

# **DataMontage™**

**Version 2.3**

## **Developer's Guide**

© 2010 Stottler Henke Associates, Inc.  
DataMontage is a trademark of Stottler Henke Associates, Inc.  
All rights reserved.

Information in this document is subject to change without notice and does not represent a commitment on the part of Stottler Henke Associates, Inc. The software described in this document is furnished only under a license or non-disclosure agreement. The software may be used only in accordance with the terms of the agreement. Any unauthorized duplication is a violation of U.S. copyright and other laws and may result in severe monetary and criminal charges.

**Stottler Henke Associates, Inc.**

951 Mariner's Island Blvd., suite 360  
San Mateo, CA 94404

Tel: 650.931.2710  
Fax: 650.931.2701  
Web: [www.stottlerhenke.com](http://www.stottlerhenke.com)  
Email: [info@stottlerhenke.com](mailto:info@stottlerhenke.com)

2/5/2010 v2.3

# Table of Contents

<b>1</b>	<b>Getting Started.....</b>	<b>1</b>
1.1	Overview.....	1
1.2	Configuration Requirements .....	1
1.3	Installing DataMontage .....	1
1.4	Uninstalling DataMontage.....	1
1.5	Launching the DataMontage Editor .....	2
1.6	Java.....	2
1.7	Third Party Software .....	2
<b>2</b>	<b>What's New in Version 2.3 .....</b>	<b>2</b>
<b>3</b>	<b>Using the DataMontage Editor .....</b>	<b>3</b>
3.1	DataMontage Editor Sample Session .....	4
3.2	DataMontage Editor Top-Level Menu .....	5
3.3	Using the Overview Pane to Edit the Layout.....	6
3.4	Using the Details Pane to Specify Appearance.....	6
3.5	Embedding SQL Query Builders within the Editor.....	11
<b>4</b>	<b>SQL Query Operations.....</b>	<b>12</b>
4.1	SQL Query Operations – Graph Containers .....	13
4.2	SQL Query Operations - Modules.....	14
4.3	SQL Query Operations - XY Graphs.....	16
4.4	SQL Query Operations - Timelines.....	16
<b>5</b>	<b>Auto Highlighting using Data Point Queries.....</b>	<b>17</b>
<b>6</b>	<b>Creating and Configuring Graph Subsets.....</b>	<b>19</b>
<b>7</b>	<b>Top-Level Container and Graph Classes .....</b>	<b>19</b>
<b>8</b>	<b>Systems Integration Options.....</b>	<b>20</b>
<b>9</b>	<b>Custom User Interactions.....</b>	<b>22</b>
<b>10</b>	<b>Developing and Deploying Your Application.....</b>	<b>23</b>
10.1	Embedding DataMontage within Client-Side Software.....	23
10.2	Embedding DataMontage within Server-Side Software .....	24
10.3	Using the DataMontage Applet Classes.....	25
10.4	Writing XML Configuration Files .....	25
10.5	Distributing DataMontage Run-Time Software Files .....	25
<b>11</b>	<b>Printing.....</b>	<b>26</b>

# 1 Getting Started

## 1.1 Overview

The DataMontage™ software system is a collection of software applications and libraries that enables you to create, display, and print interactive, synchronized, information-dense collections of timelines, XY graphs, and notes that share a common X or time axis. DataMontage can be used to display xy graphs and scatter plots, timelines and Gantt charts, bar graphs, dot plots, stock price graphs, box plots, and bubble charts. The appearance and interactive behavior of these graphical displays are described in the *DataMontage Version 2.3 User's Guide*.

This document describes how to use the *DataMontage Developer's Kit*, comprised of the *DataMontage Editor* and the application programming interface (API) to the *DataMontage software library*. The DataMontage Editor is a desktop application that enables you to create, edit, and save configuration files that specify the content and appearance of graphical displays interactively, without programming, as described in sections 3 through 5. You can also develop software applications that create, modify, load, save, and show DataMontage graphical displays by calling Java methods provided by the DataMontage software library. Use of the DataMontage library's application programming interface (API) is summarized in sections 6 through 11 and is described in detail by on-line documentation in Javadoc format that is included in the Developer's Kit.

## 1.2 Configuration Requirements

DataMontage runs on Intel-compatible PCs running the Windows Windows XP, Windows Vista, and Windows 7 operating systems which support the following configuration requirements:

CPU	Intel Pentium-compatible processors
RAM	2 GBytes
Free disk	11 Mbytes (this does not include disk space needed for the Java runtime system which is installed separately).
Display monitor resolution	1024 x 768 pixels or higher
Display monitor colors	8 bit color or higher
Video memory	2 Mbytes

## 1.3 Installing DataMontage

Before installing a new version of DataMontage, uninstall any previous versions of DataMontage. If you need to run multiple versions of DataMontage on your computer, copy the entire previous installation folder to another location on your computer's file system before uninstalling a previous version.

To install a new version of the software, run the DataMontage installation program file (e.g., `datamontage_2.3.exe`). By default, the installation program installs the files in the *installation folder* whose default location is `c:\Program Files\Stottler Henke\DataMontage 2.3`.

## 1.4 Uninstalling DataMontage

To uninstall the DataMontage software, select the Add or Remove Programs menu choice from the Windows Control Panel. In the list of programs, select *DataMontage* and press the *Change/Remove* button.

## 1.5 Launching the DataMontage Editor

To invoke the DataMontage Editor, select *DataMontage/Editor* from the Windows *Start/All Programs* menu. This action runs DOS command file *runDataMontageEditor.bat* in the DataMontage installation directory. It invokes the Java run-time system and launches the DataMontage Editor application.

**Note:** If you are editing or displaying DataMontage graph containers that contain custom user interactions (section 9) that rely on certain Java classes and methods, edit this command file to include the appropriate Java libraries using the Java *-cp* switch. For example, if you use the SQL query operations feature to populate a graph container with data from a SQL database, edit the command file to include any Java Database Connectivity (JDBC) driver library files needed to access the database. To run procedures that require large amounts of memory, use the Java *-Xms* and *-Xmx* switches to control memory allocation. If you use DataMontage in combination with different Java libraries, you may find it useful to create additional DOS command files with different settings.

## 1.6 Java

DataMontage 2.3 requires pre-installation of Java Standard Edition or Enterprise Edition version 1.6. You can download the Java software distribution from <http://java.sun.com>.

## 1.7 Third Party Software

Unlike previous versions, DataMontage 2.3 requires no third party software other than the Java run-time system. In particular, DataMontage 2.3 no longer uses the Java Architecture for XML Binding (JAXB) software library.

# 2 What's New in Version 2.3

DataMontage version 2.3 provides the following new capabilities for developers:

- **JAXB no longer used** – DataMontage no longer uses the 3<sup>rd</sup> party JAXB Java library to load and save configuration files. This change reduces the size of the Java library files needed to run DataMontage from 2.2 MBytes to 1 MByte.
- **Reference Line and Reference Region Functions** – Within XY graphs, you can specify the values of reference lines as functions of X. You can specify upper and lower Y bounds of reference regions as functions of X.
- **Zip Files** – The *DataMontage Editor* application reads and writes compressed and uncompressed DataMontage configuration files. The ZIP file format is used to store compressed files.
- **Data Point Size** – DataMontage enables you to specify the size of individual data points within XY graphs to encode an additional dimension associated with each data point. Since the data point size specifies the width of each symbol, set the size of each data point so that it varies with the square root of the dimension. DataMontage SQL Operations that create or populate XY graphs have been extended to accept an additional optional column in the query set that specifies the size of each data point.
- **Translucence** - You can specify the translucence of data points to make it easier to distinguish them when they overlap. This feature is useful when making bubble charts that can include large circular symbols that are more likely to overlap than small symbols.
- **Fill Color** – You can specify the interior color of symbols and intervals separately from the outline color.
- **Background and Default Foreground Color** – DataMontage lets you specify the DataMontage container's background color and default foreground color used to draw text, graph borders, graph labels, and tick marks. For example you can design graphs that display a white foreground and black background for use in low-light conditions.

- **Removed Redundant Methods** – In previous versions of DataMontage, some methods contained multiple signatures: one signature accepted a primitive value (e.g., int) and another signature accepted an object (e.g., Integer). Because Java 1.5 and later boxes primitive values into the appropriate wrapper class automatically, some DataMontage method signatures that accept primitive values are no longer needed and have been removed in version 2.3.
- **Export to Image file Formats** – The DataMontage application programming interface provides container-level Java methods that enable you to export DataMontage displays to files using the following image file formats: JPEG at various image qualities, PNG, BMP, and GIF.
- **Regression Lines** - The DataMontage application programming interface provides a new Java method named *addRegressionLine* that calculates a linear regression line and adds this line as a reference line to an XY graph. For details, consult the Javadoc documentation for the *XYGraph*, *XYGraphXDouble*, and *XYGraphXDateTime* classes. In addition, an example custom user interaction class named *com.stottlerhenke.datamontage.interaction.AddRegressionLineUserInteractions* provides user access to these methods via context menu. The source code for this class is provided in the *code\_samples* folder.

### 3 Using the DataMontage Editor

You can use the *DataMontage Editor* to:

- *Create graphical displays for data review* - You can specify the layout and appearance of the graphs and timelines and enter *operational data* to be displayed or printed for analysis and review. You can save and distribute DataMontage configuration files to other users<sup>1</sup>, so they can review them using the DataMontage Viewer or another application that uses the DataMontage API.
- *Create graph mockups* - You can specify the layout and appearance of graphs and timelines and enter *sample data* to create mockups that illustrate a particular way of presenting data. By enabling easy creation of mockups, the Editor enables rapid, iterative design.
- *Create graph templates* - You can specify the layout and appearance of graphs and timelines to create a template that is used as a *starting point* for a graph container. Using the DataMontage Viewer or the DataMontage software library, you can populate the graph container by replacing the sample data in it, if any, with operational data retrieved from databases.
- To invoke the DataMontage Editor, select *DataMontage/Editor* from the Windows *Start* menu. Figure 1 shows the DataMontage Editor when it is first invoked.



Figure 1 – DataMontage Editor when first invoked.

<sup>1</sup> To view interactive DataMontage graph displays specified in configuration files, users must possess and be licensed to operate DataMontage run-time software.

Each DataMontage graph container contains modules, and each module contains timelines or graphs. The left-hand side Overview pane shows an icon for the graph container and an icon for each module, time series graph, and timeline. Icons are displayed in a hierarchy to show the order and grouping of the modules, graphs, and timelines that they represent. To open the graph container icon and view its modules, or to open a module icon and view its graphs or timelines, click on the plus (+) sign button next to the icon. To close a graph container or module icon, click on the minus (-) sign button. Figure 2 shows a graph container that contains five modules.

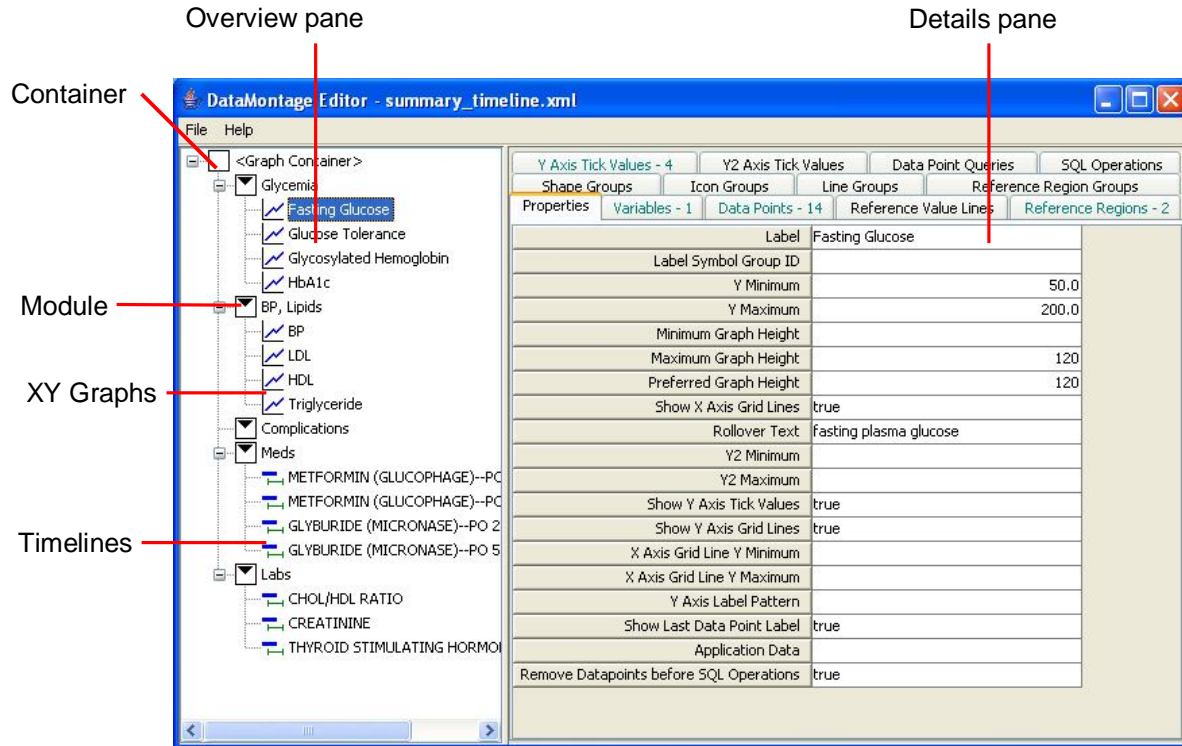


Figure 2 – DataMontage Editor used to edit a DataMontage graph container

The right-hand side Details pane shows tabbed windows that let you view and edit the details of the graph object selected in the Overview pane.

### 3.1 DataMontage Editor Sample Session

The following script describes how you can create a simple DataMontage display comprised of one module and one timeline with two shape groups and two data points.

1. Launch the DataMontage Editor by selecting *DataMontage/Editor* from the *Windows Start/All Programs* menu.
2. Create a new DataMontage graph container with date time X values by selecting *New/DateTime* from the top-level *File* menu.
3. Click on the node in the Overview Pane labeled *<Graph Container>*. A set of tabbed windows for editing the graph container should appear in the Details Pane on the right side of the window.
4. Click on the cell to the right of *Title* in the *Properties* tabbed window. Type "Sample Graph Container" and press the *return* key. The Editor updates the label of the graph container node in the Overview Pane with this title.
5. In the Details Pane, click on the tab labeled *Shape Groups*.
6. Click the right mouse button anywhere in the area below the column headers in the Details Pane. Select *Append Row* twice from the context menu. The Editor creates two new rows.

7. Click on the cell in row 1 under the column header *Group ID*. Enter "normal".
8. Click on the cell in row 2 under the column header *Group ID*. Enter "high".
9. *Double-click* on the cell in row 2 under the column header *Color*. Select a red color swatch and press *OK*. Because these two new shape groups were defined at the container level, they can be referenced by any data point in any XY graph or timeline in the container.
10. Click the right mouse button anywhere on the Overview Pane. Select *Insert/Module* from the context menu that appears. The Editor adds a module icon to the Overview Pane.
11. Click the right mouse button anywhere on the Overview Pane. Select *Insert/Timeline* from the context menu that appears. The Editor adds a timeline icon to the Overview Pane, and the timeline icon is selected.
12. Click on the cell to the right of *Label* in the *Properties* tabbed window. Type in "Sample Timeline" and press the *return* key. The Editor updates the label of the graph container node in the Overview Pane with this title.
13. Click on the *Data Points* tab in the Details Pane.
14. Click the right mouse button anywhere in the area below the column headers in the Details Pane. Select *Append Row* twice from the context menu. The Editor creates two new rows.
15. In the cell in row 1 under the *DateTime* column, enter "1/2/07 0200" to specify January 2, 2007 at 2:00AM. To widen a column, move your mouse to the right of the column until you see the cursor change shape to a two-headed arrow. Then, click and drag your mouse to the right to change the column width.
16. In the cell in row 1 under the *DateTime* column, enter "Jan 3, 2007" to specify January 3, 2007.
17. In the cell in row 1 under the *Symbol Group ID* column, double-click and select "normal" from the pull-down list.
18. In the cell in row 2 under the *Symbol Group ID* column, double-click and select "high" from the pull-down list.
19. Select *Preview* from the *File* menu to launch the DataMontage Viewer application and see a display of the graph container.
20. Click the right mouse button over the DataMontage Viewer display. Select *Show Graph Key/Container* from the context menu to display a graph key describing all graphical elements in the graph container.

### 3.2 DataMontage Editor Top-Level Menu

The File menu of the DataMontage Editor contains the following menu items:

<i>New</i>	Creates a new DataMontage graph container. When you create a new container, the Editor asks you whether X axis values will be date times or floating point values.
<i>Open</i>	Opens a DataMontage graph container stored in a configuration file. You can select either a configuration file (with .xml file extension) or a Zip file (with .zip file extension) that contains the XML configuration file. In the latter case, the Zip file and the configuration file must share the same name, with different file extensions.
<i>Save</i>	Saves the currently open DataMontage graph container to a file.
<i>Save As...</i>	Saves the currently open DataMontage graph container to a different file. If you specify a file name that ends with ".zip", DataMontage will save the file in compressed Zip format.
<i>Preview</i>	Invokes the DataMontage Viewer application to display the current graph container.
< <i>Recent Files</i> >	Opens a previously saved configuration file.
<i>Exit</i>	Exits the Editor application.

### 3.3 Using the Overview Pane to Edit the Layout

You can interact with the Overview Pane to create, delete, or rearrange modules, XY graphs, and timelines. When you click the right mouse button over the Overview pane, the Editor displays a context menu that contains the following menu items:

<i>Undo</i>	Undoes the last editing operation performed in the Overview pane.
<i>Redo</i>	Redoes the last editing operation performed in the Overview pane.
<i>Cut</i>	Deletes the currently selected module, XY graph, or timeline and stores this object in the clipboard.
<i>Copy</i>	Copies the currently selected module, XY graph, or timeline and stores this object in the clipboard.
<i>Paste</i>	Inserts the module, time series graph, or timeline currently stored in the clipboard into the graph container.
<i>Delete</i>	Deletes the currently selected module, XY graph, or timeline.
<i>Move</i>	Moves the currently selected module, XY graph, or timeline Up or Down or to the Top or Bottom of the graph container.
<i>Insert</i>	Inserts a module, XY graph, or timeline into the graph container.
<i>Rename</i>	Prompts you to enter a new Label property for the currently selected module, XY graph, timeline, or graph container.

### 3.4 Using the Details Pane to Specify Appearance

The Details Pane displays tabbed windows, so you can view and edit the details of the module, XY graph, timeline, or container that is selected in the Overview pane. The Editor displays a different set of tabbed windows for each of the four types of graph objects.

#### 3.4.1 Details Pane – Common Tabbed Windows

Some of the tabbed windows enable you to specify properties of shape groups, icon groups, line groups, and reference interval and regions that control how data and reference data are displayed.

Shape Groups	Shape groups specify the appearance of symbols drawn within XY graphs and timelines to indicate data points. Graph symbols can also be drawn next to the labels of XY graph and timelines and at the beginning and/or end of timeline intervals to provide additional information about the interval. Within XY graphs, you can use different symbol groups to encode additional information about particular data points. For example, you can use a particular symbol shape or color to highlight data point values that are lower or higher than expected. A shape group can be specified for use within a graph, timeline, module, or container. For example, if a shape group is specified for a module, it can be used to display data points in any timeline or XY graph in that module. If a shape group is specified for a timeline, it can be used only within the timeline. Some shape groups are multivariate and display more than one value. For example, the BoxPlot shape group displays the median, upper and lower quartile, and range. The OpenCloseTicks shape group shows the opening, closing, low, and high value of a stock price.
Icon Groups	You can display data points by drawing icons instead of symbols. Each icon group object specifies a type of icon that is rendered by a Java class that implements a Java interface named IconSupplier. DataMontage provides

two built-in implementations of the IconSupplier interface. The `com.stottlerhenke.datamontage.icon.TextIconMaker` Java class renders data points by drawing a single character or short text string. The `com.stottlerhenke.datamontage.icon.ImageIconLoader` Java class load icons represented as image files in GIF or JPEG format. An advantage of encoding icons in GIF format is that you can set the image's background to be transparent. An icon group can be specified for a graph, timeline, module, or container.

Line Groups	Line groups specify the color and thickness of reference lines and lines that connect data points within XY graphs that belong to the same variable. A line group can be specified for a graph, timeline, module, or container.
Interval Groups	Each interval group object specifies the color, style, and height of horizontal bars used to display a set of timeline intervals.
Reference Region Groups	Reference region groups specify the color used to fill a region within XY graphs to show a datetime/X value range and/or a Y value range. The region group also specifies the layer number. When two reference regions overlap, regions with larger layer number are drawn over regions with smaller layer numbers.
Reference Interval Groups	Reference interval groups specify the color used to fill a region within timelines to show a datetime/X value range.

The following table shows the types of appearance groups that can be defined for each type of sub-object: containers, modules, graphs, and timelines.

<i>Type of Appearance Group</i>	<i>Container</i>	<i>Module</i>	<i>XY Graph</i>	<i>Timeline</i>
<i>Shape Groups</i>	●	●	●	●
<i>Icon Groups</i>	●	●	●	●
<i>Line Groups</i>	●	●	●	
<i>Reference Region Groups</i>	●	●	●	
<i>Interval Groups</i>	●	●		●
<i>Reference Interval Groups</i>	●	●		●

The DataMontage Editor also provides tabbed windows that enable you to specify SQL query operations (section 4) and data point queries (section 5) for all four types of graph sub-objects.

### 3.4.2 Details Pane – Graph Containers

In addition to the common tabbed windows described in section 3.4.1, the DataMontage Editor displays the following tabbed windows in the Details pane when the graph container icon is selected:

Properties	This tabbed window displays and prompts for single-valued property values of the container.
Note Group	The note group object specifies the color, size, and font family used to display time-stamped notes.
Zoom Periods	A DataMontage graph container can contain zero or more zoom periods. Each zoom period specifies the graph container's X value range that is displayed when the zoom period is selected by the user using the DataMontage Viewer or Applet context menu.
X Axis Tick Values	You can specify one or more X axis tick values and labels that override the default tick values and labels.

SQL Parameters	A graph container can contain zero or more SQL Query Operations that populate the graphical display with data retrieved from a SQL database. Each SQL Query Operation contains a SQL query with optional SQL parameters. Before defining a SQL Query Operation with one or more parameterized SQL queries, you should use this tabbed window to declare the parameters by entering their names, data types, and labels. The name is used each SQL Query Operation to refer to this parameter. More than one SQL Query Operation can refer to the same parameter. When the DataMontage Viewer populates a graph container with SQL data, it prompts the user for the value of each SQL parameter value, using the parameter label as a prompt. A software application can supply a value for each SQL parameter when it uses the DataMontage software library to instruct the graph container to populate itself with SQL data.
Notes	Each Note object represents a text note that is displayed in a list within the graph container. When the user clicks on a note display, DataMontage displays a vertical highlight line in all timelines and graphs to indicate the note's time stamp or X axis value.
Data Point Queries	Each data point query specifies criteria for selecting and highlighting data points and/or time intervals in graphs and timelines, based on the value of their applicationData property. Each query can be invoked by selecting its menu item from the context menu.

### 3.4.3 Details Pane – Modules

The DataMontage Editor displays tabbed windows in the Details pane for each type of appearance group. Additional tabbed windows enable you to edit Module property values, SQL query operations, and data point queries. The X Axis Tick Values tabbed window enables you to specify X axis tick values for the module that may be different than those for the graph container.

### 3.4.4 Details Pane – XY Graphs

The Editor displays the following tabbed windows in the Details pane when an XY graph icon is selected:

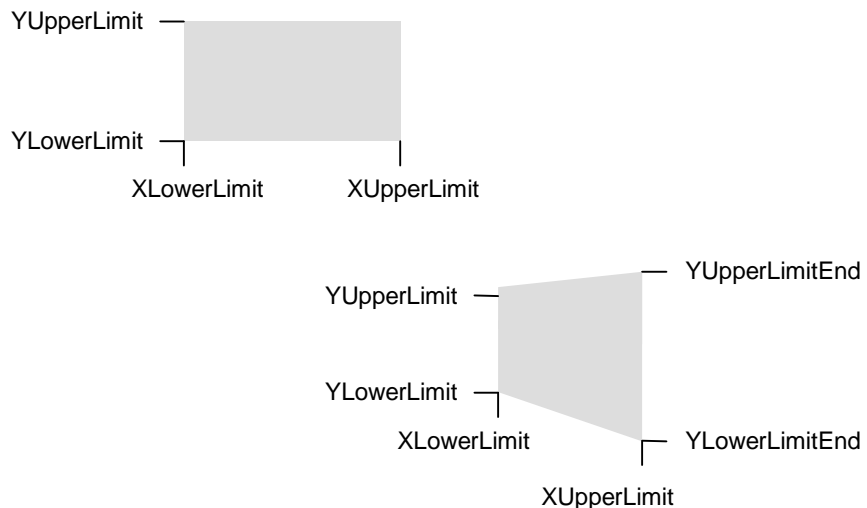
Variables	Each variable groups zero or more data points in an XY graph that represent different measurements over time of the same thing. Each graph can contain data points for one or more variables.
Data Points	Each data point describes a quantitative measurement that is associated with a variable. The data point's X value specifies when the measurement was made, and the Y value specifies the measured value. Optionally, each data point can be associated with a symbol group that describes its appearance. Some multivariate shape groups display more than one value, stored in other properties of the data point object. A variable can be associated with a line group that describes the appearance of the line that connects the variable's data points.
Reference Value Lines	DataMontage XY graphs can display <i>reference value lines</i> that help you compare data points to Y reference values. The color and thickness of each reference line is specified by its associated <i>line group</i> . One or more reference value lines can belong to the same line group and share the same meaning and appearance. A reference line can span the entire width of the graph, or it can be bounded by lower and/or upper X bounds. The Y value of each reference value line can be specified by a single constant Y value. If lower and upper X bounds are both specified, Y values can also be linearly interpolated from the Y values specified for the lower and upper X bounds.

You can also specify the reference line as a function of X using JavaScript. For example, if this property value equals "x\*x", the value of the reference line equals 1 at x=1, 2.25 at x=1.5, etc. DataMontage computes and displays a piece-wise linear function that approximates the function specified by the Function property. The optional Pixel Spacing and X Spacing properties control the spacing of the sample points of the piece-wise linear function.

Reference Regions

DataMontage XY graphs can also display *reference regions* as colored backgrounds that help you compare data points to ranges of Y values that can vary across time. The color of each reference region is specified by its associated *reference region group*. One or more reference regions can belong to the same reference region group and share the same meaning and appearance.

Each reference region can span the entire width of the graph, or it can be bounded by lower and/or upper X bounds. If lower and upper X bounds are both specified, upper and lower Y bounds can be linearly interpolated from the Y values specified for the lower and upper X bounds. The lower and upper Y bounds of a reference region can also be specified as functions of X using JavaScript. The optional Pixel Spacing and X Spacing properties control spacing of the piece-wise linear function's sample points. If no upper Y bound is specified, the region spans all parts of the graph above the lower bound. If no lower Y bound is specified, the region spans all areas below the upper bound.



- Y Axis Tick Values      Each object specifies the value and label of a (left hand) Y axis tick mark.
- Y2 Axis Tick Values    Each object specifies the value and label of a (right hand) Y2 axis tick mark.

**3.4.5 Details Pane – Timelines**

The Editor displays the following tabbed windows in the Details pane when a timeline icon is selected:

- Data Points              Each timeline data point describes an event or observation at a point in time. Observations displayed within time lines are usually qualitative, but quantitative values can be displayed as text labels that appear next to each data point's symbol in the timeline. For example, heart rate measurements can be displayed as text labels above each data point (e.g., "135/80").

Intervals	Each interval describes an event or condition that spans an interval bounded by start and/or end datetimes or by lower and/or upper X axis values.
Reference Intervals	Each reference interval is displayed as a colored background region that spans the interval's range, delimited by start and/or end times or by X axis lower and/or upper bounds. Each reference interval belongs to a reference interval group that specifies the reference interval's color and meaning.

### 3.4.6 Details Pane – Editing Tables

In most of the tabbed windows, the Details pane enables you to view and edit data in tables. Each table row displays an object, and each column displays values for a property. To enter or change the value of an object's property, double-click the left mouse button over the table cell. Depending upon which property you select, you will be able to enter a new value by typing into the table cell itself, selecting a value from a pull-down list, or entering a value using a popup dialog.

A context menu provides editing operations that help you edit values in table cells. To select this menu, click the right mouse button over the Details pane. When the Properties tabbed window is displayed, the context menu contains the following items:

<i>Undo</i>	Undoes the last editing operation
<i>Redo</i>	Redoes the last editing operation
<i>Cut Cells</i>	Deletes the values in the currently selected table cells and copies these values to the clipboard.
<i>Copy Cells</i>	Copies the values in the currently selected table cells to the clipboard.
<i>Paste</i>	Pastes the values stored in the clipboard to the table.
<i>Clear</i>	Clears the values of the currently selected table cells.

Other tabbed windows display rows and columns of data that you can edit. The context menu for these windows contains items that enable you to edit tabular data:

<i>Undo</i>	Undoes the last editing operation.
<i>Redo</i>	Redoes the last editing operation.
<i>Cut</i>	Deletes the values in the currently selected cell and copies them to the clipboard.
<i>Copy</i>	Copies the values in the currently selected table cell to the clipboard.
<i>Paste</i>	Pastes the values stored in the clipboard to the currently selected table cell.
<i>Paste as New Rows</i>	Pastes the values stored in the clipboard to the currently selected table cell.
<i>Clear</i>	Clears the values of the currently selected table cell.
<i>Delete Rows</i>	Deletes the currently selected rows.
<i>Append Row</i>	Adds a new row to the bottom of the table.
<i>Insert Row</i>	Inserts a new row before the currently selected row.
<i>Sort</i>	Sorts the rows so that values in the selected column are in ascending or descending order. You can also sort in ascending order by clicking on the column header while pressing the control key. You can sort in descending order by clicking on the column header while pressing the control and shift keys.
<i>Move</i>	Moves the currently selected row Up or Down or to the Top or Bottom of the table.

### 3.4.7 Details Pane – Entering Dates

Some of the DataMontage properties store date or date time values. You can enter a date using the following formats:

- YYYY-MM-DD – 4 digit year, a number between 1 and 12 to specify the month, and the day of the month, separated by dash (-) characters.
- MM/DD/YY or MM.DD.YYYY – In locales such as the United States, you can enter the month number, day of month, and two or four digit year, separated by slash (/) characters or periods (.).
- DD/MM/YY or DD.MM.YYYY – In locales such as Europe, you can enter the day of month, month number, and a two or four digit year, separated by slash (/) characters or periods (.).
- Mon DD, YYYY or Month DD, YYYY - In locales such as the United States, you can enter the month name or three letter abbreviation, day of month, comma, and four digit year.
- DD Month YYYY - In locales such as Europe, you can enter the day of the month, month name, and four digit year. In the UK, you can enter a three letter month abbreviation.

To enter a date time, enter a date, followed by a time, separated by a single space or letter T. You can enter the time portion of the date time using the following formats.

- 24 hour time - "1:45" specifies 45 minutes past 1:00 AM, and "13:45:50" specifies 45 minutes 50 seconds past 1:00 PM. You can also enter 24 hour times with 2 digit hours without colons such as "0145" or "134550".
- 12 hour time – e.g., "1:45am" specifies 45 minutes past 1:00 AM, and "1:45:50 pm" specifies 45 minutes 50 seconds past 1:00 PM.

Use a space character or 'T' character to separate the date and time portions of a date time value. For example, in the United States, you can enter either "3/4/07 13:45:50" or "2007-03-04T1:45 pm" to specify 1:45 PM on March 4, 2007.

After you enter a date or date time, DataMontage redisplay the value in its standard format. It displays dates as a four digit year, two digit month of the year, and two digit day of the month, separated by dashes (e.g., 2007-01-05). It displays times as the hour of the day (between 0 and 23), followed by minutes (between 0 and 59) and seconds (between 0 and 59), separate by colons. The Editor truncates the hours, minutes, or seconds portion of a date time if those values equal 0 *and* the graph container's Precision property is set to Days. The Editor truncates the seconds portion of the date time if it equals 0 and the Precision property is set to Minutes or Days.

## 3.5 Embedding SQL Query Builders within the Editor

To create a custom SQL Query Builder, a Java software developer can create a Java class that implements the *SQLQueryBuilder* Java interface defined in Java package *com.stottlerhenke.com.datamontage.editor*. This interface is described in the DataMontage on-line Javadoc documentation. To use your SQL Query Builder, edit the DOS command that launches the DataMontage Editor (e.g., in command file *runDataMontageEditor.bat*) to include:

```
-qcQueryBuilderClassname
```

where *QueryBuilderClassname* is a placeholder for the full name of the Java class that implements the custom SQL query builder. Optionally, you can also supply a comma-separated list of String constructor arguments when creating an instance of the query builder Java class by including the following command-line switch and argument:

```
-qaQueryBuilderArgumentList
```

## 4 SQL Query Operations

A DataMontage container can optionally specify SQL query operations that populate the graph container with data retrieved from a relational database. Using the *SQL Operations* tabbed window, you can enter SQL query operations for each module, XY graph, timeline, and graph container. Each SQL query operation specifies the:

- **Operation Type** – determines how the SQL data is used to populate the graph container by creating data points, intervals, graphs, timelines, and/or modules.
- **SQL Query** – determines which data should be retrieved from a relational database. Double-click the cell in the *SQL Query* column to display a text editor dialog, shown at right, that accepts your SQL query. Press the *Insert SQL Template* button to insert a SQL query template for the selected SQL operation type. Convert the SQL query template to a valid SQL query by editing each template placeholder, enclosed between angle brackets.

DataMontage's custom SQL query builder feature enables you to create a custom SQL Query Builder and configure the DataMontage Editor to invoke it. Using a custom SQL query builder, you can specify SQL queries more easily for your particular database. Section 3.5 describes how to create custom SQL query builders.

- **SQL Parameters** – specifies the names of zero or more parameters whose values are bound to placeholders in a parameterized SQL query statement. There should be one SQL parameter for each placeholder in the SQL query. Parameter names for each SQL operation are selected from the list of SQL parameters specified in the graph container's *SQL Parameters* tabbed window.

The *Remove Data before SQL Operations* property, specified in the graph container's *Properties* tabbed window, controls whether any data in the container, module, timeline, or XY graph should be removed before executing SQL query operations. Usually, this property should be set to true. For example, you can create a graph container that is used as a template that is populated with data queried from a database. Set the *Remove Data before SQL Operations* property to true to ensure that any example data in the template is removed before the template is populated.

The *Database Connection URL* property, specified in the graph container's *Properties* tabbed window, specifies a Java Database Connectivity (JDBC) Uniform Resource Locator (URL) that identifies the location of the database to be accessed. For example, to access an Oracle database, you could specify a URL of the form:

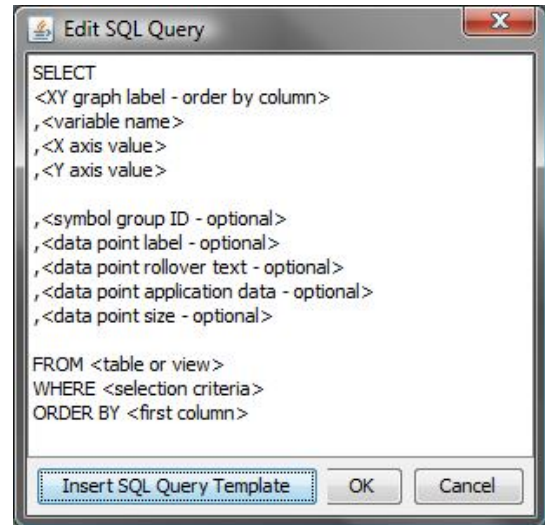
```
jdbc:oracle-thin:@host_name:ip_port_number:db_name
```

where *host\_name*, *ip\_port\_number*, and *db\_name* are placeholders for the name of the database's host computer, the database server's Internet Protocol port number, and the name of the Oracle database. To access data in an Excel spreadsheet using an Open Connectivity Database (ODBC) driver, you can specify:

```
jdbc:odbc:Driver={Microsoft Excel Driver (*.xls)};DBQ=Excel_file_path
```

where *Excel\_file\_path* is a placeholder for the full path name of the Excel file, or a path name that is relative to the folder that contains the DataMontage.jar software library file. When using some versions of Microsoft Excel, such as Excel 2007, it may be necessary to open the Excel file using the Excel software application while DataMontage accesses the data via JDBC.

When executing SQL operations, DataMontage first executes container-level SQL operations. Then, it executes SQL operations for the first module and SQL operations for each timeline or graph within the module. Then, it



executes SQL operations for the second module and SQL operations for each timeline or graph within the second module, and so on.

#### 4.1 SQL Query Operations – Graph Containers

A graph container can contain four types of SQL query operations.

The **Add Modules with Graphs** operation creates zero or more modules, and each module contains one or more graphs each containing one or more data points. The SQL query must return a result set that contains six required columns with non-null values and up to three optional columns with null values allowed. Rows in this table must be grouped so that rows with the same module header are adjacent and rows with the same module header and XY graph label are adjacent. DataMontage creates a module for each distinct value of the module header column, and it creates an XY graph for each combination of module header and XY graph label.

Column #	Required?	Nulls allowed?	Description of contents
1	Yes	No	Module header
2	Yes	No	XY graph label
3	Yes	No	Variable name
4	Yes	No	X axis value
5	Yes	No	Y axis value
6	No	Yes	Symbol group ID
7	No	Yes	Data point label
8	No	Yes	Data point rollover text
9	No	Yes	Data point application data
10	No	Yes	Data point size (overrides size specified by symbol group)

The **Add Modules with Timelines with Data Points** operation creates zero or more modules, and each module contains one or more timelines each containing one or more data points. The SQL query must return a result set that contains four required columns with non-null values and up to three optional columns with null values allowed. Rows in this table must be grouped so that rows with the same module header are adjacent and rows with the same module header and timeline label are adjacent. DataMontage creates a module for each distinct value of the module header column, and it creates a timeline for each combination of module header and timeline label.

Column #	Required?	Nulls allowed?	Description of contents
1	Yes	No	Module header
2	Yes	No	Timeline label
3	Yes	No	X axis value
4	Yes	No	Symbol group ID
5	No	Yes	Data point label
6	No	Yes	Data point rollover text
7	No	Yes	Data point application data

The **Add Modules with Timelines with Intervals** operation creates zero or more modules, and each module contains one or more timelines each containing one or more time intervals. The SQL query must return a result set that contains five required columns with non-null values and up to three optional columns with null values allowed. Rows in this table must be grouped so that rows with the same module header are adjacent and rows with the same module header and timeline label are adjacent. DataMontage creates a module for each distinct value of the module header column, and it creates a timeline for each combination of module header and

timeline label.

Column #	Required?	Nulls allowed?	Description of contents
1	Yes	No	Module header
2	Yes	No	Timeline label
3	Yes	No	Start X axis value of the interval
4	Yes	No	End X axis value of the interval
5	Yes	No	Interval group ID of the interval
6	No	Yes	Interval label
7	No	Yes	Interval rollover text
8	No	Yes	Interval application data

The **Add Modules with Timelines with Data Points and Intervals** operation creates zero or more modules, and each module contains one or more timelines each containing one or more data points and/or time intervals. The SQL query must return a result set that contains six required columns and up to three optional columns with null values allowed. Rows in this table must be grouped so that rows with the same module header are adjacent and rows with the same module header and timeline label are adjacent. DataMontage creates a module for each distinct value of the module header column, and it creates a timeline for each combination of module header and timeline label.

Column #	Required?	Nulls allowed?	Description of contents
1	Yes	No	Module header
2	Yes	No	Timeline label
3	Yes	Yes	X axis value of the datapoint, if not null. If null, this row specifies a time interval.
4	Yes	Yes	Start X axis value of the interval, if not null
5	Yes	Yes	End X axis value of the interval, if not null
6	Yes	No	Symbol group or interval group ID
7	No	Yes	Data point or time interval label
8	No	Yes	Data point or time interval rollover text
9	No	Yes	Data point or time interval application data

If the container contains any modules before SQL operations are executed, the first module is used as a template when creating new modules, and the first graph or timeline in that module is used as a template for creating new graphs and timelines. Thus, new modules, graphs, or timelines created by the SQL operations will contain the same properties and sub-objects specified for the template module, graph, or timeline. For example, you could specify symbol groups in a module or graph template that would be copied and able to be referenced by data points in modules and graphs created by SQL operations.

## 4.2 SQL Query Operations - Modules

A module can contain four types of SQL query operations. The **Add Graphs** operation creates zero or more graphs each containing one or more data points. The SQL query must return a result set that contains six required columns with non-null values and up to three optional columns with null values allowed. Rows in this table must be grouped so that rows with the same XY graph label are adjacent. DataMontage creates a new graph for each distinct value of XY graph label.

Column #	Required?	Nulls allowed?	Description of contents
----------	-----------	----------------	-------------------------

1	Yes	No	XY graph label
2	Yes	No	Variable name
3	Yes	No	X axis value
4	Yes	No	Y axis value
5	No	Yes	Symbol group ID
6	No	Yes	Data point label
7	No	Yes	Data point rollover text
8	No	Yes	Data point application data
9	No	Yes	Data point size (overrides size specified by symbol group)

The **Add Timelines with Data Points** operation creates zero or more timelines each containing one or more data points. The SQL query must return a result set that contains three required columns with non-null values and up to three optional columns with null values allowed. Rows in this table must be grouped so that rows for the same timeline label are adjacent. DataMontage creates a timeline for each distinct value of timeline label.

Column #	Required?	Nulls allowed?	Description of contents
1	Yes	No	Timeline label
2	Yes	No	X axis value
3	Yes	No	Symbol group ID
4	No	Yes	Data point label
5	No	Yes	Data point rollover text
6	No	Yes	Data point application data

The **Add Timelines with Intervals** operation creates one or more timelines each containing one or more time intervals. The SQL query must return a result set that contains four required columns and up to three optional columns with null values allowed. Rows in this table must be grouped so that rows for the same timeline label are adjacent. DataMontage creates a timeline for each distinct value of timeline label.

Column #	Required?	Nulls allowed?	Description of contents
1	Yes	No	Timeline label
2	Yes	Yes	Start X axis value of the interval
3	Yes	Yes	End X axis value of the interval
4	Yes	No	Interval group ID of the interval
5	No	Yes	Interval label
6	No	Yes	Interval rollover text
7	No	Yes	Interval application data

The **Add Timelines with Data Points and Intervals** operation creates zero or more timelines each containing one or more data points and/or time intervals. The SQL query must return a result set that contains five required columns and up to three optional columns with null values allowed. Rows in this table must be grouped so that rows for the same timeline label are adjacent. DataMontage creates a timeline for each distinct value of timeline label.

Column #	Required?	Nulls allowed?	Description of contents
----------	-----------	----------------	-------------------------

1	Yes	No	Timeline label
2	Yes	Yes	X axis value of the datapoint, if not null. If null, this row specifies a time interval.
3	Yes	Yes	Start X axis value of the interval, if not null
4	Yes	Yes	End X axis value of the interval, if not null
5	Yes	No	Symbol group or interval group ID
6	No	Yes	Data point or time interval label
7	No	Yes	Data point or time interval rollover text
8	No	Yes	Data point or time interval application data

If the modules contains any timelines or graphs before SQL operations are executed, the first graph or timeline is used as a template when creating new graphs or timelines. Thus, new graphs, or timelines created by the SQL operations will contain the same properties and sub-objects specified for the template graph, or timeline.

### 4.3 SQL Query Operations - XY Graphs

A module can contain one type of SQL query operations. The **Add Data Points** operation adds zero or more data points to the XY graph. The SQL query must return a result set that contains six required columns with non-null values and up to three optional columns with null values allowed.

Column #	Required?	Nulls allowed?	Description of contents
1	Yes	No	Variable name
2	Yes	No	X axis value
3	Yes	No	Y axis value
4	No	Yes	Symbol group ID
5	No	Yes	Data point label
6	No	Yes	Data point rollover text
7	No	Yes	Data point application data
8	No	Yes	Data point size (overrides size specified by symbol group)

### 4.4 SQL Query Operations - Timelines

A timeline can contain three types of SQL query operations. The **Add Data Points** operation adds zero or more data points to the timeline. The SQL query must return a result set that contains two required columns with non-null values and up to three optional columns with null values allowed.

Column #	Required?	Nulls allowed?	Description of contents
1	Yes	No	X axis value
2	Yes	No	Symbol group ID
3	No	Yes	Data point label
4	No	Yes	Data point rollover text
5	No	yes	Data point application data

The **Add Time Intervals** operation adds one or more time intervals to the timeline. The SQL query must return a result set that contains three required columns and up to three optional columns with null values allowed.

Column #	Required?	Nulls allowed?	Description of contents
1	Yes	Yes	Start X axis value of the interval
2	Yes	Yes	End X axis value of the interval
3	Yes	No	Interval group ID of the interval
4	No	Yes	Interval label
5	No	Yes	Interval rollover text
6	No	Yes	Interval application data

The **Add Data Points and Intervals** operation adds one or more data points and/or time intervals to the timeline. The SQL query must return a result set that contains four required columns and up to three optional columns with null values allowed.

Column #	Required?	Nulls allowed?	Description of contents
1	Yes	Yes	X axis value of the data point, if not null.
2	Yes	Yes	Start X axis value of the interval, if not null
3	Yes	Yes	End X axis value of the interval, if not null
4	Yes	No	Symbol group or interval group ID
5	No	Yes	Data point or time interval label
6	No	Yes	Data point or time interval rollover text
7	No	Yes	Data point or time interval application data

## 5 Auto Highlighting using Data Point Queries

DataMontage graph containers, modules, XY graphs, and timelines can be configured to contain `DatapointQuery` objects that specify how to select and highlight data points and/or time intervals based on the value of each *comparison object* stored in its *applicationData* property. Each query can be invoked by selecting a context menu item. The data point query's *selectionMethod* property specifies the name of the Boolean Java method that returns *true* if the comparison object's data point or time interval should be highlighted. The selection method can specify a required constraint on each comparison object, or it can specify a required relationship between each comparison object and a *reference object*. DataMontage provides three ways of specifying the source of the reference object. The reference object can be:

- The value of the *applicationData* property of the data point or time interval currently selected by the user. The *applicationData* object must be an instance of the Java class named by the query's *referenceClass* property.
- The value of the query's *referenceValue* property,
- The value of a text string entered by the user into a text edit GUI control whose label is specified by the query's *referenceValuePrompt* property.

The selection method can be a static method of the class specified by the query's *staticSelectionMethodClass* property. Static selection methods can accept either one argument (the comparison object) or two arguments (the *reference object* and the comparison object). If the selection method is not static, the method must be defined for the Java class of the reference object.

	Source of reference object			
Property	No reference object	<i>applicationData</i> of user-selected	Value of <i>referenceValue</i>	Text string entered by

		data point or time interval	property	user
menuLabel	required	required	required	required
comparisonClass	required	required	required	required
selectionMethod	required	required	required	required
staticSelectionMethodClass	required	optional	optional	optional
highlightMethod	optional	optional	optional	optional
color	optional	optional	optional	optional
referenceClass		required		
referenceValue			required	
referenceValuePrompt				required

The following table above shows whether a data point query property is required (req), optional (opt), or not accepted, depending upon the source of the query's reference object. Note that only one of the properties *referenceClass*, *referenceValue*, or *referenceValuePrompt* should be specified. If none of the properties are specified, no reference object is used by the query.

The example *DatapointQuery* specified below is invoked when the user selects the context menu item labelled "Compare strings". Because the *referenceClass* property is set, the reference object is the *applicationData* property of the currently selected data point or time interval. The *referenceClass* property specifies that the menu item is enabled only if the reference object is an instance of the *java.lang.String* class. The *comparisonClass* property specifies that comparison objects must be instances of class *java.lang.Object*. Each comparison object is compared to the reference object by invoking the "equals" method of the reference object. This method accepts a single argument, the comparison object which, according to the *comparisonClass* property, must be an instance of class *java.lang.Object*.

If the *staticSelectionMethodClass* property had been set, DataMontage would compare the comparison object with the reference object by invoking a static method named "equals" defined for the class specified by the *staticSelectionMethodClass* property. This method would accept two arguments, the reference object (1<sup>st</sup> argument) and the comparison object (2<sup>nd</sup> argument). The *color* property specifies the highlighting color in #RRGGBB hexadecimal format. In this example, selecting data points and time intervals are highlighted in green (red = #00, green = #FF, blue = #00). The *highlightMethod* property specifies how data points are highlighted. A value of "Circle", the default, specifies that an unfilled circle is drawn around the data point. A value of "Flash" specifies that the data point symbol flashes.

Property	Value
menuLabel	Compare strings
comparisonClass	java.lang.Object
selectionMethod	equals
staticSelectionMethodClass	
color	#00FF00
highlightMethod	Flash
referenceClass	java.lang.String
referenceValue	
referenceValuePrompt	

## 6 Creating and Configuring Graph Subsets

DataMontage graph containers and modules can be optionally configured to specify one or more graph subset manager classes. The *graphSubsetsManagerClasses* property, defined for containers and modules, stores a comma-separated list of fully-qualified graph subset manager class names. Each container-level graph subset manager specifies one or more container graph subsets, and each module-level graph subset manager specifies one or more module graph subsets.

To create a graph subset manager, create a Java class that implements the DataMontage API Java interface named *GraphSubsetsManager* in the *com.stottlerhenke.datamontage.interaction* package. This interface specifies methods that return the list of graph subset IDs, return a graph subset label, and determine whether a graph or module belongs to a particular graph subset, identified by its ID. The DataMontage API documentation specifies this interface in detail in Javadoc format. File *GraphSubsetsManagerApplicationData.txt* in the *code\_samples* folder contains an example implementation of a graph subset manager class. This class is included in file *DataMontage.jar* and supports selection of subsets based on the String values stored in the *applicationData* property of each graph or timeline.

## 7 Top-Level Container and Graph Classes

The independent (X) axis values of DataMontage timeline datapoints, timeline intervals, and XY data points can be double-precision floating point values or datetime values. The DataMontage library provides one set of classes for displaying data with double-precision X values, and a second set of classes for displaying data with datetime X values. Some classes are shared among these two sets. Names of classes that support double-precision data end with "XDouble", and the names of the classes for datetime data end with "XDateTime".

Each DataMontage graphical display is configured by a top-level DataMontage container object and its sub-objects. Double-precision data is displayed within *GraphContainerXDouble* container objects and datetime data is displayed within *GraphContainerXDateTime* container objects. Each container object contains zero or more module objects, and each module object contains zero or more timeline and/or time series graph objects. The container object also contains zero or more note objects, one per time-stamped note. The following table summarizes the highest-level classes.

GraphContainerXDateTime GraphContainerXDouble	Each graph container object contains module objects. Each GraphContainerXDateTime object contains GraphModuleXDateTime objects, and each GraphContainerXDouble object contains GraphModuleXDouble objects.
GraphModuleXDateTime GraphModuleXDouble	Each module object contains timeline and/or XY graph objects. Each GraphModuleXDateTime object contains TimeLineXDateTime and/or XYGraphXDateTime objects, and each GraphModuleXDouble object contains TimeLineXDouble and/or XYGraphXDouble objects.  The number of graphs per row is specified for each module. Graphs and timelines can be vertically stacked (number of graphs per row = 1) or arranged in rows and columns (number of graphs per row > 1). Different modules can contain different numbers of graphs per row. A module can contain a mixture of XY graphs and timelines if they are vertically stacked, but only XY graphs can be arranged in rows and columns.
TimeLineXDateTime TimeLineXDouble	Each timeline object represents a Gantt chart-style timeline that contains data point and time interval objects. Each TimeLineXDateTime object contains TimeLineDataPointXDateTime and/or TimeLineIntervalXDateTime objects. Each TimeLineXDouble object contains TimeLineDataPointXDouble and/or

TimeLineIntervalXDouble objects.

XYGraphXDateTime  
XYGraphXDouble

Each object represents an XY (time series) graph displayed within a graph module. XYGraphXDateTime objects contain XYGraphDataPointXDateTime objects. XYGraphXDouble objects contain XYGraphDataPointXDouble objects.

## 8 Systems Integration Options

To specify the data (content) and appearance of a DataMontage graphical display, you create a DataMontage graph container Java object and sub-objects by calling Java methods provided by the DataMontage Java API. You can also create a DataMontage graph container object by deserializing a DataMontage configuration file in XML format using the static *loadXML* method. You can serialize a DataMontage graph container object to an XML file by calling the container's *saveXML* method. For example, your Java web server application can create, configure, and serialize DataMontage objects to create configuration files that are then loaded by a Java applet running on a client computer that deserialize these files into DataMontage objects, as shown in Figure 3.

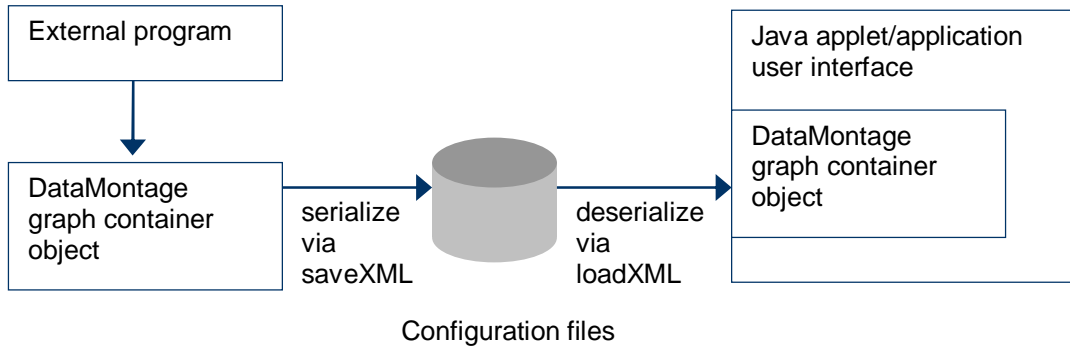


Figure 3 – Model objects simplify serialization and deserialization of DataMontage objects to/from configuration files

Programs written in a language other than Java can create configuration files directly by writing the XML configuration file directly, without calling DataMontage's Java methods. For example, a Visual Basic program can create a DataMontage configuration file to display data in Excel workbooks, or a SAS program can create a configuration file from data stored in SAS datasets.

You may find it useful to create graph templates as serialized DataMontage graph container objects as follows:

1. Create an XML configuration file that specifies a DataMontage container object and its sub-objects. You may find it useful to use the DataMontage Editor.
2. Use the DataMontage Viewer application to view the DataMontage display. Edit the configuration file until you are satisfied with the graph's appearance.

Your applet or application can use this template as follows:

1. Deserialize the XML configuration file to create a DataMontage graph container object,
2. Call Java methods provided by the DataMontage API to add or change data and appearance attributes as required by your applet or application, and
3. Display the DataMontage graph container within your applet or application or serialize it so that it can be loaded and displayed by another Java program (e.g., within a Java applet running within a web browser).

You can embed DataMontage graph containers within your Java application, applet, or web server application using the following integration options:

<div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">Custom Java applet or application</div> <div style="border: 1px solid black; padding: 5px;">DataMontage graph container objects</div>	<p>A custom applet or application calls Java methods provided by the DataMontage API to create, configure, and display DataMontage graph container objects.</p>
<div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">Java web server application</div> <div style="border: 1px solid black; padding: 5px; margin-top: 20px;">Applet with DataMontage container object</div> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">DataMontage XML configuration file</div> <p style="margin-top: 10px;">Web server app uses file I/O to write a configuration file in XML format</p> <p style="margin-top: 10px; text-align: center;">↑ Deserialize file into DataMontage container object</p>	<p>A web server application uses file I/O to generate a DataMontage configuration file in XML format. The standard DataMontage applet or a custom applet deserializes this file into a DataMontage graph container object and then displays it.</p>
<div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">Java web server application</div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">DataMontage graph container object</div> <div style="border: 1px solid black; padding: 5px; margin-top: 20px;">Applet with DataMontage container objects</div> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">DataMontage XML configuration file</div> <p style="margin-top: 10px;">Web server app serializes a container object into a configuration file</p> <p style="margin-top: 10px; text-align: center;">↑ Deserialize file into DataMontage container object</p>	<p>A web server application calls methods provided by the DataMontage API to configure a DataMontage graph container object and sub-objects. The web server application then serializes this graph container object to generate a DataMontage configuration file in XML format. The standard DataMontage applet or a custom applet deserializes this file into a DataMontage graph container object and then displays it.</p>
<div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">Application (C++, Visual Basic, SAS)</div> <div style="border: 1px solid black; padding: 5px; margin-top: 20px;">Application with DataMontage container object</div> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">DataMontage XML configuration file</div> <p style="margin-top: 10px;">Custom application writes a DataMontage configuration file in XML format via file I/O</p> <p style="margin-top: 10px; text-align: center;">↑ Deserialize file into DataMontage objects</p>	<p>A custom application written in any language (e.g., C++, Visual Basic, SAS) generates a DataMontage XML configuration file. The standard DataMontage application or a custom application deserializes this file into a DataMontage graph container object and then displays it.</p>

## 9 Custom User Interactions

You can customize the user interactions of a DataMontage graph container by creating a Java class that implements the *UserInteractions* interface in the *com.stottlerhenke.datamontage.interaction* package. This interface specifies a number of methods named *getRolloverText* that enable customization of the information displayed in the tool tip window. These methods accept as arguments:

- one or more DataMontage graphical objects that lie under the mouse cursor, such as an *XYGraph* and *XYGraphDataPoint*.
- a *Java.awt.Point* object that specifies the location of the mouse cursor within an *XYGraph* or *Timeline*, if the mouse cursor lies within one of these types of objects.

As the user's mouse moves over the graphical display, DataMontage calls the *getRolloverText* method with the most specific signature. For example, if the mouse cursor is over an *XYGraph* but not over any sub-object within the graph (e.g., a data point, reference value line, or reference region), DataMontage calls the method:

```
java.lang.String getRolloverText(XYGraph graph, java.awt.Point point)
```

If the mouse cursor is over an *XYGraphDataPoint* within an *XYGraph*, DataMontage calls the method:

```
java.lang.String getRolloverText(XYGraph graph, java.awt.Point point,
    XYGraphDataPoint dp)
```

The *UserInteractions* interface also enables you to add custom menu choices to the Montage popup menu by providing a number of methods named *getCustomMenu*. Different method signatures correspond to the different types of graphical objects. When the user presses the right mouse button, DataMontage calls every *getCustomMenu* method with a signature that matches the objects that lie under the mouse cursor. For example, if the user presses the right click button when the mouse cursor is over a *GraphModule*, DataMontage calls method:

```
java.util.Vector getCustomMenu(GraphModule module)
```

If the mouse cursor is also over a *Graph* object, DataMontage also calls method:

```
java.util.Vector getCustomMenu(Graph graph)
```

Your implementation of each *getCustomMenu* method should return a *Vector* of *JMenu* objects that specify the label and action of each menu choice. DataMontage combines all of the *JMenu* objects returned by each *getCustomMenu* method and adds a custom menu choice to the popup menu for each *JMenu* object. You can get the selected DataMontage sub-object (e.g., data point or interval) by calling the *GraphContainer.getSelectedObject* method. You can call the *GraphModule.getGraphContainer()* and the *Graph.getGraphModule* methods to get a handle to the graph container.

You can also specify actions to be carried out when the user presses and releases the left mouse button over an xy graph or timeline by overriding the default implementation of the *leftMouseReleased* method which has the following signature:

```
public void leftMouseReleased(Graph graph, Point point)
```

The *graph* parameter specifies the xy graph or timeline in which the mouse release event occurred, and the *point* parameter specifies the location of the mouse button release event in pixels. You can call the graph's *getObjectAt* method to determine whether the mouse button was released over a sub-object, such as a data point or interval, and operate on the sub-object.

You can provide a method that specifies custom initialization operations on the container. This method is called after the container is initialized. The signature of this method is:

```
public void initialize(GraphContainer container)
```

Within your *UserInteractions* methods, you might find it convenient to query information stored in the *applicationData* property of the container, module, graphs, or data points and/or intervals, to determine the object's identity and other information.

The *DefaultUserInteractions* class provides a default implementation of *UserInteractions* interface. This class implements the default rollover text behavior as described in the DataMontage User's Guide. You may find it convenient to implement custom user interactions by creating a class that extends *DefaultUserInteractions* class in order to inherit some methods and override others. The *UserInteractionsDemo* class provides a simple example implementation of the *UserInteractions* class. The Developer's Kit includes source code for both of these classes.

To instruct DataMontage to use a particular class to control user interactions when displaying a graph container, call the *GraphContainer.setUserInteractions* method. You can also configure a DataMontage graph container to use a custom user interactions class by setting the value of the XML *userInteractions* tag or by using the DataMontage Editor to set the value of the *userInteractions* property.

You may find it helpful to create a Java application development project, so you can step through the methods in this class when testing and debugging it. The DataMontage Developer's Kit includes source code for the Viewer application that you can use to create this project.

## 10 Developing and Deploying Your Application

The DataMontage run-time library is contained within a file named *DataMontage.jar* which is included in the DataMontage Developer's Kit software distribution. When developing a Java application using an integrated development environment (IDE) such as JBuilder or Eclipse, configure your project to include this jar file. If you do not use an IDE, place this jar file in the class path.

When deploying your Java application, place this jar file in the class path or invoke your Java application using the "-cp" switch (e.g., "java -cp <directories and/or zip and jar files>".) When developing or deploying a Java applet that uses the DataMontage library, set the CODEBASE and ARCHIVE attributes in the APPLET tag so that it can find file *DataMontage.jar*.

If your DataMontage container object invokes Java classes that implement customizations, use the -cp switch to specify the folder or JAR file(s) that contain those Java classes. When using the SQL Operations feature, include any JDBC library files in the class path that are needed to access SQL databases.

### 10.1 Embedding DataMontage within Client-Side Software

Your Java applet or application can embed and display DataMontage graph container objects as follows:

1. Create an empty DataMontage graph container object by calling a constructor method for the *GraphContainerXDateTime* or *GraphContainerXDouble* class.
2. Alternatively, you can use static methods *GraphContainerXDateTime.loadXML* or *GraphContainerXDouble.loadXML* to create an instance of a graph container that is already populated with sub-objects by de-serializing a DataMontage XML configuration file. This approach is useful for creating a DataMontage graph container template that is already populated with sub-objects. This approach can also be used to de-serialize data that has been serialized at execution time by a server program and then transferred over a network to a client-side program that displays the DataMontage container object.
3. Call Java methods provided by the DataMontage software library to specify the content and appearance of the DataMontage container object by creating DataMontage sub-objects and setting property values.

If you already carried out step 2 to create a graph container template, you can call API methods to populate the graph container.

4. Call the `initialize()` method of the DataMontage graph container (*GraphContainerXDateTime* or *GraphContainerXDouble*) object,
5. DataMontage graph container classes descend from the Java Swing *JPanel* class. DataMontage container objects may be embedded contained within a *Java.awt.Container* object. Display the DataMontage container object by calling the `setVisible` and/or `show` methods of the *Java.awt.Container* object. Either of these method calls will indirectly generate a call to the `repaint()` method of the DataMontage container object.
6. In general, if your application modifies the content or organization of an embedded DataMontage container, it should call `recalculateLayout()` instead of calling `repaint()` after making the changes. This ensures that DataMontage recalculates the position and dimensions of the subwindows within a DataMontage graph container. For example, changes to the lengths of y axis labels may change the widths of the graphs or timelines. However, if your program adds, deletes, or modifies data points in an XY graph that has specified (vs. automatically calculated) lower and upper X and Y limits, your program can redisplay the graph container by calling its `repaint()` method without calling the `recalculate()` method. Bypassing the call to `recalculateLayout()` enables you to add data in real time and redisplay the container more quickly.
7. Your program can call method *GraphContainerXDateTime.cleanup()* or *GraphContainerXDouble.cleanup()* after the graph container is no longer used to release resources (such as listeners) that were created to support the container. Data Montage provides additional signatures for the `cleanup()` method, so you can release resources associated with individual timelines and graphs that have been deleted from a *GraphContainer*.
8. The application programming interface (API) provided by these rendering classes is described in on-line documentation in Javadoc format that is included with the DataMontage Developer's Kit.

## 10.2 Embedding DataMontage within Server-Side Software

Your Java (web) server application can create DataMontage graph container objects and serialize them into XML configuration files as follows:

- Create an empty DataMontage graph container object by calling a constructor method for the *GraphContainerXDateTime* or *GraphContainerXDouble* class.
- Alternatively, you can use static methods *GraphContainerXDateTime.loadXML()* or *GraphContainerXDouble.loadXML()* to create an instance of a graph container that is already populated with sub-objects by deserializing a DataMontage XML configuration file. This approach is useful for creating a DataMontage graph container template that is already populated with sub-objects. This approach can also be used to deserialize data that has been serialized at execution time by a server program and then transferred over a network to a client-side program that displays the DataMontage container object.
- Call Java methods provided by the DataMontage API to specify the content and appearance of the DataMontage container object by creating DataMontage sub-objects and setting property values.
- Call the container-level `saveXML()` method to serialize the DataMontage graph container object to an XML configuration file. This file can be deserialized by another client-side or server-side program. For example, you can create a web application server that responds to requests by creating an HTML file with an embedded DataMontage applet that deserializes and displays the generated XML configuration file. Or, your server application can print the DataMontage graph container object using formats such as GIF, JPEG or Adobe Acrobat (PDF) for display by a client-side application or web browser.
- Call method *GraphContainerXDateTime.cleanup()* or *GraphContainerXDouble.cleanup()* after the graph container is no longer used to release resources such as listener objects that were created to support the container.

### 10.3 Using the DataMontage Applet Classes

The DataMontage library provides two built-in applet classes that provide wrappers around the top-level DataMontage graph container classes. The `GraphContainerXDateTimeApplet` class enables the display of data with datetime X values, and the `GraphContainerXDoubleApplet` class enables the display of data with double-precision X values. These classes load a configuration file, identified as a URL by an applet parameter named `URLConfiguration`. This URL can be a full URL, or it can be relative to the path specified by the `CODEBASE` attribute. The `URLConfiguration` applet parameter can also identify a Zip archive file (with `.zip` file extension) that contains the configuration file. In this case, the Zip file and the configuration file must share the same name, with different file extensions.

The following code shows an example applet tag that creates a `GraphContainerXDateTimeApplet` object. This example also shows the JAR files that must be specified by the `ARCHIVE` attribute.

```
<applet
  CODEBASE = "."
  CODE     = "com.stottlerhenke.datamontage.applet.GraphContainerXDateTimeApplet.class"
  ARCHIVE  = "DataMontage.jar"
  NAME     = "DataMontage Applet"
  WIDTH    = 600    HEIGHT   = 550
  HSPACE   = 0      VSPACE   = 0
  MAYSCRIPT = true>
  <param NAME = "URLConfiguration" VALUE = "cardiac_view.xml">
</applet>
```

Thus, you can create DataMontage displays and view them in a browser by creating an HTML file that includes an appropriate applet tag that references a DataMontage class, jar files, and a configuration file that specifies the content and format of the DataMontage graphical display.

The optional `defaultBaseURL` applet parameter resets the `defaultBaseURL` property of the DataMontage `GraphContainer`. The `URLPopup` or `URLNavigate` properties associated with DataMontage sub-objects specify the URLs of web pages that are displayed in a popup window or navigated towards and displayed in the browser window. If DataMontage is embedded within a Java applet, the HTML page that contains the applet must define several JavaScript functions. The DataMontage Developer's Kit provides this JavaScript code in file `DataMontageJavaScript.txt` in the `code_samples` folder.

### 10.4 Writing XML Configuration Files

You can write DataMontage configuration files directly without using model objects. This option enables you to create DataMontage XML files using a program written in any language, such as C++, PERL, Microsoft Visual Basic, or SAS. Configuration files use the eXtensible Meta Language (XML) metafile format. The syntax of these files is specified by XML Schema Definition (XSD) files which are included with the DataMontage Developer's Kit. In general, each XML complex type corresponds to a DataMontage class, and each XML element corresponds to a property of a DataMontage class.

### 10.5 Distributing DataMontage Run-Time Software Files

If you have purchased DataMontage run-time licenses, you may distribute run-time software files and install them onto single-user desktop computers or onto server computers. Files that may be distributed to DataMontage run-time licensees include:

- Software: `DataMontage.jar`
- End user Documentation: `DataMontage_2.3_users_guide.pdf`, `DataMontage_2.3_overview.pdf`

A *DataMontage Developer's License* is required to run the software in `DataMontageEditor.jar` or to compile software applications with the DataMontage software library. Use of the following files requires a *DataMontage Developer's License*:

- Software: `DataMontageEditor.jar`
- API documentation: files in folders *java-doc* and *xml\_schema*

## 11 Printing

You can use the DataMontage Viewer application to print graphs and create JPEG file output.

*DataMontagePrint.txt* and *DataMontagePrintDecorator.txt* in the *code\_samples* folder provide Java source code examples that print DataMontage graphs. Method *GraphContainer.createJPEG()* creates a JPEG file from a graph container. Command files *DataMontagePrintJpeg.bat* and *DataMontagePrint.bat* in the top-level installation folder provide examples of invoking these methods. These examples might be helpful if you are embedding DataMontage within another application.