=E03, USAGE=A10, ACTUAL=A10, $
    FIELDNAME=CURR_SAL, ALIAS=E04, USAGE=D12.2M, ACTUAL=A12, $
    FIELDNAME=DEDLEN, ALIAS=E05, USAGE=I5, ACTUAL=A05, $
    DEFINE NUMOCC/I2 = DEDLEN/12;,$
  SEGNAME=DEDUCTION, SEGTYPE=S0, PARENT=OCCURS1, OCCURS=NUMOCC,$
    FIELDNAME=DED_AMT, ALIAS=E06,   USAGE=D12.2M, ACTUAL=A12, $
    FIELDNAME=ORDER,   ALIAS=ORDER, USAGE=I2    , ACTUAL=I4 , $
```

The following request uses the ORDER field to select and print the first occurrence of the repeating field for each employee. Since every employee has at least one deduction, every employee is represented on the report output:

```
TABLE FILE OCCURS1
  PRINT NUMOCC LAST_NAME CURR_SAL DED_AMT
  WHERE ORDER EQ 1
END
```

The output is:

```
NUMOCC  LAST_NAME              CURR_SAL          DED_AMT
------  ---------              --------          -------
    10  STEVENS              $11,000.00           $64.17
     8  SMITH                $13,200.00          $100.10
     4  JONES                $18,480.00          $247.94
     8  SMITH                 $9,500.00           $60.96
     1  BANNING              $29,700.00          $519.75
     8  IRVING               $26,862.00          $626.78
     4  ROMANS               $21,120.00          $317.62
     1  MCCOY                $18,480.00          $161.70
     5  BLACKWOOD            $21,780.00          $444.67
     7  MCKNIGHT             $16,100.00          $187.88
     4  GREENSPAN             $9,000.00           $52.50
    10  CROSS                $27,062.00          $631.40
```

If you print the tenth occurrence of the repeating field, only two employees are displayed on the report output:

```
TABLE FILE OCCURS1
  PRINT NUMOCC LAST_NAME CURR_SAL DED_AMT
  WHERE ORDER EQ 10
END
```

The output is:

```
NUMOCC  LAST_NAME                 CURR_SAL          DED_AMT
------  ---------                 --------          -------
    10  STEVENS                 $11,000.00           $58.33
    10  CROSS                   $27,062.00          $526.20
```

Defining Attributes and Creating Expressions for Custom Fields

Reference:

Custom Field Attributes

Calculators for Custom Fields

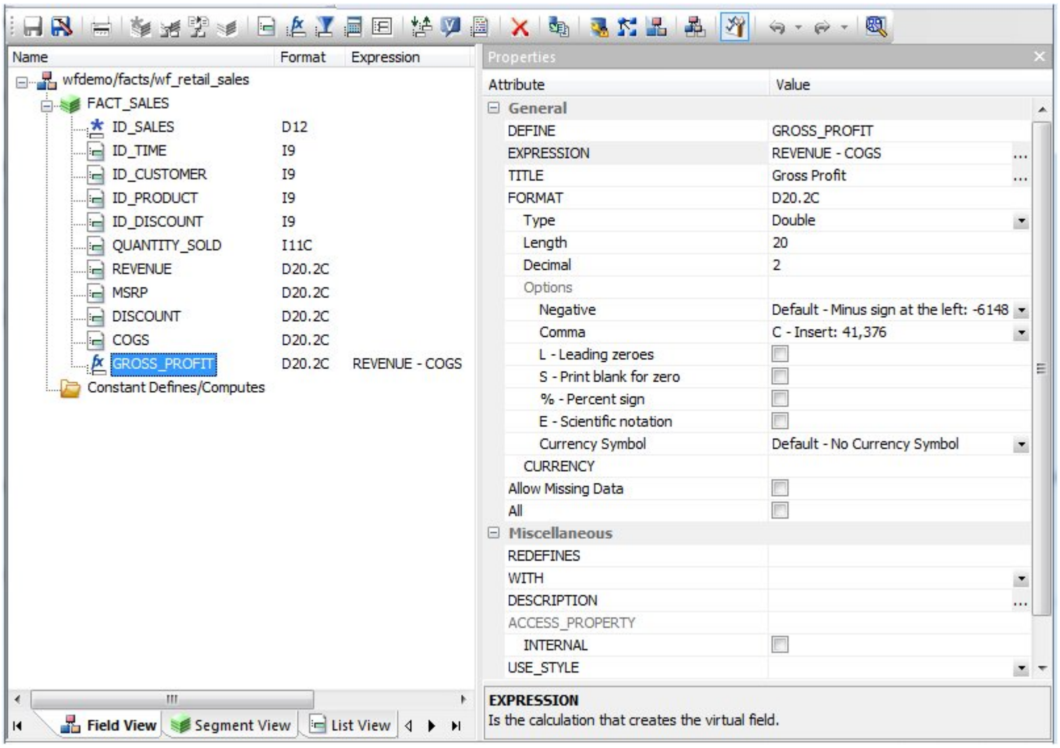
A custom field is a field whose value is not stored in the data source but can be calculated from the data that is there. You can create a custom field in your synonym by adding a virtual column (DEFINE), Master File filter (FILTER), and a Computed Field (COMPUTE). The fields are available whenever you access the corresponding data source in a reporting tool.

You can define attribute values and create expressions for custom fields by using the Synonym Editor.

Reference: Custom Field Attributes

The following attributes may be available for custom fields (DEFINE, FILTER, and COMPUTE) in the Synonym Editor.

Note: The attributes available depend on the type of synonym and the type of custom field selected. The following image is an example of an SQL data source with a virtual column (DEFINE) selected.



Name	Format	Expression
wfdemo/facts/wf_retail_sales		
FACT_SALES		
ID_SALES	D12	
ID_TIME	I9	
ID_CUSTOMER	I9	
ID_PRODUCT	I9	
ID_DISCOUNT	I9	
QUANTITY_SOLD	I11C	
REVENUE	D20.2C	
MSRP	D20.2C	
DISCOUNT	D20.2C	
COGS	D20.2C	
GROSS_PROFIT	D20.2C	REVENUE - COGS
Constant Defines/Computes		

Attribute	Value
General	
DEFINE	GROSS_PROFIT
EXPRESSION	REVENUE - COGS
TITLE	Gross Profit
FORMAT	D20.2C
Type	Double
Length	20
Decimal	2
Options	
Negative	Default - Minus sign at the left: -6148
Comma	C - Insert: 41,376
L - Leading zeroes	☐
S - Print blank for zero	☐
% - Percent sign	☐
E - Scientific notation	☐
Currency Symbol	Default - No Currency Symbol
CURRENCY	
Allow Missing Data	☐
All	☐
Miscellaneous	
REDEFINES	
WITH	
DESCRIPTION	
ACCESS_PROPERTY	
INTERNAL	☐
USE_STYLE	

EXPRESSION
Is the calculation that creates the virtual field.

Custom fields (DEFINE, FILTER, COMPUTE) typically have the following attributes:

General

DEFINE

Is the name of the virtual column.

Note: This attribute only appears when a virtual column (DEFINE) is selected.

FILTER

Is the name of the Master File Filter field.

Note: This attribute only appears when a virtual filter field is selected.

COMPUTE

Is the name of the computed field.

Note: This attribute only appears when a virtual computed field is selected.

EXPRESSION

Is the expression that creates the virtual column.

TITLE

Supplies a title to replace the column name that is normally used in reports and enables you to specify multiple language titles for the virtual column.

FORMAT

Describes the data type and format for the virtual column.

Note: This attribute only appears for DEFINE and COMPUTE custom fields.

Allow Missing Data

Allows missing data. If not, the transaction value is supplied.

Note: This attribute only appears for DEFINE and COMPUTE custom fields.

All

Allows all missing data. If not, the transaction value is supplied.

Note: This attribute only appears for DEFINE and COMPUTE custom fields.

Miscellaneous

WITH

If no columns from the synonym are used in the expression or have been defined, you can use the WITH option to identify the logical home of the defined calculation. You can also use the WITH option to move the logical home for the virtual column to a lower segment than it would otherwise be assigned (for example, to count instances in a lower segment).

Note: This attribute only appears for DEFINE and FILTER custom fields.

DESCRIPTION

Contains a description or comments about the virtual column.

WITHIN

Contains the name of a field to be included in a dimension.

These WITHIN statements are added to the synonym through the Dimension Builder to OLAP-enable relational tables. This enables you to perform OLAP analysis using the OLAP Control Panel.

Note: This attribute only appears for DEFINE and FILTER custom fields.

SCD Type

Sets slowly changing dimensions. This option is only available for existing relational targets.

A **surrogate key** is the first column in the table and has an SCD type of blank. Other columns with a blank SCD type have no SCD processing done to them.

Logical Key Field is the database key.

Activation Flag indicates that the row is current.

Begin Date/End Date indicates the date range for the row values. A null end date indicates that the row is current.

Type I (overwriting history) designates columns whose database values are overwritten with new values.

Type II (preserving history) designates columns whose database rows are flagged as inactive or assigned an end date. New rows are inserted with the new values.

blank (non-key columns) indicates that database values are not changed.

USE_STYLE

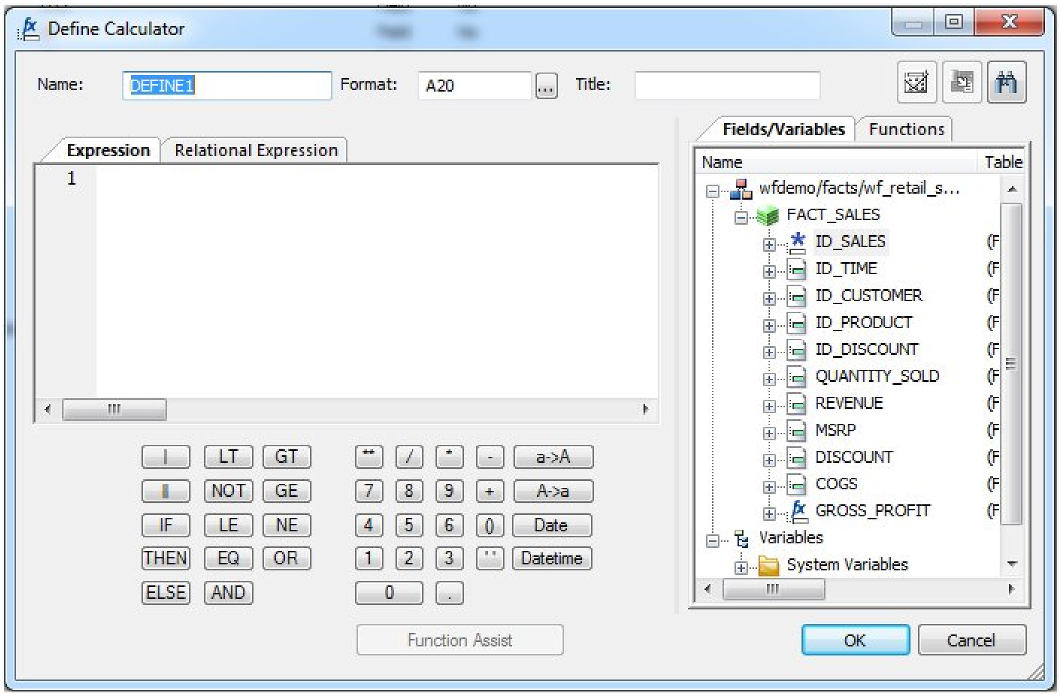
Is the name of the stylesheet applied to a field.

Note: The attributes available depend on the type of synonym.

Reference: Calculators for Custom Fields

To launch the Define Calculator, Filter Calculator, or Compute Calculator, click the browse (...) button at the right of the EXPRESSION value field in the Properties section of the Synonym Editor.

The selected calculator opens, depending on the type of custom field that you are creating.



The calculator has the following fields and options:

Name

Is the name of the object being created (virtual field (DEFINE), filter, computed field).

Format

Is the field format.

Expression tab

Location for typing an expression. You can add data source fields from the Columns/Variables tab, functions from the Functions tab, and numbers and operators from the calculator as you type.

Relational Expression tab

Displays the expression building window from which you can add and delete columns, choose the relation and type, and select values for your filter.

Fields/Variables tab

Displays a hierarchical list of available source columns and System Variable folders that you can use in creating an expression.

Functions tab

A function is a program that returns a value. This tab lists the built-in functions that you can use to derive the value of a temporary field.

Function Assist button

Enables you to specify parameters for the function through a dialog box when creating or editing a transformation.

Calculator buttons

Enables you to insert numbers and operators.

The following operators are available:

| (single concatenation bar)

Concatenates two values, retaining any trailing blanks after the first one. For example, if FIRST_NAME and LAST_NAME were both in A15 format, the expression

```
FULL_NAME = FIRST_NAME | LAST_NAME
```

would produce a column like the following:

```
MICHAEL      SMITHSONJ
ANE         JONES
.
.
.
```

|| (double concatenation bar)

Concatenates two values, suppressing any trailing blanks in the first. For example, to construct the full name and insert a comma (,), the syntax

```
FULL_NAME = LAST_NAME || (', ' | FIRST_NAME)
```

would produce a column like the following:

```
SMITHSON, MICHAEL
JONES, JANE
.
.
.
```

The concatenation in the parentheses is done first (preserving the blank space after the comma), and the result is then concatenated to LAST_NAME, suppressing the trailing blanks of LAST_NAME.

IF

Establishes a conditional test.

THEN

Specifies the action to perform if the result of a conditional test is TRUE.

ELSE

Specifies the action to perform if the result of a conditional test is FALSE.

LT

Returns the value TRUE if the value on the left is less than the value on the right.

NOT

Returns the value TRUE if the operand is false.

LE

Returns the value TRUE if the value on the left is less than or equal to the value on the right.

EQ

Returns the value TRUE if the value on the left is equal to the value on the right.

AND

Returns the value TRUE if both operands are true.

GT

Returns the value TRUE if the value on the left is greater than the value on the right.

GE

Returns the value TRUE if the value on the left is greater than or equal to the value on the right.

NE

Returns the value TRUE if the value on the left is not equal to the value on the right.

OR

Returns the value TRUE if either operand is true.

******

Raises a value to the specified power.

( )

Adds parentheses.

''

Inserts two single quotation marks. Enter alphanumeric test values between the quotation marks.

a->A

Converts selected text to uppercase.

A->a

Converts selected text to lowercase.

Check expression button

Verifies the validity of the expression.

Sample data button

Produces sample data for the expression.

Adding Group Fields in a Synonym

How to:

Add a Group Field to a Segment

Add a New Field to a Group Field

Add an Existing Field to a Group Field

Delete a Group Field From a Segment

Delete a Field From a Group Field

Reference:

Group Field Attributes

For data sources that support groups, you can assign a unique name to multiple fields to create a group field. A group field is created by two or more alphanumeric fields, physically next to each other. A group field provides an efficient means for grouping similar or logically connected fields that will be accessed as a single unit, but do not warrant a separate segment.

Note: In the Maintain environment, group fields are supported in a Master File as long as they are not group keys. That is, groups that are actual fields, as supported by VSAM. If you create a group field in the Maintain environment, the group will not be visible, only the fields that make up the group will be visible.

Procedure: How to Add a Group Field to a Segment

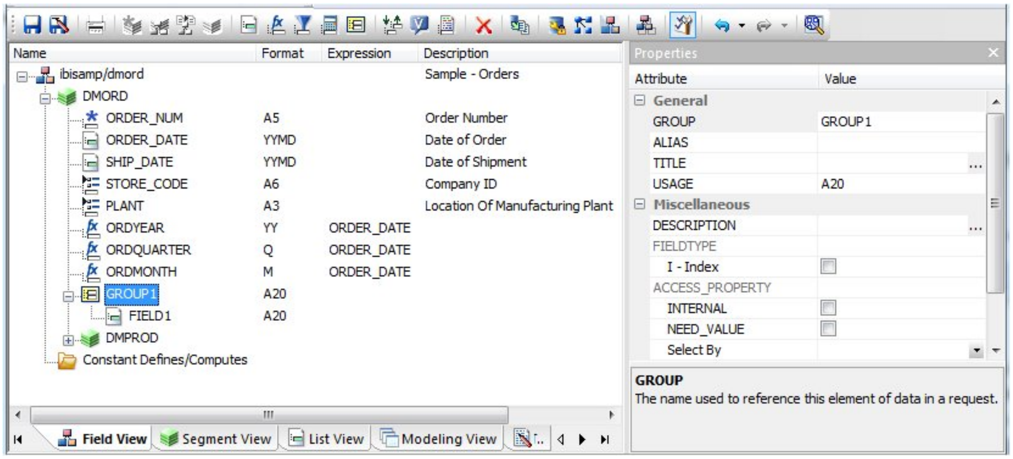
1. From the Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   The Synonym Editor opens.

2. Right-click a segment (or column), select *Insert*, and click *Group*.

   Note: The group option may not be available for some data sources.

   A group is added to the synonym and its attributes and values appear on the right.



Name	Format	Expression	Description
ibisamp/dmord			Sample - Orders
DMORD			
ORDER_NUM	A5		Order Number
ORDER_DATE	YYMD		Date of Order
SHIP_DATE	YYMD		Date of Shipment
STORE_CODE	A6		Company ID
PLANT	A3		Location Of Manufacturing Plant
ORDYEAR	YY	ORDER_DATE	
ORDQUARTER	Q	ORDER_DATE	
ORDMONTH	M	ORDER_DATE	
GROUP1	A20		
FIELD1	A20		
DMPROD			
Constant Defines/Computes			

Properties

Attribute	Value
General	
GROUP	GROUP1
ALIAS	
TITLE	...
USAGE	A20
Miscellaneous	
DESCRIPTION	...
FIELDTYPE	
I - Index	☐
ACCESS_PROPERTY	
INTERNAL	☐
NEED_VALUE	☐
Select By	

GROUP
The name used to reference this element of data in a request.

3. Type a name for the group in the *GROUP* field.

4. Select the USAGE *Type* value for the group. Specify the length, the decimal places (if applicable) and any display options for the group.

5. Select the *I - Index* check box if you want the group to be indexed.

6. Select the ACCESS_PROPERTY values to specify access options for the group data.

7. Optionally, you can specify the TITLE and DESCRIPTION display options.

8. Click *Save* from the File menu to save the synonym.

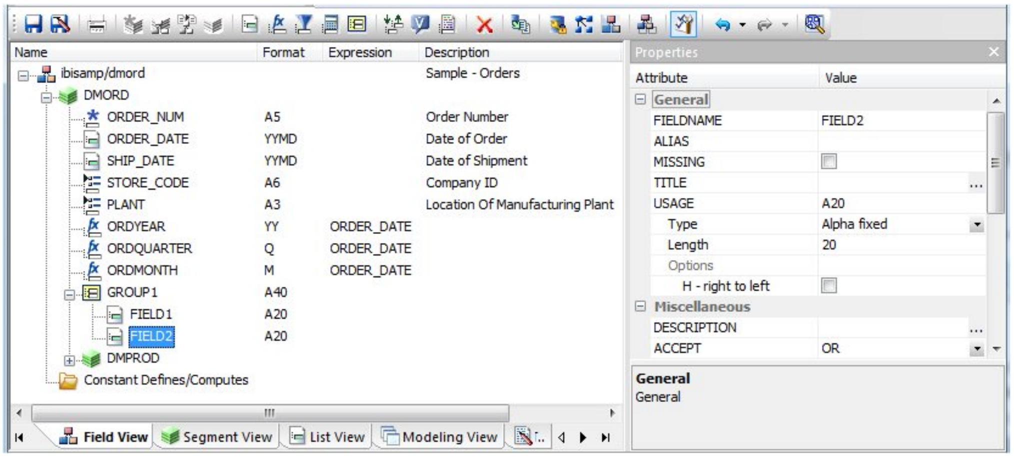
9. To close the Synonym Editor, select *Close* from the File menu or click the control button in the upper-right corner.

Note: When you add a group, a field is automatically added to the group.

Procedure: How to Add a New Field to a Group Field

1. Right-click the group field to which you want to add a new field, select *Insert*, and click *Field*.

   A field is added to the group and its attributes and values appear on the right, as shown in the following image.



Name	Format	Expression	Description
ibisamp/dmord			Sample - Orders
DMORD			
ORDER_NUM	A5		Order Number
ORDER_DATE	YYMD		Date of Order
SHIP_DATE	YYMD		Date of Shipment
STORE_CODE	A6		Company ID
PLANT	A3		Location Of Manufacturing Plant
ORDYEAR	YY	ORDER_DATE	
ORDQUARTER	Q	ORDER_DATE	
ORDMONTH	M	ORDER_DATE	
GROUP1	A40		
FIELD1	A20		
FIELD2	A20		
DMPROD			
Constant Defines/Computes			

Attribute	Value
General	
FIELDNAME	FIELD2
ALIAS	
MISSING	☐
TITLE	
USAGE	A20
Type	Alpha fixed
Length	20
Options	
H - right to left	☐
Miscellaneous	
DESCRIPTION	
ACCEPT	OR

General
General

2. Supply the required information for the group field. For more information about field attributes, see *Viewing and Editing Synonym Attributes* on page 100.

Procedure: How to Add an Existing Field to a Group Field

1. Click the field you want to add to the group field.

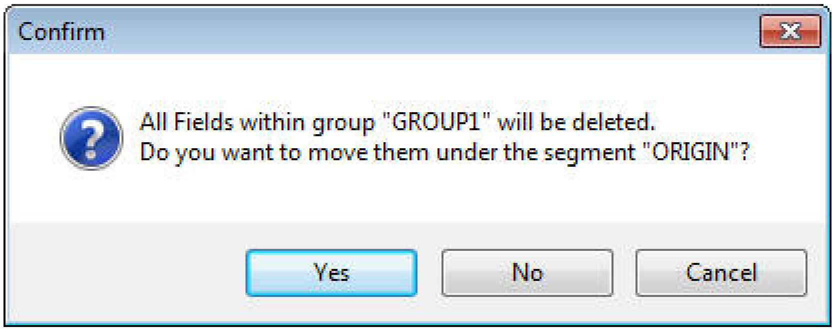
2. While holding the left mouse button down, drag the field and drop it on the group field name.

   The field is added to the group field.

Procedure: How to Delete a Group Field From a Segment

1. Right-click the group and select *Delete*.

   A confirmation appears stating that all columns (fields) within the group will be deleted.

   Confirm

   All Fields within group "GROUP1" will be deleted.
   Do you want to move them under the segment "ORIGIN"?

2. Click *Yes* to delete the group and move the fields (within the group) under the root segment of the synonym.

   Click *No* to delete the group and all of the fields within the group.

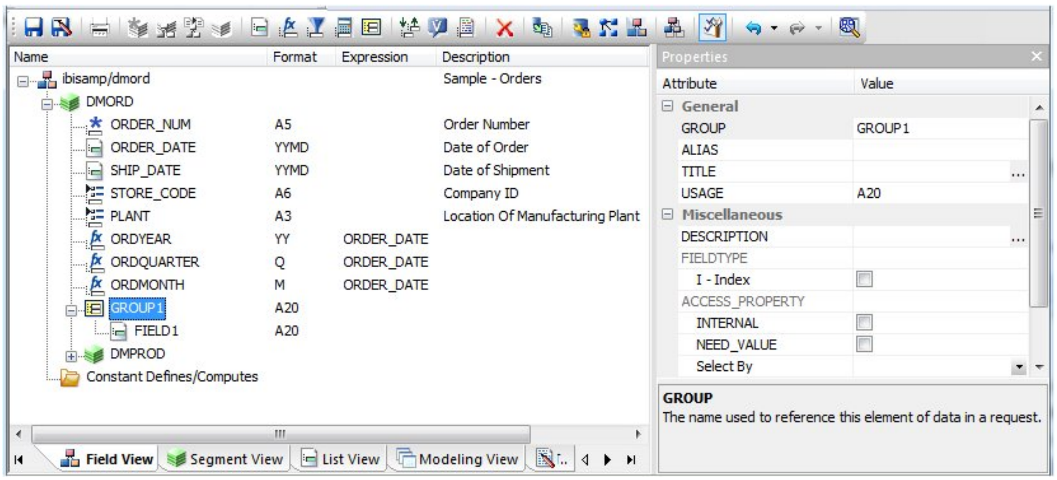
   Click *Cancel* to close the Confirm Delete dialog box and return to the Synonym Editor.

Procedure: How to Delete a Field From a Group Field

Right-click the field, then select *Delete*.

Reference: Group Field Attributes

Group fields in a synonym can have the following attributes:



Name	Format	Expression	Description
ibisamp/dmord			Sample - Orders
DMORD			
ORDER_NUM	A5		Order Number
ORDER_DATE	YYMD		Date of Order
SHIP_DATE	YYMD		Date of Shipment
STORE_CODE	A6		Company ID
PLANT	A3		Location Of Manufacturing Plant
ORDYEAR	YY	ORDER_DATE	
ORDQUARTER	Q	ORDER_DATE	
ORDMONTH	M	ORDER_DATE	
GROUP1	A20		
FIELD1	A20		
DMPROD			
Constant Defines/Computes			

Attribute	Value
General	
GROUP	GROUP1
ALIAS	
TITLE	
USAGE	A20
Miscellaneous	
DESCRIPTION	
FIELDTYPE	
I - Index	☐
ACCESS_PROPERTY	
INTERNAL	☐
NEED_VALUE	☐
Select By	

GROUP
The name used to reference this element of data in a request.

General

GROUP

Is the name of the group.

ALIAS

Assigns an alternative name for a group.

If you create a report, the group name appears as a column heading unless you have specified an alternate title for the group. Aliases cannot be used as column titles.

TITLE

Supplies a title to replace the group name that is normally used in reports and enables you to specify multiple language titles for the group.

USAGE

Contains the format for the group field. Since the group field is made by concatenating together several other fields, the Synonym Editor determines what this format needs to be. For example, if the group field has two alphanumeric fields in it, each 20 characters long (A20), then the group field must be alphanumeric and 40 characters long (A40). The group field is always alphanumeric, regardless of the fields that make it up.

Miscellaneous

DESCRIPTION

Contains a description or comments about the group. The description displays in Field lists and on the status bar.

Field descriptions also appear as bubble help in OLAP-enabled reports. If you do not include a description, bubble help shows the field name (column title).

FIELDTYPE

Identifies an indexed group. You can index the values of a field to enhance data retrieval performance. To do so, select the Index check box when you add a field and before you add the data. An index is an internally stored and maintained table of data values and locations that enhance the performance of data retrieval. A Master File can have several associated indexes, but the combined total of indices and segments cannot exceed 64.

Note: FIELDTYPE=R indicates a read-only column. This setting is useful for columns that are automatically assigned a value by the RDBMS.

Tip: You can turn on the index after adding data to a field, however, you will have to use the Rebuild Index option to create the index.

ACCESS_PROPERTY

Specifies access options for the column data.

INTERNAL defines a column that does not appear in sample data or in the list of available columns. Restricts the field from showing in any of the Field Lists in the reporting tools.

NEED_VALUE defines a column that requires a value to access the data. Indicates that a selection is needed in the report request (WHERE condition).

Select By defines a column by value, range, or multivalues:

- If Value is checked, only one value should be defined for selection in the report request.
- If Range is checked, a range selection should be defined in the report request.
- If Multivalues is checked, multiple values are allowed for selection in the report request.

USE_STYLE

Is the name of the stylesheet applied to a group.

Note: The attributes available depend on the type of synonym.

Applying Database Administrator Security

You can secure Master Files on a file-by-file basis. For each data source, security can be maintained at two different levels.

- **Database Administrator Level.** You specify the Database Administrator (DBA) password for the data source. The DBA has unlimited access to the Master File and data source and can set up or change security restrictions for individual users.
- **User Level.** You specify the DBA and user passwords for the data source. The user password represents a user who has access to that data source. When you specify a user password, you must also set at least the type of file access: read, write, read/write, or update. Security for each user can be further limited by restricting access to segments, fields, or field values. For more information, see *Restricting Access to Segments, Fields, Field Values, and Noprint* on page 183. Once a user password has been established, you can apply the same restrictions to multiple users. For more information, see *Applying Security Restrictions for Multiple Users* on page 186.

Note: You cannot specify a Database Administrator (DBA) password during the Create Synonym process. You must use the Synonym Editor.

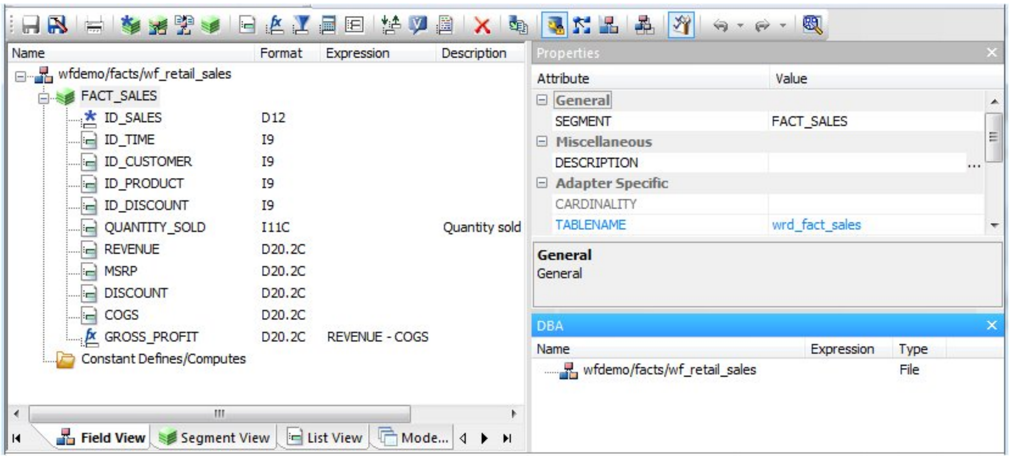
When security is specified, the Database Administrator or user, must enter a password to get access to the data source. When the DBA or user no longer needs access to the data source, you can delete their security.

Before adding any type of security to a data source, the Database Administrator must be aware of certain DBA guidelines. See *DBA Guidelines* on page 181.

Procedure: How to Set Up Security for the Database Administrator

1. In the Synonym Editor, click *DBA* from the Tools menu or click the *DBA* button from the Synonym toolbar.

   The DBA pane opens in the workspace, as shown in the following image.



Name	Format	Expression	Description
wfdemo/facts/wf_retail_sales			
FACT_SALES			
ID_SALES	D12		
ID_TIME	I9		
ID_CUSTOMER	I9		
ID_PRODUCT	I9		
ID_DISCOUNT	I9		
QUANTITY_SOLD	I11C		Quantity sold
REVENUE	D20.2C		
MSRP	D20.2C		
DISCOUNT	D20.2C		
COGS	D20.2C		
GROSS_PROFIT	D20.2C	REVENUE - COGS	
Constant Defines/Computes			

Attribute	Value
General	
SEGMENT	FACT_SALES
Miscellaneous	
DESCRIPTION	
Adapter Specific	
CARDINALITY	
TABLENAME	wrd_fact_sales

General
General

DBA

Name	Expression	Type
wfdemo/facts/wf_retail_sales		File

2. Right-click the file name in the DBA window and select *Insert*, then *DBA*.

   A default DBA password will be created for the Master File. You can change this value, delete it, add users to specify file restrictions, or add file names to specify data source-specific restrictions to the current data source. You can also specify a separate DBA file that contains DBA security restrictions.

   Note: When the password is created and the cursor is in that field, you can right-click and use the edit options to undo, select all, cut, copy, paste, or delete the password.

Procedure: How to Set Up Security for the User

1. In the DBA pane, right-click the DBA icon to insert user restrictions or specify a DBA file.
2. Once you add a user you can continue to insert file access restrictions by right-clicking the user field and selecting insert.
3. Select the type of access: Read, Write, Read/Write, or Update.
4. Specify the type of restriction for each option: Restriction to Field, Value, Segment, Noprint, or Same Restriction.

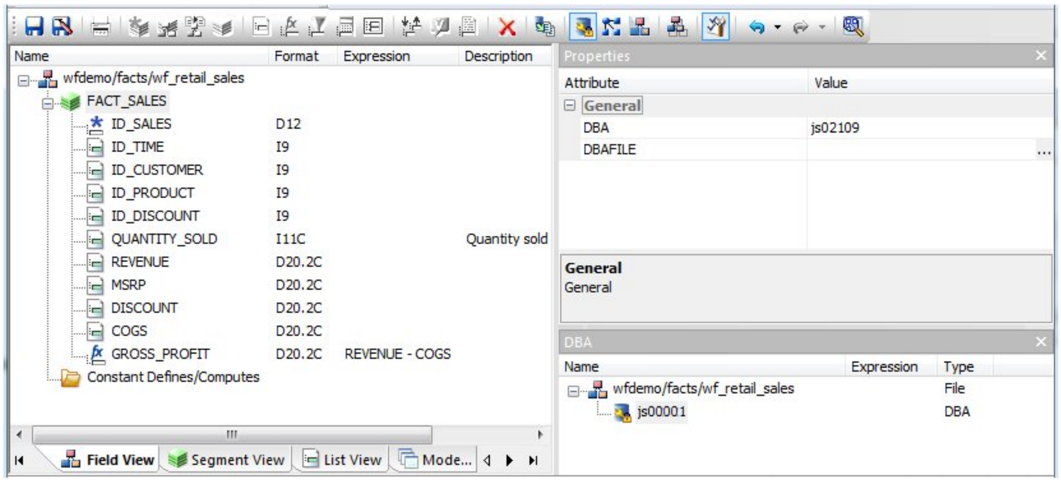
   Note: The Same Restriction option is activated when there are multiple users.
5. Click *OK* to save the Master File with the user password and restrictions.

Reference: DBA Guidelines

You can ensure that the security restrictions you place on Master Files are correct by adhering to the following guidelines:

- Every file with access limits must have a DBA password.
- No segment, field, or field value restrictions may be specified at the Database Administrator level. The Database Administrator should have unlimited access to the data source and all cross-referenced data sources.
- Once security restrictions have been applied, the Database Administrator should conduct thorough testing of every restriction before the data source is used. It is particularly important to check field values to make sure they do not contain errors. If they are in error, user access to the field data will be unnecessarily restricted.
- All groups of cross-referenced data sources must have the same security restrictions.
- The Database Administrator can change any type of security restriction.
- Access levels affect the fields users can access. The Database Administrator must consider what commands each user will need. If a user does not have access rights, that user will receive a message.

Reference: DBA Pane



Name	Format	Expression	Description
wfdemo/facts/wf_retail_sales			
FACT_SALES			
ID_SALES	D12		
ID_TIME	I9		
ID_CUSTOMER	I9		
ID_PRODUCT	I9		
ID_DISCOUNT	I9		
QUANTITY_SOLD	I11C		Quantity sold
REVENUE	D20.2C		
MSRP	D20.2C		
DISCOUNT	D20.2C		
COGS	D20.2C		
GROSS_PROFIT	D20.2C	REVENUE - COGS	
Constant Defines/Computes			

Properties

Attribute	Value
General	
DBA	js02109
DBAFILE	...

General
General

DBA

Name	Expression	Type
wfdemo/facts/wf_retail_sales		File
js00001		DBA

The following options are available from the DBA pane when the DBA password is selected.

DBA password

By default, the DBA password is the same as the user ID used to connect to the reporting server. Using the Rename option from the DBA password Context menu, you may enter a different password of up to sixty-four characters. This is the password of the DBA who will be creating and maintaining the current data source.

The DBA has full access to the data source and the corresponding Master File and controls the access rights of other users.

DBAFILE

Select the name of the Master File that contains your DBA security restrictions. Other Master Files can use the DBA security restrictions in this Master File.

Insert Filename

Enter the name of the Master File to which user security will be applied. This option is used to add data source-specific restrictions to the current data source. It includes a FILENAME attribute for the selected Master File. The FILENAME attribute in the referenced Master File must be the same as the FILENAME attribute in the DBA section of the current data source.

Insert Users

Enter the names (up to sixty-four characters) of users whose access rights will be granted for the current data source.

File Access

For user access, select one of the following options:

- Choose *Read* for full viewing rights.
- Choose *WRITE* to permit additions or changes to the data source.
- Choose *READ/WRITE* for both of the above.
- Choose *UPDATE* to permit changes to field values.

Restrictions: Segment, Field, Value, Noprint, Same

When the file access is selected, continue to select the type of restriction you wish to apply.

- Choose *Segment* to grant access to all or individual segments.
- Choose *Field* to grant access to all or individual fields.
- Choose *Value* to limit access to values that meet a test condition. See *Restricting Access to Segments, Fields, Field Values, and Noprint* on page 183.

- Choose *Noprint* to specify fields you do not want to display in a report.
- Choose *Same* to apply the same restrictions as other users that are already set up.

Access Restrictions

- **User.** Is the user name written to the Master File.
- **Name.** Is the name of the Master File component selected (for example, the segment or field name).
- **Access.** Is the type of access restriction.
- **Restrict.** Is an option for File access restriction.
- **Value.** Is the value for which to restrict access.

Selecting the Type of Access

When you assign a user password, the *type of file access* and *access restrictions* options are available. You must specify at least the type of access the user is permitted to have for the data source. The type of file access can be specified in the File Access group on the DBA pane. In this group, there are four file access options:

- **Read.** Allows the user only to read (to view) the data source.
- **WRITE.** Allows the user only to write (add or to make changes) to the data source.
- **READ/WRITE.** Allows the user to read and write to the data source.
- **UPDATE.** Allows the user to update (make changes to) existing field values.

The type of file access determines what a user can do to the entire data source:

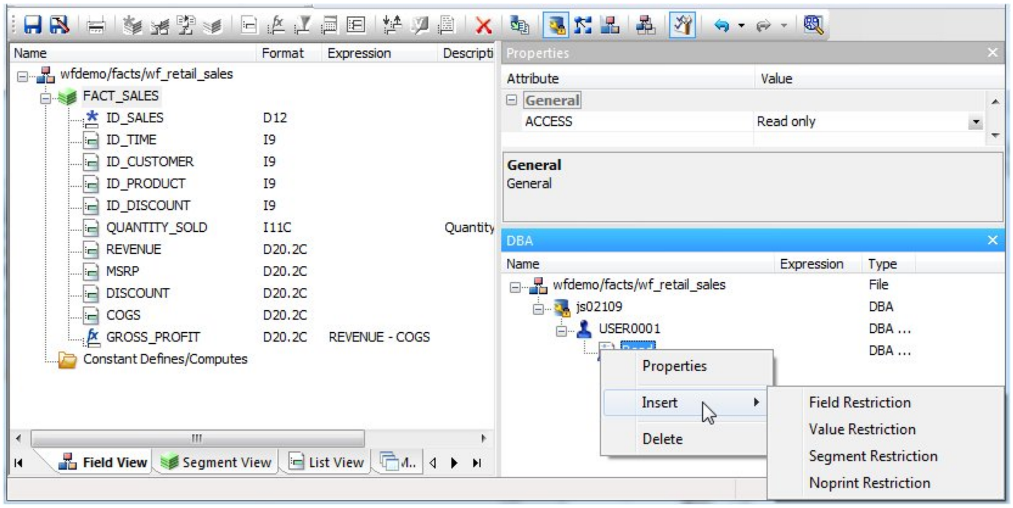
- If you specify only the type of file access, the user will have the specified access to the entire data source.
- If you want to impose additional limitations you can restrict access to segments, fields, and field values. See *Restricting Access to Segments, Fields, Field Values, and Noprint* on page 183.

Restricting Access to Segments, Fields, Field Values, and Noprint

You can restrict access to segments, fields, field values, and Noprint fields in a Master File by specifying access restrictions for a user. When you specify what is to be restricted, such as segment, field, or value, you can then specify the type of access that will be restricted.

Right-click the file access restriction and select the *Segment*, *Field*, or *Value*, or *Noprint* option from the Context menu.

- **Segment.** You specify the type of access for individual segments, as shown in the following image.



Name	Format	Expression	Descripti
wfdemo/facts/wf_retail_sales			
FACT_SALES			
ID_SALES	D12		
ID_TIME	I9		
ID_CUSTOMER	I9		
ID_PRODUCT	I9		
ID_DISCOUNT	I9		
QUANTITY_SOLD	I11C		Quantity
REVENUE	D20.2C		
MSRP	D20.2C		
DISCOUNT	D20.2C		
COGS	D20.2C		
GROSS_PROFIT	D20.2C	REVENUE - COGS	
Constant Defines/Computes			

Properties

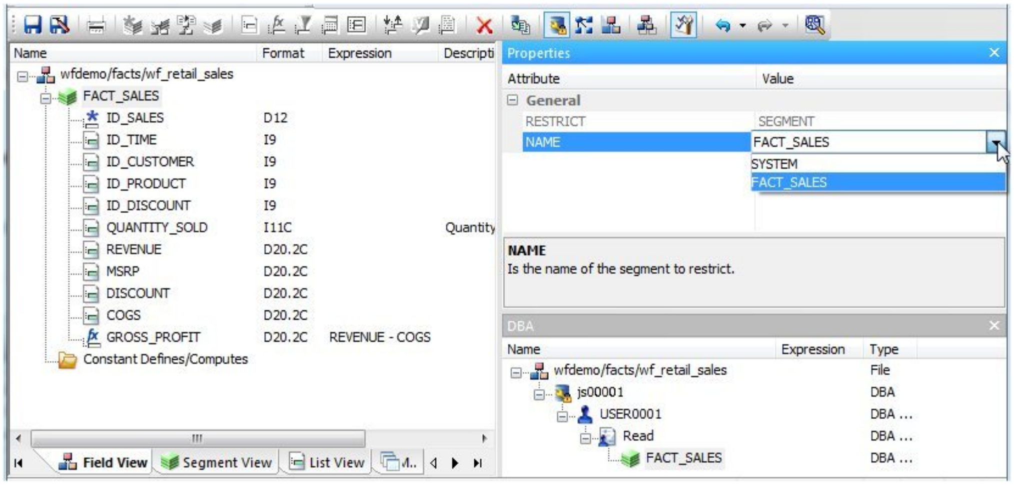
Attribute	Value
General	
ACCESS	Read only

General
General

DBA

Name	Expression	Type
wfdemo/facts/wf_retail_sales		File
js02109		DBA
USER0001		DBA ...
Read		DBA ...

The following image illustrates how a user can change a segment name.



Name	Format	Expression	Descripti
wfdemo/facts/wf_retail_sales			
FACT_SALES			
ID_SALES	D12		
ID_TIME	I9		
ID_CUSTOMER	I9		
ID_PRODUCT	I9		
ID_DISCOUNT	I9		
QUANTITY_SOLD	I11C		Quantity
REVENUE	D20.2C		
MSRP	D20.2C		
DISCOUNT	D20.2C		
COGS	D20.2C		
GROSS_PROFIT	D20.2C	REVENUE - COGS	
Constant Defines/Computes			

Properties

Attribute	Value
General	
RESTRICT	SEGMENT
NAME	FACT_SALES

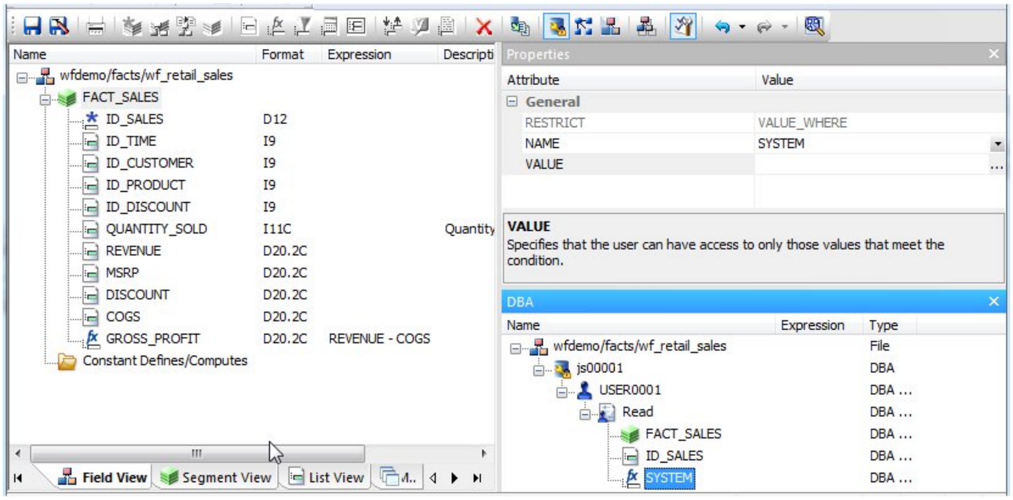
NAME
Is the name of the segment to restrict.

DBA

Name	Expression	Type
wfdemo/facts/wf_retail_sales		File
js00001		DBA
USER0001		DBA ...
Read		DBA ...
FACT_SALES		DBA ...

- **Field.** You specify the type of access for individual fields.
- **Value.** You specify the type of access (read or write) and the test condition. The user is restricted to using only those values that satisfy the test condition.

The following image illustrates how to change a field name used in a value field.



Name	Format	Expression	Descripti
wfdemo/facts/wf_retail_sales			
FACT_SALES			
ID_SALES	D12		
ID_TIME	I9		
ID_CUSTOMER	I9		
ID_PRODUCT	I9		
ID_DISCOUNT	I9		
QUANTITY_SOLD	I11C		Quantity
REVENUE	D20.2C		
MSRP	D20.2C		
DISCOUNT	D20.2C		
COGS	D20.2C		
GROSS_PROFIT	D20.2C	REVENUE - COGS	
Constant Defines/Computes			

Properties

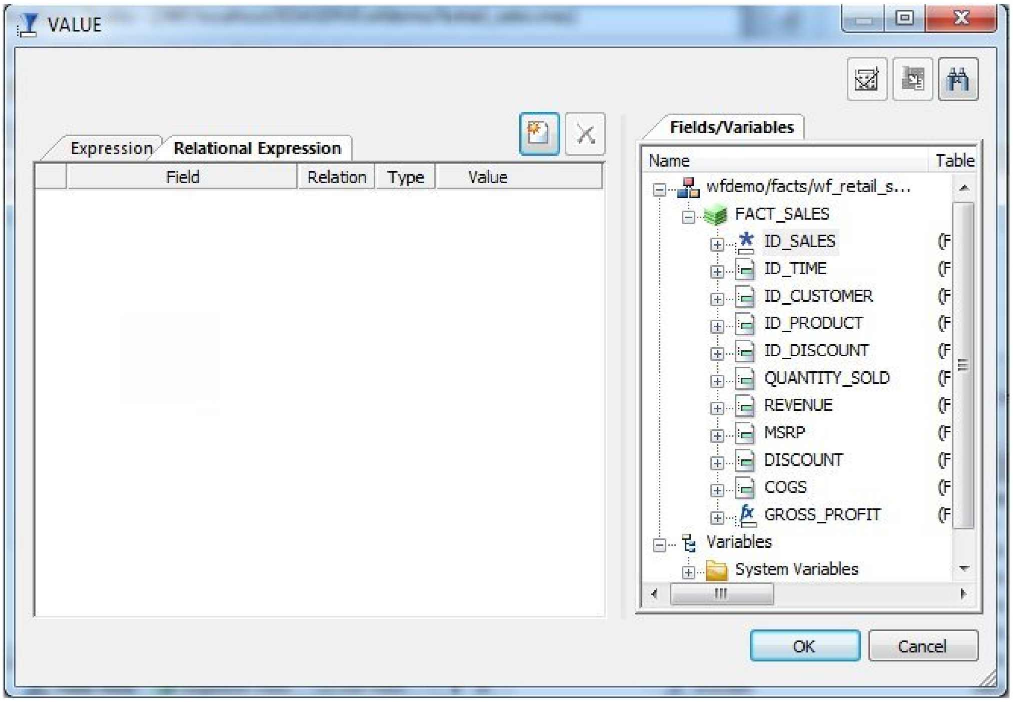
Attribute	Value
General	
RESTRICT	VALUE_WHERE
NAME	SYSTEM
VALUE	...

VALUE
Specifies that the user can have access to only those values that meet the condition.

DBA

Name	Expression	Type
wfdemo/facts/wf_retail_sales		File
js00001		DBA
USER0001		DBA ...
Read		DBA ...
FACT_SALES		DBA ...
ID_SALES		DBA ...
SYSTEM		DBA ...

The following image illustrates how to create a condition. This dialog box is presented after pressing the ellipsis next to the value field.



Field	Relation	Type	Value

Name	Table
wfdemo/facts/wf_retail_s...	
FACT_SALES	
ID_SALES	(F
ID_TIME	(F
ID_CUSTOMER	(F
ID_PRODUCT	(F
ID_DISCOUNT	(F
QUANTITY_SOLD	(F
REVENUE	(F
MSRP	(F
DISCOUNT	(F
COGS	(F
GROSS_PROFIT	(F
Variables	
System Variables	

- **Noprint.** You can also specify not to display the data in that field using Noprint. If you specify Noprint for a field, the data will appear as blanks for alphanumeric format or zeros for numeric format whenever the user tries to retrieve it.

Applying Security Restrictions for Multiple Users

How to:

Apply Previously Defined Restrictions to Another User

You can specify restrictions for one user and apply the same restrictions to other users. This helps when you want to set the same restrictions for a group of users.

Procedure: How to Apply Previously Defined Restrictions to Another User

1. In the DBA pane, right-click the DBA password and select *Insert*, then *User*.
2. Right-click the newly added user and select *Insert* to specify the desired type of access restriction you would like to apply.

   Available access types are Write Access, Read/Write Access, and Update Access.
3. Right-click an access type and select *Insert*, then *Same Restriction*.

   Note: The Same Restriction option is only available when there are multiple users. A drop-down combo box is activated in the Properties pane with a NAME attribute.
4. Click the arrow on the drop-down combo box next to the NAME attribute in the Properties pane, and then select the user with the security restrictions that would apply to the new user.

   Security restrictions from the user selected in the drop-down combo box are applied to the new user. You can apply the security restrictions to other users by repeating steps 1 to 4.

   Note: You must have created at least one user security restriction to apply security restrictions to multiple users.

Deleting a DBA or User Password

How to:

Delete a User Password

Delete a DBA Password

You can delete a DBA password or security for a user when it is no longer needed.

Procedure: How to Delete a User Password

1. On the DBA pane, select the user password you wish to delete.
2. Right-click and select *Delete* or press *Delete* on the keyboard.

If you delete the user based upon whom you have assigned security restrictions for other users, you must reset security restrictions for all users attached to the user you deleted.

Procedure: How to Delete a DBA Password

Deleting a DBA password will delete all user security for that data source.

On the DBA pane select the DBA password, then right-click and select *Delete* or press *Delete* on the keyboard.

All security information is removed.

IBM

4 Analyzing Metadata and Procedures

This chapter describes how to analyze procedures using Impact Analysis, and how to view Data Profiling for the columns in a synonym.

Note: Impact Analysis is accessible only through the DB2 Web Query Developer Workbench.

Topics:

Analyzing Procedures With the Impact Analysis Tool

How to:

View Impact Analysis Results From the Synonym Editor

Use the Impact Analysis Tool From the Interface

You can use the Impact Analysis tool to generate a report that identifies the procedures that access a specific Master File or field within a Master File. This tool helps you analyze the potential impact of modifying or deleting Master Files or fields. The Impact Analysis tool enables you to analyze data, control search criteria, save reports, and interactively open and edit procedures based on search results.

Impact Analysis searches Business Views in addition to Web Query procedures. This enables you to see if changes in the original Master File will impact fields used in the Business View.

The Impact Analysis tool is accessible from Developer Workbench or from within the tools in the Synonym Editor and Business Views. When Impact Analysis is launched from the Synonym Editor or Business Views, it searches files based on the application path of the reporting server. From the interface, it enables you to select applications or domains to be searched.

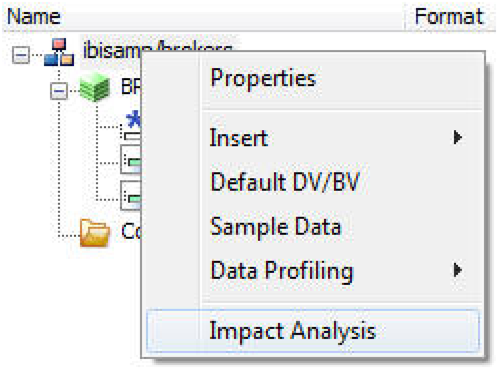
Procedure: How to View Impact Analysis Results From the Synonym Editor

When you launch Impact Analysis from within the Synonym Editor, it searches files based on the application path of the reporting server.

1. From the Data Servers area, double-click the Master File or select *Edit in Synonym Editor* from the File menu.

   The Master File opens to the Tree View tab in the Synonym Editor.

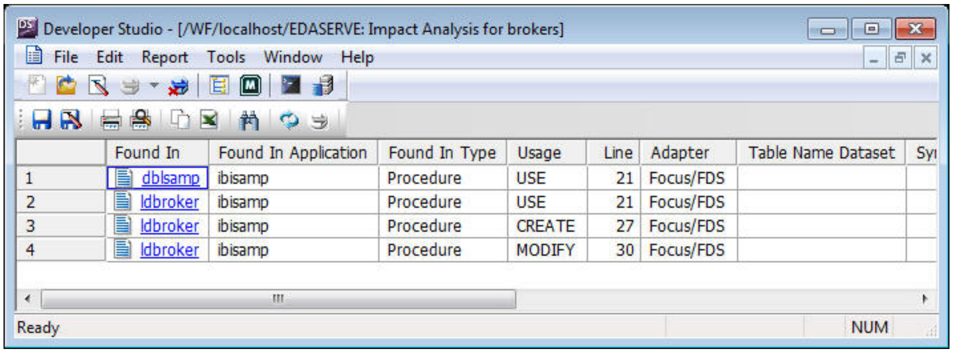
2. Select the Synonym File name, or a column field, then right-click and select *Impact Analysis* from the context menu, as shown in the following image.



Note: Impact Analysis is also available from the Dimension Builder pane.

The Impact Analysis results are displayed in a report spreadsheet.

3. You may use the results toolbar to view server messages, print the report, copy data as text, and export the report.


Developer Studio - [/WF/localhost/EDASERVE: Impact Analysis for brokers]
File
Edit
Report
Tools
Window
Help
Ready
NUM

	Found In	Found In Application	Found In Type	Usage	Line	Adapter	Table Name Dataset	Sy
1	dblsamp	ibisamp	Procedure	USE	21	Focus/FDS		
2	ldbroker	ibisamp	Procedure	USE	21	Focus/FDS		
3	ldbroker	ibisamp	Procedure	CREATE	27	Focus/FDS		
4	ldbroker	ibisamp	Procedure	MODIFY	30	Focus/FDS		

4. Click *Save* from the File menu to save the Impact Analysis results.

5. Click *Close* from the File menu to close the Impact Analysis results window and return to the Synonym Editor.

Procedure: How to Use the Impact Analysis Tool From the Interface

1. From the Data Servers area, highlight the Master File folder, select a Master File, and select *Impact Analysis* from the File menu.

   Note: You may also select *Impact Analysis* from the right-click context menu of a Master File. This interface enables you to select applications or folders to search.

The Impact Analysis tool opens with the New Report tab displaying the selected (Master) File Name and default Search Paths.

2. To search for all procedures that access a specific Master File or field, perform one of the following:

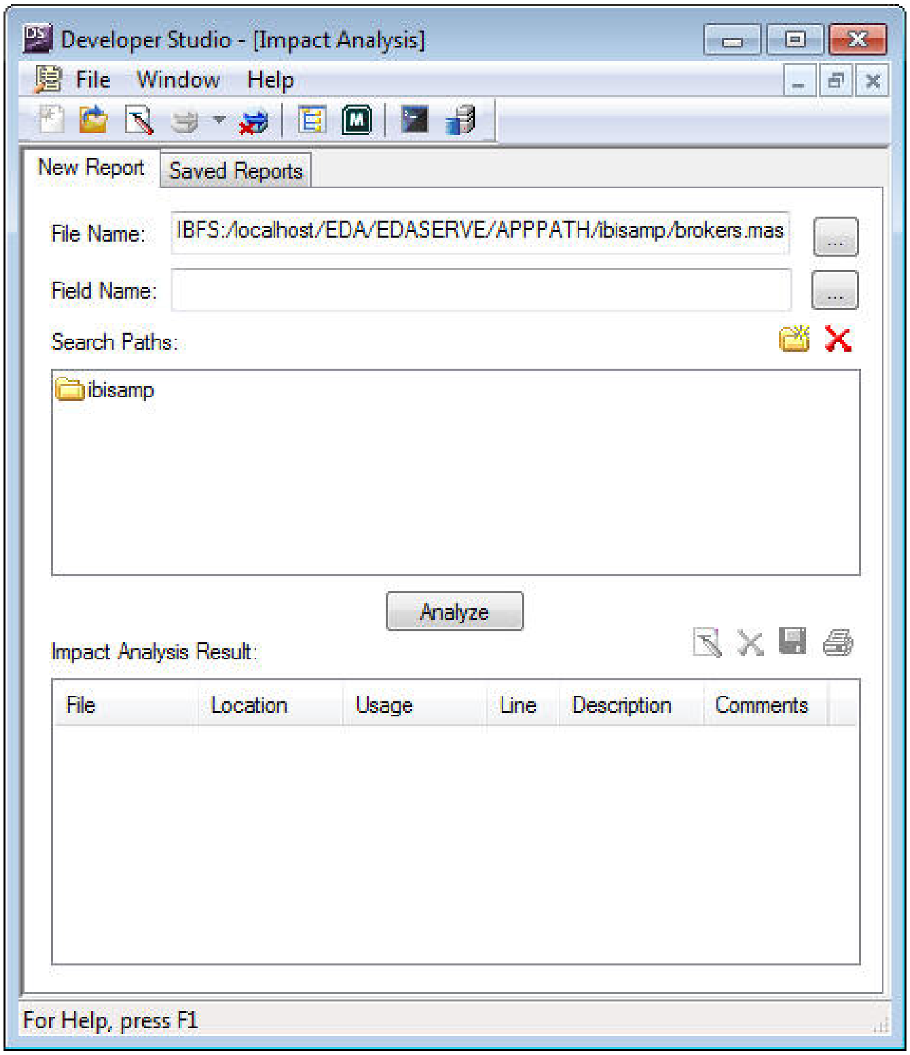
   - ❑ **Selected Master File.** The Master File you selected when opening the tool will be searched by default. Proceed to step 3.
   - ❑ **Different Master File.** Click the ellipsis button (...) to the right of the File Name search field and select a different Master File in the Open dialog box that opens.
   - ❑ **Single Field Within a Master File.** After you select the desired Master File, click the ellipsis button (...) to the right of the Field Name search field and double-click a field name in the Master File context menu.

3. Optionally, to search for procedures in directory paths not listed by default in the Search Paths pane, add more search paths by clicking the folder icon above the Search Paths area and selecting one or more folders in the Browse for Folder dialog box that opens.

   Note: You can also delete search paths by highlighting a search path and clicking the *Delete* icon above the Search Paths area.

4. Click *Analyze* to display a report in the Impact Analysis Result pane.

The following image shows the New Report tab of the Impact Analysis dialog box populated with File Name search criteria, multiple Search Paths, and a report displayed in the Impact Analysis Result pane.



File Name: IBFS:/localhost/EDA/EDASERVE/APPPATH/ibisamp/brokers.mas

Field Name:

Search Paths:

ibisamp

For Help, press F1

You have options to edit procedures, print the report, delete an item in the report, and export a report.

Exported reports are XML-formatted, have an .IAR extension, and are saved in the following default directory (unless you specify a different location):

drive:\ibi\DevStudio*releasenumber*\bin

5. Click the *Saved Reports* tab to access all previously created reports.

   All Impact Analysis reports are automatically saved in the following XML-formatted file:

   `drive:\ibi\DevStudio`*`releasenumber`*`\bin\IARepository.xml`

   Information is appended to this file as new analysis reports are performed. You have options to view reports, import previously exported reports, and delete reports.

Viewing Data Profiling Characteristics

In this section:

Data Profiling a Synonym or Segment

Data Profiling a Single Column

Data Profiling provides data characteristics for the columns in a synonym. You can display the characteristics for all the columns in a synonym or segment, or for an individual column.

Note: Data Profiling is not available if your adapter is not configured correctly.

For alphanumeric columns, Data Profiling provides the segment, format, count of distinct values, total count, patterns count, maximum, minimum, average length, minimum and maximum values, and number of nulls. Patterns count shows the number of patterns found in each alphanumeric column.

For numeric columns, Data Profiling provides the segment, format, count of distinct values, total count, maximum, minimum, average values, and number of nulls.

Data Profiling for an individual column provides access to Statistics, Patterns, Values, and Outliers reports.

Data Profiling a Synonym or Segment

How to:

View Data Profiling for a Synonym or Segment

Data Profiling provides information on all the columns in a synonym or segment. You can also drill down to the Values or Patterns reports for an individual column from a synonym or segment Data Profiling report.

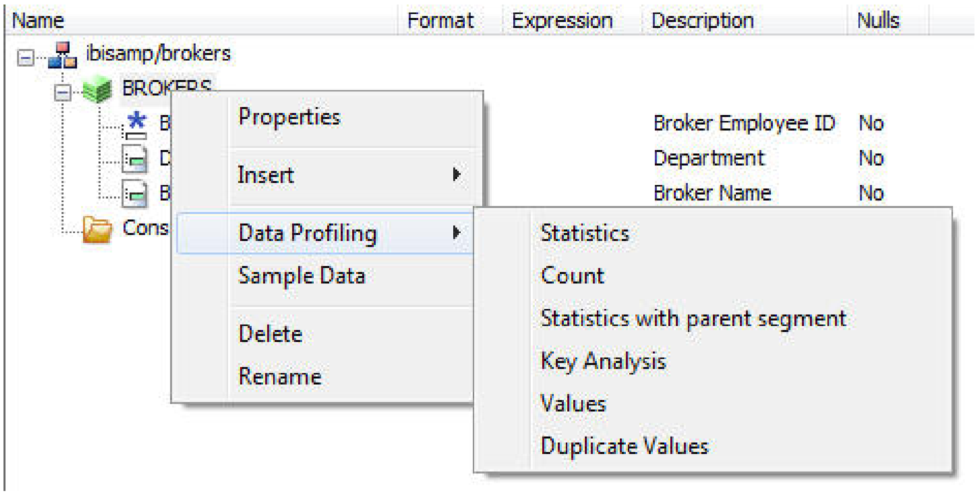
Procedure: How to View Data Profiling for a Synonym or Segment

To view the Data Profiling information for a synonym or segment:

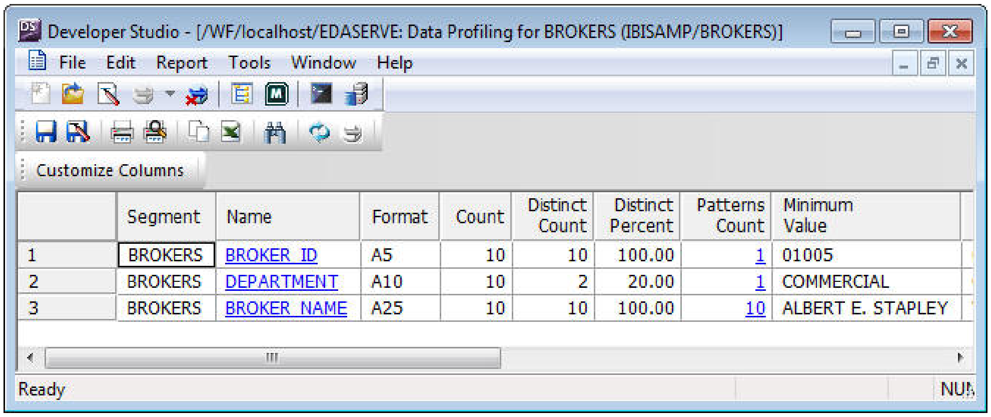
1. From the Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   The Synonym Editor opens to the Field View tab.

2. Right-click the synonym or segment name and select *Data Profiling,* and then select either *Statistics*, *Count*, *Statistics with parent segment*, *Key Analysis*, *Values*, or *Duplicate Values,* as shown in the following image.



The Data Profiling information displays in the workspace.



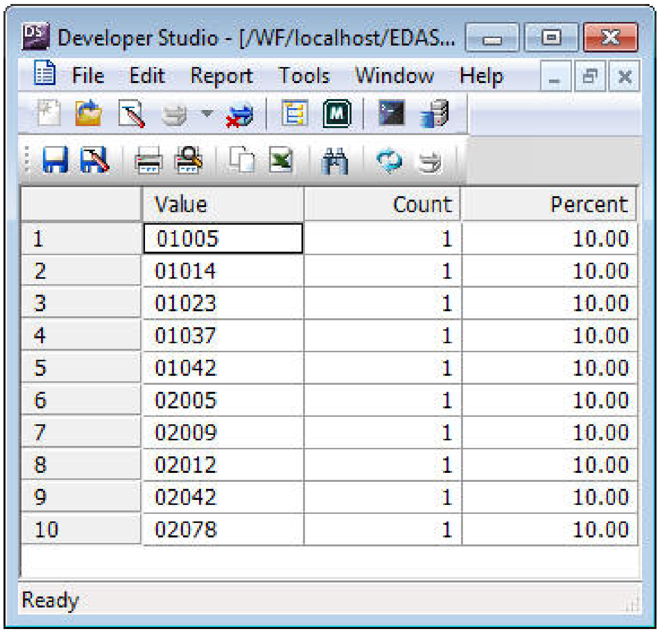
Developer Studio - [/WF/localhost/EDASERVE: Data Profiling for BROKERS (IBISAMP/BROKERS)]

	Segment	Name	Format	Count	Distinct Count	Distinct Percent	Patterns Count	Minimum Value
1	BROKERS	BROKER_ID	A5	10	10	100.00	1	01005
2	BROKERS	DEPARTMENT	A10	10	2	20.00	1	COMMERCIAL
3	BROKERS	BROKER_NAME	A25	10	10	100.00	10	ALBERT E. STAPLEY

You may use the Data Profiling Results toolbar to view server messages, print the report, copy data as text, and export the report.

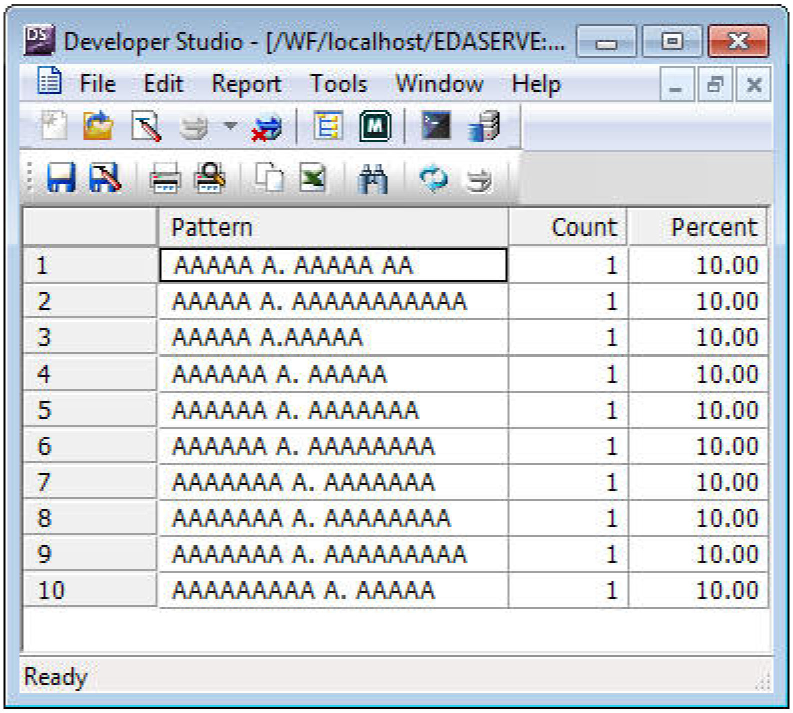
3. Optionally, you can click a column name or patterns count (for alphanumeric columns) to drill down to the Values or Patterns reports, respectively.

   This is a partial Values report produced by clicking a column name.



	Value	Count	Percent
1	01005	1	10.00
2	01014	1	10.00
3	01023	1	10.00
4	01037	1	10.00
5	01042	1	10.00
6	02005	1	10.00
7	02009	1	10.00
8	02012	1	10.00
9	02042	1	10.00
10	02078	1	10.00

The image below is an example of an address column.


Developer Studio - [/WF/localhost/EDASERVE:...
File Edit Report Tools Window Help

	Pattern	Count	Percent
1	AAAAA A. AAAAA AA	1	10.00
2	AAAAA A. AAAAAAAAAAA	1	10.00
3	AAAAA A.AAAAA	1	10.00
4	AAAAAA A. AAAAA	1	10.00
5	AAAAAA A. AAAAAAA	1	10.00
6	AAAAAA A. AAAAAAAA	1	10.00
7	AAAAAAA A. AAAAAAA	1	10.00
8	AAAAAAA A. AAAAAAAA	1	10.00
9	AAAAAAA A. AAAAAAAAA	1	10.00
10	AAAAAAAAA A. AAAAA	1	10.00

Ready

For pattern analysis, a *9* represents a digit, an *A* represents any uppercase letter, and an *a* represents any lowercase letter. All printable special characters are represented by themselves, and unprintable characters are represented by an *X*.

Data Profiling a Single Column

How to:

View Data Profile Statistics

View Data Profile Patterns

View Data Profile Values

View the Data Profile Values Graph

View Data Profile Duplicate Values

View Data Profile Outliers

Data Profiling for an individual column provides access to four reports:

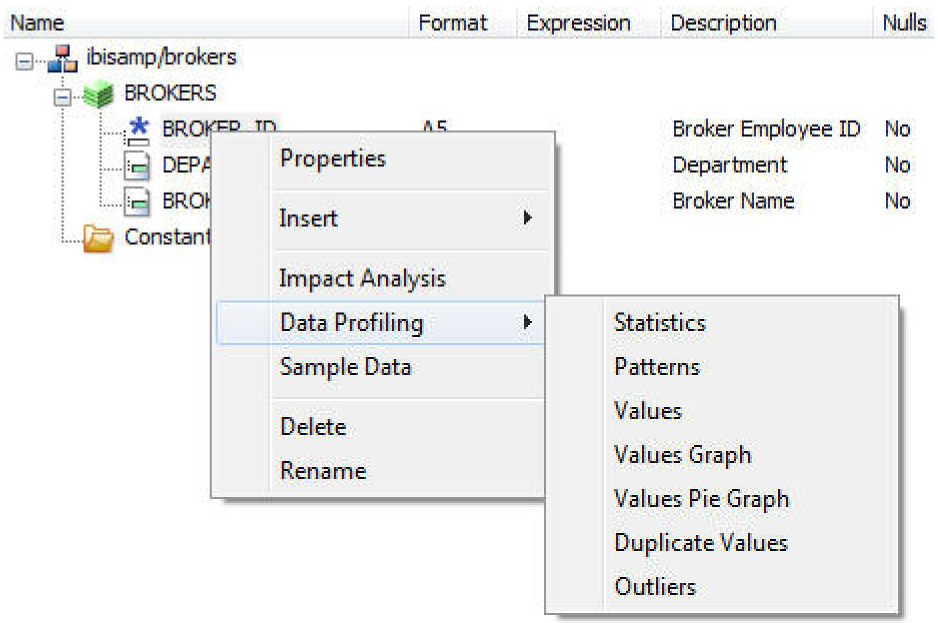
- **Statistics.** Shows the same information as a Data Profile report for a synonym or segment.

  For alphanumeric columns, the Statistics report provides the segment, format, count of distinct values, total count, patterns count, maximum, minimum, average length, minimum and maximum values, and number of nulls.

  For numeric columns, the Statistics report provides the segment, format, count of distinct values, total count, maximum, minimum, average values, and number of nulls.

- **Patterns.** Only available for alphanumeric columns. This shows patterns of letters, digits, and special characters, as well as counts and their percents.
- **Values.** Shows unique values and their percents.
- **Values Graph.** Displays a graph for alphanumeric field types.
- **Duplicate Values.** Shows identical values.
- **Outliers.** Shows the ten highest and lowest distinct values and their counts.

These reports are available by right-clicking a column in the Synonym Editor and selecting *Data Profiling*, as shown in the following image.



Procedure: How to View Data Profile Statistics

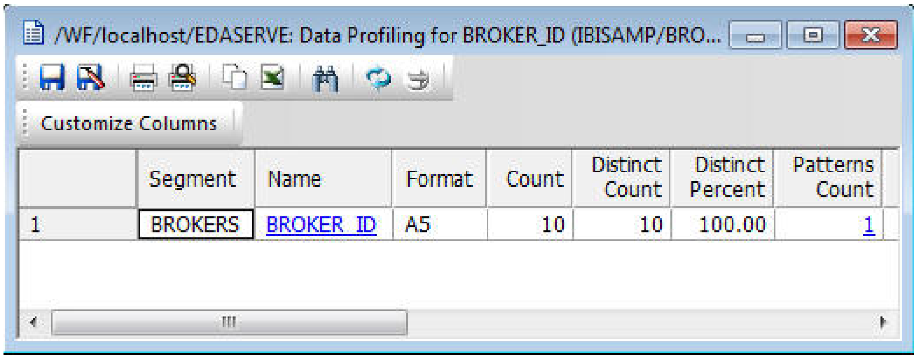
To view the Statistical Data Profiling information for a single column:

1. From the Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   The Synonym Editor opens to the Field View tab.

2. Right-click a column and select *Statistics* from the Data Profiling context menu.

   The Statistical Data Profiling information displays in the workspace, as shown in the following image.


/WF/localhost/EDASERVE: Data Profiling for BROKER_ID (IBISAMP/BRO...
Customize Columns

	Segment	Name	Format	Count	Distinct Count	Distinct Percent	Patterns Count
1	BROKERS	BROKER_ID	A5	10	10	100.00	1

3. Optionally, you can click a column name or patterns count (for alphanumeric columns) to drill down to the Values or Patterns reports, respectively.

Procedure: How to View Data Profile Patterns

Data Profile Patterns shows patterns of letters, digits, and special characters, as well as counts and their percentages.

To view the Patterns Data Profiling information for a single column:

1. From the Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   The Synonym Editor opens to the Field View tab.

2. Right-click a column and select *Patterns* from the Data Profiling context menu.

   The Patterns Data Profiling information displays in the workspace.

3. Optionally, you can click a pattern count to display the actual patterns.

   For pattern analysis, a *9* represents a digit, an *A* represents any uppercase letter, and an *a* represents any lowercase letter. All printable special characters are represented by themselves, and unprintable characters are represented by an*X*.

Procedure: How to View Data Profile Values

Data Profile Values shows unique values.

To view the Values Data Profiling information for a single column:

1. From the Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   The Synonym Editor opens to the Field View tab.

2. Right-click a column and select *Values* from the Data Profiling context menu.

   The Values Data Profiling information displays.

Procedure: How to View the Data Profile Values Graph

The Data Profile Values Graph shows

1. From the Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   The Synonym Editor opens to the Field View tab.

2. Right-click a column and select *Values Graph* from the Data Profiling context menu.

   The Values Graph Data Profiling information displays.

Procedure: How to View Data Profile Duplicate Values

Data Profile Duplicate Values shows identical values.

To view the Duplicate Values Data Profiling information for a single column:

1. From the Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

   The Synonym Editor opens to the Field View tab.

2. Right-click a column and select *Duplicate Values* from the Data Profiling submenu.

   The Duplicate Values Data Profiling information displays.

Procedure: How to View Data Profile Outliers

Data Profile Outliers shows the 10 highest and 10 lowest distinct values.

To view the Outliers Data Profiling information for a single column:

1. From the Data Servers area, open a synonym by double-clicking a Master File from the Master Files folder.

The Synonym Editor opens to the Field View tab.

2. Right-click a column and select *Outliers* from the Data Profiling context menu.

   The Outliers Data Profiling information displays.

   Note: Outliers produce a maximum of 10 highest and 10 lowest distinct values, if they exist.

IBM

5 Using Functions

The following topics offer an introduction to functions and explain how to use them. Functions provide a convenient way to perform certain calculations and manipulations.

In order to use any of the specified functions described in this reference manual, you need to create a temporary field which is discussed later.

Topics:

Types of Functions

In this section:

You can access any of the following types of functions:

- **Character functions.** Manipulate alphanumeric fields or character strings.
- **Data source and decoding functions.** Retrieve data source values and assign values based on the value of an input field.
- **Date and time functions.** Manipulate dates and times.
- **Format conversion functions.** Convert fields from one format to another.
- **Numeric functions.** Perform calculations on numeric constants and fields.
- **System functions.** Call the operating system to obtain information about the operating environment.

Character Functions

The following functions manipulate alphanumeric fields or character strings.

ARGLEN
Measures the length of a character string within a field, excluding trailing blanks.

BITSON
Evaluates an individual bit within a character string to determine whether it is on or off.

BYTVAL
Translates a character to its corresponding ASCII or EBCDIC decimal value.

CHKFMT
Checks a character string for incorrect characters or character types.

CTRAN
Translates a character within a character string to another character based on its decimal value.

CTRFLD
Centers a character string within a field.

EDIT
Extracts characters from or adds characters to a character string.

GETTOK
Divides a character string into substrings, called tokens, where a specific character, called a delimiter, occurs in the string.

LCWORD
Converts the letters in a character string to mixed case.

LCWORD2
Converts the letters in a character string to mixed-case by converting every alphanumeric character to lowercase except the first letter of each new word.

LCWORD3
Converts the letters in a character string to mixed-case by converting the first letter of each word to uppercase and converting every other letter to lowercase.

LJUST
Left-justifies a character string within a field.

LOCASE
Converts alphanumeric text to lowercase.

OVRLAY
Overlays a base character string with a substring.

PARAG
Divides a line of text into smaller lines by marking them with a delimiter.

POSIT
Finds the starting position of a substring within a larger string.

PTOA
Converts a number from numeric format to alphanumeric format. It retains the decimal positions of the number and right-justifies it with leading spaces. You can also add edit options to a number converted by PTOA.

REVERSE
Reverses the characters in a character string.

RJUST
Right-justifies a character string.

SOUNDEX
Searches for a character string phonetically without regard to spelling.

SPELLNUM
Takes an alphanumeric string or a numeric value with two decimal places and spells it out with dollars and cents.

SUBSTR
Extracts a substring based on where it begins and its length in the parent string.

UPCASE
Converts a character string to uppercase.

Data Source and Decoding Functions

The following functions retrieve data source values and assign values.

DECODE
Assigns values based on the coded value of an input field.

LAST
Retrieves the preceding value for a field.

Date and Time Functions

In this section:

Standard Date and Time Functions

The following functions manipulate dates and times.

Standard Date and Time Functions

AYM
Adds or subtracts months from dates that are in year-month format.

AYMD
Adds or subtracts days from dates that are in year-month-day format.

CHGDAT
Rearranges the year, month, and day portions of alphanumeric dates, and converts dates between long and short date formats.

DA
Converts dates to the corresponding number of days elapsed since December 31, 1899.

`DADMY` converts dates in day-month-year format.

`DADYM` converts dates in day-year-month format.

`DAMDY` converts dates in month-day-year format.

`DAMYD` converts dates in month-year-day format.

`DAYDM` converts dates in year-day-month format.

`DAYMD` converts dates in year-month-day format.

DATEADD
Adds a unit to or subtracts a unit from a date format.

DATECVT
Converts date formats.

DATEDIF
Returns the difference between two dates in units.

DATEMOV
Moves a date to a significant point on the calendar.

DATEPATTERN
Stores date values in alphanumeric format without any particular standard, with any combination of components such as year, quarter, and month, and with any delimiter.

DATETRAN
Formats dates in international formats.

DMY, MDY, and YMD
Calculates the difference between two dates.

DOWK and DOWKL
Finds the day of the week that corresponds to a date.

DT
Converts the number of days elapsed since December 31, 1899 to the corresponding date.

`DTDMY` converts numbers to day-month-year dates.

`DTDYM` converts numbers to day-year-month dates.

`DTMDY` converts numbers to month-day-year dates.

`DTMYD` converts numbers to month-year-day dates.

`DTYDM` converts numbers to year-day-month dates.

`DTYMD` converts numbers to year-month-day dates.

FIYR
Returns the financial year, also known as the fiscal year, corresponding to a given calendar date based on the financial year starting date and the financial year numbering convention.

FIQTR
Returns the financial quarter corresponding to a given calendar date based on the financial year starting date and the financial year numbering convention.

FIYYQ
Returns a financial date containing both the financial year and quarter that corresponds to a give calendar date.

GREGDT
Converts dates in Julian format to year-month-day format.

HADD
Increments a date-time field by a given number of units.

HCNVRT
Converts a date-time field to a character string.

HDATE
Extracts the date portion of a date-time field, converts it to a date format, and returns the result in the format YYMD.

HDIFF
Calculates the number of units between two date-time values.

HDTTM
Converts a date field to a date-time field. The time portion is set to midnight.

HGETC
Stores the current date and time in a date-time field.

HHMMSS
Retrieves the current time from the system.

HINPUT
Converts an alphanumeric string to a date-time value.

HMIDNT
Changes the time portion of a date-time field to midnight (all zeros).

HNAME
Extracts a specified component from a date-time field and returns it in alphanumeric format.

HPART
Extracts a specified component from a date-time field and returns it in numeric format.

HSETPT
Inserts the numeric value of a specified component into a date-time field.

HTIME
Converts the time portion of a date-time field to the number of milliseconds or microseconds.

JULDAT
Converts dates from year-month-day format to Julian (year-day format).

TIMETOTS
Converts a time to a timestamp.

TODAY
Retrieves the current date from the system.

YM
Calculates the number of months that elapse between two dates. The dates must be in year-month format.

Format Conversion Functions

The following functions convert fields from one format to another.

ATODBL
Converts a number in alphanumeric format to double-precision format.

EDIT
Converts an alphanumeric field that contains numeric characters to numeric format or converts a numeric field to alphanumeric format.

FTOA
Converts a number in a numeric format to alphanumeric format.

HEXBYT
Obtains the ASCII or EBCDIC character equivalent of a decimal integer value.

ITONUM
Converts a large binary integer in a data source to double-precision format.

ITOPACK
Converts a large binary integer in a data source to packed-decimal format.

ITOZ
Converts a number in numeric format to zoned format.

PCKOUT
Writes a packed number of variable length to an extract file.

Numeric Functions

The following functions perform calculations on numeric constants or fields.

ABS
Returns the absolute value of a number.

BAR
Produces a horizontal bar chart.

CHKPCK
Validates the data in a field described as packed format.

DMOD, FMOD, and IMOD
Calculates the remainder from a division.

EXP
Raises the number "e" to a specified power.

INT
Returns the integer component of a number.

LOG
Returns the natural logarithm of a number.

MAX and MIN
Returns the maximum or minimum value, respectively, from a list of values.

PRDNOR and PRDUNI
Generates reproducible random numbers.

RDNORM and RDUNIF
Generates random numbers.

SQRT
Calculates the square root of a number.

System Functions

The following functions call the operating system to obtain information about the operating environment.

FGETENV
Retrieves the value of an environment variable and returns it as an alphanumeric string.

GETUSER
Retrieves the ID of the connected user.

HHMMSS
Retrieves the current time from the system.

TODAY
Retrieves the current date from the system.

Supplying an Argument in a Function

In this section:

When supplying an argument in a function, you must understand which types of arguments are acceptable, the formats and lengths for these arguments, and the number and order of these arguments.

Argument Types

The following are acceptable arguments for a function:

- Numeric constant, such as 6 or 15.
- Date constant, such as 022802.
- Date in alphanumeric, numeric, or date format.
- Alphanumeric literal, such as STEVENS or NEW YORK NY. A literal must be enclosed in single quotation marks.
- Number in alphanumeric format.
- Field name, such as FIRST_NAME or HIRE_DATE. A field can be a data source field or temporary field. The field name can be up to 66 characters long or a qualified field name, unique truncation, or alias.

- Expression, such as a numeric, date, or alphanumeric expression. An expression can use arithmetic operators and the concatenation sign (|). For example, the following are valid expressions:

  ```
  CURR_SAL * 1.03
  ```

  and

  ```
  FN || LN
  ```

- Format of the output value enclosed in single quotation marks.
- Another function.

Increased Number of Function Arguments

The number of arguments supported for user-written subroutines has been increased from 28 to 200. All other rules regarding arguments for user-written subroutines remain the same.

Argument Formats

Depending on the function, an argument can be in alphanumeric, numeric, or date format. If you supply an argument in the wrong format, you will cause an error or the function will not return correct data. To obtain the valid formats, click the Format button on the tool for a list of possible types and lengths. The following are the types of argument formats:

- **Alphanumeric argument.** An alphanumeric argument is stored internally as one character per byte. An alphanumeric argument can be a literal, an alphanumeric field, a number or date stored in alphanumeric format, an alphanumeric expression, or the format of an alphanumeric field.
- **Numeric argument.** A numeric argument is stored internally as a binary or packed number. A numeric argument includes integer (I), floating-point single-precision (F), floating-point double-precision (D), and packed-decimal (P) formats. A numeric argument can be a numeric constant, field, or expression, or the format of a numeric field. All numeric arguments are converted to floating-point double-precision format when used with a function, but results are returned in the format specified for the output field.
- **Date argument.** A date argument can be in either alphanumeric, numeric, or date format. The list of arguments for the individual function will specify what type of format the function accepts. A date argument can be a date in alphanumeric, numeric, or date format; a date field or expression; or the format of a date field. If you supply an argument with a two-digit year, the function assigns a century based on the DATEFNS, YRTHRESH, and DEFCENT parameter settings.

Using a Function in a Temporary Field

In order to use any of the specified functions described in this reference manual, you need to create 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. When you create a temporary field, you determine its value by writing an expression. You can combine fields, constants, and operators in an expression to produce a single value. For example, if your data contains salary and deduction amounts, you can calculate the ratio of deductions to salaries using the following expression: deduction / salary.

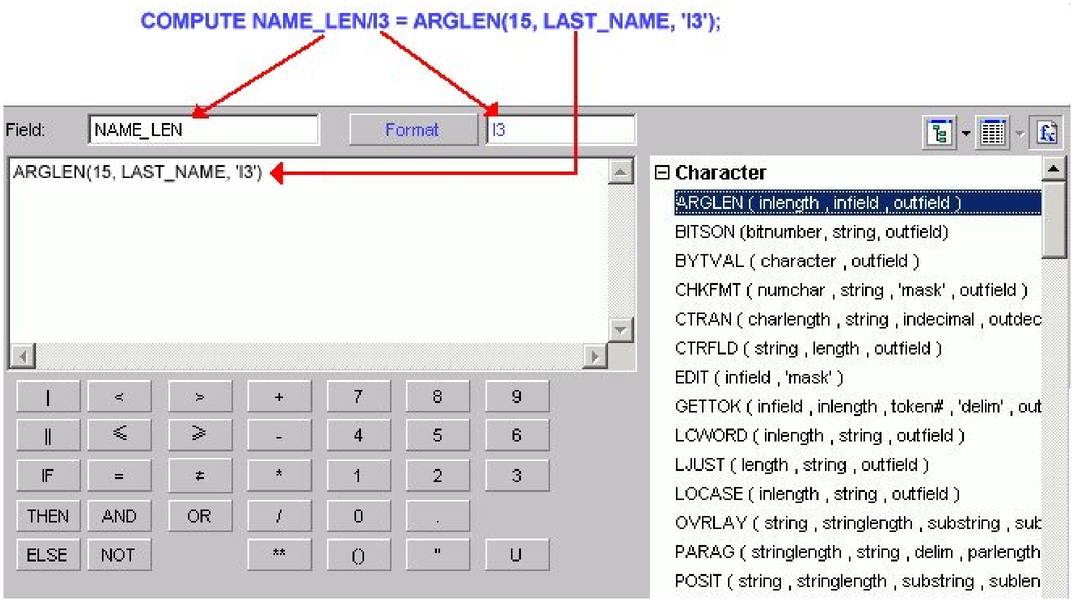
You can specify the expression yourself, or you can use one of the many supplied functions that perform specific calculations or manipulations. In addition, you can use expressions and functions as building blocks for more complex expressions, as well as use one temporary field to evaluate another. You can create temporary fields from the Field selection tab by selecting the Define or Compute tool.

All examples in this reference manual show the complete syntax for Computed fields, as seen in a text editor. For example, `COMPUTE NAME_LEN/I3 = ARGLEN(15, LAST_NAME, 'I3');`

When creating a temporary field using the Compute tool, you do *not* need to:

- Include COMPUTE in the expression, which is implicit.
- End the expression with a semicolon, which is implicit.
- Include AND when creating more than one expression; it is implicit. Each COMPUTE expression must be created separately as a new temporary field.

To illustrate, the following image in InfoAssist shows the function, ARGLEN in its full syntax. In the Compute tool, the Field name is NAME_LEN, the Format is I3, and the expression window contains ARGLEN(15, LAST_NAME, 'I3').


COMPUTE NAME_LEN/I3 = ARGLEN(15, LAST_NAME, 'I3');
Field: NAME_LEN
Format I3
ARGLEN(15, LAST_NAME, 'I3')
Character
ARGLEN (inlength , infield , outfield)
BITSON (bitnumber, string, outfield)
BYTVAL (character , outfield)
CHKFMT (numchar , string , 'mask' , outfield)
CTRAN (charlength , string , indecimal , outdec
CTRFLD (string , length , outfield)
EDIT (infield , 'mask')
GETTOK (infield , inlength , token# , 'delim' , out
LCWORD (inlength , string , outfield)
LJUST (length , string , outfield)
LOCASE (inlength , string , outfield)
OVRLAY (string , stringlength , substring , sub
PARAG (stringlength , string , delim , parlength
POSIT (string , stringlength , substring , sublen
| < > + 7 8 9
|| ≤ ≥ - 4 5 6
IF = ≠ * 1 2 3
THEN AND OR / 0 .
ELSE NOT ** () " U

Character Functions

In this section:

Character functions manipulate alphanumeric fields and character strings.

ARGLEN: Measuring the Length of a String

How to:

Measure the Length of a Character String

The ARGLEN function measures the length of a character string within a field, excluding trailing spaces. The field format in a Master File specifies the length of a field, including trailing spaces.

Syntax: How to Measure the Length of a Character String

```
ARGLEN(inlength, infield, 'outfield')
```

where:

inlength

Integer

Is the length of the field containing the character string, or a field that contains the length.

infield

Alphanumeric

Is the name of the field containing the character string.

outfield

Integer

Is the format of the output value enclosed in single quotation marks.

Example: Measuring the Length of a Character String

ARGLEN determines the length of the character string in LAST_NAME and stores the result in NAME_LEN:

```
COMPUTE NAME_LEN/I3 = ARGLEN(15, LAST_NAME, 'I3');
```

BITSON: Determining If a Bit Is On or Off

How to:

Determine If a Bit Is On or Off

The BITSON function evaluates an individual bit within a character string to determine whether it is on or off. If the bit is on, BITSON returns a value of 1; if the bit is off, it returns a value of 0. This function is useful in interpreting multi-punch data, where each punch conveys an item of information.

Syntax: How to Determine If a Bit Is On or Off

```
BITSON(bitnumber, string, 'outfield')
```

where:

bitnumber
Integer

Is the number of the bit to be evaluated, counted from the left-most bit in the character string.

string
Alphanumeric

Is the character string to be evaluated, enclosed in single quotation marks, or a field that contains the character string. The character string is in multiple eight-bit blocks.

outfield
Integer

Is the format of the output value enclosed in single quotation marks.

Example: Evaluating a Bit in a Field

BITSON evaluates the 24th bit of LAST_NAME and stores the result in BIT_24:

```
COMPUTE BIT_24/I1 = BITSON(24, LAST_NAME, 'I1');
```

BYTVAL: Translating a Character to a Decimal Value

How to:

Translate a Character

The BYTVAL function translates a character to the ASCII, EBCDIC, or Unicode decimal value that represents it, depending on the operating system.

Syntax: How to Translate a Character

```
BYTVAL(character, 'outfield')
```

where:

character

Alphanumeric

Is the character to be translated. You can specify a field that contains the character, or the character itself enclosed in single quotation marks. If you supply more than one character, the function evaluates the first.

outfield

Integer

Is the format of the output value enclosed in single quotation marks.

Example: Translating the First Character of a Field

BYTVAL translates the first character of LAST_NAME into its ASCII or EBCDIC decimal value and stores the result in LAST_INIT_CODE. Since the input string has more than one character, BYTVAL evaluates the first one.

```
COMPUTE LAST_INIT_CODE/I3 = BYTVAL(LAST_NAME, 'I3');
```

CHKFMT: Checking the Format of a String

How to:

Check the Format of a Character String

The CHKFMT function checks a character string for incorrect characters or character types. It compares each character string to a second string, called a mask, by comparing each character in the first string to the corresponding character in the mask. If all characters in the character string match the characters or character types in the mask, CHKFMT returns the value 0. Otherwise, CHKFMT returns a value equal to the position of the first character in the character string not matching the mask.

If the mask is shorter than the character string, the function checks only the portion of the character string corresponding to the mask. For example, if you are using a four-character mask to test a nine-character string, only the first four characters in the string are checked; the rest are returned as a no match with CHKFMT giving the first non-matching position as the result.

Syntax: How to Check the Format of a Character String

```
CHKFMT(numchar, string, 'mask', 'outfield')
```

where:

numchar

Integer

Is the number of characters being compared to the mask.

string

Alphanumeric

Is the character string to be checked enclosed in single quotation marks, or a field that contains the character string.

mask

Alphanumeric

Is the mask, which contains the comparison characters enclosed in single quotation marks.

Some characters in the mask are generic and represent character types. If a character in the string is compared to one of these characters and is the same type, it matches. Generic characters are:

`A` is any letter between A and Z (uppercase or lowercase).

`9` is any digit between 0-9.

`X` is any letter between A-Z or any digit between 0-9.

`$` is any character.

Any other character in the mask represents only that character. For example, if the third character in the mask is B, the third character in the string must be B to match.

outfield

Integer

Is the format of the output value enclosed in single quotation marks.

Example: Checking the Format of a Field

CHKFMT examines EMP_ID for nine numeric characters starting with 11 and stores the result in CHK_ID:

```
COMPUTE CHK_ID/I3 = CHKFMT(9, EMP_ID, '119999999', 'I3');
```

CTRAN: Translating One Character to Another

How to:

Translate One Character to Another

The CTRAN function translates a character within a character string to another character based on its decimal value. This function is especially useful for changing replacement characters to unavailable characters, or to characters that are difficult to input or unavailable on your keyboard.

To use CTRAN, you must know the decimal equivalent of the characters in internal machine representation.

In Unicode configurations, this function uses values in the range:

- 0 to 255 for 1-byte characters.
- 256 to 65535 for 2-byte characters.
- 65536 to 16777215 for 3-byte characters.
- 16777216 to 4294967295 for 4-byte characters (primarily for EBCDIC).

Syntax: **How to Translate One Character to Another**

```
CTRAN(charlen, string, decimal, decvalue, 'outfield')
```

where:

charlen

Integer

Is the number of characters in the string, or a field that contains the length.

string

Alphanumeric

Is the character string to be translated enclosed in single quotation marks, or the field that contains the character string.

decimal

Integer

Is the ASCII or EBCDIC decimal value of the character to be translated.

decvalue

Integer

Is the ASCII or EBCDIC decimal value of the character to be used as a substitute for *decimal*.

outfield

Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Example: Translating Spaces to Underscores on an EBCDIC Platform

CTRAN translates the spaces in ADDRESS_LN3 (EBCDIC decimal value 64) to underscores (EBCDIC decimal value 109) and stores the result in ALT_ADDR:

```
COMPUTE ALT_ADDR/A20 = CTRAN(20, ADDRESS_LN3, 64, 109, 'A20');
```

CTRFLD: Centering a Character String

How to:

Center a Character String

The CTRFLD function centers a character string within a field. The number of leading spaces is equal to or one less than the number of trailing spaces.

CTRFLD is useful for centering the contents of a field and its report column, or a heading that consists only of an embedded field. HEADING CENTER centers each field value including trailing spaces. To center the field value without the trailing spaces, first center the value within the field using CTRFLD.

Limit: Using CTRFLD in a styled report (StyleSheets feature) generally negates the effect of CTRFLD unless the item is also styled as a centered element. Also, if you are using CTRFLD on a platform for which the default font is proportional, either use a non-proportional font, or issue SET STYLE=OFF before running the request.

Syntax: How to Center a Character String

```
CTRFLD(string, length, 'outfield')
```

where:

string

Alphanumeric

Is the character string enclosed in single quotation marks, or a field that contains the character string.

length

Integer

Is the number of characters in *string* and *outfield*, or a field that contains the length. This argument must be greater than 0. A length less than 0 can cause unpredictable results.

`outfield`

Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Example: Centering a Field

CTRFLD centers LAST_NAME and stores the result in CENTER_NAME:

```
COMPUTE CENTER_NAME/A12 = CTRFLD(LAST_NAME, 12, 'A12');
```

EDIT: Extracting or Adding Characters

How to:

Extract or Add Characters

The EDIT function extracts characters from or adds characters to an alphanumeric string. It can extract a substring from different parts of the parent string, and can also insert characters from a parent string into another substring. For example, it can extract the first two characters and the last two characters of a string to form a single substring.

EDIT works by comparing the characters in a mask to the characters in a source field. When it encounters a nine in the mask, EDIT copies the corresponding character from the source field to the new field. When it encounters a dollar sign in the mask, EDIT ignores the corresponding character in the source field. When it encounters any other character in the mask, EDIT copies that character to the corresponding position in the new field. EDIT does not require an outfield argument because the result is obviously alphanumeric and its size is determined from the mask value.

EDIT can also convert the format of a field. For information on converting a field with EDIT, see *EDIT: Converting the Format of a Field* on page 307.

Syntax: How to Extract or Add Characters

```
EDIT(fieldname, 'mask');
```

where:

`fieldname`

Alphanumeric

Is the source field.

Is the string to extract characters from. It should be at least as long as the mask.

mask
Alphanumeric

Is a character string enclosed in single quotation marks. The length of the mask, excluding any characters other than nine and $, determines the length of the output field.

Example: Extracting and Adding a Character to a Field

EDIT extracts the first initial from the FIRST_NAME field and stores the result in FIRST_INIT. EDIT also adds dashes to the EMP_ID field and stores the result in EMPIDEDIT:

```
COMPUTE FIRST_INIT/A1 = EDIT(FIRST_NAME, '9$$$$$$$$$'); AND
COMPUTE EMPIDEDIT/A11 = EDIT(EMP_ID, '999-99-9999');
```

GETTOK: Extracting a Substring (Token)

How to:

Extract a Substring (Token)

The GETTOK function divides a character string into substrings, called tokens. A specific character, called a delimiter, occurs in the string and separates the string into tokens. GETTOK returns the token specified by the token_number. GETTOK ignores leading and trailing blanks in the parent character string.

For example, suppose you want to extract the fourth word from a sentence. Use the space character for a delimiter and four for the token_number. GETTOK divides the sentence into words using this delimiter, then extracts the fourth word. If the string is not divided by the delimiter, use the PARAG function for this purpose.

Syntax: How to Extract a Substring (Token)

```
GETTOK(infield, inlen, token_number, 'delim', outlen, 'outfield')
```

where:

infield
Alphanumeric

Is the field containing the parent character string.

inlen
Integer

Is the length of the parent string in characters. If this argument is less than or equal to 0, the function returns spaces.

token_number
Integer

Is the number of the token to extract. If this argument is positive, the tokens are counted from left to right. If this argument is negative, the tokens are counted from right to left. For example, -2 extracts the second token from the right. If this argument is 0, the function returns spaces. Leading and trailing null tokens are ignored.

delim
Alphanumeric

Is the delimiter in the parent string enclosed in single quotation marks. If you specify more than one character, only the first character is used.

outlen
Integer

Is the maximum size of the token. If this argument is less than or equal to 0, the function returns spaces. If the token is longer than this argument, it is truncated; if it is shorter, it is padded with trailing spaces.

outfield
Alphanumeric

Is the format of the output value enclosed in single quotation marks. The delimiter is not included in the token.

Example: Extracting a Token From a Field

GETTOK extracts the last token from ADDRESS_LN3 and stores the result in LAST_TOKEN:

```
COMPUTE LAST_TOKEN/A10 = GETTOK(ADDRESS_LN3, 20, -1, ' ', 10, 'A10');
```

LCWORD: Converting a Character String to Mixed Case

How to:

Convert a Character String to Mixed Case

The LCWORD function converts the letters in a character string to mixed case. It converts every alphanumeric character to lowercase except the first letter of each new word and the first letter after a single or double quotation mark. For example, O'CONNOR is converted to O'Connor and JACK'S to Jack'S.

If LCWORD encounters a number in the character string, it treats it as an uppercase character and continues to convert the following alphabetic characters to lowercase. The result of LCWORD is a word with an initial uppercase character followed by lowercase characters.

Syntax: How to Convert a Character String to Mixed Case

```
LCWORD(length, string, 'outfield')
```

where:

length

Integer

Is the length in characters of the character string or field to be converted, or a field that contains the length.

string

Alphanumeric

Is the character string to be converted enclosed in single quotation marks, or a field containing the character string.

outfield

Alphanumeric

Is the format of the output value enclosed in single quotation marks. The length must be greater than or equal to the length of *length*.

Example: Converting a Character String to Mixed-Case

LCWORD converts the LAST_NAME field to mixed-case and stores the result in MIXED_CASE:

```
COMPUTE MIXED_CASE/A15 = LCWORD(15, LAST_NAME, 'A15');
```

LCWORD2: Converting a Character String to Mixed-Case

How to:

Convert a Character String to Mixed-Case

The LCWORD2 function converts the letters in a character string to mixed-case by converting every alphanumeric character to lowercase except the first letter of each new word. If LCWORD2 encounters a lone single quotation mark, the next letter is converted to lowercase. For example, 'SMITH' would be changed to 'Smith' and JACK'S would be changed to Jack's.

Syntax: How to Convert a Character String to Mixed-Case

```
LCWORD2(length, string, 'outfield')
```

where:

`length`
Integer

Is the length in characters of the character string or field to be converted, or a field that contains the length.

`string`
Alphanumeric

Is the character string to be converted, or a temporary field that contains the string.

`outfield`
Alphanumeric

Is the name of the field that contains the result, or the format of the output value enclosed in single quotation marks. The length must be greater than or equal to the length of *length*.

Example: Converting a Character String to Mixed-Case

Use the COMPUTE or DEFINED dialog in the report development tools or the Synonym Editor in Developer Workbench to perform a conversion where LCWORD2 converts the string O'CONNOR's to mixed-case:

```
MYVAL1/A10='O'CONNOR'S';
LC2/A10 = LCWORD2(10, MYVAL1, 'A10');
```

The report output for these fields:

```
MYVAL1      LC2
------      ---
O'CONNOR'S  O'Connor's
```

LCWORD3: Converting a Character String to Mixed-Case

How to:

Convert a Character String to Mixed-Case Using LCWORD3

The LCWORD3 function converts the letters in a character string to mixed-case by converting the first letter of each word to uppercase and converting every other letter to lowercase. In addition, a single quotation mark indicates that the next letter should be converted to uppercase as long as it is neither followed by a blank nor the last character in the input string.

For example, 'SMITH' would be changed to 'Smith' and JACK'S would be changed to Jack's.

Syntax: How to Convert a Character String to Mixed-Case Using LCWORD3

```
LCWORD3(length, string, output)
```

where:

length

Integer

Is the length, in characters, of the character string or field to be converted, or a field that contains the length.

string

Alphanumeric

Is the character string to be converted, or a field that contains the string.

output

Alphanumeric

Is the name of the field that contains the result, or the format of the output value enclosed in single quotation marks. The length must be greater than or equal to the length of *length*.

Example: Converting a Character String to Mixed-Case Using LCWORD3

Use the Compute or Define dialog box in the report development tools or the Synonym Editor in Developer Workbench to perform a conversion where LCWORD3 converts the strings O'CONNOR’s and o’connor’s to mixed-case:

```
MYVAL1/A10='O'CONNOR'S';
MYVAL2/A10='o'connor's';
LC1/A10 = LCWORD3(10, MYVAL1, 'A10');
LC2/A10 = LCWORD3(10, MYVAL2, 'A10');
```

On the output, the letter *C* after the first single quotation mark is in uppercase because it is not followed by a blank and is not the final letter in the input string. The letter *s* after the second single quotation mark is in lowercase because it is the last character in the input string:

```
MYVAL1      LC1         MYVAL2      LC2
------      ---         ------      ---
O'CONNOR'S  O'Connor's  o'connor's  O'Connor's
```

LJUST: Left-Justifying a Character String

How to:

Left-Justify a Character String

The LJUST function left-justifies a character string within a field. All leading spaces become trailing spaces.

LJUST will not have any visible effect in a report that uses StyleSheets (SET STYLE=ON) unless you center the item.

Syntax: How to Left-Justify a Character String

```
LJUST(length, string, 'outfield')
```

where:

length

Integer

Is the length in characters of *string* and *outfield*, or a field that contains the length.

string

Alphanumeric

Is the character string to be justified, or a field that contains the string.

outfield

Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Example: Left-Justifying a Field

LJUST left-justifies the XNAME field and stores the result in YNAME:

```
COMPUTE YNAME/A25 = LJUST(15, XNAME, 'A25');
```

LOCASE: Converting Text to Lowercase

How to:

Convert Text to Lowercase

The LOCASE function converts alphanumeric text to lowercase.

Syntax: How to Convert Text to Lowercase

```
LOCASE(length, string, 'outfield')
```

where:

length
Integer

Is the length in characters of *string* and *outfield*, or a field that contains the length. The length must be greater than 0 and the same for both arguments; otherwise, an error occurs.

string
Alphanumeric

Is the character string to be converted in single quotation marks, or a field that contains the string.

outfield
Alphanumeric

Is the format of the output value enclosed in single quotation marks. The field name can be the same as *string*.

Example: Converting a Field to Lowercase

LOCASE converts the LAST_NAME field to lowercase and stores the result in LOWER_NAME:

```
COMPUTE LOWER_NAME/A15 = LOCASE(15, LAST_NAME, 'A15');
```

OVRLAY: Overlaying a Character String

How to:

Overlay a Character String

The OVRLAY function overlays a base character string with a substring.

Syntax: **How to Overlay a Character String**

```
OVRLAY(string1, stringlen, string2, sublen, position, 'outfield')
```

where:

`string1`

Alphanumeric

Is the base character string.

`stringlen`

Integer

Is the length in characters of *string1* and *outfield*, or a field that contains the length. If this argument is less than or equal to 0, unpredictable results occur.

`string2`

Alphanumeric

Is the substring that will overlay string1.

`sublen`

Integer

Is the length of string2, or a field that contains the length. If this argument is less than or equal to 0, the function returns spaces.

`position`

Integer

Is the position in the base string at which the overlay begins. If this argument is less than or equal to 0, the function returns spaces. If this argument is larger than *stringlen*, the function returns the base string.

`outfield`

Alphanumeric

Is the format of the output value enclosed in single quotation marks. If the overlaid string is longer than the output field, the string is truncated to fit the field.

Example: Replacing Characters in a Character String

OVRLAY replaces the last three characters of EMP_ID with CURR_JOBCODE to create a new security identification code and stores the result in NEW_ID:

```
COMPUTE NEW_ID/A9 = OVRLAY(EMP_ID, 9, CURR_JOBCODE, 3, 7, 'A9');
```

PARAG: Dividing Text Into Smaller Lines

How to:

Divide Text Into Smaller Lines

The PARAG function divides a line of text into smaller lines by marking them with a delimiter. It scans a specific number of characters from the beginning of the line and replaces the last space in the group scanned with the delimiter. It then scans the next group of characters in the line, starting from the delimiter, and replaces the last space in this group with a second delimiter. It repeats this process until reaching the end of the line.

Each group of characters marked off by the delimiter becomes a sub-line. The GETTOK function can then place the sub-lines into different fields. If the function does not find any spaces in the group it scans, it replaces the first character after the group with the delimiter. Therefore, make sure that no word of text is longer than the number of characters scanned (the maximum sub-line length).

If the input lines of text are roughly equal in length, you can keep the sub-lines equal by specifying a sub-line length that evenly divides into the length of the text lines. For example, if the text lines are 120 characters long, divide each of them into two sub-lines of 60 characters or three sub-lines of 40 characters. This technique enables you to print lines of text in paragraph form.

However, if you divide the lines evenly, you may create more sub-lines than you intend. For example, suppose you divide 120-character text lines into two lines of 60 characters maximum, but one line is divided so that the first sub-line is 50 characters and the second is 55. This leaves room for a third sub-line of 15 characters. To correct this, insert a space (using weak concatenation) at the beginning of the extra sub-line, then append this sub-line (using strong concatenation) to the end of the one before it.

Syntax: How to Divide Text Into Smaller Lines

```
PARAG(length, string, 'delim', subsize, 'outfield')
```

where:

length

Integer

Is the length in characters of *string* and *outfield*, or a field that contains the length.

string

Alphanumeric

Is the text enclosed in single quotation marks, or a field that contains the text.

delim

Alphanumeric

Is the delimiter enclosed in single quotation marks. Choose a character that does not appear in the text.

subsize

Integer

Is the maximum length of each sub-line.

outfield

Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Example: Dividing Text Into Smaller Lines

PARAG divides ADDRESS_LN2 into smaller lines of not more than ten characters using a comma as the delimiter. It then stores the result in PARA_ADDR:

```
COMPUTE PARA_ADDR/A20 = PARAG(20, ADDRESS_LN2, ',', 10, 'A20');
```

POSIT: Finding the Beginning of a Substring

How to:

Find the Beginning of a Substring

The POSIT function finds the starting position of a substring within a larger string. For example, the starting position of the substring DUCT in the string PRODUCTION is four. If the substring is not in the parent string, the function returns the value 0.

Syntax: How to Find the Beginning of a Substring

```
POSIT(parent, inlength, substring, sublength, 'outfield')
```

where:

parent

Alphanumeric

Is the parent character string enclosed in single quotation marks, or a field that contains the parent character string.

`inlength`

Integer

Is the length of the parent character string in characters, or a field that contains the length. If this argument is less than or equal to 0, the function returns a 0.

`substring`

Alphanumeric

Is the substring whose position you want to find. This can be the substring enclosed in single quotation marks, or the field that contains the string.

`sublength`

Integer

Is the length of *substring*. If this argument is less than or equal to 0, or if it is greater than *inlength*, the function returns a 0.

`outfield`

Integer

Is the format of the output value enclosed in single quotation marks.

Example: Finding the Position of a Letter

POSIT determines the position of the first capital letter I in LAST_NAME and stores the result in I_IN_NAME:

```
COMPUTE I_IN_NAME/I2 = POSIT(LAST_NAME, 15, 'I', 1, 'I2');
```

PTOA: Packed Decimal to Alphanumeric Conversion

The PTOA function converts a number from numeric format to alphanumeric format. It retains the decimal positions of the number and right-justifies it with leading spaces. You can also add edit options to a number converted by PTOA.

When using PTOA to convert a number containing decimals to a character string, you must specify an alphanumeric format large enough to accommodate both the integer and decimal portions of the number. For example, a P12.2C format is converted to A14. If the output format is not large enough, the right-most characters are truncated.

Reference: Packed Decimal to Alphanumeric Conversion

```
 PTOA(number, '(format)', output)
```

where:

`number`
: Numeric P (packed-decimal) or F or D (single and double precision floating-point)
: Is the number to be converted.

`format`
: Alphanumeric
: Is the format of the number enclosed in parenthesis.

`output`
: Alphanumeric

The length of this argument must be greater than the length of number and must account for edit options and a possible negative sign. For example, converting from packed to alphanumeric format converts PGROSS from packed-decimal to alphanumeric format.

```
PTOA(PGROSS, '(P12.2)', 'A14');
```

REVERSE: Reversing Characters in a Character String

How to:

Reverse Characters in a Character String

The REVERSE function reverses the characters in a character string.

Syntax: How to Reverse Characters in a Character String

```
REVERSE(length, string, 'outfield')
```

where:

`length`

Integer

Is the length in characters of *string* and *outfield*, or a field that contains the length.

`string`

Alphanumeric

Is the text enclosed in single quotation marks, or a field that contains the text.

`outfield`

Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Example: Reversing the Characters in a String

REVERSE reverses the characters in PRODCAT and stores the result in REVERSE_NAME:

```
COMPUTE REVERSE_NAME/A15 = REVERSE(15, PRODCAT, 'A15');
```

RJUST: Right-Justifying a Character String

How to:

Right-Justify a Character String

The RJUST function right-justifies a character string. All trailing blacks become leading blanks. This is useful when you display alphanumeric fields containing numbers.

RJUST does not have any visible effect in a report that uses StyleSheets (SET STYLE=ON) unless you center the item. Also, if you use RJUST on a platform on which StyleSheets are turned on by default, issue SET STYLE=OFF before running the request.

Syntax: How to Right-Justify a Character String

```
RJUST(length, string, 'outfield')
```

where:

length

Integer

Is the length in characters of *string* and *outfield*, or a field that contains the length. The lengths must be the same to avoid justification problems.

string

Alphanumeric

Is the character string, or a field that contains the character string enclosed in single quotation marks.

outfield

Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Example: Right-Justifying a Field

RJUST right-justifies the LAST_NAME field and stores the result in RIGHT_NAME:

```
COMPUTE RIGHT_NAME/A15 = RJUST(15, LAST_NAME, 'A15');
```

SOUNDEX: Comparing Character Strings Phonetically

How to:

Compare Character Strings Phonetically

The SOUNDEX function searches for a character string phonetically without regard to spelling. It converts character strings to four character codes. The first character must be the first character in the string. The last three characters represent the next three significant sounds in the character string.

To conduct a phonetic search, do the following:

1. Use SOUNDEX to translate data values from the field you are searching for to the phonetic codes.
2. Use SOUNDEX to translate your best guess target string to a phonetic code. Remember that the spelling of your target string need be only approximate; however, the first letter must be correct.
3. Use WHERE or IF criteria to compare the temporary fields created in Step 1 to the temporary field created in Step 2.

Syntax: How to Compare Character Strings Phonetically

```
SOUNDEX(inlength, string, 'outfield')
```

where:

inlength

2-byte Alphanumeric

Is the length, in characters, of *string*, or a field that contains the length. It can be a number enclosed in single quotation marks, or a field containing the number. The number must be from 01 to 99, expressed with two digits (for example '01'); a number larger than 99 causes the function to return asterisks (*) as output.

string

Alphanumeric

Is the character string enclosed in single quotation marks, or a field that contains the character string.

outfield

Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Example: Comparing Character Strings Phonetically

The following request creates three fields:

- PHON_NAME contains the phonetic code of employee last names.
- PHON_COY contains the phonetic code of your guess, MICOY.
- PHON_MATCH compares the guess with the phonetic code.

```
COMPUTE PHON_NAME/A4 = SOUNDEX('15', LAST_NAME, 'A4'); AND
COMPUTE PHON_COY/A4 WITH LAST_NAME = SOUNDEX('15', 'MICOY', 'A4'); AND
COMPUTE PHON_MATCH/A3 = IF PHON_NAME IS PHON_COY THEN 'YES' ELSE 'NO';
```

SPELLNM: Spelling Out a Dollar Amount

How to:

Spell Out a Dollar Amount

The SPELLNM function spells out an alphanumeric string or numeric value containing two decimal places as dollars and cents. For example, the value 32.50 is THIRTY TWO DOLLARS AND FIFTY CENTS.

Syntax: How to Spell Out a Dollar Amount

```
SPELLNM(outlength, number, 'outfield')
```

where:

outlength

Integer

Is the length of *outfield* in characters, or a field that contains the length.

If you know the maximum value of *number*, use the following table to determine the value of *outlength*:

If number is less than...	...outlength should be
$10	37
$100	45
$1,000	59
$10,000	74

If number is less than...	...outlength should be
$100,000	82
$1,000,000	96

`number`
: Alphanumeric or Decimal (9.2)

Is the number to be spelled out.

`outfield`
: Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Example: Spelling Out a Dollar Amount

SPELLNM spells out the values in CURR_SAL and stores the result in AMT_IN_WORDS:

```
COMPUTE AMT_IN_WORDS/A82 = SPELLNM(82, CURR_SAL, 'A82');
```

SUBSTR: Extracting a Substring

How to:

Extract a Substring

The SUBSTR function extracts a substring based on where it begins and its length in the parent string. SUBSTR can vary the position of the substring depending on the values of other fields.

Syntax: How to Extract a Substring

```
SUBSTR(inlength, parent, start, end, sublength, 'outfield')
```

where:

`inlength`
: Integer

Is the length of the parent string in characters, or a field that contains the length.

`parent`
: Alphanumeric

Is the parent string enclosed in single quotation marks, or the field containing the parent string.

`start`

Integer

Is the starting position of the substring in the parent string. If this argument is less than one, the function returns spaces.

`end`

Integer

Is the ending position of the substring. If this argument is less than *start* or greater than *inlength*, the function returns spaces.

`sublength`

Integer

Is the length of the substring (normally end - start + 1). If *sublength* is longer than *end* - *start* +1, the substring is padded with trailing spaces. If it is shorter, the substring is truncated. This value should be the declared length of *outfield*. Only *sublength* characters will be processed.

`outfield`

Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Example: Extracting a String

POSIT determines the position of the first letter I in LAST_NAME and stores the result in I_IN_NAME. SUBSTR then extracts three characters beginning with the letter I from LAST_NAME, and stores the results in I_SUBSTR.

```
COMPUTE I_IN_NAME/I2 = POSIT(LAST_NAME, 15, 'I', 1, 'I2'); AND
COMPUTE I_SUBSTR/A3 = SUBSTR(15, LAST_NAME, I_IN_NAME, I_IN_NAME+2, 3,
'A3');
```

UPCASE: Converting Text to Uppercase

How to:

Convert Text to Uppercase

The UPCASE function converts a character string to uppercase. It is useful for sorting on a field that contains both mixed-case and uppercase values. Sorting on a mixed-case field produces incorrect results because the sorting sequence in EBCDIC always places lowercase letters before uppercase letters, while the ASCII sorting sequence always places uppercase letters before lowercase. To obtain correct results, define a new field with all of the values in uppercase, and sort on that.

Syntax: How to Convert Text to Uppercase

```
UPCASE(length, input, 'outfield')
```

where:

`length`
: Integer

Is the length in characters of *input* and *outfield*.

`input`
: Alphanumeric

Is the character string enclosed in single quotation marks, or the field containing the character string.

`outfield`
: Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Example: Converting a Mixed-Case Field to Uppercase

UPCASE converts the LAST_NAME_MIXED field to uppercase and stores the result in LAST_NAME_UPPER:

```
COMPUTE LAST_NAME_UPPER/A15 = UPCASE(15, LAST_NAME_MIXED, 'A15') ;
```

Data Source and Decoding Functions

In this section:

DECODE: Decoding Values

LAST: Retrieving the Preceding Value

Restricting Access to Values Using WHERE Criteria

Data source and decoding functions retrieve data source values and assign values based on the value of an input field.

DECODE: Decoding Values

How to:

Supply Values in the Function

Read Values From a File

Reference:

Guidelines for Reading Values From a File

The DECODE function assigns values based on the coded value of an input field. DECODE is useful for giving a more meaningful value to a coded value in a field. For example, the field GENDER may have the code F for female employees and M for male employees for efficient storage (for example, one character instead of six for female). DECODE expands (decodes) these values to ensure correct interpretation on a report.

You can use DECODE by supplying values directly in the function or by reading values from a separate file.

Syntax: How to Supply Values in the Function

```
DECODE fieldname(code1 result1 code2 result2...[ELSE default]);
```

where:

fieldname
Alphanumeric or Numeric

Is the name of the input field.

code
Alphanumeric or Numeric

Is the coded value for which DECODE searches. If the value has embedded blanks, commas, or other special characters, enclose it in single quotation marks. When DECODE finds the specified value, it assigns the corresponding result.

result

Alphanumeric or Numeric

Is the value assigned to a code. If the value has embedded blanks or commas or contains a negative number, enclose it in single quotation marks.

default

Alphanumeric or Numeric

Is the value assigned if the code is not found. If you omit a default value, DECODE assigns a blank or zero to non-matching codes.

You can use up to 40 lines to define the code and result pairs for any given DECODE function, or 39 lines if you also use an ELSE phrase. Use either a comma or blank to separate the code from the result, or one pair from another.

Example: Supplying Values in the Function

EDIT extracts the first character of the CURR_JOBCODE field, then DECODE returns either ADMINISTRATIVE or DATA PROCESSING depending on the value extracted.

```
COMPUTE DEPX_CODE/A1 = EDIT(CURR_JOBCODE, '9$$'); AND
COMPUTE JOB_CATEGORY/A15 = DECODE DEPX_CODE(A 'ADMINISTRATIVE'
B 'DATA PROCESSING') ;
```

Syntax: How to Read Values From a File

```
DECODE fieldname(ddname [ELSE default]);
```

where:

fieldname

Alphanumeric or Numeric

Is the name of the input field.

ddname

Alphanumeric

Is a logical name or a shorthand name that points to the physical file containing the decoded values.

default

Alphanumeric or Numeric

Is the value assigned if the code is not found. If you omit a default, DECODE assigns a blank or zero to non-matching codes.

Reference: Guidelines for Reading Values From a File

- Each record in the file is expected to contain pairs of elements separated by a comma or blank.
- If each record in the file consists of only one element, this element is interpreted as the code, and the result becomes either a blank or zero, as needed.

  This makes it possible to use the file to hold screening literals referenced in the screening condition

  ```
  IF field IS (filename)
  ```

  and as a file of literals for an IF criteria specified in a computational expression. For example:

  ```
  TAKE = DECODE SELECT (filename ELSE 1);
  VALUE = IF TAKE IS 0 THEN... ELSE...;
  ```

  TAKE is 0 for SELECT values found in the literal file and 1 in all other cases. The VALUE computation is carried out as if the expression had been:

  ```
  IF SELECT (filename) THEN... ELSE...;
  ```

- The file can contain up to 32,767 characters in the file.
- Leading and trailing blanks are ignored.
- The remainder of each record is ignored and can be used for comments or other data. This convention applies in all cases, except when the file name is HOLD. In that case, the file is presumed to have been created by the HOLD command, which writes fields in the internal format, and the DECODE pairs are interpreted accordingly. In this case, extraneous data in the record is ignored.

Example: Reading Values From a File

DECODE assigns the value 0 to an employee whose EMP_ID appears in the HOLD file and 1 when EMP_ID does not appear in the file.

```
COMPUTE NOT_IN_LIST/I1 = DECODE EMP_ID(HOLD ELSE 1);
```

LAST: Retrieving the Preceding Value

How to:

Retrieve the Preceding Value

The LAST function retrieves the preceding value for a field.

The effect of LAST depends on whether it appears in a DEFINE or COMPUTE command:

- In a DEFINE command, the LAST value applies to the previous record retrieved from the data source before sorting takes place.
- In a COMPUTE command, the LAST value applies to the record in the previous line of the internal matrix.

Syntax: How to Retrieve the Preceding Value

```
LAST fieldname
```

where:

fieldname
Alphanumeric or Numeric

Is the field name.

Example: Retrieving the Preceding Value

LAST retrieves the previous value of the DEPARTMENT field to determine whether to restart the running total of salaries by department. If the previous value equals the current value, CURR_SAL is added to RUN_TOT to generate a running total of salaries within each department.

```
COMPUTE RUN_TOT/D12.2M = IF DEPARTMENT EQ LAST DEPARTMENT THEN
                (RUN_TOT + CURR_SAL) ELSE CURR_SAL ;
```

Restricting Access to Values Using WHERE Criteria

How to:

Restrict Access to Values Using WHERE Criteria

Using DBA security, you can restrict access to specific values in a data source. The RESTRICT=VALUE attribute supports those criteria that are supported by the IF phrase. The RESTRICT=VALUE_WHERE attribute supports all criteria supported in a WHERE phrase, including comparison between fields and use of functions. The WHERE expression will be passed to a configured adapter when possible.

Syntax: How to Restrict Access to Values Using WHERE Criteria

```
USER=password, ACCESS={R|RW|W},
   RESTRICT=VALUE_WHERE, NAME=name,
   VALUE=expression; ,$
```

where:

`password`

Is the user password, up to 64 characters.

`ACCESS={R|RW|W}`

Is the type of access the user is granted: Read, Read/Write, or Write.

`RESTRICT=VALUE_WHERE`

Specifies that the user can have access to only those values that satisfy the expression described in the VALUE attribute.

`name`

Is the name of the field or segment to restrict. NAME=SYSTEM, which can only be used with value tests, restricts every segment in the data source, including descendant segments. Multiple fields or segments can be specified by issuing the RESTRICT attribute several times for one user.

`VALUE=`*`expression`*

Specifies an expression that describes the values to which the user has access. The expression can use all elements supported by a WHERE test in a TABLE request.

Date and Time Functions

In this section:

Date and time functions manipulate date and time values.

When using standard date and time functions, you need to understand the settings that alter the behavior of these functions, as well as the acceptable formats and how to supply values in these formats.

You can affect the behavior of date and time functions by defining which days of the week are work days and which are not. Then, when you use a date function involving work days, dates that are not work days are ignored.

AYM: Adding or Subtracting Months to or From Dates

How to:

Add or Subtract Months to or From a Date

The AYM function adds months to or subtracts months from a date in year-month format. You can convert a date to this format using the CHGDAT or EDIT function.

Syntax: **How to Add or Subtract Months to or From a Date**

```
AYM(indate, months, 'outfield')
```

where:

indate

Integer (I4, I4YM, I6, or I6YYM)

Is the original date in year-month format, the name of a field that contains the date, or an expression that returns the date. If the date is not valid, the function returns a 0.

months

Integer

Is the number of months you are adding to or subtracting from the date. To subtract months, use a negative number.

outfield

Integer (I4YM or I6YYM)

Is the format of the output value enclosed in single quotation marks.

Tip: If the input date is in integer year-month-day format (I6YMD or I8YYMD), divide the date by 100 to convert to year-month format and set the result to an integer. This drops the day portion of the date, which is now after the decimal point.

Example: Adding Months to a Date

The COMPUTE command converts the dates in HIRE_DATE from year-month-day to year-month format and stores the result in HIRE_MONTH. AYM then adds six months to HIRE_MONTH and stores the result in AFTER6MONTHS.

```
COMPUTE HIRE_MONTH/I4YM = HIRE_DATE/100; AND
COMPUTE AFTER6MONTHS/I4YM = AYM(HIRE_MONTH, 6, 'I4YM');
```

AYMD: Adding or Subtracting Days to or From a Date

How to:

Add or Subtract Days to or From a Date

The AYMD function adds days to or subtracts days from a date in year-month-day format. You can convert a date to this format using the CHGDAT or EDIT function.

If the addition or subtraction of days crosses forward or backward into another century, the century digits of the output year are adjusted.

Syntax: How to Add or Subtract Days to or From a Date

```
AYMD(indate, days, 'outfield')
```

where:

indate

Integer (I6, I6YMD, I8, I8YYMD)

If the date is not valid, the function returns a 0.

days

Integer

Is the number of days you are adding to or subtracting from *indate*. To subtract days, use a negative number.

outfield

Integer (I6, I6YMD, I8, or I8YYMD)

Is the format of the output value enclosed in single quotation marks. If *indate* is a field, *outfield* must have the same format.

Example: Adding Days to a Date

AYMD adds 35 days to each value in the HIRE_DATE field, and stores the result in AFTER35DAYS:

```
COMPUTE AFTER35DAYS/I6YMD = AYMD(HIRE_DATE, 35, 'I6YMD');
```

CHGDAT: Changing How a Date String Displays

How to:

Change the Date Display String

Reference:

Short to Long Conversion

Usage Notes for CHGDAT

The CHGDAT function rearranges the year, month, and day portions of an input character string representing a date. It may also convert the input string from long to short or short to long date representation. Long representation contains all three date components: year, month, and day; short representation omits one or two of the date components, such as year, month, or day. The input and output date strings are described by display options that specify both the order of date components (year, month, day) in the date string and whether two or four digits are used for the year (for example, 97 or 1997). CHGDAT reads an input date character string and creates an output date character string that represents the same date in a different way.

Note: CHGDAT requires a date character string as input, not a date itself. Convert the input to a date character string (using the EDIT or DATECVT functions, for example) before applying CHGDAT.

The order of date components in the date character string is described by display options comprised of the following characters in your chosen order:

Character	Description
`D`	Day of the month (01 through 31).
`M`	Month of the year (01 through 12).
`Y[Y]`	Year. `Y` indicates a two-digit year (such as 94); `YY` indicates a four-digit year (such as 1994).

To spell out the month rather than use a number in the resulting string, append one of the following characters to the display options for the resulting string:

Character	Description
T	Displays the month as a three-letter abbreviation.
X	Displays the full name of the month.

Display options can consist of up to five display characters. Characters other than those display options are ignored.

For example: The display options 'DMYY' specify that the date string starts with a two digit day, then two digit month, then four digit year.

Note: Display options are *not* date formats.

Reference: Short to Long Conversion

If you are converting a date from short to long representation (for example, from year-month to year-month-day), the function supplies the portion of the date missing in the short representation, as shown in the following table:

Portion of Date Missing	Portion Supplied by Function
Day (for example, from YM to YMD)	Last day of the month.
Month (for example, from Y to YM)	Last month of the year (December).
Year (for example, from MD to YMD)	The year 99.
Converting year from two-digit to four-digit (for example, from YMD to YYMD)	If DATEFNS=ON, the century will be determined by the 100-year window defined by DEFCENT and YRTHRESH. If DATEFNS=OFF, the year 19*xx* is supplied, where *xx* is the last two digits in the year.

Syntax: How to Change the Date Display String

```
CHGDAT('in_display_options', 'out_display_options', date_string, 'outfield')
```

where:

`in_display_options`

Alphanumeric (A1 to A5)

Is a series of up to five display options that describe the layout of *date_string*. These options can be stored in an alphanumeric field or supplied as a literal enclosed in single quotation marks.

`out_display_options`

Alphanumeric (A1 to A5)

Is a series of up to five display options that describe the layout of the converted date string. These options can be stored in an alphanumeric field or supplied as a literal enclosed in single quotation marks.

`date_string`

Alphanumeric (A2 to A8)

Is the input date character string with date components in the order specified by *in_display_options*.

Note that if the original date is in numeric format, you must convert it to a date character string. If *date_string* does not correctly represent the date (the date is invalid), the function returns blank spaces.

`outfield`

Alphanumeric

Is the format of the output value enclosed in single quotation marks.

A*xx*, where *xx* is a number of characters large enough to fit the date string specified by *out_display_options*. A17 is long enough to fit the longest date string.

Reference: Usage Notes for CHGDAT

Since CHGDAT uses a date string (as opposed to a date) and returns a date string with up to 17 characters, use the EDIT or DATECVT functions or any other means to convert the date to or from a date character string.

Example: Converting the Date Display From YMD to MDYYX

The EDIT function changes HIRE_DATE from numeric to alphanumeric format. CHGDAT then converts each value in ALPHA_HIRE from displaying the components as YMD to MDYYX and stores the result in HIRE_MDY, which has the format A17. The option X in the output value displays the full name of the month.

```
COMPUTE ALPHA_HIRE/A17 = EDIT(HIRE_DATE); AND
COMPUTE HIRE_MDY/A17 = CHGDAT('YMD', 'MDYYX', ALPHA_HIRE, 'A17');
```

DA Functions: Converting a Date to an Integer

How to:

Convert a Date to an Integer

The DA functions convert a date to the number of days between December 31, 1899 and that date. By converting a date to the number of days, you can add and subtract dates and calculate the intervals between them.

There are six DA functions; each one accepts a date in a different format.

Syntax: How to Convert a Date to an Integer

```
function(indate, 'outfield')
```

where:

`function`

Is one of the following:

`DADMY` converts a date in day-month-year format.

`DADYM` converts a date in day-year-month format.

`DAMDY` converts a date in month-day-year format.

`DAMYD` converts a date in month-year-day format.

`DAYDM` converts a date in year-day-month format.

`DAYMD` converts a date in year-month-day format.

`indate`

Integer or Packed

I or P format with date display options.

Is the date to be converted, or the name of a field that contains the date. The date is truncated to an integer before conversion.

To specify the year, enter only the last two digits; the function assumes the century component. If the date is invalid, the function returns a 0.

outfield

Integer

Is the format of the output value enclosed in single quotation marks. The format of the date returned depends on the function.

Example: Converting Dates and Calculating the Difference Between Them

DAYMD converts the DAT_INC and HIRE_DATE fields to the number of days since December 31, 1899, and the smaller number is then subtracted from the larger number:

```
COMPUTE DAYS_HIRED/I8 = DAYMD(DAT_INC, 'I8') - DAYMD(HIRE_DATE, 'I8');
```

DATEADD: Adding or Subtracting a Date Unit to or From a Date

How to:

Add or Subtract a Date Unit to or From a Date

The DATEADD function adds a unit to or subtracts a unit from a date format. A unit is one of the following:

- **Year.**
- **Month.** If the calculation using the month unit creates an invalid date, DATEADD corrects it to the last day of the month. For example, adding one month to October 31 yields November 30, not November 31 since November has 30 days.
- **Day.**
- **Weekday.** When using the weekday unit, DATEADD does not count Saturday or Sunday. For example, if you add one day to Friday, the result is Monday.
- **Business day.** When using the business day unit, DATEADD uses the BUSDAYS parameter setting and holiday file to determine which days are working days and disregards the rest. If Monday is not a working day, then one business day past Sunday is Tuesday.

You add or subtract non day-based dates (for example, YM or YQ) directly without using DATEADD.

DATEADD works only with full component dates.

Syntax: How to Add or Subtract a Date Unit to or From a Date

```
DATEADD(date, 'unit', #units)
```

where:

date

Date

Is a full component date.

unit

Alphanumeric

Is one of the following enclosed in single quotation marks:

`Y` indicates a year unit.

`M` indicates a month unit.

`D` indicates a day unit.

`WD` indicates a weekday unit.

`BD` indicates a business day unit.

#units

Integer

Is the number of date units added to or subtracted from *date*. If this number is not a whole unit, it is rounded down to the next largest integer.

Example: Adding Weekdays to a Date

DATEADD adds three weekdays to NEW_DATE. In some cases, it adds more than three days because HIRE_DATE_PLUS_THREE would otherwise be on a weekend.

```
COMPUTE NEW_DATE/YYMD = HIRE_DATE; AND
COMPUTE HIRE_DATE_PLUS_THREE/YYMD = DATEADD(NEW_DATE, 'WD', 3);
```

DATECVT: Converting the Format of a Date

How to:

Convert a Date Format

The DATECVT function converts the format of a date in an application without requiring an intermediate calculation. If you supply an invalid format, DATECVT returns a zero or a blank.

The DATECVT function is optimized when using the following date formats: A8YYMD, A8MDYY, A8DMYY, I8YYMD, I8MDYY, I8DMYY, P8YYMD, P8MDYY, and P8DMYY. Optimized SQL is now passed to the DB2 Engine for processing when converting dates.

Note: You can use simple assignment instead of calling this function.

Syntax: How to Convert a Date Format

```
DATECVT(date, 'infmt', 'outfmt')
```

where:

date

Date

Is the date to be converted. If you supply an invalid date, DATECVT returns zero. When the conversion is performed, a legacy date obeys any DEFCENT and YRTHRESH parameter settings supplied for that field.

infmt

Alphanumeric

Is the format of the date enclosed in single quotation marks. It is one of the following:

- ❑ A non-legacy date format (for example, YYMD, YQ, M, DMY, JUL).
- ❑ A legacy date format (for example, I6YMD or A8MDYY).
- ❑ A non-date format (such as I8 or A6). A non-date format in *infmt* functions as an offset from the base date of a YYMD field (12/31/1900).

outfmt

Alphanumeric

Is the output format enclosed in single quotation marks. It is one of the following:

- ❑ A non-legacy date format (for example, YYMD, YQ, M, DMY, JUL).
- ❑ A legacy date format (for example, I6YMD or A8MDYY).
- ❑ A non-date format (such as I8 or A6). A non-date format in *infmt* functions as an offset from the base date of a YYMD field (12/31/1900).

Example: Converting a YYMD Date to DMY

DATECVT converts 19991231 to 311299 and stores the result in CONV_FIELD:

```
COMPUTE CONV_FIELD/DMY = DATECVT(19991231, 'I8YYMD', 'DMY');
```

or

```
COMPUTE CONV_FIELD/DMY = DATECVT('19991231', 'A8YYMD', 'DMY');
```

Example: Converting a Legacy Date to Date Format

DATECVT converts HIRE_DATE from I6YMD legacy date format to YYMD date format:

```
COMPUTE NEW_HIRE_DATE/YYMD = DATECVT(HIRE_DATE, 'I6YMD', 'YYMD');
```

DATEDIF: Finding the Difference Between Two Dates

How to:

Find the Difference Between Two Dates

The DATEDIF function returns the difference between two dates in units. A unit is one of the following:

- **Year.** Using the year unit with DATEDIF yields the inverse of DATEADD. If subtracting one year from date X creates date Y, then the count of years between X and Y is one. Subtracting one year from February 29 produces the date February 28.
- **Month.** Using the month unit with DATEDIF yields the inverse of DATEADD. If subtracting one month from date X creates date Y, then the count of months between X and Y is one. If the to-date is the end-of-month, then the month difference may be rounded up (in absolute terms) to guarantee the inverse rule.

  If one or both of the input dates is the end of the month, DATEDIF takes this into account. This means that the difference between January 31 and April 30 is three months, not two months.
- **Day.**
- **Weekday.** With the weekday unit, DATEDIF does not count Saturday or Sunday when calculating days. This means that the difference between Friday and Monday is one day.
- **Business day.** With the business day unit, DATEDIF uses the BUSDAYS parameter setting and holiday file to determine which days are working days and disregards the rest. This means that if Monday is not a working day, the difference between Friday and Tuesday is one day.

DATEDIF returns a whole number. If the difference between two dates is not a whole number, DATEDIF truncates the value to the next largest integer. For example, the number of years between March 2, 2001, and March 1, 2002, is zero. If the end date is before the start date, DATEDIF returns a negative number.

You can find the difference between non-day based dates (for example YM or YQ) directly without using DATEDIF.

Syntax: How to Find the Difference Between Two Dates

`DATEDIF(`*`from_date`*`, `*`to_date`*`, '`*`unit`*`')`

where:

`from_date`

Date

Is the start date from which to calculate the difference. Is a full component date.

`to_date`

Date

Is the end date from which to calculate the difference.

`unit`

Alphanumeric

Is one of the following enclosed in single quotation marks:

`Y` indicates a year unit.

`M` indicates a month unit.

`D` indicates a day unit.

`WD` indicates a weekday unit.

`BD` indicates a business day unit.

Example: Finding the Number of Weekdays Between Two Dates

DATECVT converts the legacy dates in HIRE_DATE and DAT_INC to the date format YYMD. DATEDIF then uses those date formats to determine the number of weekdays between NEW_HIRE_DATE and NEW_DAT_INC:

```
COMPUTE NEW_HIRE_DATE/YYMD = DATECVT(HIRE_DATE, 'I6YMD', 'YYMD'); AND
COMPUTE NEW_DAT_INC/YYMD = DATECVT(DAT_INC, 'I6YMD', 'YYMD'); AND
COMPUTE WDAYS_HIRED/I8 = DATEDIF(NEW_HIRE_DATE, NEW_DAT_INC, 'WD');
```

DATEMOV: Moving a Date to a Significant Point

How to:

Move a Date to a Significant Point

The DATEMOV function moves a date to a significant point on the calendar.

DATEMOV works only with full component dates.

Syntax: How to Move a Date to a Significant Point

`DATEMOV(date, 'move-point')`

where:

`date`

Date

Is a full component date. Is the date to be moved.

`move-point`

Alphanumeric

Is the significant point the date is moved to enclosed in single quotation marks. An invalid point results in a return code of zero. Valid values are:

`EOM` is the end of month.

`BOM` is the beginning of month.

`EOQ` is the end of quarter.

`BOQ` is the beginning of quarter.

`EOY` is the end of year.

`BOY` is the beginning of year.

`EOW` is the end of week.

`BOW` is the beginning of week.

`NWD` is the next weekday.

`NBD` is the next business day.

`PWD` is the prior weekday.

`PBD` is the prior business day.

`WD-` is a weekday or earlier.

`BD-` is a business day or earlier.

`WD+` is a weekday or later.

`BD+` is a business day or later.

A business day calculation is affected by the BUSDAYS and HDAY parameter settings.

Example: Determining the End of the Week

DATEMOV determines the end of the week for each date in NEW_DATE and stores the result in EOW:

```
COMPUTE NEW_DATE/YYMDWT = DATECVT(HIRE_DATE, 'I6YMD', 'YYMDWT'); AND
COMPUTE EOW/YYMDWT = DATEMOV(NEW_DATE, 'EOW');
```

Returning a Date Component as an Integer

How to:

Extract a Date Component and Return It in Integer Format

The DPART function extracts a specified component from a date field and returns it in numeric format.

Syntax: How to Extract a Date Component and Return It in Integer Format

```
DPART(datevalue, 'component', outfield)
```

where:

datevalue

Date

Is a full component date.

component

Alphanumeric

Is the name of the component to be retrieved, enclosed in single quotation marks. Valid values are the following:

For year: YEAR, YY

For month: MONTH, MM

For day: DAY, for day of month: DAY-OF-MONTH.

For quarter: QUARTER, QQ

outfield

Integer

Is the field that contains the result, or the integer format of the output value enclosed in single quotation marks.

Example: Extracting Date Components in Integer Format

The following request against the VIDEOTRK data source uses the DPART function to extract the year, month, and day component from the TRANSDATE field.

```
DEFINE FILE
 VIDEOTRK
 YEAR/I4 = DPART(TRANSDATE, 'YEAR', 'I4');
 MONTH/I4 = DPART(TRANSDATE, 'MM', 'I4');
 DAY/I4 = DPART(TRANSDATE, 'DAY', 'I4');
END

TABLE FILE VIDEOTRK
PRINT TRANSDATE YEAR MONTH DAY
BY LASTNAME BY FIRSTNAME
WHERE LASTNAME LT 'DIAZ'
END
```

The output is:

```
LASTNAME         FIRSTNAME   TRANSDATE  YEAR  MONTH   DAY
--------         ---------   ---------  ----  -----   ---
ANDREWS          NATALIA     91/06/19   1991      6    19
                             91/06/18   1991      6    18
BAKER            MARIE       91/06/19   1991      6    19
                             91/06/17   1991      6    17
BERTAL           MARCIA      91/06/23   1991      6    23
                             91/06/18   1991      6    18
CHANG            ROBERT      91/06/28   1991      6    28
                             91/06/27   1991      6    27
                             91/06/26   1991      6    26
COLE             ALLISON     91/06/24   1991      6    24
                             91/06/23   1991      6    23
CRUZ             IVY         91/06/27   1991      6    27
DAVIS            JASON       91/06/24   1991      6    24
```

DATETRAN: Formatting Dates in International Formats

How to:

Format Dates in International Formats

Reference:

Usage Notes for the DATETRAN Function

The DATETRAN function formats dates in international formats.

Syntax: How to Format Dates in International Formats

```
DATETRAN (indate, '(intype)', '([formatops])', 'lang', outlen, 'outfield')
```

where:

indate

Is the input date to be formatted. Note that the date format cannot be an alphanumeric or numeric format with date display options.

intype

Is one of the following character strings indicating the input date components and the order in which you want them to display, enclosed in single quotation marks and parentheses.

These are the single component input types:

Single Component Input Type	Description
`'(W)'`	Day of week component only (original format must have only W component).
`'(M)'`	Month component only (original format must have only M component).

These are the two-component input types:

Two-Component Input Type	Description
`'(YYM)'`	Four-digit year followed by month.
`'(YM)'`	Two-digit year followed by month.
`'(MYY)'`	Month component followed by four-digit year.
`'(MY)'`	Month component followed by two-digit year.

These are the three-component input types:

Three- Component Input Type	Description
`'(YYMD)'`	Four-digit year followed by month followed by day.
`'(YMD)'`	Two-digit year followed by month followed by day.
`'(DMYY)'`	Day component followed by month followed by four-digit year.
`'(DMY)'`	Day component followed by month followed by two-digit year.
`'(MDYY)'`	Month component followed by day followed by four-digit year.
`'(MDY)'`	Month component followed by day followed by two-digit year.
`'(MD)'`	Month component followed by day (derived from three-component date by ignoring year component).
`'(DM)'`	Day component followed by month (derived from three-component date by ignoring year component).

`formatops`

Is a string of zeros or more formatting options enclosed in parentheses and single quotation marks. The parentheses and quotation marks are required even if you do not specify formatting options. Formatting options are as follows:

- Options for suppressing initial zeros in month or day numbers.
- Options for translating month or day components to full or abbreviated uppercase or default case (mixed case or lowercase depending on the language) names.
- Date delimiter options and options for punctuating a date with commas.

Valid options for suppressing initial zeros in month or day numbers are:

Format Option	Description
`m`	Zero-suppresses months (displays numeric months before October as 1 through 9 rather than 01 through 09).
`d`	Displays days before the tenth of the month as 1 through 9 rather than 01 through 09.

Format Option	Description
`dp`	Displays days before the tenth of the month as 1 through 9 rather than 01 through 09 with a period after the number.
`do`	Displays days before the tenth of the month as 1 through 9. For English (langcode EN) only, displays an ordinal suffix (st, nd, rd, or th) after the number.

Valid month and day name translation options are:

Format Option	Description
`T`	Displays month as an abbreviated name with no punctuation, all uppercase.
`TR`	Displays month as a full name, all uppercase.
`Tp`	Displays month as an abbreviated name followed by a period, all uppercase.
`t`	Displays month as an abbreviated name with no punctuation. The name is all lowercase or initial uppercase, depending on language code.
`tr`	Displays month as a full name. The name is all lowercase or initial uppercase, depending on language code.
`tp`	Displays month as an abbreviated name followed by a period. The name displays in the default case of the specified language (for example, all lowercase for French and Spanish, initial uppercase for English and German).
`W`	Includes an abbreviated day of the week name at the start of the displayed date, all uppercase with no punctuation.
`WR`	Includes a full day of the week name at the start of the displayed date, all uppercase.
`Wp`	Includes an abbreviated day of the week name at the start of the displayed date, all uppercase, followed by a period.

Format Option	Description
w	Includes an abbreviated day of the week name at the start of the displayed date with no punctuation. The name displays in the default case of the specified language (for example, all lowercase for French and Spanish, initial uppercase for English and German).
wr	Includes a full day of the week name at the start of the displayed date. The name displays in the default case of the specified language (for example, all lowercase for French and Spanish, initial uppercase for English and German).
wp	Includes an abbreviated day of the week name at the start of the displayed date followed by a period. The name displays in the default case of the specified language (for example, all lowercase for French and Spanish, initial uppercase for English and German).
X	Includes an abbreviated day of the week name at the end of the displayed date, all uppercase with no punctuation.
XR	Includes a full day of the week name at the end of the displayed date, all uppercase.
Xp	Includes an abbreviated day of the week name at the end of the displayed date, all uppercase, followed by a period.
x	Includes an abbreviated day of the week name at the end of the displayed date with no punctuation. The name displays in the default case of the specified language (for example, all lowercase for French and Spanish, initial uppercase for English and German).
xr	Includes a full day of the week name at the end of the displayed date. The name displays in the default case of the specified language (for example, all lowercase for French and Spanish, initial uppercase for English and German).
xp	Includes an abbreviated day of the week name at the end of the displayed date followed by a period. The name displays in the default case of the specified language (for example, all lowercase for French and Spanish, initial uppercase for English and German).

Valid date delimiter options are:

Format Option	Description
B	Uses a blank as the component delimiter. This is the default if the month or day of week is translated or if comma is used.
.	Uses a period as the component delimiter.
-	Uses a minus sign as the component delimiter. This is the default when the conditions for a blank default delimiter are not satisfied.
/	Uses a slash as the component delimiter.
|	Omits component delimiters.
K	Uses appropriate Asian characters as component delimiters.
c	Places a comma after the month name (following T, Tp, TR, t, tp, or tr). Places a comma and blank after the day name (following W, Wp, WR, w, wp, or wr). Places a comma and blank before the day name (following X, XR, x, or xr).
e	Displays the Spanish or Portuguese word de or DE between the day and month and between the month and year. The case of the word de is determined by the case of the month name. If the month is displayed in uppercase, DE is displayed; otherwise de is displayed. Useful for formats DMY, DMYY, MY, and MYY.
D	Inserts a comma after the day number and before the general delimiter character specified.
Y	Inserts a comma after the year and before the general delimiter character specified.

lang

Is the two-character standard ISO code for the language into which the date should be translated, enclosed in single quotation marks. Valid language codes are:

'AR' Arabic

'CS' Czech

'DA'	Danish
'DE'	German
'DU'	Dutch
'EN'	English
'ES'	Spanish
'FI'	Finnish
'FR'	French
'EL'	Greek
'IW'	Hebrew
'IT'	Italian
'JA'	Japanese
'KO'	Korean
'LT'	Lithuanian
'NO'	Norwegian
'PO'	Polish
'PT'	Portuguese
'RU'	Russian
'SV'	Swedish
'TH'	Thai
'TR'	Turkish
'TW'	Chinese (Traditional)
'ZH'	Chinese (Simplified)

outlen

Numeric

Is the length of the output field in bytes. If the length is insufficient, an all blank result is returned. If the length is greater than required, the field is padded with blanks on the right.

outfield

Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Reference: Usage Notes for the DATETRAN Function

- The type A output field may contain variable length information, since the lengths of month names and day names can vary. Also month and day numbers may be either one or two bytes long if a zero-suppression option is chosen. Unused bytes are filled with blanks.
- All invalid and inconsistent inputs result in all blank output strings. Missing data also results in blank output.
- The base dates (1900-12-31 and 1900-12 or 1901-01) are treated as though the DATEDISPLAY setting were ON (that is, not automatically shown as blanks). To suppress the printing of base dates, which have an internal integer value of 0, test for 0 before calling DATETRAN. For example:

```
RESULT/A40 = IF DATE EQ 0 THEN ' ' ELSE
                DATETRAN (DATE, '(YYMD)', '(.t)', 'FR', 40, 'A40');
```

- Valid translated date components are contained in files named DTLNG*lng* where *lng* is a three-character code that specifies the language. These files must be accessible for each language into which you want to translate dates.

Example: Using the DATETRAN Function

DATETRAN prints the day of the week in the default case for French:

```
COMPUTE OUT/A8=DATETRAN(DATEW, '(W)', '(wr)', 'FR', 8 , 'A8') ;
```

Precision for Date-Time Values

How to:

Specify Precision for Date-Time Values

In prior releases, the seconds component of a date-time value could be displayed with zero, three, or six decimal digits. It can now be displayed with zero through nine decimal digits.

In order to control the precision of the seconds component in a date-time format, you can specify a digit from 1 to 9 in the format.

Syntax: How to Specify Precision for Date-Time Values

Special format components exist for three decimal digits (milliseconds, s), six decimal digits (microseconds, m), and nine decimal digits (nanoseconds, n). To display:

- **A Time-only value**, when the format includes more than one time display component:

    - The components must appear in the order hour, minute, second, millisecond, microsecond, or nanosecond.
    - The first component must be either hour, minute, or second.
    - No intermediate component can be skipped. If hour is specified, the next component must be minute; it cannot be second.
    - To display a specific number of digits, include the format for seconds (S) followed by the number of digits to display. Alternatively, you can specify one or more of the following components: milliseconds (*s*), microseconds (*m*), nanoseconds (*n*).
- **Both a date component and a time component**, the time component is represented by one character, which specifies the smallest unit of time to be displayed. This character can be either:
    - A number from 1 to 9, which specifies the number of decimal digits to display for the seconds component.
    - One of the supported time components. In this case, all higher-order time components will also be included in the display.

Example: Specifying Precision for Date-Time Values

Use the COMPUTE or DEFINE dialog in the report development tools or the Synonym Editor in Developer Workbench to perform these transformations. Assume the date is February 5, 1999 and the time is 02:05:25.123456789 a.m. The following defines use of the nanosecond component or a precision in display formats and function calls:

```
TRANSDT/HYYMDn      = DT(19990205 02:05:25.123456789);
 O_HSsmn/HSsmn       = TRANSDT;
 O_HHIS2/HHIS2       = TRANSDT;
 O_HYYMDn/HYYMDn     = TRANSDT;
 O_HYYMD1/HYYMD1     = TRANSDT;
 O_HADD/HYYMD9       = HADD(TRANSDT, 'NS', 2, 12, 'HYYMD9');
 O_HCNVRT/A26        = HCNVRT(TRANSDT, '(H23)', 23, 'A26');
 O_HDIFF/D12.2       = HDIFF(O_HADD, TRANSDT, 'NS', 'D12.2');
 TRANSDATE_DATE/YYMD = HDATE(TRANSDT, 'YYMD');
 O_HDTTM/HYYMDn      = HDTTM(TRANSDATE_DATE,12, 'HYYMDn');
 O_HEXTR/HHIS9       = HEXTR(TRANSDT, 'n',12, 'HHIS9');
 O_HGETC/HYYMDn      = HGETC(12, 'HYYMDn');
 O_HINPUT/HYYMDn     = HINPUT(14, O_HCNVRT,12, 'HYYMDn');
 O_HMASK/HYYMDn      = HMASK(O_HEXTR, 'HISsmn', TRANSDT, 12, 'HYYMDn');
 O_HMIDNT/HYYMDn     = HMIDNT(TRANSDT,12, 'HYYMDn');
 O_HNAME/A10         = HNAME(TRANSDT, 'NANOSECOND', 'A10');
 O_HPART/I10         = HPART(TRANSDT, 'NANOSECOND', 'I10');
 O_HSETPT/HYYMDn     = HSETPT(TRANSDT, 'NS', 28, 12,'HYYMDn');
 O_HTIME/P20.2C      = HTIME(12,TRANSDT, 'D16');
```

Output of these Define fields in a report:

```
TRANSDT   1999/02/05 02:05:25.123456789
O_HSsmn   25.123456789
O_HHIS2   02:05:25.12
O_HYYMDn  1999/02/05 02:05:25.123456789
O_HYYMD1  1999/02/05 02:05:25.1
O_HADD    1999/02/05 02:05:25.123456791
O_HCNVRT  19990205020525123456789
O_HDIFF                          2.00
O_HDTTM   1999/02/05 00:00:00.000000000
O_HEXTR   00:00:00.000000789
O_HGETC   2008/06/25 10:26:52.343644000
O_HINPUT  1999/02/05 02:05:25.000000000
O_HMASK   1999/02/05 00:00:00.000456789
O_HMIDNT  1999/02/05 00:00:00.000000000
O_HNAME   123456789
O_HPART                     123456789
O_HSETPT  1999/02/05 02:05:25.000000028
O_HTIME            7,525,123,456,789.00
```

Note:

- Field O_HSsmn displays the seconds (S), milliseconds (s), microseconds (m), and nanoseconds (n) from TRANSDT.
- Field O_HHIS2 displays the hours (H), minutes (I), and seconds (S) from TRANSDT. The seconds are displayed with two decimal digits (2).
- Field O_HYYMDn displays TRANSDT with the date in YYMD format and the time down to nanoseconds (n).
- Field O_HYYMD1 displays TRANSDT with the date in YYMD format and the time down to seconds with one decimal digit (1).
- Field O_HADD is created by calling the HADD function to add 2 nanoseconds to the date-time value in TRANSDT.
- Field O_HCNVRT is created by calling the HCNVRT function to convert the date-time value in TRANSDT to alphanumeric format.
- Field O_HDIFF is created by calling the HDIFF function to subtract the date-time value in TRANSDT from the date-time version in O_HADD.
- Field O_HDTTM is created by calling the HDTTM function to create a date-time field by taking the date from TRANSDATE_DATE and setting the time components to zero.
- Field O_HEXTR is created by calling the HEXTR function to extract the nanoseconds (low order three digits of nine) from TRANSDT and set the remaining components to zero.

- Field HGETC is created by calling the HGETC function to retrieve the current date and time and display them with 9 decimal digits for the seconds component. Note that not all operating systems return all 9 decimal digits, in which case the digits not returned display as zeros.
- Field O_HINPUT is created by calling the HINPUT function to convert the alphanumeric date-time string stored in field O_HCNVRT to a date-time value and display it down to the nanosecond.
- Field O_HMASK is created by calling the HMASK function to extract the hours, minutes, seconds, milliseconds, microseconds, and nanoseconds from O_HEXTR and take the remaining components from TRANSDT.
- Field O_HMIDNT is created by calling the HMIDNT function to retrieve the date from TRANSDT and set the time to midnight.
- Field O_HNAME is created by calling the HNAME function to retrieve the nanosecond component from TRANSDT in alphanumeric format.
- Field O_HPART is created by calling the HPART function to retrieve the nanosecond component from TRANSDT in numeric format.
- Field O_HSETPT is created by calling the HSETPT function to set the nanoseconds component to the value 28.
- Field O_HTIME is created by calling the HTIME function to convert the time portion of TRANSDT to the number of nanoseconds.

Reference: Usage Notes for Nanosecond Date-Time Format Component

- ACTUAL formats for date-time fields can be up to H12. USAGE formats can be up to H23.
- The following date-time functions take a component name as an argument: HADD, HDIFF, HNAME, HPART, and HSETPT. The component name for nanoseconds for use in these date-time functions is *nanosecond*, which can also be abbreviated as *ns*. In addition, the HEXTR and HMASK functions have a new component, *n*, that represents the low order three digits of nine decimal digits.
- The following functions have arguments whose length depends on the precision of the date-time format:
  - HADD, HDIFF, HINPUT, HMIDNT, HSETPT, and HTIME have a length argument. The length can be 8, 10, or 12, where 12 is needed if the seconds value has more than six decimal digits.
  - HDTTM and HGETC have an argument for the length of the date-time value, which can be 8, 10, or 12, where 12 is needed if the seconds value has more than six decimal digits.

- HCNVRT takes one argument that specifies the format of the date-time field to be converted to alphanumeric and one for the length of that field. The length of these arguments can be up to 23. The format for the output must be long enough to hold all of the characters returned.
- HTIME converts the time portion of a date-time value to nanoseconds if the first argument is 12.

DATEPATTERN in the Master File

In this section:

In some data sources, date values are stored in alphanumeric format without any particular standard, with any combination of components such as year, quarter, and month, and with any delimiter. In a sorted report, if such data is sorted alphabetically, the sequence does not make business sense. To ensure adequate sorting, aggregation, and reporting on date fields, DB2 Web Query can convert the alphanumeric dates into standard DB2 Web Query date format using a conversion pattern that you can specify in the Master File attribute called DATEPATTERN.

Each element in the pattern is either a constant character which must appear in the actual input or a variable that represents a date component. You must edit the USAGE attribute in the Master File so that it accounts for the date elements in the date pattern. The maximum length of the DATEPATTERN string is 64.

Specifying Variables in a Date Pattern

How to:

Specify Years in a Date Pattern

Specify Month Numbers in a Date Pattern

Specify Month Names in a Date Pattern

Specify Days of the Month in a Date Pattern

Specify Julian Days in a Date Pattern

Specify Day of the Week in a Date Pattern

Specify Quarters in a Date Pattern

The valid date components (variables) are year, quarter, month, day, and day of week. In the date pattern, variables are enclosed in square brackets (these brackets are not part of the input or output). Note that if the data contains brackets, you must use an escape character in the date pattern to distinguish the brackets in the data from the brackets used for enclosing variables.

Syntax: How to Specify Years in a Date Pattern

`[YYYY]`
: Specifies a 4-digit year.

`[YY]`
: Specifies a 2-digit year.

`[yy]`
: Specifies a zero-suppressed 2-digit year (for example, 8 for 2008).

`[by]`
: Specifies a blank-padded 2-digit year.

Syntax: How to Specify Month Numbers in a Date Pattern

`[MM]`
: Specifies a 2-digit month number.

`[mm]`
: Specifies a zero-suppressed month number.

`[bm]`
: Specifies a blank-padded month number.

Syntax: How to Specify Month Names in a Date Pattern

`[MON]`
: Specifies a 3-character month name in uppercase.

`[mon]`
: Specifies a 3-character month name in lowercase.

`[Mon]`
: Specifies a 3-character month name in mixed-case.

`[MONTH]`
: Specifies a full month name in uppercase.

`[month]`
: Specifies a full month name in lowercase.

`[Month]`
: Specifies a full month name in mixed-case.

Syntax: How to Specify Days of the Month in a Date Pattern

`[DD]`
: Specifies a 2-digit day of the month.

`[dd]`
: Specifies a zero-suppressed day of the month.

`[bd]`
: Specifies a blank-padded day of the month.

Syntax: How to Specify Julian Days in a Date Pattern

`[DDD]`
: Specifies a 3-digit day of the year.

`[ddd]`
: Specifies a zero-suppressed day of the year.

`[bdd]`
: Specifies a blank-padded day of the year.

Syntax: How to Specify Day of the Week in a Date Pattern

`[WD]`
: Specifies a 1-digit day of the week.

`[DAY]`
: Specifies a 3-character day name, uppercase.

`[day]`
Specifies a 3-character day name, lowercase.

`[Day]`
Specifies a 3-character day name, mixed-case.

`[WDAY]`
Specifies a full day name, uppercase.

`[wday]`
Specifies a full day name, lowercase.

`[Wday]`
Specifies a full day name, mixed-case.

For the day of the week, the WEEKFIRST setting defines which day is day 1.

Syntax: **How to Specify Quarters in a Date Pattern**

`[Q]`
Specifies a 1-digit quarter number (1, 2, 3, or 4).

For a string like Q2 or Q02, use constants before [Q], for example, Q0[Q].

Specifying Constants in a Date Pattern

Between the variables, you can insert any constant values.

If you want to insert a character that would normally be interpreted as part of a variable, use the backslash character as an escape character. For example:

- Use \[to specify a left square bracket constant character.
- Use \\ to specify a backslash constant character.

For a single quotation mark, use two consecutive single quotation marks ('').

Example: **Sample Date Patterns**

If the date in the data source is of the form CY 2001 Q1, the DATEPATTERN attribute is:

```
DATEPATTERN = 'CY [YYYY] Q[Q]'
```

If the date in the data source is of the form Jan 31, 01, the DATEPATTERN attribute is:

```
DATEPATTERN = '[Mon] [DD], [YY]'
```

If the date in the data source is of the form APR-06, the DATEPATTERN attribute is:

```
DATEPATTERN = '[MON]-[YY]'
```

If the date in the data source is of the form APR - 06, the DATEPATTERN attribute is:

```
DATEPATTERN = '[MON] - [YY]'
```

If the date in the data source is of the form APR '06, the DATEPATTERN attribute is:

```
DATEPATTERN = '[MON] ''[YY]'
```

If the date in the data source is of the form APR [06], the DATEPATTERN attribute is:

```
DATEPATTERN = '[MON] \[[YY]\]' (or '[MON] \[[YY]]'
```

Note the right square bracket does not require an escape character.

Example: Sorting By an Alphanumeric Date

In the following example, date1.ftm is a sequential file containing the following data:

```
June 1, '02
June 2, '02
June 3, '02
June 10, '02
June 11, '02
June 12, '02
June 20, '02
June 21, '02
June 22, '02
June 1, '03
June 2, '03
June 3, '03
June 10, '03
June 11, '03
June 12, '03
June 20, '03
June 21, '03
June 22, '03
June 1, '04
June 2, '04
June 3, '04
June 4, '04
June 10, '04
June 11, '04
June 12, '04
June 20, '04
June 21, '04
June 22, '04
```

In the DATE1 Master File, the DATE1 field has alphanumeric USAGE and ACTUAL formats, each A18:

```
FILENAME=DATE1, SUFFIX=FIX,
  DATASET = c:\tst\date1.ftm, $
  SEGMENT=FILE1, SEGTYPE=S0, $
    FIELDNAME=DATE1, ALIAS=E01, USAGE=A18, ACTUAL=A18, $
```

The following request sorts by the DATE1 FIELD:

```
TABLE FILE DATE1
PRINT DATE1 NOPRINT
BY DATE1
ON TABLE SET PAGE NOPAGE
END
```

The output shows that the alphanumeric dates are sorted alphabetically, not chronologically:

```
DATE1
-----
June 1, '02
June 1, '03
June 1, '04
June 10, '02
June 10, '03
June 10, '04
June 11, '02
June 11, '03
June 11, '04
June 12, '02
June 12, '03
June 12, '04
June 2, '02
June 2, '03
June 2, '04
June 20, '02
June 20, '03
June 20, '04
June 21, '02
June 21, '03
June 21, '04
June 22, '02
June 22, '03
June 22, '04
June 3, '02
June 3, '03
June 3, '04
June 4, '04
```

In order to sort the data correctly, you can add a DATEPATTERN attribute to the Master File that enables DB2 Web Query to convert the date to a DB2 Web Query date field. You must also edit the USAGE format to make it a DB2 Web Query date format. To construct the appropriate pattern, you must account for all of the components in the stored date. The alphanumeric date has the following variables and constants:

- Variable: full month name in mixed-case, [Month].
- Constant: blank space.
- Variable: zero-suppressed day of the month number, [dd].
- Constant: comma followed by a blank space followed by an apostrophe (coded as two apostrophes in the pattern).
- Variable: two-digit year, [YY].

The edited Master File follows. Note the addition of the DEFCENT attribute to convert the two-digit year to a four-digit year:

```
FILENAME=DATE1, SUFFIX=FIX,
  DATASET = c:\tst\date1.ftm, $
  SEGMENT=FILE1, SEGTYPE=S0, $
    FIELDNAME=DATE1, ALIAS=E01, USAGE=A18, ACTUAL=A18, DEFCENT=20,
    DATEPATTERN = '[Month] [dd], ''[YY]', $
```

Now, issuing the same request produces the following output. Note that DATE1 has been converted to a DB2 Web Query date in MtrDYY format (as specified in the USAGE format):

```
DATE1
-----
June  1, 2002
June  2, 2002
June  3, 2002
June 10, 2002
June 11, 2002
June 12, 2002
June 20, 2002
June 21, 2002
June 22, 2002
June  1, 2003
June  2, 2003
June  3, 2003
June 10, 2003
June 11, 2003
June 12, 2003
June 20, 2003
June 21, 2003
June 22, 2003
June  1, 2004
June  2, 2004
June  3, 2004
June  4, 2004
June 10, 2004
June 11, 2004
June 12, 2004
June 20, 2004
June 21, 2004
June 22, 2004
```

DMY, MDY, YMD: Calculating the Difference Between Two Dates

How to:

Calculate the Difference Between Two Dates

The DMY, MDY, and YMD functions calculate the difference between two dates in integer, alphanumeric, or packed format.

Syntax: How to Calculate the Difference Between Two Dates

```
function(begin, end)
```

where:

function

Is one of the following:

DMY calculates the difference between two dates in day-month-year format.

MDY calculates the difference between two dates in month-day-year format.

YMD calculates the difference between two dates in year-month-day format.

begin

Integer, Packed, or Alphanumeric

I, P, or A format with date display options.

Is the beginning date, or the name of a field that contains the date.

end

Integer, Packed, or Alphanumeric

I, P, or A format with date display options.

Is the end date, or the name of a field that contains the date.

Example: Calculating the Number of Days Between Two Dates

YMD calculates the number of days between the dates in HIRE_DATE and DAT_INC:

```
COMPUTE DIFF/I4 = YMD(HIRE_DATE, FST.DAT_INC);
```

DOWK and DOWKL: Finding the Day of the Week

How to:

Find the Day of the Week

The DOWK and DOWKL functions find the day of the week that corresponds to a date. DOWK returns the day as a three letter abbreviation; DOWKL displays the full name of the day.

Syntax: How to Find the Day of the Week

```
{DOWK|DOWKL}(indate, 'outfield')
```

where:

indate
: Integer (I6YMD or I8 YMD)

  Is the input date in year-month-day format. If the date is not valid, the function returns spaces. If the date specifies a two-digit year and DEFCENT and YRTHRESH values have not been set, the function assumes the 20th century.

outfield
: DOWK: Alphanumeric

  DOWKL: Alphanumeric

  Is the format of the output value enclosed in single quotation marks.

Example: Finding the Day of the Week

DOWK determines the day of the week that corresponds to the value in the HIRE_DATE field and stores the result in DATED:

```
COMPUTE DATED/A3 = DOWK(HIRE_DATE, 'A3');
```

DT Functions: Converting an Integer to a Date

How to:

Convert an Integer to a Date

The DT functions convert an integer representing the number of days elapsed since December 31, 1899 to the corresponding date. They are useful when you are performing arithmetic on a date converted to the number of days. The DT functions convert the result back to a date.

There are six DT functions; each one converts a number into a date of a different format.

Note: When USERFNS is set to LOCAL, DT functions only display a six-digit date.

Syntax: How to Convert an Integer to a Date

```
function(number, 'outfield')
```

where:

function
: Is one of the following:

`DTDMY` converts a number to a day-month-year date.

`DTDYM` converts a number to a day-year-month date.

`DTMDY` converts a number to a month-day-year date.

`DTMYD` converts a number to a month-year-day date.

`DTYDM` converts a number to a year-day-month date.

`DTYMD` converts a number to a year-month-day date.

`number`

Integer

Is the number of days since December 31, 1899. The number is truncated to an integer.

`outfield`

Integer

I6*xxx*, where *xxx* corresponds to the function DT*xxx* in the above list.

Is the format of the output value enclosed in single quotation marks. The output format depends on the function being used.

Example: Converting an Integer to a Date

DTMDY converts the NEWF field (which was converted to the number of days by DAYMD) to the corresponding date and stores the result in NEW_HIRE_DATE:

```
COMPUTE NEWF/I8 WITH EMP_ID = DAYMD(HIRE_DATE, NEWF); AND
COMPUTE NEW_HIRE_DATE/I8MDYY WITH EMP_ID = DTMDY(NEWF, NEW_HIRE_DATE);
```

FIYR: Obtaining the Financial Year

How to:

Obtain the Financial Year

The FIYR function returns the financial year, also known as the fiscal year, corresponding to a given calendar date based on the financial year starting date and the financial year numbering convention.

Syntax: How to Obtain the Financial Year

```
FIYR(inputdate, lowcomponent, startmonth, startday, yrnumbering, output)
```

where:

`inputdate`

Date

Is the date for which the financial year is returned. The date must be a standard date stored as an offset from the base date.

If the financial year does not begin on the first day of a month, the date must have Y(Y), M, and D components, or Y(Y) and JUL components (note that JUL is equivalent to YJUL). Otherwise, the date only needs Y(Y) and M components or Y(Y) and Q components.

`lowcomponent`

Alphanumeric

Is one of the following:

- `D` if the date contains a D or JUL component.
- `M` if the date contains an M component, but no D component.
- `Q` if the date contains a Q component.

`startmonth`

Numeric

1 through 12 are used to represent the starting month of the financial year, where 1 represents January and 12 represents December. If the low component is Q, the start month must be 1, 4, 7, or 10.

`startday`

Numeric

Is the starting day of the starting month, usually 1. If the low component is M or Q, 1 is required.

`yrnumbering`

Alphanumeric

Valid values are:

FYE to specify the *Financial Year Ending* convention. The financial year number is the calendar year of the ending date of the financial year. For example, when the financial year starts on October 1, 2008, the date, 2008 November 1 is in FY 2009 Q1 because that date is in the financial year that ends on 2009 September 30.

FYS to specify the *Financial Year Starting* convention. The financial year number is the calendar year of the starting date of the financial year. For example, when the financial year starts on April 6, 2008, the date, 2008 July 6 is in FY 2008 Q2 because that date is in the financial year that starts on 2008 April 6.

`output`

I, Y, or YY

The result will be in integer format, or Y or YY. This function returns a year value. In case of an error, zero is returned.

Note: February 29 cannot be used as a start day for a financial year.

Example: Obtaining the Financial Year

The following obtains the financial year corresponding to an account period (field PERIOD, format YYM) and returns the values in each of the supported formats: Y, YY, and I4.

```
FISCALYY/YY=FIYR(PERIOD,'M', 4,1,'FYE',FISCALYY);
FISCALY/Y=FIYR(PERIOD,'M', 4,1,'FYE',FISCALY);
FISCALI/I4=FIYR(PERIOD,'M', 4,1,'FYE',FISCALI);
END
```

On the output, note that the period April 2002 (2002/04) is in fiscal year 2003 because the starting month is April (4), and the FYE numbering convention is used:

```
Ledger
Account  PERIOD   FISCALYY  FISCALY  FISCALI
-------  ------   --------  -------  -------
1000     2002/01  2002      02          2002
         2002/02  2002      02          2002
         2002/03  2002      02          2002
         2002/04  2003      03          2003
         2002/05  2003      03          2003
         2002/06  2003      03          2003
2000     2002/01  2002      02          2002
         2002/02  2002      02          2002
         2002/03  2002      02          2002
         2002/04  2003      03          2003
         2002/05  2003      03          2003
         2002/06  2003      03          2003
```

FIQTR: Obtaining the Financial Quarter

How to:

Obtain the Financial Quarter

The FIQTR function returns the financial quarter corresponding to a given calendar date based on the financial year starting date and the financial year numbering convention.

Syntax: How to Obtain the Financial Quarter

`FIQTR(`*`inputdate, lowcomponent, startmonth, startday, yrnumbering, output`*`)`

where:

`inputdate`

Date

Is the date for which the financial year is returned. The date must be a standard date stored as an offset from the base date.

If the financial year does not begin on the first day of a month, the date must have Y(Y), M, and D components, or Y(Y) and JUL components (note that JUL is equivalent to YJUL). Otherwise, the date only needs Y(Y) and M components or Y(Y) and Q components.

`lowcomponent`

Alphanumeric

Is one of the following:

- `D` if the date contains a D or JUL component.
- `M` if the date contains an M component, but no D component.
- `Q` if the date contains a Q component.

`startmonth`

Numeric

1 through 12 are used to represent the starting month of the financial year, where 1 represents January and 12 represents December. If the low component is Q, the start month must be 1, 4, 7, or 10.

`startday`

Numeric

Is the starting day of the starting month, usually 1. If the low component is M or Q, 1 is required.

`yrnumbering`

Alphanumeric

Valid values are:

FYE to specify the *Financial Year Ending* convention. The financial year number is the calendar year of the ending date of the financial year. For example, when the financial year starts on October 1, 2008, the date, 2008 November 1 is in FY 2009 Q1 because that date is in the financial year that ends on 2009 September 30.

FYS to specify the *Financial Year Starting* convention. The financial year number is the calendar year of the starting date of the financial year. For example, when the financial year starts on April 6, 2008, the date, 2008 July 6 is in FY 2008 Q2 because that date is in the financial year that starts on 2008 April 6.

output

I or Q

The result will be in integer format, or Q. This function will return a value of 1 through 4. In case of an error, zero is returned.

Note: February 29 cannot be used as a start day for a financial year.

Example: Obtaining the Financial Quarter

The following obtains the financial quarter corresponding to an employee starting date (field START_DATE, format YYMD) and returns the values in each of the supported formats: Q and I1.

```
FISCALQ/Q=FIQTR(START_DATE,'D',10,1,'FYE',FISCALQ);
FISCALI/I1=FIQTR(START_DATE,'D',10,1,'FYE',FISCALI);
```

On the output, note that the date, November 12, 1998 (1998/11/12) is in fiscal quarter Q1 because the starting month is October (10):

```
Last            First          Starting
Name            Name           Date        FISCALQ  FISCALI
----            -----          --------    -------  -------
CHARNEY         ROSS           1998/09/12  Q4             4
CHIEN           CHRISTINE      1997/10/01  Q1             1
CLEVELAND       PHILIP         1996/07/30  Q4             4
CLINE           STEPHEN        1998/11/12  Q1             1
COHEN           DANIEL         1997/10/05  Q1             1
CORRIVEAU       RAYMOND        1997/12/05  Q1             1
COSSMAN         MARK           1996/12/19  Q1             1
CRONIN          CHRIS          1996/12/03  Q1             1
CROWDER         WESLEY         1996/09/17  Q4             4
CULLEN          DENNIS         1995/09/05  Q4             4
CUMMINGS        JAMES          1993/07/11  Q4             4
CUTLIP          GREGG          1997/03/26  Q2             2
```

FIYYQ: Converting a Calendar Date to a Financial Date

How to:

Convert a Calendar Date to a Financial Date

The FIYYQ function returns a financial date containing both the financial year and quarter that corresponds to a given calendar date. The returned financial date is based on the financial year starting date and the financial year numbering convention.

Syntax: How to Convert a Calendar Date to a Financial Date

```
FIYYQ(inputdate, lowcomponent, startmonth, startday, yrnumbering, output)
```

where:

inputdate

Date

Is the date for which the financial year is returned. The date must be a standard date stored as an offset from the base date.

If the financial year does not begin on the first day of a month, the date must have Y(Y), M, and D components, or Y(Y) and JUL components (note that JUL is equivalent to YJUL). Otherwise, the date only needs Y(Y) and M components or Y(Y) and Q components.

lowcomponent

Alphanumeric

Is one of the following:

- `D` if the date contains a D or JUL component.
- `M` if the date contains an M component, but no D component.
- `Q` if the date contains a Q component.

startmonth

Numeric

1 through 12 are used to represent the starting month of the financial year, where 1 represents January and 12 represents December. If the low component is Q, the start month must be 1, 4, 7, or 10.

startday

Numeric

Is the starting day of the starting month, usually 1. If the low component is M or Q, 1 is required.

`yrnumbering`

Alphanumeric

Valid values are:

FYE to specify the *Financial Year Ending* convention. The financial year number is the calendar year of the ending date of the financial year. For example, when the financial year starts on October 1, 2008, the date, 2008 November 1 is in FY 2009 Q1 because that date is in the financial year that ends on 2009 September 30.

FYS to specify the *Financial Year Starting* convention. The financial year number is the calendar year of the starting date of the financial year. For example, when the financial year starts on April 6, 2008, the date, 2008 July 6 is in FY 2008 Q2 because that date is in the financial year that starts on 2008 April 6.

`output`

Y[Y]Q or QY[Y]

In case of an error, zero is returned.

Note: February 29 cannot be used as a start day for a financial year.

Example: Converting a Calendar Date to a Financial Date

The following converts each employee starting date (field START_DATE, format YYMD) to a financial date containing year and quarter components in all the supported formats: YQ, YYQ, QY, and QYY.

```
FISYQ/YQ=FIYYQ(START_DATE,'D',10,1,'FYE',FISYQ);
FISYYQ/YYQ=FIYYQ(START_DATE,'D',10,1,'FYE',FISYYQ);
FISQY/QY=FIYYQ(START_DATE,'D',10,1,'FYE',FISQY);
FISQYY/QYY=FIYYQ(START_DATE,'D',10,1,'FYE',FISQYY);
```

On the output, note that the date, November 12, 1998 (1998/11/12) is converted to Q1 1999 because the starting month is October (10), and the FYE numbering convention is used:

```
Last            First          Starting
Name            Name           Date        FISYQ  FISYYQ   FISQY  FISQYY
----            -----          --------    -----  ------   -----  ------
CHARNEY         ROSS           1998/09/12  98 Q4  1998 Q4  Q4 98  Q4 1998
CHIEN           CHRISTINE      1997/10/01  98 Q1  1998 Q1  Q1 98  Q1 1998
CLEVELAND       PHILIP         1996/07/30  96 Q4  1996 Q4  Q4 96  Q4 1996
CLINE           STEPHEN        1998/11/12  99 Q1  1999 Q1  Q1 99  Q1 1999
COHEN           DANIEL         1997/10/05  98 Q1  1998 Q1  Q1 98  Q1 1998
CORRIVEAU       RAYMOND        1997/12/05  98 Q1  1998 Q1  Q1 98  Q1 1998
COSSMAN         MARK           1996/12/19  97 Q1  1997 Q1  Q1 97  Q1 1997
CRONIN          CHRIS          1996/12/03  97 Q1  1997 Q1  Q1 97  Q1 1997
CROWDER         WESLEY         1996/09/17  96 Q4  1996 Q4  Q4 96  Q4 1996
CULLEN          DENNIS         1995/09/05  95 Q4  1995 Q4  Q4 95  Q4 1995
CUMMINGS        JAMES          1993/07/11  93 Q4  1993 Q4  Q4 93  Q4 1993
CUTLIP          GREGG          1997/03/26  97 Q2  1997 Q2  Q2 97  Q2 1997
```

GREGDT: Converting From Julian to Gregorian Format

How to:

Convert From Julian to Gregorian Format

Reference:

DATEFNS Settings for GREGDT

The GREGDT function converts a date in Julian format to Gregorian format (year-month-day).

A date in Julian format is a five- or seven-digit number. The first two or four digits are the year; the last three digits are the number of the day, counting from January 1. For example, January 1, 1999 in Julian format is either 99001 or 1999001.

Reference: DATEFNS Settings for GREGDT

GREGDT converts a Julian date to either YMD or YYMD format using the DEFCENT and YRTHRESH parameter settings to determine the century, if required. GREGDT returns a date as follows:

DATEFNS Setting	I6 or I7 Format	I8 Format or Greater
ON	YMD	YYMD

DATEFNS Setting	I6 or I7 Format	I8 Format or Greater
OFF	YMD	YMD

Syntax: How to Convert From Julian to Gregorian Format

```
GREGDT(indate, 'outfield')
```

where:

indate

Integer (I5 or I7)

Is the Julian date, which is truncated to an integer before conversion. Each value must be a five- or seven-digit number after truncation. If the date is invalid, the function returns a 0.

outfield

Integer (I6, I8, I6YMD, or I8YYMD)

Is the format of the output value enclosed in single quotation marks.

Example: Converting From Julian to Gregorian Format

GREGDT converts the JULIAN field to YYMD (Gregorian) format.

```
COMPUTE GREG_DATE/I8 = GREGDT(JULIAN, 'I8');
```

HADD: Incrementing a Date-Time Value

How to:

Increment a Date-Time Value

The HADD function increments a date-time value by a given number of units.

Syntax: How to Increment a Date-Time Value

```
HADD(value, 'component', increment, length, 'outfield')
```

where:

value

Date-time

Is the date-time value to be incremented, the name of a date-time field that contains the value, or an expression that returns the value.

component

Alphanumeric

Is the name of the component to be incremented enclosed in single quotation marks.

Note: WEEKDAY is not a valid component for HADD.

increment

Integer

Is the number of units by which to increment the component, the name of a numeric field that contains the value, or an expression that returns the value.

length

Integer

Is the length of the returned date-time value. Valid values are:

`8` indicates a time value that includes milliseconds.

`10` indicates a time value that includes microseconds.

outfield

Date-time

Is the format of the output value enclosed in single quotation marks.

Example: Incrementing the Month Component of a Date-Time Field

HADD adds two months to each value in TRANSDATE and stores the result in ADD_MONTH. If necessary, the day is adjusted so that it is valid for the resulting month.

```
COMPUTE ADD_MONTH/HYYMDS = HADD(TRANSDATE, 'MONTH', 2, 8, 'HYYMDS');
```

HCNVRT: Converting a Date-Time Value to Alphanumeric Format

How to:

Convert a Date-Time Value to Alphanumeric Format

The HCNVRT function converts a date-time value to alphanumeric format for use with operators such as EDIT, CONTAINS, and LIKE.

Syntax: How to Convert a Date-Time Value to Alphanumeric Format

```
HCNVRT(value, '(fmt)', length, 'outfield')
```

where:

value

Date-time

Is the date-time value to be converted, the name of a date-time field that contains the value, or an expression that returns the value.

fmt

Alphanumeric

Is the format of the date-time field enclosed in single quotation marks and parentheses.

length

Integer

Is the length of the alphanumeric field that is returned. You can supply the actual value, the name of a numeric field that contains the value, or an expression that returns the value. If *length* is smaller than the number of characters needed to display the alphanumeric field, the function returns a blank.

outfield

Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Example: Converting a Date-Time Field to Alphanumeric Format

HCNVRT converts the TRANSDATE field to alphanumeric format. The first function does not include date-time display options for the field; the second function does for readability. It also specifies the display of seconds in the input field.

```
COMPUTE ALPHA_DATE_TIME1/A20 = HCNVRT(TRANSDATE, '(H17)', 17, 'A20'); AND
COMPUTE ALPHA_DATE_TIME2/A20 = HCNVRT(TRANSDATE, '(HYYMDS)', 20, 'A20');
```

HDATE: Converting the Date Portion of a Date-Time Value to a Date Format

How to:

Convert the Date Portion of a Date-Time Value to a Date Format

The HDATE function converts the date portion of a date-time value to the date format YYMD. You can then convert the result to other date formats.

Syntax: How to Convert the Date Portion of a Date-Time Value to a Date Format

```
HDATE(value, 'YYMD')
```

where:

value

Date-time

Is the date-time value to be converted, the name of a date-time field that contains the value, or an expression that returns the value.

`YYMD`

Date

Is the output format. The value must be YYMD. YYMD is a constant value and cannot be changed in this syntax, although you can change the format in subsequent DEFINEs or COMPUTEs.

Example: Converting the Date Portion of a Date-Time Field to a Date Format

HDATE converts the date portion of the TRANSDATE field to the date format YYMD:

```
COMPUTE TRANSDATE_DATE/YYMD = HDATE(TRANSDATE, 'YYMD');
```

HDIFF: Finding the Number of Units Between Two Date-Time Values

How to:

Find the Number of Units Between Two Date-Time Values

The HDIFF function calculates the number of units between two date-time values.

Syntax: How to Find the Number of Units Between Two Date-Time Values

```
HDIFF(value1, value2, 'component', 'outfield')
```

where:

value1

Date-time

Is the end date-time value, the name of a date-time field that contains the value, or an expression that returns the value.

value2

Date-time

Is the start date-time value, the name of a date-time field that contains the value, or an expression that returns the value.

component

Alphanumeric

Is the name of the component to be used in the calculation enclosed in single quotation marks. If the component is a week, the WEEKFIRST parameter setting is used in the calculation.

outfield

Floating point or Decimal

Is the format of the output value enclosed in single quotation marks.

Example: Finding the Number of Days Between Two Date-Time Fields

HDIFF calculates the number of days between the TRANSDATE and ADD_MONTH fields and stores the result in DIFF_PAYS, which has the format D12.2:

```
COMPUTE ADD_MONTH/HYYMDS = HADD(TRANSDATE, 'MONTH', 2, 8, 'HYYMDS'); AND
COMPUTE DIFF_DAYS/D12.2 = HDIFF(ADD_MONTH, TRANSDATE, 'DAY', 'D12.2');
```

HDTTM: Converting a Date Value to a Date-Time Value

How to:

Convert a Date Value to a Date-Time Value

The HDTTM function converts a date value to a date-time field. The time portion is set to midnight.

Syntax: How to Convert a Date Value to a Date-Time Value

```
HDTTM(date, length, 'outfield')
```

where:

date

Date

Is the date value to be converted, the name of a date field that contains the value, or an expression that returns the value.

length

Integer

Is the length of the returned date-time value. Valid values are:

`8` indicates a time value that includes milliseconds.

`10` indicates a time value that includes microseconds.

`outfield`

Date-time

Is the format of the output value enclosed in single quotation marks.

Example: Converting a Date Field to a Date-Time Field

HDTTM converts the date field TRANSDATE_DATE to a date-time field:

```
COMPUTE TRANSDATE_DATE/YYMD = HDATE(TRANSDATE, 'YYMD'); AND
COMPUTE DT2/HYYMDIA = HDTTM(TRANSDATE_DATE, 8, 'HYYMDIA');
```

HGETC: Storing the Current Date and Time in a Date-Time Field

How to:

Store the Current Date and Time in a Date-Time Field

The HGETC function stores the current date and time in a date-time field. If millisecond or microsecond values are not available in your operating environment, the function retrieves the value zero for these components.

Syntax: How to Store the Current Date and Time in a Date-Time Field

```
HGETC(length, 'outfield')
```

where:

`length`

Integer

Is the length of the returned date-time value. Valid values are:

`8` indicates a time value that includes milliseconds.

`10` indicates a time value that includes microseconds.

`outfield`

Date-time

Is the format of the output value enclosed in single quotation marks.

Example: Storing the Current Date and Time in a Date-Time Field

HGETC stores the current date and time in DT2:

```
COMPUTE DT2/HYYMDm = HGETC(10, 'HYYMDm');
```

HHMMSS: Retrieving the Current Time

How to:

Retrieve the Current Time

The HHMMSS function retrieves the current time from the operating system as an eight character string, separating the hours, minutes, and seconds with periods.

Syntax: How to Retrieve the Current Time

```
HHMMSS('outfield')
```

where:

outfield

Alphanumeric

Is the format of the output value enclosed in single quotation marks. The field format must be at least A8.

Example: Retrieving the Current Time

HHMMSS retrieves the current time:

```
COMPUTE NOWTIME/A8 = HHMMSS('A8');
```

HINPUT: Converting an Alphanumeric String to a Date-Time Value

How to:

Convert an Alphanumeric String to a Date-Time Value

The HINPUT function converts an alphanumeric string to a date-time value.

Syntax: How to Convert an Alphanumeric String to a Date-Time Value

```
HINPUT(inputlength, 'inputstring', length, 'outfield')
```

where:

inputlength

Integer

Is the length of the alphanumeric string to be converted. You can supply the actual value, the name of a numeric field that contains the value, or an expression that returns the value.

inputstring

Alphanumeric

Is the alphanumeric string to be converted enclosed in single quotation marks, the name of an alphanumeric field that contains the string, or an expression that returns the string. The string can consist of any valid date-time input value.

length

Integer

Is the length of the returned date-time value. Valid values are:

`8` indicates a time value that includes milliseconds.

`10` indicates a time value that includes microseconds.

outfield

Date-time

Is the format of the output value enclosed in single quotation marks.

Example: Converting an Alphanumeric String to a Date-Time Value

HCNVRT converts the TRANSDATE field to alphanumeric format, then HINPUT converts the alphanumeric string to a date-time value:

```
COMPUTE ALPHA_DATE_TIME/A20 = HCNVRT(TRANSDATE, '(H17)', 17, 'A20'); AND
COMPUTE DT_FROM_ALPHA/HYYMDS = HINPUT(14, ALPHA_DATE_TIME, 8, 'HYYMDS');
```

HMIDNT: Setting the Time Portion of a Date-Time Value to Midnight

How to:

Set the Time Portion of a Date-Time Value to Midnight

The HMIDNT function changes the time portion of a date-time value to midnight (all zeros by default). This allows you to compare a date field with a date-time field.

Syntax: How to Set the Time Portion of a Date-Time Value to Midnight

```
HMIDNT(value, length, 'outfield')
```

where:

value

Date-time

Is the date-time value whose time is to be set to midnight, the name of a date-time field that contains the value, or an expression that returns the value.

length
: Integer

  Is the length of the returned date-time value. Valid values are:

  `8` indicates a time value that includes milliseconds.

  `10` indicates a time value that includes microseconds.

outfield
: Date-time

  Is the format of the output value enclosed in single quotation marks.

Example: Setting the Time to Midnight

HMIDNT sets the time portion of the TRANSDATE field to midnight first in the 24-hour system and then in the 12-hour system:

```
COMPUTE TRANSDATE_MID_24/HYYMDS  = HMIDNT(TRANSDATE, 8, 'HYYMDS'); AND
COMPUTE TRANSDATE_MID_12/HYYMDSA = HMIDNT(TRANSDATE, 8, 'HYYMDSA');
```

HNAME: Retrieving a Date-Time Component in Alphanumeric Format

How to:

Retrieve a Date-Time Component in Alphanumeric Format

The HNAME function extracts a specified component from a date-time value in alphanumeric format.

Syntax: How to Retrieve a Date-Time Component in Alphanumeric Format

```
HNAME(value, 'component', 'outfield')
```

where:

value
: Date-time

  Is the date-time value from which a component is to be extracted, the name of a date-time field that contains the value, or an expression that returns the value.

component
: Alphanumeric

  Is the name of the component to be retrieved enclosed in single quotation marks.

outfield
: Alphanumeric

Is the format of the output value enclosed in single quotation marks. The field format must be at lease A2.

The function converts all other components to strings of digits only. The year is always four digits, and the hour assumes the 24-hour system.

Example: Retrieving the Week Component in Alphanumeric Format

HNAME returns the week in alphanumeric format from the TRANSDATE field. Changing the WEEKFIRST parameter setting changes the value of the component.

```
COMPUTE WEEK_COMPONENT/A10 = HNAME(TRANSDATE, 'WEEK', 'A10');
```

Example: Retrieving the Day Component in Alphanumeric Format

HNAME retrieves the day in alphanumeric format from the TRANSDATE field:

```
COMPUTE DAY_COMPONENT/A2 = HNAME(TRANSDATE, 'DAY', 'A2');
```

HPART: Retrieving a Date-Time Component in Numeric Format

How to:

Retrieve a Date-Time Component in Numeric Format

The HPART function extracts a specified component from a date-time value and returns it in numeric format.

Syntax: How to Retrieve a Date-Time Component in Numeric Format

```
HPART(value, 'component', 'outfield')
```

where:

value

Date-time

Is a date-time value, the name of a date-time field that contains the value, or an expression that returns the value.

component

Alphanumeric

Is the name of the component to be retrieved enclosed in single quotation marks.

outfield

Integer

Is the format of the output value enclosed in single quotation marks.

Example: Retrieving the Day Component in Numeric Format

HPART retrieves the day in integer format from the TRANSDATE field:

```
COMPUTE DAY_COMPONENT/I2 = HPART(TRANSDATE, 'DAY', 'I2');
```

HSETPT: Inserting a Component Into a Date-Time Value

How to:

Insert a Component Into a Date-Time Value

The HSETPT function inserts the numeric value of a specified component into a date-time value.

Syntax: How to Insert a Component Into a Date-Time Value

```
HSETPT(dtfield, 'component', value, length, 'outfield')
```

where:

dtfield

Date-time

Is a date-time value, the name of a date-time field that contains the value, or an expression that returns the value.

component

Alphanumeric

Is the name of the component to be inserted enclosed in single quotation marks.

value

Integer

Is the numeric value to be inserted for the requested component, the name of a numeric field that contains the value, or an expression that returns the value.

length

Integer

Is the length of the returned date-time value. Valid values are:

`8` indicates a time value that includes milliseconds.

`10` indicates a time value that includes microseconds.

outfield

Date-time

Is the format of the output value enclosed in single quotation marks.

Example: Inserting the Day Component Into a Date-Time Field

HSETPT inserts the day as 28 into the ADD_MONTH field and stores the result in INSERT_DAY:

```
COMPUTE ADD_MONTH/HYYMDS = HADD(TRANSDATE, 'MONTH', 2, 8, 'HYYMDS'); AND
COMPUTE INSERT_DAY/HYYMDS = HSETPT(ADD_MONTH, 'DAY', 28, 8, 'HYYMDS');
```

HTIME: Converting the Time Portion of a Date-Time Value to a Number

How to:

Convert the Time Portion of a Date-Time Field to a Number

The HTIME function converts the time portion of a date-time value to the number of milliseconds if the first argument is eight, or microseconds if the first argument is ten. To include microseconds, the input date-time value must be 10-bytes.

Syntax: How to Convert the Time Portion of a Date-Time Field to a Number

```
HTIME(length, value, 'outfield')
```

where:

length

Integer

Is the length of the input date-time value. Valid values are:

8 indicates a time value that includes milliseconds.

10 indicates a time value that includes microseconds.

value

Date-time

Is the date-time value from which to convert the time, the name of a date-time field that contains the value, or an expression that returns the value.

outfield

Floating point or Decimal

Is the format of the output value enclosed in single quotation marks.

Example: Converting the Time Portion of a Date-Time Field to a Number

HTIME converts the time portion of the TRANSDATE field to the number of milliseconds:

```
COMPUTE MILLISEC/D12.2 = HTIME(8, TRANSDATE, 'D12.2');
```

JULDAT: Converting From Gregorian to Julian Format

How to:

Convert From Gregorian to Julian Format

Reference:

DATEFNS Settings for JULDAT

The JULDAT function converts a date from Gregorian format (year-month-day) to Julian format (year-day). A date in Julian format is a five- or seven-digit number. The first two or four digits are the year; the last three digits are the number of the day, counting from January 1. For example, January 1, 1999 in Julian format is either 99001 or 1999001.

Reference: DATEFNS Settings for JULDAT

JULDAT converts a Gregorian date to either YYNNN or YYYYNNN format, using the DEFCENT and YRTHRESH parameter settings to determine if the century is required.

JULDAT returns dates as follows:

DATEFNS Setting	I6 or I7 Format	I8 Format or Greater
ON	YYNNN	YYYYNNN
OFF	YYNNN	YYNNN

Syntax: How to Convert From Gregorian to Julian Format

```
JULDAT(indate, 'outfield')
```

where:

indate

Integer (I6, I8, I6YMD, I8YYMD)

Is the date or the name of the field that contains the date in year-month-day format (YMD or YYMD).

outfield

Integer (I5 or I7)

Is the format of the output value enclosed in single quotation marks.

Example: Converting From Gregorian to Julian Format

JULDAT converts the HIRE_DATE field to Julian format.

```
COMPUTE JULIAN/I7 = JULDAT(HIRE_DATE, 'I7');
```

TIMETOTS: Converting a Time to a Timestamp

How to:

Convert a Time to a Timestamp

The TIMETOTS function converts a time to a timestamp, using the current date to supply the date component of its value. The first argument must be in H (date-time) format. The DATE component will be set to the current date.

Syntax: How to Convert a Time to a Timestamp

```
TIMETOTS (time, length, 'outfield')
```

where:

time

Date-time

Is the time in a date-time format.

length

Integer

Is the length of the result. This can be one of the following:

8 for time values including milliseconds.

10 for input time values including microseconds.

outfield

Date-time

Is the format of the output value enclosed in single quotation marks.

Example: Converting a Time to a Timestamp

TIMETOTS converts a time argument to a timestamp:

```
COMPUTE TSTMPSEC/HYYMDS = TIMETOTS(TMSEC, 8, 'HYYMDS'); AND
COMPUTE TSTMPMILLI/HYYMDm = TIMETOTS(TMMILLI, 10, 'HYYMDm');
```

TODAY: Returning the Current Date

How to:

Retrieve the Current Date

The TODAY function retrieves the current date from the operating system in the format MM/DD/YY or MM/DD/YYYY. It always returns a date that is current. Therefore, if you are running an application late at night, use TODAY. You can remove the default embedded slashes with the EDIT function.

Syntax: How to Retrieve the Current Date

```
TODAY('outfield')
```

where:

outfield

Alphanumeric

Is the format of the output value enclosed in single quotation marks. The field format must be at least A8. The following apply:

- If DATEFNS=ON and the format is A8 or A9, TODAY returns the 2-digit year.
- If DATEFNS=ON and the format is A10 or greater, TODAY returns the 4-digit year.
- If DATEFNS=OFF, TODAY returns the 2-digit year, regardless of the format of *outfield*.

Example: Retrieving the Current Date

TODAY retrieves the current date and stores it in the DATE field.

```
COMPUTE DATE/A10 = TODAY('A10');
```

YM: Calculating Elapsed Months

How to:

Calculate Elapsed Months

The YM function calculates the number of months that elapse between two dates. The dates must be in year-month format. You can convert a date to this format by using the CHGDAT or EDIT function.

Syntax: How to Calculate Elapsed Months

```
YM(fromdate, todate, 'outfield')
```

where:

fromdate

Integer (I4YM or I6YYM)

Is the start date in year-month format (for example, I4YM). If the date is not valid, the function returns a 0.

todate

Integer (I4YM or I6YYM)

Is the end date in year-month format. If the date is not valid, the function returns a 0.

outfield

Integer

Is the format of the output value enclosed in single quotation marks.

Note: If *fromdate* or *todate* is in integer year-month-day format (I6YMD or I8YYMD), simply divide by 100 to convert to year-month format and set the result to an integer. This drops the day portion of the date, which is now after the decimal point.

Example: Calculating Elapsed Months

The COMPUTE commands convert the dates from year-month-day to year-month format; then YM calculates the difference between the values in the HIRE_DATE/100 and DAT_INC/100 fields:

```
COMPUTE HIRE_MONTH/I4YM = HIRE_DATE/100; AND
COMPUTE MONTH_INC/I4YM = DAT_INC/100; AND
COMPUTE MONTHS_HIRED/I3 = YM(HIRE_MONTH, MONTH_INC, 'I3');
```

Format Conversion Functions

In this section:

Format conversion functions convert fields from one format to another.

ATODBL: Converting an Alphanumeric String to Double-Precision Format

How to:

Convert an Alphanumeric String to Double-Precision Format

The ATODBL function converts a number in alphanumeric format to decimal (double-precision) format.

Syntax: **How to Convert an Alphanumeric String to Double-Precision Format**

```
ATODBL(string, length, 'outfield')
```

where:

string

Alphanumeric

Is the alphanumeric string to be converted, or a field that contains the string.

length

Alphanumeric

Is the two-character length of *infield* in bytes. This can be a numeric constant, or a field that contains the value. If you specify a numeric constant, enclose it in single quotation marks. The maximum value is 15.

`outfield`

Decimal

Is the format of the output value enclosed in single quotation marks.

Example: Converting an Alphanumeric Field to Double-Precision Format

ATODBL converts the EMP_ID field into double-precision format and stores the result in D_EMP_ID:

```
COMPUTE D_EMP_ID/D12.2 = ATODBL(EMP_ID, '09', 'D12.2');
```

EDIT: Converting the Format of a Field

How to:

Convert the Format of a Field

The EDIT function converts an alphanumeric field that contains numeric characters to numeric format or converts a numeric field to alphanumeric format. It is useful when you need to manipulate a field using a command that requires a particular format.

When EDIT assigns a converted value to a new field, the format of the new field must correspond to the format of the returned value. For example, if EDIT converts a numeric field to alphanumeric format, you must give the new field an alphanumeric format:

```
DEFINE ALPHAPRICE/A6 = EDIT(PRICE);
```

EDIT deals with a symbol in the following way:

- When an alphanumeric field is converted to numeric format, a sign or decimal point in the field is acceptable and is stored in the numeric field.
- When converting a floating-point or packed-decimal field to alphanumeric format, EDIT removes the sign, the decimal point, and any number to the right of the decimal point. It then right-justifies the remaining digits and adds leading zeros to achieve the specified field length. Converting a number with more than nine significant digits in floating-point or packed-decimal format may produce an incorrect result.

EDIT also extracts characters from or adds characters to an alphanumeric string. For more information, see *EDIT: Extracting or Adding Characters* on page 222.

Syntax: How to Convert the Format of a Field

```
EDIT(fieldname);
```

where:

fieldname

Alphanumeric or Numeric

Is the field name.

Example: Converting From Numeric to Alphanumeric Format

EDIT converts HIRE_DATE (a legacy date format) to alphanumeric format. CHGDAT is then able to use the field, which it expects in alphanumeric format:

```
COMPUTE ALPHA_HIRE/A17 = EDIT(HIRE_DATE); AND
COMPUTE HIRE_MDY/A17 = CHGDAT('YMD', 'MDYYX', ALPHA_HIRE, 'A17');
```

FTOA: Converting a Number to Alphanumeric Format

How to:

Convert a Number to Alphanumeric Format

The FTOA function converts a number up to 16 digits long from numeric format to alphanumeric format. It retains the decimal positions of a number and right-justifies it with leading spaces. You can also add edit options to a number converted by FTOA.

When using FTOA to convert a number containing decimals to a character string, you must specify an alphanumeric format large enough to accommodate both the integer and decimal portions of the number. For example, a D12.2 format is converted to A14. If the output format is not large enough, decimals are truncated.

Syntax: How to Convert a Number to Alphanumeric Format

```
FTOA(number, '(format)', 'outfield')
```

where:

number

Numeric F or D (single and double-precision floating-point)

Is the number to be converted, or the name of the field that contains the number.

format

Alphanumeric

Is the output format of the number enclosed in both single quotation marks and parentheses. Only floating point single-precision and double-precision formats are supported. Include any edit options that you want to appear in the output. The D (floating-point double-precision) format automatically supplies commas.

If you use a field name for this argument, specify the name without quotation marks or parentheses. If you specify a format, the format must be enclosed in parentheses.

`outfield`

Alphanumeric

Is the format of the output value enclosed in single quotation marks. The length of this argument must be greater than the length of *number* and must account for edit options and a possible negative sign.

Example: Converting From Numeric to Alphanumeric Format

FTOA converts the GROSS field from floating point double-precision to alphanumeric format and stores the result in ALPHA_GROSS:

```
COMPUTE ALPHA_GROSS/A15 = FTOA(GROSS, '(D12.2)', 'A15');
```

HEXBYT: Converting a Decimal Integer to a Character

How to:

Convert a Decimal Integer to a Character

The HEXBYT function obtains the ASCII, EBCDIC, or Unicode character equivalent of a decimal integer, depending on your configuration and operating environment. It returns a single alphanumeric character in the ASCII, EBCDIC, or Unicode character set. You can use this function to produce characters that are not on your keyboard, similar to the CTRAN function.

In Unicode configurations, this function uses values in the range:

- 0 to 255 for 1-byte characters.
- 256 to 65535 for 2-byte characters.
- 65536 to 16777215 for 3-byte characters.
- 16777216 to 4294967295 for 4-byte characters (primarily for EBCDIC).

The display of special characters depends on your software and hardware; not all special characters may appear.

Syntax: How to Convert a Decimal Integer to a Character

```
HEXBYT(input, 'outfield')
```

where:

`input`

Integer

Is the decimal integer to be converted to a single character. In non-Unicode environments, a value greater than 255 is treated as the remainder of *input* divided by 256.

`outfield`

Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Example: Converting a Decimal Integer to a Character

HEXBYT converts LAST_INIT_CODE to its character equivalent and stores the result in LAST_INIT:

```
COMPUTE LAST_INIT_CODE/I3 = BYTVAL(LAST_NAME, 'I3'); AND
COMPUTE LAST_INIT/A1 = HEXBYT(LAST_INIT_CODE, 'A1');
```

ITONUM: Converting a Large Binary Integer to Double-Precision Format

How to:

Convert a Large Binary Integer to Double-Precision Format

The ITONUM function converts a large binary integer in a data source to double-precision format. Some programming languages and some data storage systems use large binary integer formats. However, large binary integers (more than 4 bytes in length) are not supported in the Master File so they require conversion to double-precision format.

You must specify how many of the right-most bytes in the input field are significant. The result is an 8-byte double-precision field.

Syntax: How to Convert a Large Binary Integer to Double-Precision Format

```
ITONUM(maxbytes, infield, 'outfield')
```

where:

`maxbytes`

Numeric

Is the maximum number of bytes in the 8-byte binary input field that have significant numeric data, including the binary sign. Valid values are:

5 ignores the left-most 3 bytes.

6 ignores the left-most 2 bytes.

7 ignores the left-most byte.

infield

Alphanumeric

Is the field that contains the binary number. Both the USAGE and ACTUAL formats of the field must be A8.

outfield

Decimal

Is the format of the output value enclosed in single quotation marks. The format must be D*n*.

Example: Converting a Large Binary Integer to Double-Precision Format

ITONUM converts the BINARYFLD field to double-precision format:

```
COMPUTE MYFLD/D14 = ITONUM(6, BINARYFLD, 'D14');
```

ITOPACK: Converting a Large Binary Integer to Packed-Decimal Format

How to:

Convert a Large Binary Integer to Packed-Decimal Format

The ITOPACK function converts a large binary integer in a data source to packed-decimal format. Some programming languages and some data storage systems use double-word binary integer formats. Large binary integers (more than 4 bytes in length) are not supported in the Master File so they require conversion to packed-decimal format.

You must specify how many of the right-most bytes in the input field are significant. The result is an 8-byte packed-decimal field of up to 15 significant numeric positions (for example, P15 or P16.2).

Limit: For a field defined as 'PIC 9(15) COMP' or the equivalent (15 significant digits), the maximum number that can be converted is 167,744,242,712,576.

Syntax: How to Convert a Large Binary Integer to Packed-Decimal Format

```
ITOPACK(maxbytes, infield, 'outfield')
```

where:

`maxbytes`

Numeric

Is the maximum number of bytes in the 8-byte binary input field that have significant numeric data, including the binary sign.

Valid values are:

`5` ignores the left-most 3 bytes (up to 11 significant positions).

`6` ignores the left-most 2 bytes (up to 14 significant positions).

`7` ignores the left-most byte (up to 15 significant positions).

`infield`

Alphanumeric

Is the field that contains the binary number. Both the USAGE and ACTUAL formats of the field must be A8.

`outfield`

Packed

Is the format of the output value enclosed in single quotation marks. The format must be P*n* or P*n.d*.

Example: Converting a Large Binary Integer to Packed-Decimal Format

ITOPACK converts the BINARYFLD field to packed-decimal format:

```
COMPUTE PACKFLD/P14.4 = ITOPACK(6, BINARYFLD, 'P14.4');
```

ITOZ: Converting a Number to Zoned Format

How to:

Convert to Zoned Format

The ITOZ function converts a number in numeric format to zoned format. Although a request cannot process zoned numbers, it can write zoned fields to an extract file for use by an external program.

Syntax: How to Convert to Zoned Format

```
ITOZ(outlength, number, 'outfield')
```

where:

outlength

Integer

Is the length of *number* in bytes. The maximum number of bytes is 15. The last byte includes the sign.

number

Numeric

Is the number to be converted, or the field that contains the number. The number is truncated to an integer before it is converted.

outfield

Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Example: Converting a Number to Zoned Format

ITOZ converts the CURR_SAL field to a zoned format:

```
COMPUTE ZONE_SAL/A8 = ITOZ(8, CURR_SAL, 'A8');
```

PCKOUT: Writing a Packed Number of Variable Length

How to:

Write a Packed Number of Variable Length

The PCKOUT function writes a packed number of variable length to an extract file. When a request saves a packed number to an extract file, it typically writes it as an 8- or 16-byte field regardless of its format specification. With PCKOUT, you can vary the field's length between 1 to 16 bytes.

Syntax: How to Write a Packed Number of Variable Length

```
PCKOUT(infield, outlength, 'outfield')
```

where:

infield

Numeric

Is the input field that contains the values. The field can be in packed, integer, floating-point, or double-precision format. If the field is not in integer format, its values are rounded to the nearest integer.

outlength

Numeric

Is the length of outfield from 1 to 16 bytes.

outfield

Alphanumeric

Is the format of the output value enclosed in single quotation marks. The function returns the field as alphanumeric although it contains packed data.

Example: Writing a Packed Number of Variable Length

PCKOUT converts the CURR_SAL field to a 5-byte packed field and stores the result in SHORT_SAL:

```
COMPUTE SHORT_SAL/A5 = PCKOUT(CURR_SAL, 5, 'A5');
```

Numeric Functions

In this section:

Numeric functions perform calculations on numeric constants and fields.

ABS: Calculating Absolute Value

How to:

Calculate Absolute Value

The ABS function returns the absolute value of a number.

Syntax: How to Calculate Absolute Value

```
ABS(argument)
```

where:

argument
Numeric

Is the value for which the absolute value is returned, the name of a field that contains the value, or an expression that returns the value. If you use an expression, use parentheses as needed to ensure the correct order of evaluation.

Example: Calculating Absolute Value

The first COMPUTE command creates the DIFF field, then ABS calculates the absolute value of DIFF:

```
COMPUTE DIFF/I5 = DELIVER_AMT - UNIT_SOLD; AND
COMPUTE ABS_DIFF/I5 = ABS(DIFF);
```

BAR: Producing a Bar Chart

How to:

Produce a Bar Chart

The BAR function produces a horizontal bar chart using repeating characters to form each bar. Optionally, you can create a scale to clarify the meaning of a bar chart by replacing the title of the column containing the bar with a scale.

Syntax: How to Produce a Bar Chart

```
BAR(barlength, infield, maxvalue, 'char', 'outfield')
```

where:

barlength
Numeric

Is the maximum length of the bar in characters. If this value is less than or equal to 0, the function does not return a bar.

`infield`

Numeric

Is the data field plotted as a bar chart.

`maxvalue`

Numeric

Is the maximum value of a bar. This value must be greater than the maximum value stored in `infield`. If `infield` is larger than `maxvalue`, the function uses `maxvalue` and returns a bar of maximum length.

`char`

Alphanumeric

Is the repeating character that creates the bars enclosed in single quotation marks. If you specify more than one character, only the first character is used.

`outfield`

Alphanumeric

Is the format of the output value enclosed in single quotation marks. The output field must be large enough to contain a bar of maximum length as defined by `barlength`.

Example: Producing a Bar Chart

BAR creates a bar chart for the CURR_SAL field, and stores the output in SAL_BAR. The bar created can be no longer than 30 characters long, and the value it represents can be no greater than 30,000.

```
COMPUTE SAL_BAR/A30 = BAR(30, CURR_SAL, 30000, '=', SAL_BAR);
```

CHKPCK: Validating a Packed Field

How to:

Validate a Packed Field

The CHKPCK function validates the data in a field described as packed format (if available on your platform). The function prevents a data exception from occurring when a request reads a field that is expected to contain a valid packed number but does not.

To use CHKPCK:

1. Ensure that the Master File (USAGE and ACTUAL attributes) defines the field as alphanumeric, not packed. This does *not* change the field data, which remains packed, but it enables the request to read the data without a data exception.
2. Call CHKPCK to examine the field. The function returns the output to a field defined as packed. If the value it examines is a valid packed number, the function returns the value; if the value is not packed, the function returns an error code.

Syntax: How to Validate a Packed Field

```
CHKPCK(inlength, infield, error, 'outfield')
```

where:

inlength

Numeric

Is the length of the packed field. It can be between 1 and 16 bytes.

infield

Alphanumeric

Is the name of the packed field. The field is described as alphanumeric, not packed.

error

Numeric

Is the error code that the function returns if a value is not packed. Choose an error code outside the range of data. The error code is first truncated to an integer, then converted to packed format. However, it may appear on a report with a decimal point because of the format of the output field.

outfield

Packed

Is the format of the output value enclosed in single quotation marks.

Example: Validating Packed Data

CHKPCK validates the values in the PACK_SAL field, and stores the result in the GOOD_PACK field. Values not in packed format return the error code -999. Values in packed format appear accurately.

```
COMPUTE GOOD_PACK/P8CM = CHKPCK(8, PACK_SAL, -999, GOOD_PACK);
```

DMOD, FMOD, and IMOD: Calculating the Remainder From a Division

How to:

Calculate the Remainder From a Division

The MOD functions calculate the remainder from a division. Each function returns the remainder in a different format.

The functions use the following formula.

`remainder = dividend - INT(dividend/divisor) * divisor`

- `DMOD` returns the remainder as a decimal number.
- `FMOD` returns the remainder as a floating point number.
- `IMOD` returns the remainder as an integer.

For information on the INT function, see *INT: Finding the Greatest Integer* on page 319.

Syntax: How to Calculate the Remainder From a Division

`function(dividend, divisor, 'outfield')`

where:

`function`
Is one of the following:

`DMOD` returns the remainder as a decimal number.

`FMOD` returns the remainder as a floating point number.

`IMOD` returns the remainder as an integer.

`dividend`
Numeric

Is the number being divided.

`divisor`
Numeric

Is the number dividing the dividend.

`outfield`
Numeric

Is the format of the output value enclosed in single quotation marks. The format is determined by the result returned by the specific function.

Example: Calculating the Remainder From a Division

IMOD divides ACCTNUMBER by 1000 and returns the remainder to LAST3_ACCT:

```
COMPUTE LAST3_ACCT/I3L = IMOD(ACCTNUMBER, 1000, LAST3_ACCT);
```

EXP: Raising *e* to the Nth Power

How to:

Raise e to the Nth Power

The EXP function raises the value "e" (approximately 2.72) to a specified power. This function is the inverse of the LOG function, which returns an argument's logarithm.

EXP calculates the result by adding terms of an infinite series. If a term adds less than .000001 percent to the sum, the function ends the calculation and returns the result as a double-precision number.

Syntax: How to Raise *e* to the Nth Power

```
EXP(power, 'outfield')
```

where:

power

Numeric

Is the power to which "e" is raised.

outfield

Floating point or Decimal

Is the format of the output value enclosed in single quotation marks.

Example: Raising *e* to the Nth Power

EXP raises e to the 2nd power:

```
COMPUTE E2/D12.2 = EXP(2, 'D12.2');
```

INT: Finding the Greatest Integer

How to:

Find the Greatest Integer

The INT function returns the integer component of a number.

Syntax: **How to Find the Greatest Integer**

```
INT(argument)
```

where:

argument

Numeric

Is the value for which the integer component is returned, the name of a field that contains the value, or an expression that returns the value. If you supply an expression, use parentheses as needed to ensure the correct order of evaluation.

Example: **Finding the Greatest Integer**

INT finds the greatest integer in the DED_AMT field and stores it in INT_DED_AMT:

```
COMPUTE INT_DED_AMT/I9 = INT(DED_AMT);
```

LOG: Calculating the Natural Logarithm

How to:

Calculate the Natural Logarithm

The LOG function returns the natural logarithm of a number.

Syntax: **How to Calculate the Natural Logarithm**

```
LOG(argument)
```

where:

argument

Numeric

Is the value for which the natural logarithm is calculated, the name of a field that contains the value, or an expression that returns the value. If you supply an expression, use parentheses as needed to ensure the correct order of evaluation. If *argument* is less than or equal to 0, LOG returns 0.

Example: **Calculating the Natural Logarithm**

LOG calculates the logarithm of the CURR_SAL field:

```
COMPUTE LOG_CURR_SAL/D12.2 = LOG(CURR_SAL);
```

MAX and MIN: Finding the Maximum or Minimum Value

How to:

Find the Maximum or Minimum Value

The MAX and MIN functions return the maximum or minimum value, respectively, from a list of values.

Syntax: How to Find the Maximum or Minimum Value

```
{MAX|MIN}(argument1, argument2, ...)
```

where:

`MAX`
: Returns the maximum value.

`MIN`
: Returns the minimum value.

`argument1, argument2`
: Numeric

  Are the values for which the maximum or minimum value is returned, the name of a field that contains the values, or an expression that returns the values. If you supply an expression, use parentheses as needed to ensure the correct order of evaluation.

Example: Determining the Minimum Value

MIN returns either the value of the ED_HRS field or the constant 30, whichever is lower:

```
COMPUTE MIN_EDHRS_30/D12.2 = MIN(ED_HRS, 30);
```

PRDNOR and PRDUNI: Generating Reproducible Random Numbers

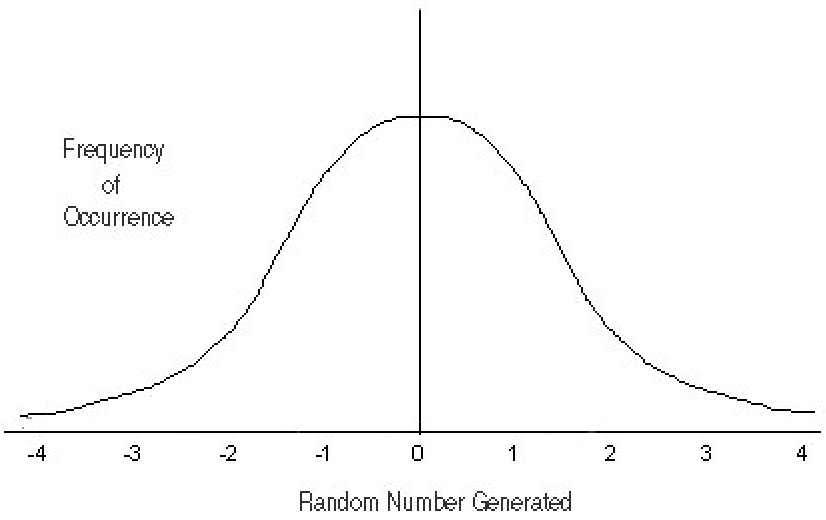
How to:

Generate Reproducible Random Numbers

The PRDNOR and PRDUNI functions generate reproducible random numbers:

❑ PRDNOR generates reproducible double-precision random numbers normally distributed with an arithmetic mean of 0 and a standard deviation of 1. If PRDNOR generates a large set of numbers, they have the following properties:

- The numbers lie roughly on a bell curve, as shown in the following figure. The bell curve is highest at the 0 mark, meaning that there are more numbers closer to 0 than farther away.



- The average of the numbers is close to 0.
- The numbers can be any size, but most are between 3 and -3.

- PRDUNI generates reproducible double-precision random numbers uniformly distributed between 0 and 1 (that is, any random number it generates has an equal probability of being anywhere between 0 and 1).

Syntax: How to Generate Reproducible Random Numbers

`{PRDNOR|PRDUNI}(`*`seed`*`, '`*`outfield`*`')`

where:

`PRDNOR`
: Generates reproducible double-precision random numbers normally distributed with an arithmetic mean of 0 and a standard deviation of 1.

`PRDUNI`
: Generates reproducible double-precision random numbers uniformly distributed between 0 and 1.

`seed`
: Numeric

Is the seed or the field that contains the seed, up to 9 digits. The seed is truncated to an integer.

`outfield`

Decimal

Is the format of the output value enclosed in single quotation marks.

Example: Generating Reproducible Random Numbers

PRDNOR assigns random numbers and stores them in RAND. The seed is 40. To produce a different set of numbers, change the seed.

```
COMPUTE RAND/D12.2 = PRDNOR(40, 'D12.2');
```

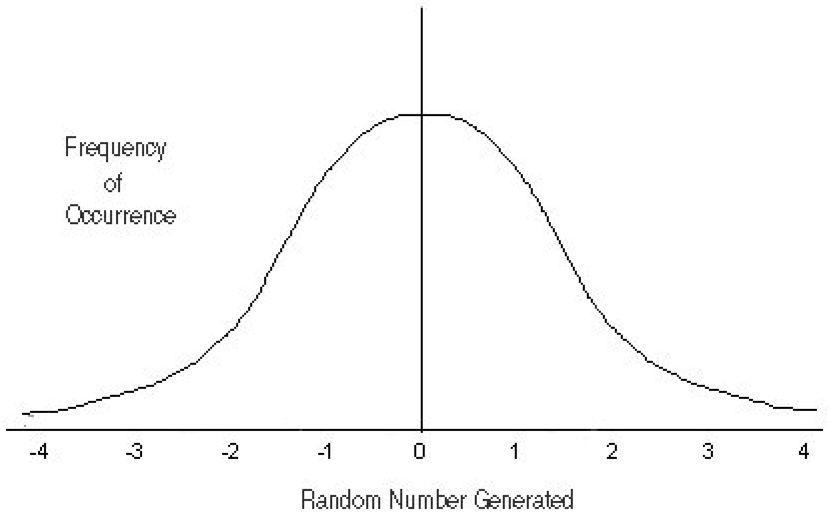
RDNORM and RDUNIF: Generating Random Numbers

How to:

Generate Random Numbers

The RDNORM and RDUNIF functions generate random numbers:

- ❑ RDNORM generates double-precision random numbers normally distributed with an arithmetic mean of 0 and a standard deviation of 1. If RDNORM generates a large set of numbers (between 1 and 32768), they have the following properties:
    - ❑ The numbers lie roughly on a bell curve, as shown in the following figure. The bell curve is highest at the 0 mark, meaning that there are more numbers closer to 0 than farther away.



    - ❑ The average of the numbers is close to 0.
    - ❑ The numbers can be any size, but most are between 3 and -3.
- ❑ RDUNIF generates double-precision random numbers uniformly distributed between 0 and 1 (that is, any random number it generates has an equal probability of being anywhere between 0 and 1).

Syntax: How to Generate Random Numbers

```
{RDNORM|RDUNIF}('outfield')
```

where:

`RDNORM`
: Generates double-precision random numbers normally distributed with an arithmetic mean of 0 and a standard deviation of 1.

`RDUNIF`
: Generates double-precision random numbers uniformly distributed between 0 and 1.

`outfield`
: Decimal

  Is the format of the output value enclosed in single quotation marks.

Example: Generating Random Numbers

RDNORM assigns random numbers and stores them in RAND.

```
COMPUTE RAND/D12.2 = RDNORM('D12.2');
```

SQRT: Calculating the Square Root

How to:

Calculate the Square Root

The SQRT function calculates the square root of a number.

Syntax: How to Calculate the Square Root

```
SQRT(argument)
```

where:

`argument`
: Numeric

  Is the value for which the square root is calculated, the name of a field that contains the value, or an expression that returns the value. If you supply an expression, use parentheses as needed to ensure the correct order of evaluation. If you supply a negative number, the result is zero.

Example: Calculating the Square Root

SQRT calculates the square root of LISTPR:

```
COMPUTE SQRT_LISTPR/D12.2 = SQRT(LISTPR);
```

System Functions

In this section:

FGETENV: Retrieving the Value of an Environment Variable

GETUSER: Retrieving a User ID

HHMMSS: Retrieving the Current Time

TODAY: Returning the Current Date

System functions call the operating system to obtain information about the operating environment.

FGETENV: Retrieving the Value of an Environment Variable

How to:

Retrieve the Value of an Environment Variable

The FGETENV function retrieves the value of an environment variable and returns it as an alphanumeric string.

Syntax: How to Retrieve the Value of an Environment Variable

```
FGETENV(varlength, 'varname', outfieldlen, 'outfield')
```

where:

varlength

Integer

Is the length of the environment variable name.

varname

Alphanumeric

Is the name of the environment variable.

outfieldlen

Integer

Is the length of the field in which the environment variable's value is stored.

outfield
Alphanumeric

Is the format of the output value enclosed in single quotation marks.

Example: Retrieving the Language Locale

Using the LANG environment variable, FGETENV retrieves the object location for the language locale:

```
COMPUTE LANG_LOCALE/A40 = FGETENV(4, 'LANG', 40, 'A40');
```

GETUSER: Retrieving a User ID

How to:

Retrieve a User ID

The GETUSER function retrieves the ID of the connected user.

Syntax: How to Retrieve a User ID

```
GETUSER('outfield')
```

where:

outfield
Alphanumeric

Is the format (at least A8) of the output value enclosed in single quotation marks. The length depends on the platform on which the function is issued. Provide a length as long as required for your platform; otherwise, the output will be truncated.

Example: Retrieving a User ID

GETUSER retrieves the user ID of the person running the request:

```
COMPUTE USERID/A8 = GETUSER('A8');
```

HHMMSS: Retrieving the Current Time

The HHMMSS function retrieves the current time from the operating system as an eight-character string, separating the hours, minutes, and seconds with periods. For details on how to use this function, see *HHMMSS: Retrieving the Current Time* on page 296.

TODAY: Returning the Current Date

The TODAY function retrieves the current date from the system. For details on how to use this function, see *TODAY: Returning the Current Date* on page 304.

A Running DB2 Web Query Reports Using the Java Batch Run Utility

The Java Batch Run utility (RUNWEBQRY) enables you to run a DB2 Web Query report from the command line so that the request can be submitted to a batch queue, without having to physically log on to DB2 Web Query. The utility accepts a single report procedure (.fex) that is in DB2 Web Query and executes the .fex via a Java program.

Topics:

Java Batch Run Utility Prerequisites

How to:

Invoke the Java Batch Run

Reference:

Retrieving Input Parameters for the RUNWEBQRY Command

The following are prerequisites for running the utility:

- ❑ DB2 Web Query must be installed and running on the system where the reporting and application servers are running.
- ❑ FOCEXECs which exist in the DB2 Web Query environment.
- ❑ An available user ID that is licensed for DB2 Web Query.

Procedure: How to Invoke the Java Batch Run

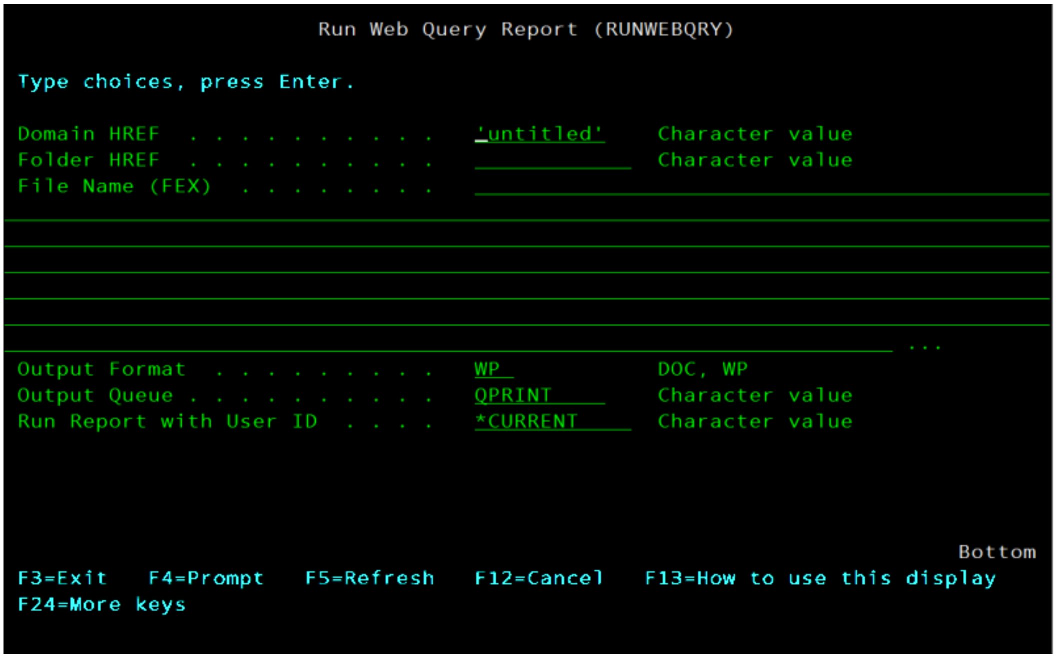
To invoke the utility from the command line:

1. Log in to the IBM i system via a 5250 terminal emulator.
2. At the command line, execute the following command:

   ```
   RUNWEBQRY
   ```

3. Press the *F4* key.

The Run Java batch in WebQuery (RUNWEBQRY) screen opens as shown in the following image.



```
                     Run Web Query Report (RUNWEBQRY)

Type choices, press Enter.

Domain HREF . . . . . . . . . . .   'untitled'     Character value
Folder HREF . . . . . . . . . . .   ____________   Character value
File Name (FEX)  . . . . . . . .   ____________________________________________
_______________________________________________________________________________
_______________________________________________________________________________
_______________________________________________________________________________
_______________________________________________________________________________
_______________________________________________________________________________
____________________________________________________________________ ...
Output Format  . . . . . . . . .   WP             DOC, WP
Output Queue . . . . . . . . . .   QPRINT____     Character value
Run Report with User ID  . . . .   *CURRENT___    Character value




                                                                        Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

4. On the screen, type values for the following input parameters:

`Domain href`

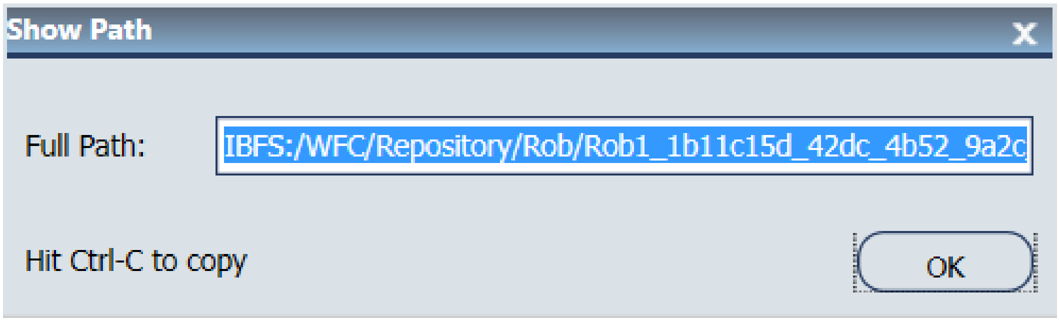
Is the domain name in Dashboard or an href. An href is an internal name given to an object. The href can be displayed via the Properties option in the Dashboard menu, which is available by right-clicking the mouse. You do not need to use the href if the name is exactly 8 characters. You need to use the href if the name is less than 8 characters or more than 8 characters. This input parameter can be retrieved from the Properties page by right-clicking the report name and choosing *Properties* from the drop-down menu.

`Folder href`

Is the folder name in Dashboard or an href. An href is an internal name given to an object. The href can be displayed via the Properties option in the Dashboard menu, which is available by right-clicking the mouse. You do not need to use the href if the name is exactly 12 characters. You need to use the href if the name is less than 12 characters or more than 12 characters. This input parameter can be retrieved from the Properties page by right-clicking the report name and choosing *Properties* from the drop-down menu.

File Name (fex)

Is the full path or name of the report FOCEXEC (fex) as displayed in the BI portal tree. This input parameter can be retrieved from the Show Path option by right-clicking the report name and choosing *Show Path* from the drop-down menu as shown in the following image. Press Ctrl+C to copy the fex's full path and report name.


Show Path

Full Path: IBFS:/WFC/Repository/Rob/Rob1_1b11c15d_42dc_4b52_9a2c

Hit Ctrl-C to copy

Output Format

Is the report format. Possible values are DOC or WP. WP is the default value.

Output Queue

Is the name of the outq on the IBM i system where the report will be sent. QPRINT is the default value.

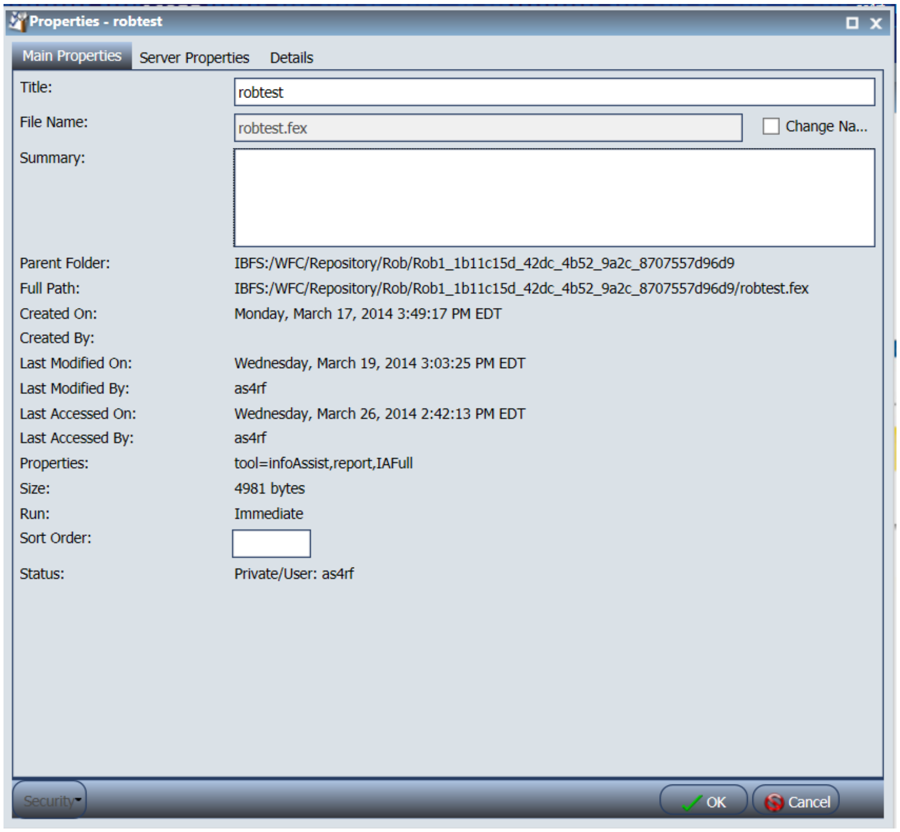
Run Report with User ID

Enables you to submit the job using another user ID. The default value is *CURRENT which means that the current user ID will be used to submit the job.

Note: The RUNWEBQRY command can be part of a job stream using SBMJOB to run multiple requests.

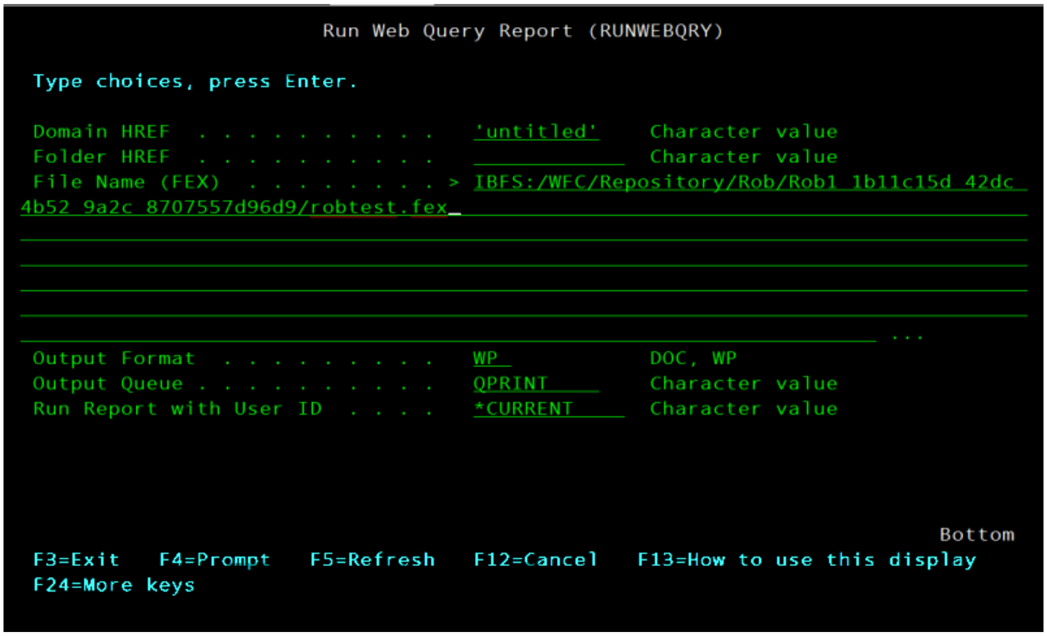
Reference: Retrieving Input Parameters for the RUNWEBQRY Command

The following is an example of a Properties page for retrieving parameter information. This page is accessed by right-clicking a report name in a report folder, and choosing *Properties* from the drop-down menu.



Title:	robtest
File Name:	robtest.fex
Summary:	
Parent Folder:	IBFS:/WFC/Repository/Rob/Rob1_1b11c15d_42dc_4b52_9a2c_8707557d96d9
Full Path:	IBFS:/WFC/Repository/Rob/Rob1_1b11c15d_42dc_4b52_9a2c_8707557d96d9/robtest.fex
Created On:	Monday, March 17, 2014 3:49:17 PM EDT
Created By:	
Last Modified On:	Wednesday, March 19, 2014 3:03:25 PM EDT
Last Modified By:	as4rf
Last Accessed On:	Wednesday, March 26, 2014 2:42:13 PM EDT
Last Accessed By:	as4rf
Properties:	tool=infoAssist,report,IAFull
Size:	4981 bytes
Run:	Immediate
Sort Order:	
Status:	Private/User: as4rf

The following image shows the input parameters populated from the example properties page.



```
                         Run Web Query Report (RUNWEBQRY)

 Type choices, press Enter.

 Domain HREF . . . . . . . . . .   'untitled'     Character value
 Folder HREF . . . . . . . . . .                  Character value
 File Name (FEX)  . . . . . . . > IBFS:/WFC/Repository/Rob/Rob1_1b11c15d_42dc_
4b52_9a2c_8707557d96d9/robtest.fex_







                                                                      ...
 Output Format  . . . . . . . . .  WP             DOC, WP
 Output Queue . . . . . . . . . .  QPRINT         Character value
 Run Report with User ID  . . . .  *CURRENT       Character value




                                                                        Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

IBM

B Describing an Individual Field

A field is the smallest meaningful element of data in a data source, but it can exhibit a number of complex characteristics. Master File attributes are used to describe these characteristics.

Topics:

Field Characteristics

The Master File describes the following field characteristics:

- The name of the field, as identified in the FIELDNAME attribute.
- Another name for the field, either its original name, as defined to its native data management system, or (for some types of data sources) a synonym of your own choosing, or (in some special cases) a pre-defined value that tells how to interpret the field, that you can use as an alternative name in requests. This alternative name is defined by the ALIAS attribute.
- How the field stores and displays data, specified by the ACTUAL, USAGE, and MISSING attributes.

  The ACTUAL attribute describes the type and length of the data as it is actually stored in the data source. For example, a field might be alphanumeric and 15 characters in length. Note that FOCUS data sources do not use the ACTUAL attribute, and instead use the USAGE attribute to describe the data as it is formatted. handles the storage.

  The USAGE attribute, also known by its alias, FORMAT, describes how a field is formatted when it appears in reports. You can also specify edit options, such as date formats, floating dollar signs, and zero suppression.

  The MISSING attribute enables null values to be entered into and read from a field in data sources that support null data, such as FOCUS data sources and most relational data sources.
- The option for a field to be virtual, rather than being stored in the data source, and have its value derived from information already in the data source. Virtual fields are specified by the DEFINE attribute.
- Optional field documentation for the developer, contained in the DESCRIPTION attribute.
- Acceptable data-entry values for the field, specified by the ACCEPT attribute.
- An alternative report column title for the field, described by the TITLE attribute.
- A 100-year window that assigns a century value to a two-digit year stored in the field. Two attributes define this window: DEFCENT and YRTHRESH.

The Field Name: FIELDNAME

Identify a field using FIELDNAME, the first attribute specified in a field declaration in the Master File. You can assign any name to a field, regardless of its name in its native data source. Likewise, for FOCUS data sources, you can assign any name to a field in a new data source.

When you generate a report, each column title in the report has the name of the field displayed in that column as its default, so assigning meaningful field names helps readers of the report. Alternatively, you can specify a different column title within a given report by using the AS phrase in the report request, as described in the manual, or a different default column title for all reports by using the TITLE attribute in the Master File, as described in *Alternative Report Column Titles: TITLE* on page 399.

Syntax: How to Identify the Field Name

```
FIELD[NAME] = field_name
```

where:

field_name

Is the name you are giving this field. It can be a maximum of 512 characters. Some restrictions apply to names longer than 12 characters, as described in *Restrictions for Field Names* on page 339. The name can include any combination of letters, digits, and underscores (_), and must contain at least one letter. Other characters are not recommended, and may cause problems in some operating environments or when resolving expressions.

It is recommended that you not use field names of the type Cn, En, and Xn (where n is any sequence of one or two digits) because these can be used to refer to report columns, HOLD file fields, and other special objects.

If you must use special characters because of a field report column title, consider using the TITLE attribute in the Master File to specify the title, as described in *Alternative Report Column Titles: TITLE* on page 399.

Reference: Usage Notes for FIELDNAME

Note the following rules when using FIELDNAME:

- **Alias.** FIELDNAME has an alias of FIELD.
- **Changes.** In a FOCUS data source, if the INDEX attribute has been set to I (that is, if an index has been created for the field), you cannot change the field name without rebuilding the data source. You may change the name in all other situations.

Reference: Restrictions for Field Names

The following restrictions apply to field names and aliases longer than 12 characters:

- You cannot use a field name longer than 12 characters to specify a cross-referenced field in a JOIN command when the cross-referenced file is a FOCUS data source.
- Indexed fields and text fields in FOCUS data sources cannot have field names longer than 12 characters. Indexed fields and text fields in XFOCUS data sources are not subject to this 12 character limitation. Long ALIAS names are supported for both types of data sources.
- A field name specified in an alternate file view cannot be qualified.
- The CHECK FILE command PICTURE and HOLD options display the first 11 characters of long names within the resulting diagram or HOLD file. A caret (>) in the 12th position indicates that the name is longer than the displayed portion.
- ?FF, ? HOLD, ? DEFINE

  These display up to 31 characters of the name, and display a caret (>) in the 32nd character to indicate a longer field name.

Using a Qualified Field Name

How to:

Specify a Qualified Field Name in a Request

Change the Qualifying Character

Requests can qualify all referenced field names and aliases with file and/or segment names, a useful technique when duplicate field names exist across segments in a Master File or in joined data sources.

The names of text fields and indexed fields in FOCUS Master Files are limited to 12 characters. Text fields and indexed fields in XFOCUS Master Files are not subject to this 12-character limitation. However, the aliases for text and indexed fields may be up to 512 characters. Field names up to 512 characters appear as column titles in TABLE reports if there is no TITLE attribute or AS phrase.

The default value for the SET FIELDNAME command, SET FIELDNAME=NEW, activates long and qualified field names. The syntax is described in the manual.

Syntax: How to Specify a Qualified Field Name in a Request

```
[filename.][segname.]fieldname
```

where:

filename

Is the name of the Master File or tag name. Tag names are used with the JOIN and COMBINE commands.

segname

Is the name of the segment in which the field resides.

fieldname

Is the name of the field.

Example: Qualifying a Field Name

The fully qualified name of the field EMP_ID in the EMPINFO segment of the EMPLOYEE data source is:

```
EMPLOYEE.EMPINFO.EMP_ID
```

Syntax: How to Change the Qualifying Character

```
SET QUALCHAR = qualcharacter
```

The period (.) is the default qualifying character.

Using a Duplicate Field Name

Field names are considered duplicates when you can reference two or more fields with the same field name or alias. Duplication may occur:

- ❑ If a name appears multiple times within a Master File.
- ❑ In a JOIN between two or more Master Files, or in a recursive JOIN.
- ❑ If you issue a COMBINE and do not specify a prefix.

Duplicate fields (those having the same field name and alias) are not allowed in the same segment. The second occurrence is never accessed, and the following message is generated when you issue CHECK and CREATE FILE:

```
(FOC1829) WARNING. FIELDNAME IS NOT UNIQUE WITHIN A SEGMENT: fieldname
```

Duplicate field names may exist across segments in a Master File. To retrieve such a field, you must qualify its name with the segment name in a request. If a field that appears multiple times in a Master File is not qualified in a request, the first field encountered in the Master File is the one retrieved.

Note: If a Master File includes duplicate field names for real fields and/or virtual fields, the following logic is used when retrieving a field:

- ❑ If only virtual fields are duplicated, the last virtual field is retrieved.
- ❑ If only real fields are duplicated, the first real field is retrieved.
- ❑ If a Master File has both a real field and one or more virtual fields with the same name, the last virtual field is retrieved.
- ❑ If a field defined outside of a Master File has the same name as a virtual or real field in a Master File, the last field defined outside of the Master File is retrieved.

Reports can include qualified names as column titles. The SET QUALTITLES command, discussed in the manual, determines whether reports display qualified column titles for duplicate field names. With SET QUALTITLES=ON, they display qualified column titles for duplicate field names even when the request itself does not specify qualified names. The default value, OFF, disables qualified column titles.

Rules for Evaluating a Qualified Field Name

The following rules are used to evaluate qualified field names:

- The maximum field name qualification is *filename.segname.fieldname*. For example:

  ```
  TABLE FILE EMPLOYEE
  PRINT EMPLOYEE.EMPINFO.EMP_ID
  END
  ```

  includes EMP_ID as a fully qualified field. The file name, EMPLOYEE, and the segment name, EMPINFO, are the field qualifiers.

  Qualifier names can also be duplicated. For example:

  ```
  FILENAME=CAR, SUFFIX=FOC
    SEGNAME=ORIGIN, SEGTYPE=S1
        FIELDNAME=COUNTRY, COUNTRY, A10, $
    SEGNAME=COMP, SEGTYPE=S1, PARENT=ORIGIN
        FIELDNAME=CAR, CARS, A16, $
          .
          .
          .
  TABLE FILE CAR
  PRINT CAR.COMP.CAR
  END
  ```

  This request prints the field with alias CARS. Both the file name and field name are CAR.

  A field name can be qualified with a single qualifier, either its file name or its segment name. For example:

  ```
  FILENAME=CAR, SUFFIX=FOC
     SEGNAME=ORIGIN, SEGTYPE=S1
        FIELDNAME=COUNTRY, COUNTRY, A10, $
     SEGNAME=COMP, SEGTYPE=S1, PARENT=ORIGIN
        FIELDNAME=CAR, CARS, A16, $
          .
          .
          .
  TABLE FILE CAR
  PRINT COMP.CAR AND CAR.CAR
  END
  ```

  This request prints the field with alias CARS twice.

  When there is a single qualifier, segment name takes precedence over file name. Therefore, if the file name and segment name are the same, the field qualified by the segment name is retrieved.

- If a field name begins with characters that are the same as the name of a prefix operator, it may be unclear whether a request is referencing that field name or a second field name prefixed with the operator. The value of the first field is retrieved, not the value calculated by applying the prefix operator to the second field. In the next example, there is a field whose unqualified field name is CNT.COUNTRY and another whose field name is COUNTRY:

  ```
  FILENAME=CAR, SUFFIX=FOC
     SEGNAME=ORIGIN, SEGTYPE=S1
        FIELDNAME=CNT.COUNTRY, ACNTRY, A10, $
        FIELDNAME=COUNTRY, BCNTRY, A10, $
  TABLE FILE CAR
  SUM CNT.COUNTRY
  END
  ```

  In this request, the string CNT.COUNTRY is interpreted as a reference to the field named CNT.COUNTRY, not as a reference to the prefix operator CNT. applied to the field named COUNTRY. Therefore, the request sums the field whose alias is ACNTRY. Although the field name CNT.COUNTRY contains a period as one of its characters, it is an unqualified field name. It is not a qualified name or a prefix operator acting on a field name, neither of which is allowed in a Master File. The request does not count instances of the field whose alias is BCNTRY.

- If a Master File has either a file name or segment name that is the same as a prefix operator, the value of the field within the segment is retrieved in requests, not the value calculated by applying the prefix operator to the field.

  For example:

  ```
  FILENAME=CAR, SUFFIX=FOC
     SEGNAME=ORIGIN, SEGTYPE=S1
        FIELDNAME=COUNTRY, COUNTRY, A10, $
     SEGNAME=PCT, SEGTYPE=S1, PARENT=ORIGIN
        FIELDNAME=CAR, CARS, I2, $
   TABLE FILE CAR
  SUM PCT.CAR PCT.PCT.CAR
  BY COUNTRY
  END
  ```

  This request sums the field with alias CARS first, and then the percent of CARS by COUNTRY.

- When a qualified field name can be evaluated as a choice between two levels of qualification, the field name with the higher level of qualification takes precedence.

  In the following example, the choice is between an unqualified field name (the field named ORIGIN.COUNTRY in the ORIGIN segment) and a field name with segment name qualification (the field named COUNTRY in the ORIGIN segment). The field with segment name qualification is retrieved:

  ```
  FILENAME=CAR, SUFFIX=FOC
     SEGNAME=ORIGIN, SEGTYPE=S1
        FIELDNAME=ORIGIN.COUNTRY, OCNTRY, A10, $
        FIELDNAME=COUNTRY, CNTRY, A10, $
  TABLE FILE CAR
  PRINT ORIGIN.COUNTRY
  END
  ```

  This request prints the field with alias CNTRY. To retrieve the field with alias OCNTRY, qualify its field name, ORIGIN.COUNTRY, with its segment name, ORIGIN:

  ```
  PRINT ORIGIN.ORIGIN.COUNTRY
  ```

- When a qualified field name can be evaluated as a choice between two field names with the same level of qualification, the field with the shortest basic field name length is retrieved. For example:

  ```
  FILENAME=CAR, SUFFIX=FOC
     SEGNAME=CAR, SEGTYPE=S1
        FIELDNAME=CAR.CAR, CAR1, A10, $
     SEGNAME=CAR.CAR, SEGTYPE=S1, PARENT=CAR
        FIELDNAME=CAR, CAR2, A10, $
  TABLE FILE CAR
  PRINT CAR.CAR.CAR
  END
  ```

  In this example, it is unclear if you intend CAR.CAR.CAR to refer to the field named CAR.CAR in the CAR segment, or the field named CAR in the CAR.CAR segment. (In either case, the name CAR.CAR is an unqualified name that contains a period, not a qualified name. Qualified names are not permitted in Master Files.)

  No matter what the intention, the qualified field name is exactly the same and there is no obvious choice between levels of qualification.

  Since the field with alias CAR2 has the shortest basic field name length, CAR2 is printed. This is different from the prior example, where the choice is between two levels of qualification. To retrieve the CAR1 field, you must specify its alias.

The Field Synonym: ALIAS

In this section:

Implementing a Field Synonym

You can assign every field an alternative name, or alias. A field alias may be its original name as defined to its native data source, any name you choose, or, in special cases, a predefined value. The way in which you assign the alias is determined by the type of data source and, in special cases, the role the field plays in the data source. After it has been assigned, you can use this alias in requests as a synonym for the regular field name. Assign this alternative name using the ALIAS attribute.

Example: Using a Field Synonym

In the EMPLOYEE data source, the name CURR_SAL is assigned to a field using the FIELDNAME attribute, and the alternative name CSAL is assigned to the same field using the ALIAS attribute:

```
FIELDNAME = CURR_SAL, ALIAS = CSAL, USAGE = D12.2M,  $
```

Both names are equally valid within a request. The following TABLE requests illustrate that they are functionally identical, refer to the same field, and produce the same result:

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL BY EMP_ID
END

TABLE FILE EMPLOYEE
PRINT CSAL BY EMP_ID
END
```

Note: In extract files (HOLD, PCHOLD), the field name is used to identify fields, not the ALIAS.

Implementing a Field Synonym

The value you assign to ALIAS must conform to the same naming conventions to which the FIELDNAME attribute is subject, unless stated otherwise. Assign a value to ALIAS in the following way for the following types of data sources:

- **Relational data sources.** ALIAS describes the field original column name as defined in the relational table.
- **Sequential data sources.** ALIAS describes a synonym, or alternative name, that you can use in a request to identify the field. You can assign any name as the alias. Many users choose a shorter version of the primary name of the field. For example, if the field name is LAST_NAME, the alias might be LN. The ALIAS attribute is required in the Master File, but it can have the value blank.

  Note that ALIAS is used in a different way for sequenced repeating fields, where its value is ORDER, as well as for RECTYPE and MAPVALUE fields when the data source includes multiple record types.
- **FOCUS data sources.** ALIAS describes a synonym, or alternative name, that you can use in a request to identify the field. You can assign any name as the alias. Many users choose a shorter version of the primary name of the field. For example, if the field name is LAST_NAME, the alias might be LN. The ALIAS attribute is required in the Master File, but it can have the value blank. Aliases can be changed without rebuilding the data source. If an alias is referred to in other data sources, similar changes may be needed in those Master Files.

The Displayed Data Type: USAGE

In this section:

This attribute, which is also known as FORMAT, describes how to format a field when displaying it in a report or using it in a calculation.

Specifying a Display Format

How to:

Specify a Display Format

Reference:

Usage Notes for USAGE

For FOCUS data sources, which do not use the ACTUAL attribute, USAGE also specifies how to store the field. For other types of data sources, assign a USAGE value that corresponds to the ACTUAL value, to identify the field as the same data type used to store it in the data source. If the data is store as alphanumeric, assign the USAGE based on how the field will be displayed in your reports. The conversion is done automatically. For instructions on which ACTUAL values correspond to which USAGE values, see the documentation for the specific data adapter.

In addition to selecting the data type and length, you can also specify display options, such as date formatting, floating dollar signs, and zero suppression. Use these options to customize how the field appears in reports.

Syntax: How to Specify a Display Format

```
USAGE = tl[d]
```

where:

t

Is the data type. Valid values are A (alphanumeric), F (floating-point single-precision), D (floating-point double-precision), I (integer), P (packed decimal), D, W, M, Q, or Y used in a valid combination (date), and TX (text).

l

Is a length specification. The specification varies according to the data type. See the section for each data type for more information. Note that you do not specify a length for date format fields.

d

Is one or more display options. Different data types offer different display options. See the section for each data type for more information.

The complete USAGE value cannot exceed eight characters.

The values that you specify for type and field length determine the number of print positions allocated for displaying or storing the field. Display options only affect displayed or printed fields. They are not active for non-display retrievals, such as extract files.

Note: If a numeric field cannot display with the USAGE format given (for example, the result of aggregation is too large), asterisks appear.

See the sections for each format type for examples and additional information.

Reference: Usage Notes for USAGE

Note the following rules when using USAGE:

- **Alias.** USAGE has an alias of FORMAT.
- **Changes.** For most data sources, you can change the type and length specifications of USAGE only to other types and lengths valid for the ACTUAL attribute of that field. You can change display options at any time.

  For FOCUS data sources, you cannot change the type specification. You can change the length specification for I, F, D, and P fields, because this affects only display, not storage. You cannot change the decimal part of the length specification for P fields. You can change the length specification of A (alphanumeric) fields only if you use the REBUILD facility. You can change display options at any time.

Data Type Formats

You can specify several types of formats:

- **Numeric.** There are four types of numeric formats: integer, floating-point single-precision, floating-point double-precision, and packed decimal. See *Numeric Display Options* on page 353 for additional information.
- **Alphanumeric.** You can use alphanumeric format for any value to be interpreted as a sequence of characters and composed of any combination of digits, letters, and other characters.
- **Date.** The date format enables you to define date components, such as year, quarter, month, day, and day of week to:
  - Sort by date.
  - Do date comparisons and arithmetic with dates.
  - Validate dates automatically in transactions.

  Note that for some applications, such as assigning a date value using the DECODE function, you may wish instead to use alphanumeric, integer, or packed-decimal fields with date display options, which provide partial date functionality.

- **Date-Time.** The date-time format supports both the date and the time, similar to the timestamp data types available in many relational data sources. Date-time fields are stored in eight, ten, or 12 bytes: four bytes for date and either four, six, or eight bytes for time, depending on whether the format specifies a microsecond or nanosecond. Computations only allow direct assignment within data types. All other operations are accomplished through a set of date-time functions.
- **Text.** Text fields can be used to store large amounts of data and display it with line breaks.

Integer Format

You can use integer format for whole numbers. An integer is any value composed of the digits zero to nine, without a decimal point.

You can also use integer fields with date display options to provide limited date support. This use of integer fields is described in the *Alphanumeric and Numeric Formats With Date Display Options* on page 371.

The integer USAGE type is I. See *Numeric Display Options* on page 353. The format of the length specification is:

n

where:

n

Is the number of digits to display. The maximum length is 11, which must include the digits and a leading minus sign if the field contains a negative value. You can also specify a number of decimal places (up to *n* - 1), and the number will display with a decimal point before that number of digits.

For example:

Format	Display
`I6`	`  4316`
`I6.2`	` 43.16`
`I2`	`22`
`I4`	` -617`

Floating-Point Double-Precision Format

You can use floating-point double-precision format for any value composed of the digits zero to nine and an optional decimal point.

The floating-point double-precision USAGE type is D. See *Numeric Display Options* on page 353 for the compatible display options. The length specification format is:

```
t[.s]
```

where:

t

Is the number of characters to display up to a maximum of 33, including the numeric digits, an optional decimal point, and a leading minus sign if the field contains a negative value. The number of significant digits supported varies with the operating environment.

s

Is the number of digits that follow the decimal point. It can be a maximum of 31 and must be less than *t*.

For example:

Format	Display
`D8.2`	`3,187.54`
`D8`	`416`

In the case of D8.2, the 8 represents the maximum number of places, including the decimal point and decimal places. The 2 represents how many of these eight places are decimal places. The commas are automatically included in the display, and are not counted in the total.

Floating-Point Single-Precision Format

You can use floating-point single-precision format for any number, including numbers with decimal positions. The number is composed of the digits 0 to 9, including an optional decimal point. This format is intended for use with smaller decimal numbers. Unlike floating-point double-precision format, its length cannot exceed nine positions.

The floating-point single-precision USAGE type is F. Compatible display options are described in *Numeric Display Options* on page 353. The length specification format is:

`t[.s]`

where:

t

Is the number of characters to display, up to a maximum of 33, including the numeric digits, an optional decimal point, and a leading minus sign if the field contains a negative value. The number of significant digits supported varies with the operating environment.

s

Is the number of digits that follow the decimal point. It can be up to 31 digits and must be less than *t*.

For example:

Format	Display
`F5.1`	`614.2`
`F4`	`318`

Packed-Decimal Format

You can use packed-decimal format for any number, including decimal numbers. A decimal number is any value composed of the digits zero to nine, including an optional decimal point.

You can also use packed-decimal fields with date display options to provide limited date support. See *Alphanumeric and Numeric Formats With Date Display Options* on page 371.

The packed-decimal USAGE type is P. The compatible display options are described in *Numeric Display Options* on page 353.

The length specification format is:

`t[.s]`

where:

t

Is the number of characters to display, up to 33, including a maximum of 31 digits, an optional decimal point, and a leading minus sign if the field contains a negative value.

s

Is the number of digits that follow the decimal point. It can be up to 31 digits and must be less than *t*.

For example:

Format	Display
`P9.3`	`4168.368`
`P7`	`617542`

Numeric Display Options

Display options may be used to edit numeric formats. These options only affect how the data in the field is printed or appears on the screen, not how it is stored in your data source.

Edit Option	Meaning	Effect
`-`	Minus sign	Displays a minus sign to the right of negative numeric data. **Note:** Not supported with format options B, E, R, T, DMY, MDY, and YMD.
`%`	Percent sign	Displays a percent sign, along with numeric data. Does not calculate the percent.
`A`	Negative suppression	Displays the absolute value of the number, but does not affect the stored value. ❑ If you propagate a field with a negative suppression USAGE attribute to a HOLD file, the HOLD file contains the signed values. The negative suppression USAGE attribute is also propagated to the HOLD file so that if you run a report request against the HOLD file, the minus signs are suppressed on the report output. ❑ The negative suppression option cannot be used in with the following display options: ❑ B (bracket negative). ❑ R (credit negative). ❑ - (right side negative).

Edit Option	Meaning	Effect
`B`	Bracket negative	Encloses negative numbers in parentheses.
`c`	Comma suppress	Suppresses the display of commas. Used with numeric format options M and N (floating and non-floating dollar sign) and data format D (floating-point double-precision).
`C`	Comma edit	Inserts a comma after every third significant digit, or a period instead of a comma if continental decimal notation is in use.
`DMY`	Day-Month-Year	Displays alphanumeric or integer data as a date in the form day/month/year.
`E`	Scientific notation	Displays only significant digits.
`L`	Leading zeroes	Adds leading zeroes.
`M`	Floating currency symbol ($ for US code page)	Places a floating currency symbol to the left of the highest significant digit. The default currency symbol depends on the code page. You can use the SET CURRSYMB=*symbol* command to specify any character as the currency symbol or one of the following currency codes: USD or '$' specifies U. S. dollars. GBP specifies the British pound. JPY specifies the Japanese yen or Chinese yuan. EUR specifies the Euro.
`MDY`	Month-Day-Year	Displays alphanumeric or integer data as a date in the form month/day/year.

Edit Option	Meaning	Effect
N	Fixed currency symbol ($ for US code page)	Places a currency symbol to the left of the field. The symbol appears only on the first detail line of each page. The default currency symbol depends on the code page. You can use the SET CURRSYMB=*symbol* command to specify any character as the currency symbol or one of the following currency codes: USD or '$' specifies U. S. dollars. GBP specifies the British pound. JPY specifies the Japanese yen or Chinese yuan. EUR specifies the Euro.
R	Credit (CR) negative	Places CR after negative numbers.
S	Zero suppress	If the data value is zero, prints a blank in its place.
T	Month translation	Displays the month as a three-character abbreviation.
YMD	Year-Month-Day	Displays alphanumeric or integer data as a date in the form year/month/day.

Example: Using Numeric Display Options

The following table shows examples of the display options that are available for numeric fields.

Option	Format	Data	Display

Minus sign	`I2-`
	`D7-`
	`F7.2-`
		`-21`
		`-6148`
		`-8878`
			`21-`
			`6148-`
			`8878.00-`

Percent sign	`I2%`
	`D7%`
	`F3.2%`
		`21`
		`6148`
		`48`
			`21%`
			`6,148%`
			`48.00%`

Comma suppression	`D6c`
	`D7Mc`
	`D7Nc`
		`41376`
		`6148`
		`6148`
			`41376`
			`$6148`
			`$   6148`

Option	Format	Data	Display
Comma inclusion	`I6C`	`41376`	`41,376`
Zero suppression	`D6S`	`0`	
Bracket negative	`I6B`	`-64187`	`(64187)`
Credit negative	`I8R`	`-3167`	`3167 CR`
Leading zeroes	`F4L`	`31`	`0031`
Floating dollar	`D7M`	`6148`	`$6,148`
Non-floating dollar	`D7N`	`5432`	`$    5,432`
Scientific notation	`D12.5E`	`1234.5`	`0.12345D+04`

Year/month/day	`I6YMD`
	`I8YYMD`
		`980421`
		`19980421`
			`98/04/21`
			`1998/04/21`

Month/day/year	`I6MDY`
	`I8MDYY`
		`042198`
		`04211998`
			`04/21/98`
			`04/21/1998`

Day/month/year	`I6DMY`
	`I8DMYY`
		`210498`
		`21041998`
			`21/04/98`
			`21/04/1998`

Month translation	`I2MT`	`07`	`JUL`

Several display options can be combined, as shown:

Format	Data	Display
`I5CB`	`-61874`	`(61,874)`

All of the options may be specified in any order. Options M and N (floating and non-floating dollar sign) and data format D (floating-point double-precision) automatically invoke option C (comma). Options L and S cannot be used together. Option T (Translate) can be included anywhere in an alphanumeric or integer USAGE specification that includes the M (month) display option. Date display options (D, M, T, and Y), which cannot be used with floating-point fields, are described in *Alphanumeric and Numeric Formats With Date Display Options* on page 371.

Extended Currency Symbol Display Options

Reference:

Extended Currency Symbol Formats

You can select a currency symbol for display in report output regardless of the default currency symbol configured for National Language Support (NLS). Use the extended currency symbol format in place of the floating dollar (M) or non-floating dollar (N) display option. When you use the floating dollar (M) or non-floating dollar (N) display option, the currency symbol associated with the default code page is displayed. For example, when you use an American English code page, the dollar sign is displayed.

The extended currency symbol format allows you to display a symbol other than the dollar sign. For example, you can display the symbol for a United States dollar, a British pound, a Japanese yen or Chinese yuan, or the euro. Extended currency symbol support is available for numeric formats (I, D, F, and P).

The extended currency symbol formats are specified as two-character combinations *in the last positions* of any numeric display format. The first character in the combination can be either an exclamation point (!) or a colon (:). The colon is the recommended character because it will work in all ASCII and EBCDIC code pages. The exclamation point is not consistent on all EBCDIC code pages and may produce unexpected behavior if the code page you are using translates the exclamation point differently. The following table lists the supported extended currency display options:

Display Option	Description	Example
:d or !d	Fixed dollar sign.	D12.2:d
:D or !D	Floating dollar sign.	D12.2:D
:e or !e	Fixed euro symbol.	F9.2:e
:E or !E	Floating euro symbol on the left side.	F9.2:E
:F or !F	Floating euro symbol on the right side.	F9.2:F
:l or !l	Fixed British pound sign.	D12.1:l
:L or !L	Floating British pound sign.	D12.1:L

Display Option	Description	Example
:y or !y	Fixed Japanese yen or Chinese yuan symbol.	I9:y
:Y or !Y	Floating Japanese yen or Chinese yuan symbol.	I9:Y

Reference: Extended Currency Symbol Formats

The following guidelines apply:

- ❏ A format specification cannot be longer than eight characters.
- ❏ The extended currency option must be the last option in the format.
- ❏ The extended currency symbol format cannot include the floating (M) or non-floating (N) display option.
- ❏ A non-floating currency symbol is displayed only on the first row of a report page. If you use field-based reformatting (as in the example that follows) to display multiple currency symbols in a report column, only the symbol associated with the first row is displayed. In this case, do not use non-floating currency symbols.
- ❏ Lowercase letters are transmitted as uppercase letters by the terminal I/O procedures. Therefore, the fixed extended currency symbols can only be specified in a procedure.
- ❏ Extended currency symbol formats can be used with fields in floating point, decimal, packed, and integer formats. Alphanumeric and variable character formats cannot be used.

Alphanumeric Format

Reference:

Usage Notes for 4K Alphanumeric Fields

You can use alphanumeric format for any value to be interpreted as a sequence of characters and composed of any combination of digits, letters, and other characters.

You can also use alphanumeric fields with date display options to provide limited date support. This use of alphanumeric fields is described in *Alphanumeric and Numeric Formats With Date Display Options* on page 371.

The alphanumeric USAGE type is A. The format of the length specification is n, where n is the maximum number of characters in the field. You can have up to 3968 bytes in an alphanumeric field in a FOCUS file segment, and up to 4096 bytes in an XFOCUS file segment. You can have up to 4095 bytes in a fixed-format sequential data source. You may define the length in the Master File, a DEFINE FILE command, or a COMPUTE command.

For example:

Format	Display
`A522`	`The minutes of today's meeting were submitted...`
`A2`	`B3`
`A24`	`127-A429-BYQ-49`

Standard numeric display options are not available for the alphanumeric data format. However, alphanumeric data can be printed under the control of a pattern that is supplied at run time. For instance, if you are displaying a product code in parts, with each part separated by a "-", include the following in a DEFINE command:

```
PRODCODE/A11 = EDIT (fieldname,'999-999-999') ;
```

where:

`fieldname`

Is the existing field name, not the newly defined field name.

If the value is 716431014, the PRODCODE appears as 716-431-014. See the manual for more information.

Reference: Usage Notes for 4K Alphanumeric Fields

- Long alphanumeric fields cannot be indexed.
- For FOCUS data sources, a segment still has to fit on a 4K page. Thus, the maximum length of an alphanumeric field depends on the length of the other fields within its segment.
- You can print or hold long alphanumeric fields, but you cannot view them online.
- Long alphanumeric fields may be used as keys.

Date Formats

Date format enables you to define a field as a date, then manipulate the field value and display that value in ways appropriate to a date. Using date format, you can:

- Define date components, such as year, quarter, month, day, and day of week, and extract them easily from date fields.
- Sort reports into date sequence, regardless of how the date appears.
- Perform arithmetic with dates and compare dates without resorting to special date-handling functions.
- Refer to dates in a natural way, such as JAN 1 1995, without regard to display or editing formats.
- Automatically validate dates in transactions.

Date Display Options

Reference:

How Field Formats Y, YY, M, and W Are Stored

Date Literals Interpretation Table

The date format does not specify type or length. Instead, it specifies date component options (D, W, M, Q, Y, and YY) and display options. These options are shown in the following chart.

Display Option	Meaning	Effect
`D`	Day	Prints a value from 1 to 31 for the day.
`M`	Month	Prints a value from 1 to 12 for the month.
`Y`	Year	Prints a two-digit year.
`YY`	Four-digit year	Prints a four-digit year.
`T`	Translate month or day	Prints a three-letter abbreviation for months in uppercase, if M is included in the USAGE specification. Otherwise, it prints day of week.

Display Option	Meaning	Effect
`t`	Translate month or day	Functions the same as uppercase T (described above), except that the first letter of the month or day is uppercase and the following letters are lowercase.*
`TR`	Translate month or day	Functions the same as uppercase T (described above), except that the entire month or day name is printed instead of an abbreviation.
`tr`	Translate month or day	Functions the same as lowercase t (described above), except that the entire month or day name is printed instead of an abbreviation.*
`Q`	Quarter	Prints Q1 - Q4.
`W`	Day-of-Week	If it is included in a USAGE specification with other date component options, prints a three-letter abbreviation of the day of the week in uppercase. If it is the only date component option in the USAGE specification, it prints the number of the day of the week (1-7; Mon=1).
`w`	Day-of-Week	Functions the same as uppercase W (described above), except that the first letter is uppercase and the following letters are lowercase.*
`WR`	Day-of-Week	Functions the same as uppercase W (described above), except that the entire day name is printed instead of an abbreviation.
`wr`	Day-of-Week	Functions the same as lowercase w (described above), except that the entire day name is printed instead of an abbreviation.*
`J[UL]or JULIAN`	Julian format	Prints date in Julian format.

Display Option	Meaning	Effect
`YYJ[UL]`	Julian format	Prints a Julian format date in the format YYYYDDD. The 7-digit format displays the four-digit year and the number of days counting from January 1. For example, January 3, 2001 in Julian format is 2001003.

***Note:** When using these display options, be sure they are actually stored in the Master File as lowercase letters.

The following combinations of date components are not supported in date formats:

`I2M, A2M, I2MD, A2MD`

Reference: How Field Formats Y, YY, M, and W Are Stored

The Y, YY, and M formats are not smart dates. Smart date formats YMD and YYMD are stored as an offset from the base date of 12/31/1900. Smart date formats YM, YQ, YYM, and YYQ are stored as an offset from the base date 01/1901. W formats are stored as integers with a display length of one, containing values 1-7 representing the days of the week. Y, YY, and M formats are stored as integers. Y and M have display lengths of two. YY has a display length of four. When using Y and YY field formats, keep in mind these two important points:

- The Y formats do not sort based on DEFCENT and YRTHRESH settings. A field with a format of Y does not equal a YY field, as this is not a displacement, but a 4-digit integer.
- It is possible to use DEFCENT and YRTHRESH to convert a field from Y to YY format.

Reference: Date Literals Interpretation Table

This table illustrates the behavior of date formats. The columns indicate the number of input digits for a date format. The rows indicate the usage or format of the field. The intersection of row and column describes the result of input and format.

Date Format	1	2	3	4
`YYMD`	*	*	`CC00/0m/dd`	`CC00/mm/dd`
`MDYY`	*	*	*	*
`DMYY`	*	*	*	*

Date Format	1	2	3	4
YMD	*	*	CC00/0m/dd	CC00/mm/dd
MDY	*	*	*	*
DMY	*	*	*	*
YYM	CC00/0m	CC00/mm	CC0y/mm	CCyy/mm
MYY	*	*	*	*
YM	CC00/0m	CC00/mm	CC0y/mm	CCyy/mm
MY	*	*	0m/CCyy	mm/CCyy
M	0m	mm	*	*
YYQ	CC00/q	CC0y/q	CCyy/q	0yyy/q
QYY	*	*	q/CCyy	*
YQ	CC00/q	CC0y/q	CCyy/q	0yyy/q
QY	*	*	q/CCyy	*
Q	q	*	*	*
JUL	00/00d	00/0dd	00/ddd	0y/ddd
YYJUL	CC00/00d	CC00/0dd	CC00/ddd	CC0y/ddd
YY	000y	00yy	0yyy	yyyy
Y	0y	yy	*	*
D	0d	dd	*	*
W	w	*	*	*

Date Format	5	6	7	8
YYMD	CC0y/mm/dd	CCyy/mm/dd	0yyy/mm/dd	yyyy/mm/dd
MDYY	0m/dd/CCyy	mm/dd/CCyy	0m/dd/yyyy	mm/dd/yyyy
DMYY	0d/mm/CCyy	dd/mm/CCyy	0d/mm/yyyy	dd/mm/yyyy

Date Format	5	6	7	8
YMD	CC0y/mm/dd	CCyy/mm/dd	0yyy/mm/dd	yyyy/mm/dd
MDY	0m/dd/CCyy	mm/dd/CCyy	0m/dd/yyyy	mm/dd/yyyy
DMY	0d/mm/CCyy	dd/mm/CCyy	0d/mm/yyyy	dd/mm/yyyy
YYM	0yyy/mm	yyyy/mm	*	*
MYY	0m/yyyy	mm/yyyy	*	*
YM	0yyy/mm	yyyy/mm	*	*
MY	0m/yyyy	mm/yyyy	*	*
M	*	*	*	*
YYQ	yyyy/q	*	*	*
QYY	q/yyyy	*	*	*
YQ	yyyy/q	*	*	*
QY	q/yyyy	*	*	*
Q	*	*	*	*
JUL	yy/ddd	*	*	*
YYJUL	CCyy/ddd	0yyy/ddd	yyyy/ddd	*
YY	*	*	*	*
Y	*	*	*	*
D	*	*	*	*
W	*	*	*	*

Note:

- CC stands for two century digits provided by DFC/YRT settings.
- * stands for message FOC177 (invalid date constant).
- Date literals are read from right to left. Date literals and fields can be used in computational expressions, as described in the manual.

Controlling the Date Separator

You can control the date separators when the date appears. In basic date format, such as YMD and MDYY, the date components appear separated by a slash character (/). The same is true for the year-month format, which appears with the year and quarter separated by a blank (for example, 94 Q3 or Q3 1994). The single component formats display just the single number or name.

The separating character can also be a period, a dash, or a blank, or can even be eliminated entirely. The following table shows the USAGE specifications for changing the separating character.

Format	Display
`YMD`	`93/12/24`
`Y.M.D`	`93.12.24`
`Y-M`	`93-12`
`YBMBD`	`93 12 24` (The letter B signifies blank spaces.)
`Y|M|D`	`931224` (The concatenation symbol (|) eliminates the separation character.)

Note:

- You can change the date separator in the following date formats: YYMD, MDYY, DMYY, YMD, MDY, DMY, YYM, MYY, YM, MY, YYQ, QYY, YQ, and QY.
- You cannot change the date separator in a format that includes date translation options.
- You cannot change the date separator (/) in an alphanumeric or numeric format with date display options (for example, I8YYMD).

Date Translation

Numeric months and days can be replaced by a translation, such as JAN, January, Wed, or Wednesday. The translated month or day can be abbreviated to three characters or fully spelled out. It can appear in either uppercase or lowercase. In addition, the day of the week (for example, Monday) can be appended to the beginning or end of the date. All of these options are independent of each other.

Translation	Display
`MT`	`JAN`
`Mt`	`Jan`
`MTR`	`JANUARY`
`Mtr`	`January`
`WR`	`MONDAY`
`wr`	`Monday`

Example: Using a Date Format

The following chart shows sample USAGE and ACTUAL formats for data stored in a non-FOCUS data source. The Value column shows the actual data value, and the Display column shows how the data appears.

USAGE	ACTUAL	Value	Display
`wrMtrDYY`	`A6YMD`	`990315`	`Monday, March 15 1999`
`YQ`	`A6YMD`	`990315`	`99 Q1`
`QYY`	`A6YMD`	`990315`	`Q1 1999`
`YMD`	`A6`	`990315`	`99/03/15`
`MDYY`	`A6YMD`	`990315`	`03/15/1999`

Note that the date attributes in the ACTUAL format specify the order in which the date is stored in the non-FOCUS data source. If the ACTUAL format does not specify the order of the month, day, and year, it is inferred from the USAGE format.

Using a Date Field

A field formatted as a date is automatically validated when entered. It can be entered as a natural date literal (for example, JAN 12 1999) or as a numeric date literal (for example, 011299).

Natural date literals enable you to specify a date in a natural, easily understandable way, by including spaces between date components and using abbreviations of month names. For example, April 25, 1999 can be specified as any of the following natural date literals:

```
APR 25 1999
25 APR 1999
1999 APR 25
```

Natural date literals can be used in all date computations, and all methods of data source updating. The following code shows examples:

```
In WHERE screening                    WHERE MYDATE IS 'APR 25 1999'
In arithmetic expressions             MYDATE - '1999 APR 25'
In computational date comparisons     IF MYDATE GT '25 APR 1999'In
comma-delimited data           ...,MYDATE = APR 25 1999, ...
```

The following chart describes the format of natural date literals.

Literal	Format
Year-month-day	Four-digit year, uppercase three-character abbreviation, or uppercase full name, of the month, and one-digit or two-digit day of the month (for example, 1999 APR 25 or APRIL 25 1999).
Year-month	Year and month as described above.
Year-quarter	Year as described above, Q plus quarter number for the quarter (for example, 1999 Q3).
Month	Month as described above.
Quarter	Quarter as described above.
Day of week	Three-character, uppercase abbreviation, or full, uppercase name, of the day (for example, MON or MONDAY).

The date components of a natural date literal can be specified in any order, regardless of their order in the USAGE specification of the target field. Date components are separated by one or more blanks.

For example, if a USAGE specification for a date field is YM, a natural date literal written to that field can include the year and month in any order. MAY 1999 and 1990 APR are both valid literals.

Numeric Date Literals

Numeric date literals differ from natural date literals in that they are simple strings of digits. The order of the date components in a numeric date literal must match the order of the date components in the corresponding USAGE specification. In addition, the numeric date literal must include all of the date components included in the USAGE specification. For example, if the USAGE specification is DMY, then April 25 1999 must be represented as:

```
250499
```

Numeric date literals can be used in all date computations and all methods of data source updating.

Date Fields in Arithmetic Expressions

The general rule for manipulating date fields in arithmetic expressions is that date fields in the same expression must specify the same date components. The date components can be specified in any order, and display options are ignored. Y or YY, Q, M, W, and D are valid components.

Note that arithmetic expressions assigned to quarters, months, or days of the week are computed modulo 4, 12, and 7, respectively, so that anomalies like fifth quarters and thirteenth months are avoided.

For example, if NEWQUARTER and THISQUARTER both have USAGE specifications of Q, and the value of THISQUARTER is 2, then the following statement gives NEWQUARTER a value of 1 (that is, the remainder of 5 divided by 4):

```
NEWQUARTER = THISQUARTER + 3
```

Converting a Date Field

How to:

Convert a Date Field

Two types of conversion are possible: format conversion and date component conversion. In the first case, the value of a date format field can be assigned to an alphanumeric or integer field that uses date display options (see the following section). The reverse conversion is also possible.

In the second case, a field whose USAGE specifies one set of date components can be assigned to another field specifying different date components.

For example, the value of REPORTDATE (DMY) can be assigned to ORDERDATE (Y). In this case, the year is being extracted from REPORTDATE. If REPORTDATE is Apr 27 99, ORDERDATE is 99.

You can also assign the value of ORDERDATE to REPORTDATE. If the value of ORDERDATE is 99, the value of REPORTDATE is Jan 1 99. In this case, REPORTDATE is given values for the missing date components.

Syntax: How to Convert a Date Field

```
field1/format = field2;
```

where:

`field1`

Is a date format field, or an alphanumeric or integer format field using date display options.

`format`

Is the USAGE (or FORMAT) specification of *field1* (the target field).

`field2`

Is a date format field, or an alphanumeric or integer format field using date display options. The format types (alphanumeric, integer, or date) and the date components (YY, Y, Q, M, W, D) of *field1* and *field2* do not need to match.

How a Date Field Is Represented Internally

Date fields are represented internally as four-byte binary integers indicating the time elapsed since the date format base date. For each field, the unit of elapsed time is that field smallest date component.

For example, if the USAGE specification of REPORTDATE is MDY, then elapsed time is measured in days, and internally the field contains the number of days elapsed between the entered date and the base date. If you enter the numeric literal for February 13, 1964 (that is, 021364), and then print the field in a report, 02/13/64 appears. If you use it in the equation:

```
NEWDATE = 'FEB 28 1964' - REPORTDATE ;
DAYS/D = NEWDATE ;
```

then the value of DAYS is 15. However, the internal representation of REPORTDATE is a four-byte binary integer representing the number of days between December 31, 1900 and February 13, 1964.

Just as the unit of elapsed time is based on a field smallest date component, so too is the base date. For example, for a YQ field, elapsed time is measured in quarters, and the base date is the first quarter of 1901. For a YM field, elapsed time is measured in months, and the base date is the first month of 1901.

To display blanks or the actual base date in a report, use the SET DATEDISPLAY command described in the manual. The default value, OFF, displays blanks when a date matches the base date. ON displays the actual base date value.

You do not need to be concerned with the date format internal representation, except to note that all dates set to the base date appear as blanks, and all date fields that are entered blank or as all zeroes are accepted during validation and interpreted as the base date. They appear as blanks, but are interpreted as the base date in date computations and expressions.

Displaying a Non-Standard Date Format

How to:

Invoke ALLOWCVTERR

By default, if a date field in a non-FOCUS data source contains an invalid date, a message appears and the entire record fails to appear in a report. For example, if a date field contains '980450' with an ACTUAL of A6 and a USAGE of YMD, the record containing that field does not appear. The SET ALLOWCVTERR command enables you to display the rest of the record that contains the incorrect date.

Note: The ALLOWCVTERR parameter is not supported for virtual fields.

Syntax: How to Invoke ALLOWCVTERR

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

where:

`ON`

Enables you to display a field containing an incorrect date.

`OFF`

Generates a diagnostic message if incorrect data is encountered, and does not display the record containing the bad data. OFF is the default value.

When a bad date is encountered, ALLOWCVTERR sets the value of the field to either MISSING or to the base date, depending on whether MISSING=ON.

The following chart shows the results of interaction between DATEDISPLAY and MISSING, assuming ALLOWCVTERR=ON and the presence of a bad date.

	MISSING=OFF	**MISSING=ON**
`DATEDISPLAY=ON`	Displays Base Date 19001231 or 1901/1	.
`DATEDISPLAY=OFF`	Displays Blanks	.

DATEDISPLAY affects only how the base date appears. See the manual for a description of DATEDISPLAY.

Date Format Support

Date format fields are used in special ways with the following facilities:

- **Dialogue Manager.** Amper variables can function as date fields if they are set to natural date literals. For example:

  ```
  -SET &NOW = 'APR 25 1960' ;
  -SET &LATER = '1990 25 APR' ;
  -SET &DELAY = &LATER - &NOW ;
  ```

  In this case, the value of &DELAY is the difference between the two dates, measured in days: 10,957.

- **Extract files.** Date fields in SAVB and unformatted HOLD files are stored as four-byte binary integers representing the difference between the field face value and the standard base date. Date fields in SAVE files and formatted HOLD files (for example, USAGE WP) are stored without any display options.

- **GRAPH.** Date fields are not supported as sort fields in ACROSS and BY phrases.

- **FML.** Date fields are not supported within the RECAP statement.

Alphanumeric and Numeric Formats With Date Display Options

In addition to the standard date format, you can also represent a date by using an alphanumeric, integer, or packed-decimal field with date display options (D, M, Y, and T). Note, however, that this does not offer the full date support that is provided by the standard date format.

Alphanumeric and integer fields used with date display options have some date functionality when used with special date functions, as described in the manual.

When representing dates as alphanumeric or integer fields with date display options, you can specify the year, month, and day. If all three of these elements are present, then the date has six digits (or eight if the year is presented as four digits), and the USAGE can be:

Format	**Display**
`I6MDY`	`04/21/98`
`I6YMD`	`98/04/21`
`P6DMY`	`21/04/98`
`I8DMYY`	`21/04/1998`

The number of a month (1 to 12) can be translated to the corresponding month name by adding the letter T to the format, immediately after the M. For instance:

Format	**Data**	**Display**
`I6MTDY`	`05/21/98`	`MAY 21 98`
`I4MTY`	`0698`	`JUN 98`
`I2MT`	`07`	`JUL`

If the date has only the month element, a format of I2MT displays the value 4 as APR, for example. This is particularly useful in reports where columns or rows are sorted by month. They then appear in correct calendar order. For example, JAN, FEB, MAR, because the sorting is based on the numerical, not alphabetical, values. (Note that without the T display option, I2M is interpreted as an integer with a floating dollar sign.)

Date-Time Formats

How to:

Enable ISO Standard Date-Time Notation

The date-time data type supports both the date and time, similar to the timestamp data types available in many relational data sources.

Date-time fields are stored in eight, ten, or 12 bytes: four bytes for date and either four, six, or eight bytes for time, depending on whether the format specifies a microsecond or nanosecond.

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. See the manual for information on subroutines for manipulating date-time fields.

Date-time formats can also produce output values and accept input values that are compatible with the ISO 8601:2000 date-time notation standard. A SET parameter and specific formatting options enable this notation.

Syntax: How to Enable ISO Standard Date-Time Notation

```
SET DTSTANDARD = {OFF|ON|STANDARD|STANDARDU}
```

where:

OFF

Does not provide compatibility with the ISO 8601:2000 date-time notation standard. OFF is the default value.

ON|STANDARD

Enables recognition and output of the ISO standard formats, including use of T as the delimiter between date and time, use of period or comma as the delimiter of fractional seconds, use of Z at the end of universal times, and acceptance of inputs with time zone information. STANDARD is a synonym for ON.

STANDARDU

Enables ISO standard formats (like STANDARD) and also, where possible, converts input strings to the equivalent universal time (formerly known as Greenwich Mean Time), thus enabling applications to store all date-time values in a consistent way.

Example: Using SET DTSTANDARD

The following request displays date-time values input in ISO 8601:2000 date-time standard formats. With SET DTSTANDARD=OFF, the request terminates with a (FOC177): INVALID DATE CONSTANT:

```
SET DTSTANDARD = &STAND
DEFINE FILE EMPLOYEE
-* The following input is format YYYY-MM-DDThh:mm:ss.sTZD
DT1/HYYMDs  =     DT(2004-06-01T19:20:30.45+01:00);
-* The following input has comma as the decimal separator
DT2/HYYMDs  =     DT(2004-06-01T19:20:30,45+01:00);
DT3/HYYMDs  =     DT(20040601T19:20:30,45);
DT4/HYYMDUs =     DT(2004-06-01T19:20:30,45+01:00);
END
TABLE FILE EMPLOYEE
HEADING CENTER
"DTSANDARD = &STAND "
" "
SUM CURR_SAL NOPRINT DT1 AS 'DT1: INPUT = 2004-06-01T19:20:30.45+01:00'
OVER DT2 AS 'DT2: INPUT = 2004-06-01T19:20:30,45+01:00'
OVER DT3 AS 'DT3: INPUT = 20040601T19:20:30,45'
OVER DT4 AS 'DT4: OUTPUT  FORMAT HYYMDUs'
END
```

With DTSTANDARD= STANDARD, the output shows that the input values were accepted, but the time zone offsets in DT1, DT2, and DT4 (+01:00) were ignored on output. The character U in the format for DT4 causes the T separator to be used between the date and the time:

```
                      DTSANDARD = STANDARD

DT1: INPUT = 2004-06-01T19:20:30.45+01:00  2004-06-01 19:20:30.450
DT2: INPUT = 2004-06-01T19:20:30,45+01:00  2004-06-01 19:20:30.450
DT3: INPUT = 20040601T19:20:30,45          2004-06-01 19:20:30.450
DT4: OUTPUT  FORMAT HYYMDUs                2004-06-01T19:20:30.450
```

With DTSTANDARD= STANDARDU, the output shows that the values DT1, DT2, and DT4 were converted to universal time by subtracting the time zone offsets (+01:00):

```
                      DTSANDARD = STANDARDU

DT1: INPUT = 2004-06-01T19:20:30.45+01:00  2004-06-01 18:20:30.450
DT2: INPUT = 2004-06-01T19:20:30,45+01:00  2004-06-01 18:20:30.450
DT3: INPUT = 20040601T19:20:30,45          2004-06-01 19:20:30.450
DT4: OUTPUT  FORMAT HYYMDUs                2004-06-01T18:20:30.450
```

Describing a Date-Time Field

How to:

Describe a Numeric Date-Time Value Without Display Options

Describe a Time-Only Value

Describe a Date-Time Value

Reference:

Display Options for a Time-Only Value

Display Options for the Date Component of a Date-Time Field

Display Options for the Time Component of a Date-Time Field

Date-Time Usage Notes

In a Master File, the USAGE (or FORMAT) attribute determines how date-time field values appear in report output and forms, and how they behave in expressions and functions. For FOCUS data sources, it also determines how they are stored.

Format type H describes date-time fields. The USAGE attribute for a date-time field contains the H format code and can identify either the length of the field or the relevant date-time display options.

The MISSING attribute for date-time fields can be ON or OFF. If it is OFF, and the date-time field has no value, it defaults to blank.

Syntax: How to Describe a Numeric Date-Time Value Without Display Options

This format is appropriate for alphanumeric HOLD files or transaction files.

```
USAGE = Hn
```

where:

n

Is the field length, from 1 to 23, including up to eight characters for displaying the date and up to nine, 12, or 15 characters for the time. For lengths less than 20, the date is truncated on the right.

An eight-character date includes four digits for the year, two for the month, and two for the day of the month, YYYYMMDD.

A nine-character time includes two digits for the hour, two for the minute, two for the second, and three for the millisecond, HHMMSSsss. The millisecond component represents the decimal portion of the second to three places.

A twelve-character time includes two digits for the hour, two for the minute, two for the second, three for the millisecond, and three for the microsecond, HHMMSSsssmmm. The millisecond component represents the decimal portion of the second value to three places. The microsecond component represents three additional decimal places beyond the millisecond value.

A fifteen-character time includes two digits for the hour, two for the minute, two for the second, three for the millisecond, three for the microsecond and three for the nanosecond, HHMMSSsssmmmnnn. The millisecond component represents the decimal portion of the second value to three places. The microsecond component represents three additional decimal places beyond the millisecond value. The nanosecond component represents three additional decimal places beyond the microsecond value.

With this format, there are no spaces between the date and time components, no decimal points, and no spaces or separator characters within either component. The time must be entered using the 24-hour system. For example, the value 19991231225725333444 represents 1999/12/31 10:57:25.333444PM.

Syntax: How to Describe a Time-Only Value

```
USAGE = Htimefmt1
```

where:

timefmt1

Is the USAGE format for displaying time only. Hour, minute, and second components are always separated by colons (:), with no intervening blanks. A time value can have a blank immediately preceding an am/pm indicator. For information, see *Display Options for a Time-Only Value* on page 376.

Reference: Display Options for a Time-Only Value

The following table lists the valid time display options for a time-only USAGE attribute. Assume the time value is 2:05:27.123456444 a.m.

Option	Meaning	Effect
`H`	Hour (two digits). If the format includes the option a, b, A, or B, the hour value is from 01 to 12. Otherwise, the hour value is from 00 to 23, with 00 representing midnight.	Prints a two-digit hour. For example: `USAGE = HH` prints `02`

Option	**Meaning**	**Effect**
`h`	Hour with zero suppression. If the format includes the option a, b, A, or B, the hour value is from 1 to 12. Otherwise, the hour is from 0 to 23.	Displays the hour with zero suppression. For example: `USAGE = Hh` prints `2`
`I`	Minute (two digits). The minute value is from 00 to 59.	Prints the two-digit minute. For example: `USAGE = HHI` prints `02:05`
`i`	Minute with zero suppression. The minute value is from 0 to 59.	Prints the minute with zero suppression. This cannot be used together with an hour format (H or h). For example: `USAGE = Hi` prints `5`

`S[x]`	Second (two digits). You can specify up to nine decimal places after the seconds value using the *x* option, where *x* is a number from 1 to 9. Alternatively, you can use the s, m, and n formats to display three, six, or nine decimal places.
	S: 00 to 59
		Prints the two-digit second. For example:
		`USAGE = HHIS` prints `02:05:27`
		`USAGE = HHIS1` prints `02:05:27.1`

`s`	Millisecond (three digits, after the decimal point in the second). 000 to 999	Prints the second to three decimal places. For example: `USAGE = HHISs` prints `02:05:27.123`
`m`	Microsecond (three additional digits after the millisecond). 000 through 999	Prints the second to six decimal places. For example: `USAGE = HSsm` prints `27.123456`

Option	Meaning	Effect
`n`	Nanosecond (three additional digits after the microsecond). 000 through 999	Prints the second to nine decimal places. For example: `USAGE = HSsn` prints `27.123456444`
`A`	12-hour time display with AM or PM in uppercase.	Prints the hour from 01 to 12, followed by AM or PM. For example: `USAGE = HHISA` prints `02:05:27AM`
`a`	12-hour time display with am or pm in lowercase.	Prints the hour from 01 to 12, followed by am or pm. For example: `USAGE = HHISa` prints `02:05:27am`
`B`	12-hour time display with AM or PM in uppercase, with a blank space before the AM or PM.	Prints the hour from 01 to 12, followed by a space and then AM or PM. For example: `USAGE = HHISB` prints `02:05:27 AM`
`b`	12-hour time display with am or pm in lowercase, with a blank space before the am or pm.	Prints the hour from 01 to 12, followed by a space followed by am or pm. For example: `USAGE = HHISb` prints `02:05:27 am`
`Z`	24-hour time display with Z to indicate universal time. Z is incompatible with AM/PM output.	Prints the hour from 01 to 24, followed by Z. For example: `USAGE = HHISZ` prints `14:30[:20.99]Z`

When the format includes more than one time display option:

- The options must appear in the order hour, minute, second, millisecond, microsecond, nanosecond.
- The first option must be either hour, minute, or second.
- No intermediate component can be skipped. If hour is specified, the next option must be minute. It cannot be second.

Note: Unless you specify one of the AM/PM time display options, the time component appears using the 24-hour system.

Syntax: How to Describe a Date-Time Value

```
USAGE = Hdatefmt [separator] [timefmt2]
```

where:

datefmt

Is the USAGE format for displaying the date portion of the date-time field. For information, see *Display Options for the Date Component of a Date-Time Field* on page 379.

separator

Is a separator between the date components. The default separator is a slash (/). Other valid separators are: period (.), hyphen (-), blank (B), or none (N). With translated months, these separators can only be specified when the k option is not used.

With the STANDARD and STANDARDU settings, the separator for dates is always hyphen. The separator between date and time is blank by default. However, if you specify the character U as the separator option, the date and time will be separated by the character T.

timefmt2

Is the format for a time that follows a date. Time is separated from the date by a blank. Time components are separated from each other by colons. Unlike the format for time alone, a time format that follows a date format consists of at most two characters: a single character to represent all of the time components that appear and, optionally, one character for an AM/PM option. For information, see *Display Options for the Time Component of a Date-Time Field* on page 382.

Reference: Display Options for the Date Component of a Date-Time Field

The date format can include the following display options, as long as they conform to the allowed combinations. In the following table, assume the date is February 5, 1999.

Option	Meaning	Example
`Y`	2-digit year	`99`
`YY`	4-digit year	`1999`
`M`	2-digit month (01 - 12)	`02`
`MT`	Full month name	`February`
`Mt`	Short month name	`Feb`
`D`	2-digit day	`05`

Option	Meaning	Example
d	Zero-suppressed day. A blank space replaces the zero.	` 5`
e	Zero-removed day. The day number is shifted to the left, and any components to the right of this are shifted to the left. Requires a date separator.	`5`
o	Zero-removed month. Automatically implements the e option for a zero-removed day. The month and day numbers are shifted to the left, and any components to the right of these are also shifted. Required a date separator.	`5`

k	For formats in which month or day is followed by year, and month is translated to a short or full name, k separates the year from the day with a comma and blank. Otherwise, the separator is a blank.	`USAGE = HMtDkYY`
		
		`prints Feb 05, 1999`

Note: Unless you specify one of the AM/PM time display options, the time component uses the 24-hour system.

Example: Using Zero Removal for Date-Time Month and Day Numbers

The following request creates the date-time value 01/01/2013. It then displays this value using:

- Normal month and day numbers, format HMDYY.
- Month removal and day removal, format HoeYY.
- Month removal without day removal (which forces day removal), format HodYY.

- Day removal without month removal, format HMeYY. Note that month removal is not forced by day removal.

```
DEFINE FILE GGSALES
DATE1A/HMDYY = DT(01/01/2013);
DATE1B/HoeYY = DATE1A;
DATE1C/HodYY = DATE1A;
DATE1D/HMeYY = DATE1A;
END
TABLE FILE GGSALES
SUM DOLLARS NOPRINT
DATE1A AS 'HMDYY'
DATE1B AS 'HoeYY'
DATE1C AS 'HodYY'
DATE1D AS 'HMeYY'
ON TABLE SET PAGE NOPAGE
END
```

The output is:

```
HMDYY       HoeYY       HodYY       HMeYY
-----       -----       -----       -----
01/01/2013  1/1/2013    1/1/2013    01/1/2013
```

Example: Comparing Zero Suppression With Zero Removal

The following request creates two dates with date-time formats in which the date component has a leading zero (01). In the first date, the day component is the first component and displays on the left. In the second date, the day component is the second component and displays in the middle. The request prints these dates:

- With all zeros displayed, format HDMYY.
- With zero suppression for the day component, format HdMYY.

- With zero removal for the day component, format HeMYY.

```
DEFINE FILE GGSALES
DATE1A/HDMYY = DT(01/12/2012);
DATE2A/HMDYY = DT(12/01/2012);
DATE1B/HdMYY = DATE1A;
DATE2B/HMdYY = DATE2A;
DATE1C/HeMYY = DATE1A;
DATE2C/HMeYY = DATE2A;
END
TABLE FILE GGSALES
SUM DOLLARS NOPRINT
DATE1A AS 'HDMYY'
DATE2A AS ''      OVER
DATE1B AS 'HdMYY'
DATE2B AS ''      OVER
DATE1C AS 'HeMYY'
DATE2C AS ''
ON TABLE SET PAGE NOPAGE
```

On the output, the first row shows the date with all zeros displayed. The second row shows zero suppression of the day number, where the zero has been replaced by a blank space so that all the components are aligned with the components on row 1. The last row shows zero removal, where the zero has been removed from the day number, and all of the remaining characters have been shifted over to the left:

```
HDMYY  01/12/2012    12/01/2012
HdMYY   1/12/2012    12/ 1/2012
HeMYY  1/12/2012     12/1/2012
```

Reference: Display Options for the Time Component of a Date-Time Field

The following table lists the valid options. Assume the date is February 5, 1999 and the time is 02:05:25.444555333 a.m.

Option	Meaning	Example
`H`	Prints hour.	`USAGE = HYYMDH` prints `1999/02/05 02`
`I`	Prints hour:minute.	`USAGE = HYYMDI` prints `1999/02/05 02:05`

Option	Meaning	Example

`S[`*x*`]`	Prints hour:minute:second. If you specify the *x* option, where *x* is a number from 1 to 9, the seconds displays with *x* decimal digits. Alternatively, you can specify the s, m, or n option to display the seconds with three, six, or nine decimal digits.	`USAGE = HYYMDS` prints `1999/02/05 02:05:25`
		`USAGE = HYYMDS1` prints `1999/02/05 02:05:25.4`

`s`	Prints hour:minute:second.millisecond.	`USAGE = HYYMDs` prints `1999/02/05 02:05:25.444`
`m`	Prints hour:minute:second.microsecond.	`USAGE = HYYMDm` prints `1999/02/05 02:05:25.444555`
`n`	Prints hour:minute:second.nanosecond.	`USAGE = HYYMDn` prints `1999/02/05 02:05:25.444555333`
`A`	Prints AM or PM. This uses the 12-hour system and causes the hour to be printed with zero suppression.	`USAGE = HYYMDSA` prints `1999/02/05 2:05:25AM`
`a`	Prints am or pm. This uses the 12-hour system and causes the hour to be printed with zero suppression.	`USAGE = HYYMDSa` prints `1999/02/05 2:05:25am`
`B`	Prints AM or PM preceded by a blank space. This uses the 12-hour system and causes the hour to be printed with zero suppression.	`USAGE = HYYMDSB` prints `1999/02/05 2:05:25 AM`
`b`	Prints am or pm preceded by a blank space. This uses the 12-hour system and causes the hour to be printed with zero suppression.	`USAGE = HYYMDSb` prints `1999/02/05 2:05:25 am`
`Z`	Prints Z to indicate universal time. This uses the 24-hour system. Z is incompatible with AM/PM output.	`USAGE = HHISZ` prints `14:30[:20.99]Z`

The date components can be in any of the following combinations and order:

- Year-first combinations: Y, YY, YM, YYM, YMD, YYMD.
- Month-first combinations: M, MD, MY, MYY, MDY, MDYY.
- Day-first combinations: D, DM, DMY, DMYY.

Reference: Date-Time Usage Notes

- In order to have a time component, you must have a day component.
- If you use the k option, you cannot change the date separator.

Character Format A*n*V

The character format A*n*V is supported in Master Files for FOCUS, XFOCUS, and relational data sources. This format is used to represent the VARCHAR (variable length character) data types supported by relational database management systems.

For relational data sources, A*n*V keeps track of the actual length of a VARCHAR column. This information is important when the value is used to populate a VARCHAR column in a different RDBMS. It affects whether trailing blanks are retained in string concatenation and, for Oracle, string comparisons (the other relational engines ignore trailing blanks in string comparisons).

In a FOCUS or XFOCUS data source, A*n*V does not provide true variable length character support. It is a fixed-length character field with two extra leading bytes to contain the actual length of the data stored in the field. This length is stored as a short integer value occupying two bytes. Trailing blanks entered as part of an AnV field count in its length.

Note: Because of the two bytes of overhead and the additional processing required to strip them, A*n*V format is not recommended for use in non-relational data sources.

Syntax: How to Specify A*n*V Fields in a Master File

```
FIELD=name, ALIAS=alias, USAGE=AnV [,ACTUAL=AnV] , $
```

where:

n

Is the size (maximum length) of the field. It can be from 1 to 4093. Note that because of the additional two bytes used to store the length, an A4093V field is actually 4095 bytes long. A size of zero (A0V) is not supported. The length of an instance of the field can be zero.

Note: HOLD FORMAT ALPHA creates an ACTUAL format of A*n*W in the Master File. See *Propagating an AnV Field to a HOLD File* on page 386.

Example: Specifying the A*n*V Format in a Master File

The following represents a VARCHAR field in a Master File for a DB2 data source with size 200:

```
$ VARCHAR FIELD USING AnV
  FIELD=VARCHAR200, ALIAS=VC200, USAGE=A200V, ACTUAL=A200V, MISSING=ON ,$
```

The following represents an A*n*V field in a Master File for a FOCUS data source with size 200:

```
FIELD=ALPHAV, ALIAS=AV200, USAGE=A200V, MISSING=ON ,$
```

If a data source has an A*n*V field, specify the following in order to create a HOLD FORMAT ALPHA file without the length designator:

```
FIELD=ALPHA, USAGE=A25, ACTUAL=A25V, $
```

or

```
DEFINE ...
  ALPHA/A25 = VARCHAR ;
END
```

or

```
COMPUTE ALPHA/A25 = VARCHAR ;
```

In order to alter or create a Master File to include A*n*V, the data must be converted and the length added to the beginning of the field. For example, issue a HOLD command when the field is described as follows:

```
FIELD=VARCHAR, ,USAGE=A25V, ACTUAL=A25, $
```

or

```
DEFINE ...
  VARCHAR/A25V = ALPHA ;
END
```

or

```
COMPUTE VARCHAR/A25V = ALPHA ;
```

Reference: Usage Notes for A*n*V Format

- A*n*V can be used anywhere that A*n* can be used, except for the restrictions listed in these notes.
- Full FOCUS and SQL operations are supported with this data type, including CREATE FILE for relational data sources.
- Joins are not supported between A*n* and A*n*V fields.
- DBCS characters are supported. As with the A*n* format, the number of characters must fit within the 4K data area.
- COMPUTE and DEFINE generate the data type specified on the left-hand side.
- Conversion between A*n*V and TX fields is not supported.
- A*n*V fields cannot have date display options.

Reference: Propagating an A*n*V Field to a HOLD File

When a user propagates an A*n*V field to a sequential data source using the HOLD FORMAT ALPHA command, the two-byte integer length is converted to a six-digit alphanumeric length. The field in the HOLD file consists of this six-digit number followed by the character data. The format attributes for this field are:

```
... USAGE=AnV, ACTUAL=AnW
```

A*n*W is created as a by-product of HOLD FORMAT ALPHA. However, it can be read and used for input as necessary. The number of bytes occupied by this field in the HOLD file is 6+*n*.

Example: Propagating an A*n*V Field to a HOLD File

The A39V field named TITLEV, is propagated to the HOLD file as:

```
FIELDNAME = TITLEV ,E03 ,A39V ,A39W ,$
```

In a binary HOLD file, the USAGE and ACTUAL formats are A*n*V, although the ACTUAL format may be padded to a full 4-byte word. The number of bytes occupied by this field in the HOLD file is 2+*n*.

When an A*n*V field is input into a data source, all bytes in the input field beyond the given length are ignored. These bytes are set to blanks as part of the input process.

When a user creates a relational data source using the HOLD FORMAT *sqlengine* command, the A*n*V field generates a VARCHAR column in the relational data source.

For example, the A39V field named TITLEV, is propagated to a HOLD FORMAT DB2 file as:

```
FIELDNAME = 'TITLEV', 'TITLEV', A39V, A39V ,$
```

Text Field Format

How to:

Specify a Text Field in a Master File

You can store any combination of characters as a text field.

Syntax: How to Specify a Text Field in a Master File

```
FIELD = fieldname, ALIAS = aliasname, USAGE = TXn[F],$
```

where:

fieldname

Is the name you assign the text field.

aliasname

Is an alternate name for the field name.

n

Is the output display length in TABLE for the text field. The display length may be between 1 and 256 characters.

All letters, digits, and special characters can be stored with this format. The following are some sample text field formats.

Format	Display
`TX50`	`This course provides the DP professional with the skills needed to create, maintain, and report from FOCUS data sources.`
`TX35`	`This course provides the DP professional with the skills needed to create, maintain, and report from FOCUS data sources.`

The standard edit options are not available for the text field format.

Reference: Usage Notes for Text Field Format

- Conversion between text and alphanumeric fields is supported in DEFINE and COMPUTE commands.
- Multiple text fields are supported, and they and be anywhere in the segment.

The Stored Data Type: ACTUAL

In this section:

ACTUAL Attribute

ACTUAL describes the type and length of data as it is actually stored in the data source. While some data types, such as alphanumeric, are universal, others differ between different types of data sources. Some data sources support unique data types. For this reason, the values you can assign to the ACTUAL attribute differ for each type of data source.

ACTUAL Attribute

How to:

Specify the ACTUAL Attribute

Reference:

ACTUAL to USAGE Conversion

COBOL Picture to USAGE Format Conversion

This attribute describes the type and length of your data as it actually exists in the data source. The source of this information is your existing description of the data source (such as a COBOL FD statement). The ACTUAL attribute is one of the distinguishing characteristics of a Master File for non-FOCUS data sources. Since this attribute exists only to describe the format of a non-FOCUS data structure, it is not used in the Master File of a FOCUS data structure.

If your data source has a date stored as an alphanumeric field and you need to convert it to a date for sorting or aggregation in a report, you can use the DATEPATTERN attribute in the Master File. then uses the pattern specified to convert the alphanumeric date to a date.

Syntax: How to Specify the ACTUAL Attribute

```
ACTUAL = format
```

where:

format

Consists of values taken from the following table, which shows the codes for the types of data that can be read.

ACTUAL Type	Meaning
DATE	Four-byte integer internal format, representing the difference between the date to be entered and the date format base date.

ACTUAL Type	Meaning
`A`*n*	Where *n* = 1-4095 for fixed-format sequential and VSAM data sources, and 1-256 for other non-FOCUS data sources. Alphanumeric characters A-Z, 0-9, and the special characters in the EBCDIC display mode. A*n* accepts all the date-time string formats, as well as the H*n* display formats. ACTUAL=A*n* also accepts a date-time field as it occurs in an alphanumeric HOLD file or SAVE file.
`D8`	Double-precision, floating-point numbers, stored internally in eight bytes.
`F4`	Single-precision, floating-point numbers, stored internally in four bytes.
`H`*n*	H8, H10, or H12 accepts a date-time field as it occurs in a binary HOLD file or SAVB file.

`I`*n*	Binary integers:
	
	I1 = single-byte binary integer.
	
	I2 = half-word binary integer (2 bytes).
	
	I4 = full-word binary integer (4 bytes).
	
	I8 = double-word binary integer (8 bytes).
	
	Note: The USAGE must be P or D. Decimals are honored, with proper conversion to the decimals of the P or D USAGE.

`P`*n*	Where *n* = 1-16. Packed decimal internal format. *n* is the number of bytes, each of which contains two digits, except for the last byte which contains a digit and the sign (+ or -). For example, P6 means 11 digits plus a sign.

ACTUAL Type	Meaning
Z*n*	Where *n* = 1-31. Zoned decimal internal format. *n* is the number of digits, each of which takes a byte of storage. The last digit contains a digit and the sign. If the field contains an assumed decimal point, represent the field with an ACTUAL format of Z*n* and a USAGE format of P*m*.*d*, where *m* is the total number of digits in the display plus the assumed decimal point, *d* is the number of decimal places, and *m* must be at least 1 greater than the value of *n*. For example, a field with ACTUAL=Z5 and one decimal place needs USAGE=P6.1 (or P7.1, or greater).

Note:

- Unless your data source is created by a program, all of the characters are either of type A (alphanumeric) or type Z (zoned decimal).
- ACTUAL formats supported for date-time values are A*n*, H8, H10, and H12. A*n* accepts all the date-time string formats as well as the H*n* USAGE display format. ACTUAL=H8, H10, or H12 accepts a date-time field as it occurs in a binary HOLD file or SAVB file. ACTUAL=A*n* accepts a date-time field as it occurs in an alphanumeric HOLD file or SAVE file.
- If you create a binary HOLD file from a data source with a date-time field, the ACTUAL format for that field is of the form H*n*. If you create an alphanumeric HOLD file from a data source with a date-time field, the ACTUAL format for that field is of the form A*n*.

Reference: ACTUAL to USAGE Conversion

The following conversions from ACTUAL format to USAGE (display) format are automatically handled and do not require invoking a function:

ACTUAL	USAGE
`A`	`A, D, F, I, P, date format, date-time format`
`D`	`D`
`DATE`	`date format`
`F`	`F`
`H`	`H`

ACTUAL	USAGE
I	I, date format
P	P, date format
Z	D, F, I, P

Reference: COBOL Picture to USAGE Format Conversion

The following table shows the USAGE and ACTUAL formats for COBOL, FORTRAN, PL1, and Assembler field descriptions.

COBOL USAGE FORMAT	BYTES OF COBOL PICTURE	INTERNAL STORAGE	ACTUAL FORMAT	USAGE FORMAT
DISPLAY DISPLAY DISPLAY DISPLAY	X(4) S99 9(5)V9 99	4 2 6 2	A4 Z2 Z6.1 A2	A4 P3 P8.1 A2

COMP
COMP
COMP*
COMP
COMP-1**
	S9
	S9(4)
	S9(5)
	S9(9)
	—
		4
		4
		4
		4
		4
			I2
			I2
			I4
			I4
			F4
				I1
				I4
				I5
				I9
				F6

COMP-2***	—	8	D8	D15
COMP-3 COMP-3 COMP-3	9 S9V99 9(4)V9(3)	8 8 8	P1 P2 P4	P1 P5.2 P8.3
FIXED BINARY(7) (COMP-4)	B or XL1	8	I4	I7

* Equivalent to INTEGER in FORTRAN, FIXED BINARY(31) in PL/1, and F in Assembler.

** Equivalent to REAL in FORTRAN, FLOAT(6) in PL/1, and E in Assembler.

*** Equivalent to DOUBLE PRECISION or REAL*8 in FORTRAN, FLOAT(16) in PL/1, and D in Assembler.

Note:

1. The USAGE lengths shown are minimum values. They may be larger if desired. Additional edit options may also be added.
2. In USAGE formats, an extra character position is required for the minus sign if negative values are expected.
3. PICTURE clauses are not permitted for internal floating-point items.
4. USAGE length should allow for maximum possible number of digits.
5. In USAGE formats, an extra character position is required for the decimal point.

Note that FOCUS data sources do not use the ACTUAL attribute, and instead rely upon the USAGE attribute to specify both how a field is stored and formatted.

Null or MISSING Values: MISSING

If a segment instance exists but no data has been entered into one of its fields, that field has no value. Some types of data sources represent this absence of data as a blank space ( ) or zero (0), but others explicitly indicate an absence of data with a null indicator or as a special null value. Null values (sometimes known as missing data) are significant in reporting applications, especially those that perform aggregating functions, such as averaging.

If your type of data source supports missing data, as do FOCUS data sources and most relational data sources, then you can use the optional MISSING attribute to enable null values to be entered into and read from a field. MISSING plays a role when you:

- **Create new segment instances.** If no value is supplied for a field for which MISSING has been turned ON in the Master File or in a DEFINE or COMPUTE definition, then the field is assigned a missing value.

- **Generate reports.** If a field with a null value is retrieved, the field value is not used in aggregating calculations, such as averaging and summing. If the report calls for the field value to display, a special character appears to indicate a missing value. The default character is a period (.), but you can change it to any character string you wish using the SET NODATA command or the SET HNODATA command for HOLD files, as described in the manual.

Syntax: How to Specify a Missing Value

```
MISSING = {ON|OFF}
```

where:

`ON`

Distinguishes a missing value from an intentionally entered blank or zero when creating new segment instances and reporting.

`OFF`

Does not distinguish between missing values and blank or zero values when creating new segment instances and reporting. OFF is the default value.

Reference: Usage Notes for MISSING

Note the following rules when using MISSING:

- **Alias.** MISSING does not have an alias.
- **Value.** It is recommended that you set the MISSING attribute to match the field predefined null characteristic (whether the characteristic is explicitly set when the data source is created, or set by default). For example, if a relational table column has been created with the ability to accept null data, describe the field with the MISSING attribute set to ON so that its null values are correctly interpreted.

  FOCUS data sources also support MISSING=ON, which assigns numeric fields the value -9998998 for NULL values and alphanumeric fields the value '.'.
- **Changes.** You can change the MISSING attribute at any time. Note that changing MISSING does not affect the actual stored data values that were entered using the old setting. However, it does affect how that data is interpreted. If null data is entered when MISSING is turned ON, and then MISSING is switched to OFF, the data originally entered as null is interpreted as blanks (for alphanumeric fields) or zeroes (for numeric fields).

Using a Missing Value

Consider the field values shown in the following four records:

		1	3

If you average these values without declaring the field with the MISSING attribute, a value of zero is automatically be supplied for the two blank records. Thus, the average of these four records is (0+0+1+3)/4, or 1. If you turn MISSING to ON, the two blank records are not used in the calculation, so the average is (1+3)/2, or 2.

Missing values in a unique segment are also automatically supplied with a zero, a blank, or a missing value depending on the MISSING attribute. What distinguishes missing values in unique segments from other values is that they are not stored. You do have to supply a MISSING attribute for fields in unique segments on which you want to perform counts or averages.

The manual contains a more thorough discussion of using null values (sometimes called missing data) in reports. It includes alternative ways of distinguishing these values in reports, such as using the WHERE phrase with MISSING selection operators, and creating virtual fields using the DEFINE FILE command with the SOME or ALL phrase.

Describing an FML Hierarchy

How to:

Specify a Hierarchy Between Fields in a Master File

Assign Descriptive Captions for Hierarchy Field Values

The Financial Modeling Language (FML) supports dynamic reporting against hierarchical data structures.

You can define the hierarchical relationships between fields in a Master File and automatically display these fields using FML. You can also provide descriptive captions to appear in reports in place of the specified hierarchy field values.

In the Master File, use the PROPERTY=PARENT_OF and REFERENCE=*hierarchyfld* attributes to define the hierarchical relationship between two fields.

The parent and child fields must have the same FORMAT or USAGE, and their relationship should be hierarchical. The formats of the parent and child fields must both be numeric or both alphanumeric.

Syntax: How to Specify a Hierarchy Between Fields in a Master File

```
FIELD=parentfield,...,PROPERTY=PARENT_OF, REFERENCE=[seg.]hierarchyfld, $
```

where:

`parentfield`

Is the parent field in the hierarchy.

`PROPERTY=PARENT_OF`

Identifies this field as the parent of the referenced field in a hierarchy.

These attributes can be specified on every field. Therefore, multiple hierarchies can be defined in one Master File. However, an individual field can have only one parent. If multiple fields have PARENT_OF attributes for the same hierarchy field, the first parent found by traversing the structure in top-down, left-to-right order is used as the parent.

`seg`

Is the segment location of the hierarchy field. Required if more than one segment has a field named *hierarchyfield*.

`hierarchyfld`

Is the child field in the hierarchy.

PARENT_OF is also allowed on a virtual field in the Master File:

```
DEFINE name/fmt=expression;,PROPERTY=PARENT_OF,REFERENCE=hierarchyfld,$
```

Syntax: How to Assign Descriptive Captions for Hierarchy Field Values

The following attributes specify a caption for a hierarchy field in a Master File

```
FIELD=captionfield,..., PROPERTY=CAPTION, REFERENCE=[seg.]hierarchyfld, $
```

where:

`captionfield`

Is the name of the field that contains the descriptive text for the hierarchy field. For example, if the employee ID is the hierarchy field, the last name may be the descriptive text that appears on the report in place of the ID.

`PROPERTY=CAPTION`

Signifies that this field contains a descriptive caption that appears in place of the hierarchy field values.

A caption can be specified for every field, but an individual field can have only one caption. If multiple fields have CAPTION attributes for the same hierarchy field, the first parent found by traversing the structure in top-down, left-to-right order is used as the caption.

seg

Is the segment location of the hierarchy field. Required if more than one segment has a field named *hierarchyfield*.

hierarchyfld

Is the hierarchy field.

CAPTION is also allowed on a virtual field in the Master File:

```
DEFINE name/format=expression;,PROPERTY=CAPTION,REFERENCE=hierarchyfld,$
```

Example: Defining a Hierarchy in a Master File

The 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:

```
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, $
```

Validating Data: ACCEPT

How to:

Validate Data

Reference:

Usage Notes for ACCEPT

ACCEPT is an optional attribute that you can use to validate data as it is entered into a field from a MODIFY procedure. The ACCEPT test is applied immediately after a FIXFORM is processed after which subsequent COMPUTE statements can manipulate the value. By including ACCEPT in a field declaration, you can define a list or range of acceptable field values. In relational terms, you are defining the domain.

Syntax: How to Validate Data

```
ACCEPT = list
ACCEPT = value1 TO value2
ACCEPT = FIND (field [AS name] IN file)
```

where:

list

Is a string of acceptable values. The syntax is:

```
value1 OR value2 OR value3...
```

For example, ACCEPT = RED OR WHITE OR BLUE. You can also use a blank as an item separator. If the list of acceptable values runs longer than one line, continue it on the next. The list is terminated by a comma.

value1 TO *value2*

Gives the range of acceptable values. For example, ACCEPT = 150 TO 1000.

FIND

Verifies the incoming data against the values in another indexed field. This option is available only for FOCUS data sources.

Any value in the ACCEPT that contains an embedded blank (for example, Great Britain) must be enclosed within single quotation marks.

If the ACCEPT attribute is included in a field declaration and the SET command parameter ACCBLN has a value of OFF, blank ( ) and zero (0) values are accepted only if they are explicitly coded into the ACCEPT. SET ACCBLN is described in the manual.

Example: Specifying a List With an Embedded Blank

```
ACCEPT = SPAIN OR ITALY OR FRANCE OR 'GREAT BRITAIN'
```

Reference: Usage Notes for ACCEPT

Note the following rules when using ACCEPT:

- **Alias.** ACCEPT does not have an alias.
- **Changes.** You can change the information in an ACCEPT attribute at any time.
- **Virtual fields.** You cannot use the ACCEPT attribute to validate virtual fields created with the DEFINE attribute.
- **HOLD files.** If you wish to propagate the ACCEPT attribute into the Master File of a HOLD file, use the SET HOLDATTR command. HOLD files are discussed in the manual.
- **ACCEPT** is used only in MODIFY procedures. It is useful for providing one central validation list to be used by several procedures. The FIND function is useful when the list of values is large or undergoes frequent change.

Alternative Report Column Titles: TITLE

How to:

Specify an Alternative Title

Reference:

Usage Notes for TITLE

When you generate a report, each column title in the report defaults to the name of the field that appears in that column. However, you can change the default column title by specifying the optional TITLE attribute for that field.

You can also specify a different column title within an individual report by using the AS phrase in that report request, as described in the manual.

Note that the TITLE attribute has no effect in a report if the field is used with a prefix operator, such as AVE. You can supply an alternative column title for fields used with prefix operators by using the AS phrase.

Master Files support TITLE attributes for multiple languages. For information, see *Multilingual Metadata* on page 402.

Syntax: How to Specify an Alternative Title

```
TITLE = 'text'
```

where:

text

Is any string of up to 512 characters. You can split the text across as many as five separate title lines by separating the lines with a comma (,). Include blanks at the end of a column title by including a slash (/) in the final blank position. You must enclose the string within single quotation marks if it includes commas or leading blanks.

Example: Replacing the Default Column Title

The following FIELD declaration:

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

replaces the default column heading, LNAME, with the following:

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

Reference: Usage Notes for TITLE

Note the following rules when using TITLE:

- **Alias.** TITLE does not have an alias.
- **Changes.** You can change the information in TITLE at any time. You can also override the TITLE with an AS name in a request, or turn it off with the SET TITLES=OFF command.
- **Virtual fields.** If you use the TITLE attribute for a virtual field created with the DEFINE attribute, the semicolon (;) terminating the DEFINE expression must be on the same line as the TITLE keyword.
- **HOLD files.** To propagate the TITLE attribute into the Master File of a HOLD file, use the SET HOLDATTR command. HOLD files are discussed in the manual.

Documenting the Field: DESCRIPTION

How to:

Supply Field Documentation

Reference:

Usage Notes for DESCRIPTION

DESCRIPTION is an optional attribute that enables you to provide comments and other documentation for a field within the Master File. You can include any comment up to 2K (2048) characters in length.

Note that you can also add documentation to a field declaration, or to a segment or file declaration, by typing a comment in the columns following the terminating dollar sign. You can even create an entire comment line by inserting a new line following a declaration and placing a dollar sign at the beginning of the line.

The DESCRIPTION attribute for a FOCUS data source can be changed at any time without rebuilding the data source.

Master Files support description attributes for multiple languages. For information, see *Multilingual Metadata* on page 402.

Syntax: How to Supply Field Documentation

```
DESC[RIPTION] = text
```

where:

`DESCRIPTION`

Can be shortened to DESC. Abbreviating the keyword has no effect on its function.

`text`

Is any string of up to 2K (2048) characters. If it contains a comma, the string must be enclosed within single quotation marks.

Example: Specifying a DESCRIPTION

The following FIELD declaration provides a DESCRIPTION:

```
FIELD=UNITS,ALIAS=QTY,USAGE=I6, DESC='QUANTITY SOLD, NOT RETURNED',$
```

Reference: Usage Notes for DESCRIPTION

Note the following rules when using the DESCRIPTION attribute:

- **Alias.** The DESCRIPTION attribute has an alias of DEFINITION.
- **Changes.** You can change DESCRIPTION at any time.
- **Virtual fields.** You can use the DESCRIPTION attribute for a virtual field created with the DEFINE attribute.

Multilingual Metadata

Master Files support column headings and descriptions in multiple languages. The heading or description used depends on the value of the LANG parameter and whether a TITLE_*ln* or DESC_*ln* attribute is specified in the Master File, where *ln* identifies the language to which the column heading or description applies.

In a Master File, column headings are taken from:

1. A heading specified in the report request using the AS phrase.
2. A TITLE attribute in the Master File, if no AS phrase is specified in the request and SET TITLES=ON.
3. The field name specified in the Master File, if no AS phrase or TITLE attribute is specified, or if SET TITLES=OFF.

Syntax: How to Specify Multilingual Metadata in a Master File

```
FIELDNAME = field, ...
   .
   .
   .
TITLE= default_column_headingTITLE_ln = column_heading_for_ln   .
   .
   .
DESC= default_descDESC_ln = desc_for_ln   .
   .
   .
```

where:

field

Is a field in the Master File.

default_column_heading

Is the column heading to use when SET TITLES=ON and either the LANG parameter is set to the default language for the server, or another language is set but the Master File has no corresponding TITLE_*ln* attribute for that field. This column heading is also used if an the *ln* value is invalid.

default_desc

Is the description to use when either the LANG parameter is set to the default language for the server, or another language is set but the Master File has no corresponding DESC_*ln* attribute for that field. This description is also used if an the *ln* value is invalid.

TITLE_*ln* = *column_heading_for_ln*

Specifies the language for which the column heading applies and the text of the column heading in that language. That column heading is used when SET TITLES=ON, the LANG parameter is set to a non-default language for the server, and the Master File has a corresponding TITLE_*ln* attribute, where *ln* is the two-digit code for the language specified by the LANG parameter Valid values for *ln* are the two-letter ISO 639 language code abbreviations. For information, see *Languages and Language Code Abbreviations* on page 404.

DESC_*ln* = *desc_for_ln*

Specifies the language for which the description applies and the description text in that language. This description is used when the LANG parameter is set to a non-default language for the server and the Master File has a corresponding DESC_*ln* attribute. Valid values for *ln* are the two-letter ISO 639 language code abbreviations.

Reference: Languages and Language Code Abbreviations

Language Name	Two-Letter Language Code	Three-Letter Language Abbreviation
Arabic	ar	ARB
Baltic	lt	BAL
Chinese - Simplified GB	zh	PRC
Chinese - Traditional Big-5	tw	ROC
Czech	cs	CZE
Danish	da	DAN
Dutch	nl	DUT
English - American	en	AME or ENG
English - UK	uk	UKE
Finnish	fi	FIN
French - Canadian	fc	FRE
French - Standard	fr	FRE
German - Austrian	at	GER
German - Standard	de	GER
Greek	el	GRE
Hebrew	iw	HEW
Italian	it	ITA
Japanese - Shift-JIS(cp942) on ascii cp939 on EBCDIC	ja	JPN
Japanese - EUC(cp10942) on ascii (UNIX)	je	JPE
Korean	ko	KOR

Language Name	Two-Letter Language Code	Three-Letter Language Abbreviation
Norwegian	no	NOR
Polish	pl	POL
Portuguese - Brazilian	br	POR
Portuguese - Portugal	pt	POR
Russian	ru	RUS
Spanish	es	SPA
Swedish	sv	SWE
Thai	th	THA
Turkish	tr	TUR

Syntax: How to Activate the Use of a Language

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

```
SET LANG = lng
```

or

```
SET LANG = ln
```

where:

lng

Is the three-letter abbreviation for the language.

ln

Is the two-letter ISO language code.

Note: If SET LANG is used in a procedure, its value will override the values set in nlscfg.err or in any profile.

Reference: Activating a Language in the NLS Configuration File

In the configuration file, issue the following command:

```
LANG = lng
```

Reference: Usage Notes for Multilingual Metadata

- To generate the correct characters, all languages used must be on the code page specified at startup.
- Master Files should be stored using the code page.
- Multilingual descriptions are supported with all fields described in the Master File, including DEFINE and COMPUTE fields.
- If you issue a HOLD command with SET HOLDATTR=ON, only one TITLE attribute is propagated to the HOLD Master File. Its value is the column heading that would have appeared on the report output.

Example: Using Multilingual Descriptions in a Master File

The following Master File for the CENTINV data source specifies French descriptions (DESC_FR) and Spanish descriptions (DESC_ES) as well as default descriptions (DESC) for the PROD_NUM and PRODNAME fields:

```
FILE=CENTINV, SUFFIX=FOC, FDFC=19, FYRT=00
 SEGNAME=INVINFO, SEGTYPE=S1, $
  FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4, INDEX=I,
   DESCRIPTION='Product Number'
   DESC='Product Number',
   DESC_ES='Numero de Producto',
   DESC_FR='Nombre de Produit', $
  FIELD=PRODNAME, ALIAS=PNAME, FORMAT=A30,
   WITHIN=PRODCAT,
   DESCRIPTION='Product Name'
   DESC_FR='Nom de Produit',
   DESC_ES='Nombre de Producto', $
  FIELD=QTY_IN_STOCK, ALIAS=QIS, FORMAT=I7,
   DESCRIPTION='Quantity In Stock', $
  FIELD=PRICE, ALIAS=RETAIL, FORMAT=D10.2,
   TITLE='Price:',
   DESCRIPTION=Price, $
```

Example: Using Multilingual Titles in a Request

The following Master File for the CENTINV data source specifies French titles (TITLE_FR) and Spanish titles (TITLE_ES) as well as default titles (TITLE) for the PROD_NUM and PRODNAME fields:

```
FILE=CENTINV, SUFFIX=FOC, FDFC=19, FYRT=00
 SEGNAME=INVINFO, SEGTYPE=S1, $
  FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4, INDEX=I,
   TITLE='Product,Number:',
   TITLE_FR='Nombre,de Produit:',
   TITLE_ES='Numero,de Producto:',
   DESCRIPTION='Product Number', $
  FIELD=PRODNAME, ALIAS=PNAME, FORMAT=A30,
   WITHIN=PRODCAT,
   TITLE='Product,Name:',
   TITLE_FR='Nom,de Produit:',
   TITLE_ES='Nombre,de Producto:'
   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, $
```

The default language is English and, by default, SET TITLES=ON. Therefore, the following request, uses the TITLE attributes to produce column headings that are all in English:

```
TABLE FILE CENTINV
PRINT PROD_NUM PRODNAME PRICE
WHERE PRICE LT 200
END
```

The output is:

```
Product  Product
Number:  Name:                                 Price:
-------  -------                               ------
1004     2 Hd VCR LCD Menu                     179.00
1008     DVD Upgrade Unit for Cent. VCR        199.00
1026     AR3 35MM Camera 10 X                  129.00
1028     AR2 35MM Camera 8 X                   109.00
1030     QX Portable CD Player                 169.00
1032     R5 Micro Digital Tape Recorder         89.00
```

Now, issue the following command to set the language to Spanish and run the same request:

```
SET LANG = SPA
```

The output now displays column headings from the TITLE_ES attributes where they exist (Product Number and Product Name). Where no Spanish title is specified (the Price field), the column heading in the TITLE attribute appears:

```
Numero         Nombre
de Producto:   de Producto:                       Price:
------------   ------------                       ------
1004           2 Hd VCR LCD Menu                  179.00
1008           DVD Upgrade Unit for Cent. VCR     199.00
1026           AR3 35MM Camera 10 X               129.00
1028           AR2 35MM Camera 8 X                109.00
1030           QX Portable CD Player              169.00
1032           R5 Micro Digital Tape Recorder      89.00
```

Describing a Virtual Field: DEFINE

In this section:

Using a Virtual Field

How to:

Define a Virtual Field

Reference:

Usage Notes for Virtual Fields in a Master File

DEFINE is an optional attribute used to create a virtual field for reporting. You can derive the virtual field value from information already in the data source (that is, from permanent fields). Some common uses of virtual data fields include:

- Computing new numerical values that are not on the data record.
- Computing a new string of alphanumeric characters from other strings.
- Classifying data values into ranges or groups.
- Invoking subroutines in calculations.

Virtual fields are available whenever the data source is used for reporting.

Syntax: How to Define a Virtual Field

```
DEFINE fieldname/format [REDEFINES field2] = expression;
  [,TITLE='title',]
  [TITLE_ln='titleln', ... ,]
  [,DESC[CRIPTION]='desc',]
  [DESC_ln='descln', ... ,]$
```

where:

`fieldname`

Is the name of the virtual field. The name is subject to the same conventions as names assigned using the FIELDNAME attribute. FIELDNAME is described in *The Field Name: FIELDNAME* on page 337.

`format`

Is the field format. It is specified in the same way as formats assigned using the USAGE attribute, which is described in *The Displayed Data Type: USAGE* on page 347. If you do not specify a format, it defaults to D12.2.

`field2`

Enables you to redefine or recompute a field whose name exists in more than one segment.

`expression`

Is a valid expression. The expression must end with a semicolon (;). Expressions are fully described in the manual.

Note that when an IF-THEN phrase is used in the expression of a virtual field, it must include the ELSE phrase.

`TITLE='title'`

Is a column title for the virtual field in the default language.

`TITLE_ln='titleln'`

Is a column title for the virtual field in the language specified by the language code *ln*.

`DESC[CRIPTION]='desc'`

Is a description for the virtual field in the default language.

`DESC_ln='descln'`

Is a description for the virtual field in the language specified by the language code *ln*.

Place each DEFINE attribute after all of the field descriptions for that segment.

Example: Defining a Field

The following shows how to define a field called PROFIT in the segment CARS:

```
SEGMENT = CARS ,SEGTYPE = S1 ,PARENT = CARREC, $
   FIELDNAME = DEALER_COST ,ALIAS = DCOST ,USAGE = D7, $
   FIELDNAME = RETAIL_COST ,ALIAS = RCOST ,USAGE = D7, $
   DEFINE PROFIT/D7 = RETAIL_COST - DEALER_COST; $
```

Reference: Usage Notes for Virtual Fields in a Master File

Note the following rules when using DEFINE:

- **Alias.** DEFINE does not have an alias.
- **Changes.** You can change the virtual field declaration at any time.
- A DEFINE FILE command takes precedence over a DEFINE in the Master with same name.
- If the expression used to derive the virtual field invokes a function, parameter numbers and types are not checked unless the USERFCHK parameter is set to FULL.

Using a Virtual Field

A DEFINE attribute cannot contain qualified field names on the left-hand side of the expression. Use the WITH phrase on the left-hand side to place the defined field in the same segment as any real field you choose. This will determine when the DEFINE expression will be evaluated.

Expressions on the right-hand side of the DEFINE can refer to fields from any segment in the same path. The expression on the right-hand side of a DEFINE statement in a Master File can contain qualified field names.

A DEFINE attribute in a Master File can refer to only fields in its own path. If you want to create a virtual field that derives its value from fields in several different paths, you have to create it with a DEFINE FILE command using an alternate view prior to a report request, as discussed in the manual. The DEFINE FILE command is also helpful when you wish to create a virtual field that is only used once, and you do not want to add a declaration for it to the Master File.

Virtual fields defined in the Master File are available whenever the data source is used, and are treated like other stored fields. Thus, a field defined in the Master File cannot be cleared in your report request.

A virtual field cannot be used for cross-referencing in a join. It can, however, be used as a host field in a join.

Note: Maintain does not support DEFINE attributes that have a constant value. Using such a field in a Maintain procedure generates the following message:

```
(FOC03605) name is not recognized.
```

Describing a Calculated Value: COMPUTE

How to:

Include a COMPUTE Command in a Master File

Reference:

Usage Notes for COMPUTE in a Master File

COMPUTE commands can be included in Master Files and referenced in subsequent TABLE requests, enabling you to build expressions once and use them in multiple requests.

Syntax: How to Include a COMPUTE Command in a Master File

```
COMPUTE fieldname/fmt=expression;
  [,TITLE='title',]
  [TITLE_ln='titleln', ... ,]
  [,DESC[CRIPTION]='desc',]
  [DESC_ln='descln', ... ,]$
```

where:

`fieldname`

Is name of the calculated field.

`fmt`

Is the format and length of the calculated field.

`expression`

Is the formula for calculating the value of the field.

`TITLE='title'`

Is a column title for the calculated field in the default language.

`TITLE_ln='titleln'`

Is a column title for the calculated field in the language specified by the language code *ln*.

`DESC[CRIPTION]='desc'`

Is a description for the calculated field in the default language.

`DESC_`*`ln`*`='`*`descln`*`'`

Is a description for the calculated field in the language specified by the language code *ln*.

Reference: Usage Notes for COMPUTE in a Master File

In all instances, COMPUTEs in the Master File have the same functionality and limitations as temporary COMPUTEs. Specifically, fields computed in the Master File must follow these rules:

- They cannot be used in JOIN, DEFINE, or ACROSS phrases, or with prefix operators.
- When used as selection criteria, syntax is either IF TOTAL field or WHERE TOTAL field.
- When used as sort fields, syntax is BY TOTAL COMPUTE field.
- To insert a calculated value into a heading or footing, you must reference it prior to the HEADING or FOOTING command.

Note: Maintain does not currently support using COMPUTEs in Master Files, and these COMPUTEs do not appear in the Update Assist Wizard.

Example: Coding a COMPUTE in the Master File and Accessing the Computed Value

Use standard COMPUTE syntax to add a calculated value to your Master File. You can then access the calculated value by referencing the computed fieldname in subsequent TABLE requests. When used as a verb object, as in the following example, the syntax is SUM (or PRINT) COMPUTE field.

The following is the SALESTES Master File (the SALES FILE modified with an embedded COMPUTE):

```
FILENAME=SALESTES, 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, $

   COMPUTE REVENUE/D12.2M=UNIT_SOLD*RETAIL_PRICE;
```

The following TABLE request uses the REVENUE field:

```
TABLE FILE SALESTES
HEADING CENTER
"NEW YORK PROFIT REPORT"
" "
SUM UNIT_SOLD AS 'UNITS,SOLD' RETAIL_PRICE AS 'RETAIL_PRICE'
COMPUTE REVENUE;
BY PROD_CODE AS 'PROD,CODE'
WHERE CITY EQ 'NEW YORK'
END
```

The output is:

```
           NEW YORK PROFIT REPORT

  PROD  UNITS
  CODE  SOLD   RETAIL_PRICE          REVENUE
  ----  ----   ------------          -------
  B10     30            $.85          $25.50
  B17     20           $1.89          $37.80
  B20     15           $1.99          $29.85
  C17     12           $2.09          $25.08
  D12     20           $2.09          $41.80
  E1      30            $.89          $26.70
  E3      35           $1.09          $38.15
```

Describing a Filter: FILTER

How to:

Declare a Filter in a Master File

Use a Master File Filter in a Request

Reference:

Usage Notes for Filters in a Master File

Boolean virtual fields (DEFINE fields that evaluate to TRUE or FALSE) can be used as record selection criteria. If the primary purpose of a virtual field is for use in record selection, you can clarify this purpose and organize virtual fields in the Master File by storing the expression using a FILTER declaration rather than a DEFINE. Filters offer the following features:

- They allow you to organize and store popular selection criteria in a Master File and reuse them in multiple requests and tools.
- For some data sources (such as VSAM and ISAM), certain filter expressions can be inserted inline into the WHERE or IF clause, enhancing optimization compared to a Boolean DEFINE.

Syntax: How to Declare a Filter in a Master File

```
FILTER  filtername = expression;
  [, DESC[CRIPTION]='desc',]
  [DESC_ln='descln', ... ,]$
```

where:

filtername

Is the name assigned to the filter. The filter is internally assigned a format of I1, which cannot be changed.

expression

Is a logical expression that evaluates to TRUE (which assigns the value 1 to the filter field) or FALSE (which assigns the value 0 to the filter field). For any other type of expression, the field becomes a standard numeric virtual field in the Master File. Dialogue Manager variables (amper variables) can be used in the filter expression in same way they are used in standard Master File DEFINEs.

DESC[CRIPTION]='*desc*'

Is a description for the sort object in the default language.

`DESC_ln='descln'`

Is a description for the sort object in the language specified by the language code *ln*.

Syntax: How to Use a Master File Filter in a Request

```
TABLE FILE filename   .
   .
   .
{WHERE|IF} expression_using_filters
```

where:

`expression_using_filters`

Is a logical expression that references a filter. In a WHERE phrase, the logical expression can reference one or more filters and/or virtual fields.

Reference: Usage Notes for Filters in a Master File

- ❑ The filter field name is internally assigned a format of I1 which cannot be changed.
- ❑ A filter can be used as a standard numeric virtual field anywhere in a report request, except that they are not supported in WHERE TOTAL tests.

Example: Defining and Using a Master File Filter

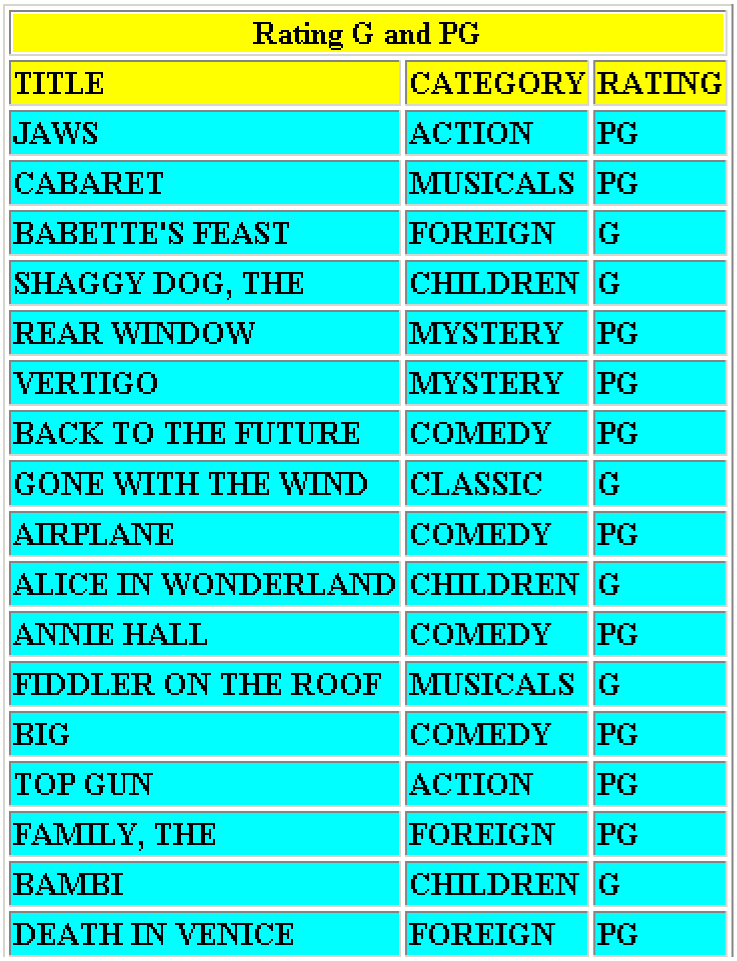
Consider the following filter declaration added to the MOVIES Master File:

```
FILTER G_RATING = RATING EQ 'G' OR 'PG'; $
```

The following request applies the G_RATING filter:

```
TABLE FILE MOVIES
HEADING CENTER
"Rating G and PG"
PRINT TITLE CATEGORY RATING
WHERE G_RATING
ON TABLE SET PAGE NOPAGE
ON TABLE SET GRID OFFON TABLE SET STYLE *
type=report, style=bold, color=black, backcolor=yellow, $
type=data, backcolor=aqua, $
ENDSTYLE
END
```

The output is:



Rating G and PG		
TITLE	CATEGORY	RATING
JAWS	ACTION	PG
CABARET	MUSICALS	PG
BABETTE'S FEAST	FOREIGN	G
SHAGGY DOG, THE	CHILDREN	G
REAR WINDOW	MYSTERY	PG
VERTIGO	MYSTERY	PG
BACK TO THE FUTURE	COMEDY	PG
GONE WITH THE WIND	CLASSIC	G
AIRPLANE	COMEDY	PG
ALICE IN WONDERLAND	CHILDREN	G
ANNIE HALL	COMEDY	PG
FIDDLER ON THE ROOF	MUSICALS	G
BIG	COMEDY	PG
TOP GUN	ACTION	PG
FAMILY, THE	FOREIGN	PG
BAMBI	CHILDREN	G
DEATH IN VENICE	FOREIGN	PG

Describing a Sort Object: SORTOBJ

How to:

Declare a Sort Object in a Master File

Reference a Sort Object in a Request

Reference:

Usage Notes for Sort Objects in a Master File

You can define sort phrases and attributes in a Master File and reference them by name in a request against the Master File. The entire text of the sort object is substituted at the point in the TABLE where the sort object is referenced. The sort phrases in the sort object are not verified prior to this substitution. The only verification is that there is a sort object name and an equal sign in the Master File SORTOBJ record.

Reference: Usage Notes for Sort Objects in a Master File

- The sort object declaration can appear anywhere after the first SEGNAME/SEGMENT record. However, it must appear after all fields mentioned by it in the Master File, including virtual fields.
- A sort object can use both Master File and local virtual fields.
- Unlimited sort object declarations may appear in a Master File, but the number referenced by a TABLE request cannot result in more than the maximum number of sort phrases in the request.
- The sort object declaration can be followed by optional attributes.
- If a sort object has the same name as a field, the sort object will be used when referenced in a request.

Syntax: How to Declare a Sort Object in a Master File

```
FILE= ...
SEG= ...
FIELD= ...
SORTOBJ sortname = {BY|ACROSS} sortfield1 [attributes]
  [{BY|ACROSS} sortfield2 ... ];
  [,DESC[CRIPTION]='desc',]
  [DESC_ln='descln', ... ,]$
```

where:

`sortname`

Is a name for the sort object.

`sortfield1, sortfield2 ..`

Are fields from the Master File or local DEFINE fields that will be used to sort the report output.

`attributes`

Are any valid sort attributes.

`;`

Is required syntax for delimiting the end of the sort object expression.

`DESC[CRIPTION]='`*`desc`*`'`

Is a description for the sort object in the default language.

`DESC_`*`ln`*`='`*`descln`*`'`

Is a description for the sort object in the language specified by the language code *ln*.

Syntax: How to Reference a Sort Object in a Request

```
TABLE FILE ...
   .
   .
   .
BY sortname   .
   .
   .
END
```

where:

`sortname`

Is the sort object to be inserted into the request.

Example: Declaring and Referencing a Sort Object

The following sort object for the GGSALES Master File is named CRSORT. It defines two sort phrases:

- BY the REGION field, with a SKIP-LINE attribute.
- ACROSS the CATEGORY field.

```
SORTOBJ CRSORT = ACROSS CATEGORY BY REGION SKIP-LINE ; ,$
```

The following request references the CRSORT sort object:

```
TABLE FILE GGSALES
SUM DOLLARS
BY CRSORT
ON TABLE SET PAGE NOPAGE
END
```

The output is:

```
                      Category
Region       Coffee        Food          Gifts
------------------------------------------------
Midwest      4178513       4404483       2931349
Northeast    4201057       4445197       2848289
Southeast    4435134       4308731       3037420
West         4493483       4204333       2977092
```

Using Date System Amper Variables in Master File DEFINEs

Reference:

Messages for Date System Amper Variables in Master File DEFINEs

Master File DEFINE fields can use Dialogue Manager system date variables to capture the system date each time the Master File is parsed for use in a request.

The format of the returned value for each date variable is the format indicated in the variable name. For example, &DATEYYMD returns a date value with format YYMD. The exceptions are &DATE and &TOD, which return alphanumeric values and must be assigned to a field with an alphanumeric format. The variable names &DATE and &TOD must also be enclosed in single quotation marks in the DEFINE expression.

The variables supported for use in Master File DEFINEs are:

- &DATE
- &TOD
- &DATEMDY
- &DATEDMY
- &DATEYMD
- &DATEMDYY
- &DATEDMYY
- &DATEYYMD

- &DMY
- &YMD
- &MDY
- &YYMD
- &MDYY
- &DMYY

Note that all other reserved amper variables are not supported in Master Files.

Example: Using the Date Variable &DATE in a Master File DEFINE

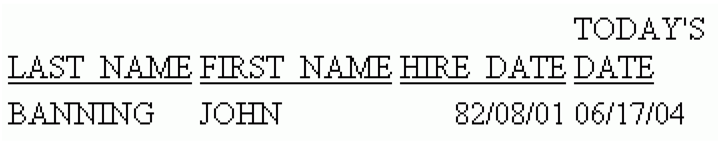
The following version of the EMPLOYEE Master File has the DEFINE field named TDATE added to it. TDATE has format A12 and retrieves the value of &DATE, which returns an alphanumeric value and must be enclosed in single quotation marks:

```
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,     $
DEFINE TDATE/A12   ='&DATE';, $
   .
   .
   .
```

The following request displays the value of TDATE:

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME HIRE_DATE TDATE AS 'TODAY''S,DATE'
WHERE LAST_NAME EQ 'BANNING'
END
```

The output is:



```
                                       TODAY'S
LAST_NAME  FIRST_NAME  HIRE_DATE  DATE
BANNING    JOHN         82/08/01  06/17/04
```

Example: Using the Date Variable &YYMD in a Master File DEFINE

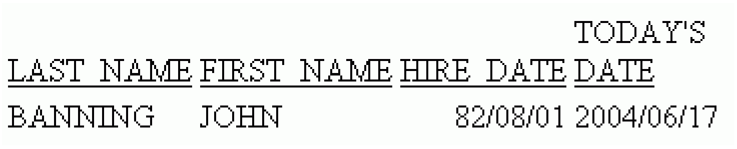
The following version of the EMPLOYEE Master File has the DEFINE field named TDATE added to it. TDATE has format YYMD and retrieves the value of &YYMD:

```
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,     $
DEFINE TDATE/YYMD   = &YYMD ;, $
   .
   .
   .
```

The following request displays the value of TDATE:

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME HIRE_DATE TDATE AS 'TODAY''S,DATE'
WHERE LAST_NAME EQ 'BANNING'
END
```

The output is:



```
                                        TODAY'S
LAST_NAME  FIRST_NAME  HIRE_DATE  DATE

BANNING    JOHN         82/08/01  2004/06/17
```

Reference: Messages for Date System Amper Variables in Master File DEFINEs

The following message appears if an attempt is made to use an unsupported amper variable in a Master File DEFINE:

`(FOC104) DEFINE IN MASTER REFERS TO A FIELD OUTSIDE ITS SCOPE:` *`var`*

Parameterizing Master and Access File Values Using Variables

How to:

Create a Master File Variable

Reference:

Support for Variables in Master and Access File Attributes

You can define global variables in a Master File and use them to parameterize certain attributes in the Master File and its corresponding Access File. For example, you can parameterize the connection attribute in the Access File with a variable you define in the Master File and then specify the actual connection name at run time.

Syntax: How to Create a Master File Variable

Add variable definitions after the FILE declaration in the Master File:

```
VARIABLE NAME=[&&]var, USAGE=Aln, [DEFAULT=defvalue,][QUOTED={OFF|ON},] $
```

where:

`[&&]var`

Is the name you are assigning to the global variable. When you reference the variable in the Master or Access File, you must prepend the name with two ampersands. However, the ampersands are optional when defining the variable.

`ln`

Is the maximum length for the variable value.

`defvalue`

Is the default value for the variable. If no value is set at run time, this value is used.

`QUOTED = {OFF|ON}`

ON adds single quotation marks around the assigned string for the variable. A single quotation mark within the string is converted to two single quotation marks. OFF is the default value.

Reference: Support for Variables in Master and Access File Attributes

In the Master File, the following attributes can be parameterized with variables: POSITION, OCCURS, REMARKS, DESCRIPTION, TITLE, HELPMESSAGE.

In the DBA section of a Master File, the following attributes can be parameterized: USER, VALUE.

In the Access File, the following attributes can be parameterized with variables: CONNECTION, TABLENAME, START, CHKPT_SAVE, CHKPT_FILE, POLLING, TIMEOUT, MAXLUWS, ACTION, MSGLIMIT, DIRECTORY, NAME, EXTENSION, DATA_ORIGIN, MAXFILES, MAXRECS, PICKUP, TRIGGER, DISCARD, ARCHIVE.

Note: You can concatenate multiple variables to create an attribute value.

Example: Parameterizing Attributes in a Master and Access File

The following request creates an Oracle table named ORAEMP from the FOCUS data source named EMPLOYEE:

```
TABLE FILE EMPLOYEE
SUM LAST_NAME FIRST_NAME CURR_SAL CURR_JOBCODE DEPARTMENT
BY EMP_ID
ON TABLE HOLD AS ORAEMP FORMAT SQLORA
END
```

The following is the Master File created by the request:

```
FILENAME=ORAEMP , SUFFIX=SQLORA , $
  SEGMENT=SEG01, SEGTYPE=S0, $
    FIELDNAME=EMP_ID, ALIAS=EID, USAGE=A9, ACTUAL=A9, $
    FIELDNAME=LAST_NAME, ALIAS=LN, USAGE=A15, ACTUAL=A15, $
    FIELDNAME=FIRST_NAME, ALIAS=FN, USAGE=A10, ACTUAL=A10, $
    FIELDNAME=CURR_SAL, ALIAS=CSAL, USAGE=D12.2M, ACTUAL=D8, $
    FIELDNAME=CURR_JOBCODE, ALIAS=CJC, USAGE=A3, ACTUAL=A3, $
    FIELDNAME=DEPARTMENT, ALIAS=DPT, USAGE=A10, ACTUAL=A10, $
```

The following is the Access File created by the request:

```
SEGNAME=SEG01, TABLENAME=ORAEMP, KEYS=01, WRITE=YES, $
```

Add the following variable definitions to the Master File in order to parameterize the TABLENAME attribute in the Access File and the TITLE attribute for the EMP_ID column in the Master File:

```
FILENAME=ORAEMP, SUFFIX=SQLORA , $
VARIABLE NAME=table, USAGE=A8, DEFAULT=EDUCFILE, $
VARIABLE NAME=emptitle, USAGE=A30, DEFAULT=empid,$
```

Now, in the Master File, add the TITLE attribute to the FIELD declaration for EMP_ID:

```
FIELDNAME=EMP_ID, ALIAS=EID, USAGE=A9, ACTUAL=A9,
      TITLE='&&emptitle', $
```

In the Access File, replace the value for the TABLENAME attribute with the variable name:

```
SEGNAME=SEG01, TABLENAME=&&table, KEYS=01, WRITE=YES, $
```

The following request sets the values of the variables and then issues a TABLE request:

```
-SET &&table = ORAEMP;
-SET &&emptitle = 'Id,number';
TABLE FILE ORAEMP
PRINT EMP_ID LAST_NAME FIRST_NAME DEPARTMENT
END
```

Note that the value for &&emptitle is enclosed in single quotation marks in the –SET command because it contains a special character (the comma). The single quotation marks are not part of the string and do not display on the report output. The column title would display enclosed in single quotation marks if the variable definition contained the attribute QUOTED=ON.

On the report output, the column title for the employee ID column displays the value set for &&emptitle, and the table accessed by the request is the ORAEMP table created as the first step in the example:

```
Id
number     LAST_NAME         FIRST_NAME  DEPARTMENT
------     ---------         ----------  ----------
071382660  STEVENS           ALFRED      PRODUCTION
112847612  SMITH             MARY        MIS
117593129  JONES             DIANE       MIS
119265415  SMITH             RICHARD     PRODUCTION
119329144  BANNING           JOHN        PRODUCTION
123764317  IRVING            JOAN        PRODUCTION
126724188  ROMANS            ANTHONY     PRODUCTION
219984371  MCCOY             JOHN        MIS
326179357  BLACKWOOD         ROSEMARIE   MIS
451123478  MCKNIGHT          ROGER       PRODUCTION
543729165  GREENSPAN         MARY        MIS
818692173  CROSS             BARBARA     MIS
```

Example: Concatenating Variables to Create an Attribute Value

In the following example, the TABLENAME attribute requires a multipart name consisting of a database name, an owner ID, a table prefix, and a static table name with a variable suffix. In this case, you can define separate variables for the different parts and concatenate them.

First, define separate variables for each part:

```
VARIABLE NAME=db,USAGE=A8,DEFAULT=mydb,$
VARIABLE NAME=usr,USAGE=A8,DEFAULT=myusrid,$
VARIABLE NAME=tprf,USAGE=A4,DEFAULT=test_,$
VARIABLE NAME=tsuf,USAGE=YYM,$
```

In the Access File, concatenate the variables to create the TABLENAME attribute. Note that the separator for between each part is a period, but to concatenate a variable name and retain the period, you must use two periods:

```
TABLENAME=&db..&usr..&tprf.table&tsuf,
```

Based on the defaults, the TABLENAME would be:

```
TABLENAME=mydb.myusrid.test_table
```

In a request, set the following values for the separate variables:

```
I-SET &&db=db1;
-SET &&tprf=prod_;
-SET &&tsuf=200801;
```

With these values, the TABLENAME used is the following:

```
TABLENAME=db1.myusrid.prod_table200801
```

Converting Alphanumeric Dates to Dates

In some data sources, date values are stored in alphanumeric format without any particular standard, with any combination of components, such as year, quarter, and month, and with any delimiter. In a sorted report, if such data is sorted alphabetically, the sequence does not make business sense. To ensure adequate sorting, aggregation, and reporting on date fields, can convert the alphanumeric dates into standard date format using a conversion pattern that you can specify in the Master File attribute called DATEPATTERN.

Each element in the pattern is either a constant character which must appear in the actual input or a variable that represents a date component. You must edit the USAGE attribute in the Master File so that it accounts for the date elements in the date pattern. The maximum length of the DATEPATTERN string is 64.

Reference: Usage Notes for DATEPATTERN

- If your original date has elements with no USAGE format equivalent, the converted date will not look like the original data. In that case, if you want to display the original data, you may be able to use an OCCURS segment to redefine the field with the original alphanumeric format and display that field in the request.
- DATEPATTERN requires an ACTUAL format to USAGE format conversion. Therefore, it is not supported for SUFFIX=FOC and SUFFIX=XFOC data sources.

Specifying Variables in a Date Pattern

How to:

Specify Years in a Date Pattern

Specify Month Numbers in a Date Pattern

Specify Month Names in a Date Pattern

Specify Days of the Month in a Date Pattern

Specify Julian Days in a Date Pattern

Specify Day of the Week in a Date Pattern

Specify Quarters in a Date Pattern

The valid date components (variables) are year, quarter, month, day, and day of week. In the date pattern, variables are enclosed in square brackets (these brackets are not part of the input or output. Note that if the data contains brackets, you must use an escape character in the date pattern to distinguish the brackets in the data from the brackets used for enclosing variables).

Syntax: How to Specify Years in a Date Pattern

`[YYYY]`

Specifies a four-digit year.

`[YYYY]`

Specifies a four-digit year.

`[YY]`

Specifies a two-digit year.

`[yy]`

Specifies a zero-suppressed two-digit year (for example, 8 for 2008).

`[by]`

Specifies a blank-padded two-digit year.

Syntax: How to Specify Month Numbers in a Date Pattern

`[MM]`

Specifies a two-digit month number.

`[mm]`

Specifies a zero-suppressed month number.

`[bm]`

Specifies a blank-padded month number.

Syntax: How to Specify Month Names in a Date Pattern

`[MON]`

Specifies a three-character month name in upper case.

`[mon]`

Specifies a three-character month name in lower case.

`[Mon]`

Specifies a three-character month name in mixed case.

`[MONTH]`

Specifies a full month name in upper case.

`[month]`

Specifies a full month name in lower case.

`[Month]`

Specifies a full month name in mixed case.

Syntax: How to Specify Days of the Month in a Date Pattern

`[DD]`

Specifies a two-digit day of the month.

`[dd]`

Specifies a zero-suppressed day of the month.

`[bd]`

Specifies a blank-padded day of the month.

Syntax: How to Specify Julian Days in a Date Pattern

`[DDD]`

Specifies a three-digit day of the year.

`[ddd]`

Specifies a zero-suppressed day of the year.

`[bdd]`

Specifies a blank-padded day of the year.

Syntax: How to Specify Day of the Week in a Date Pattern

`[WD]`

Specifies a one-digit day of the week.

`[DAY]`

Specifies a three-character day name in upper case.

`[day]`

Specifies a three-character day name in lower case.

`[Day]`

Specifies a three-character day name in mixed case.

`[WDAY]`

Specifies a full day name in upper case.

`[wday]`

Specifies a full day name in lower case.

`[Wday]`

Specifies a full day name in mixed case.

For the day of the week, the WEEKFIRST setting defines which day is day 1.

Syntax: How to Specify Quarters in a Date Pattern

```
[Q]
```

Specifies a one-digit quarter number (1, 2, 3, or 4).

For a string like Q2 or Q02, use constants before [Q], for example, Q0[Q].

Specifying Constants in a Date Pattern

Between the variables, you can insert any constant values.

If you want to insert a character that would normally be interpreted as part of a variable, use the backslash character as an escape character. For example:

- Use \[to specify a left square bracket constant character.
- Use \\ to specify a backslash constant character.

For a single quotation mark, use two consecutive single quotation marks ('').

Sample Date Patterns

If the date in the data source is of the form CY 2001 Q1, the DATEPATTERN attribute is:

```
DATEPATTERN = 'CY [YYYY] Q[Q]'
```

If the date in the data source is of the form Jan 31, 01, the DATEPATTERN attribute is:

```
DATEPATTERN = '[Mon] [DD], [YY]'
```

If the date in the data source is of the form APR-06, the DATEPATTERN attribute is:

```
DATEPATTERN = '[MON]-[YY]'
```

If the date in the data source is of the form APR - 06, the DATEPATTERN attribute is:

```
DATEPATTERN = '[MON] - [YY]'
```

If the date in the data source is of the form APR '06, the DATEPATTERN attribute is:

```
DATEPATTERN = '[MON] ''[YY]'
```

If the date in the data source is of the form APR [06], the DATEPATTERN attribute is:

```
DATEPATTERN = '[MON] \[[YY]\]' (or '[MON] \[[YY]]'
```

Note that the right square bracket does not have to be escaped.

Example: Sorting By an Alphanumeric Date

In the following example, is a sequential file containing the following data:

```
June 1, '02
June 2, '02
June 3, '02
June 10, '02
June 11, '02
June 12, '02
June 20, '02
June 21, '02
June 22, '02
June 1, '03
June 2, '03
June 3, '03
June 10, '03
June 11, '03
June 12, '03
June 20, '03
June 21, '03
June 22, '03
June 1, '04
June 2, '04
June 3, '04
June 4, '04
June 10, '04
June 11, '04
June 12, '04
June 20, '04
June 21, '04
June 22, '04
```

In the DATE1 Master File, the DATE1 field has alphanumeric USAGE and ACTUAL formats, each A18:

```
FILENAME=DATE1   , SUFFIX=FIX ,
  DATASET =     , $
  SEGMENT=FILE1, SEGTYPE=S0, $
    FIELDNAME=DATE1, ALIAS=E01, USAGE=A18, ACTUAL=A18, $
```

The following request sorts by the DATE1 FIELD:

```
TABLE FILE DATE1
PRINT DATE1 NOPRINT
BY DATE1
ON TABLE SET PAGE NOPAGE
END
```

The output shows that the alphanumeric dates are sorted alphabetically, not chronologically:

```
DATE1
-----
June 1, '02
June 1, '03
June 1, '04
June 10, '02
June 10, '03
June 10, '04
June 11, '02
June 11, '03
June 11, '04
June 12, '02
June 12, '03
June 12, '04
June 2, '02
June 2, '03
June 2, '04
June 20, '02
June 20, '03
June 20, '04
June 21, '02
June 21, '03
June 21, '04
June 22, '02
June 22, '03
June 22, '04
June 3, '02
June 3, '03
June 3, '04
June 4, '04
```

In order to sort the data correctly, you can add a DATEPATTERN attribute to the Master File that enables to convert the date to a date field. You must also edit the USAGE format to make it a date format. To construct the appropriate pattern, you must account for all of the components in the stored date. The alphanumeric date has the following variables and constants:

- Variable: full month name in mixed case, [Month].
- Constant: blank space.
- Variable: zero-suppressed day of the month number, [dd].
- Constant: comma followed by a blank space followed by an apostrophe (coded as two apostrophes in the pattern).
- Variable: two-digit year, [YY].

The edited Master File follows. Note the addition of the DEFCENT attribute to convert the two-digit year to a four-digit year:

```
FILENAME=DATE1   , SUFFIX=FIX ,
  DATASET =     , $
  SEGMENT=FILE1, SEGTYPE=S0, $
    FIELDNAME=DATE1, ALIAS=E01, USAGE=MtrDYY, ACTUAL=A18,
      DEFCENT=20,
      DATEPATTERN = '[Month] [dd], ''[YY]', $
```

Now, issuing the same request produces the following output. Note that DATE1 has been converted to a date in MtrDYY format (as specified in the USAGE format):

```
DATE1
-----
June  1, 2002
June  2, 2002
June  3, 2002
June 10, 2002
June 11, 2002
June 12, 2002
June 20, 2002
June 21, 2002
June 22, 2002
June  1, 2003
June  2, 2003
June  3, 2003
June 10, 2003
June 11, 2003
June 12, 2003
June 20, 2003
June 21, 2003
June 22, 2003
June  1, 2004
June  2, 2004
June  3, 2004
June  4, 2004
June 10, 2004
June 11, 2004
June 12, 2004
June 20, 2004
June 21, 2004
June 22, 2004
```

IBM

Index