RTView Core® User's Guide

Version 7.3

SL Corporation
871 Marlborough Ave, Suite 100A
Riverside, CA 92507 USA

Phone: 415.927.8400
Fax: 415.927.8401
Web: https://www.rtview.com

Preface

Welcome to the *RTView Core® User's Guide*.

Read this preface for an overview of the information provided in this guide and the documentation conventions used throughout, additional reading, and contact information. This preface includes the following sections:

About This Guide

The *RTView Core® User's Guide* describes how to install, configure and use RTView Core®.

Document Conventions

This guide uses the following standard set of typographical conventions.

Convention	Meaning
italics	Within text, new terms and emphasized words appear in italic typeface.
boldface	Within text, directory paths, file names, commands and GUI controls appear in bold typeface.

Courier	Code examples appear in Courier font:
	`amnesiac > enable`
	`amnesiac # configure terminal`

< >	Values that you specify appear in angle brackets: **interface <ipaddress>**

Additional Resources

This section describes resources that supplement the information in this guide. It includes the following information:

Release Notes

The following online file supplements the information in this user guide. It is available on the SL Technical Support site at http://www.sl.com/services/techsupport.shtml.

Documentation and Support Knowledge Base

For a complete list and the most current version of SL documentation, visit the SL Support Web site located at http://www.sl.com/services/support_rtviewdocs.shtml. The SL Knowledge Base is a database of known issues, how-to documents, system requirements, and common error messages. You can browse titles or search for keywords and strings. To access the SL Knowledge Base, log in to the SL Support site located at http://www.sl.com/services/techsupport.shtml.

Contacting SL

This section describes how to contact departments within SL.

Internet

You can learn about SL products at http://www.sl.com.

Technical Support

If you have problems installing, using, or replacing SL products, contact SL Support or your channel partner who provides support. To contact SL Support, open a trouble ticket by calling 415 927 8400 in the United States and Canada or +1 415 927 8400 outside the United States.

You can also go to http://www.sl.com/services/techsupport.shtml.

CHAPTER 1 Getting Started

This section contains the following:

Overview

RTView is an open business information delivery platform that is used by IT Managers to rapidly configure, deploy, maintain, and extend real-time and other critical enterprise information systems. This enables management and operational personnel to monitor the health and status of business operations. The features of this platform extensively reduce time, cost, and risk associated with developing these systems.

RTView includes the Display Builder and the Display Viewer Application. The Display Builder is used to quickly build customized displays with objects that connect easily to live and archived data. Displays can be deployed in the Display Viewer Application. RTView supports a variety of built in data sources in addition to custom data sources. See the "RTView Data Sources" section of this documentation for more information.

Note: You may not be licensed to run all RTView data sources.

Following "Setup and Registration", we suggest you begin by reviewing the Product Overview and completing the Quick Start Example before viewing the Sample Displays. Once you familiarize yourself with RTView, read through the documentation and complete the Examples for each component.

Product Overview

The "Product Overview" describes each component and explains how they interact within RTView.

Examples

These step-by-step guides will help you use each component more effectively. The "Quick Start Tutorial" will show you how to use the Display Builder, run the data simulators, attach objects to a variety of data sources, create drill down displays, and view your display in the Display Viewer Application.

Demos

See "RTView Demos" for how to utilize RTView in order to take full advantage of real-time dashboard displays of data.

CHAPTER 2 Setup and Registration

Note: Please review the "System Requirements" before attempting installation and setup.

This section includes:

- "Installation": If you have not already installed RTView, please do so now.
- "Setup": Your system environment must be setup properly to run RTView.
- "Registration": RTView requires a license to run.

Installation

RTView may be used as a stand-alone application or in conjunction with SL-GMS J/Developer. If you are using RTView in conjunction with J/Developer, first install J/Developer and then install RTView in the J/Developer installation directory.

UNIX or Windows

Create an RTView installation directory and extract your .zip file in that directory. You will need to know the exact name and location of your RTView installation directory when it is necessary to work from a command window. Once installation is complete, please refer to the "Setup" section for instructions on how to setup your environment in order to run RTView.

RTView_X.X.X.X.zip or **RTView_X.X.X.X_simple.zip***

*Created for users who only require application files, the **RTView_X.X.X.X_simple.zip** file is compatible with all RTView supported operating systems and excludes the following directories and files:

- demos
- custom
- docs
- servers

Setup

Following installation, you will need to setup your environment to run RTView. If necessary, contact your system administrator for assistance.

Depending on which data sources are licensed in your RTView application, additional setup may be required. For information on setup for your data source, refer to the "RTView Data Sources" section of this documentation.

Once setup is complete, please refer to the "Registration" section for instructions on how to obtain a license key.

Setup Environment

Append the following to the PATH environment variable:

Name	Description	Example
<java installation directory>\bin	The bin subdirectory of the Java installation subdirectory.	**c:\Program Files\Java\jdk1.7.0_51\bin**

Set RTV_HOME Locally or Globally

Locally

Use the following initialization scripts to set the **RTV_HOME** environment variable and append **RTV_HOME\bin** to the PATH environment variable.

Note: You must initialize each command or terminal window you open.

Windows

Select **Start-> Programs-> Accessories-> Command Prompt**, go to your installation directory, and type:

rtv_init

UNIX

csh	bsh
Open a terminal window, go to your installation directory, and type:	Open a terminal window, go to your installation directory, and type:
source rtv_init	**../rtv_init.ksh**

Globally

Alternatively, you may set **RTV_HOME** and append **RTV_HOME\bin** globally.

Name	Description	Example
RTV_HOME	RTView installation directory. Set "Locally" with rtv_init scripts. If you installed RTView using the Windows installer, this may already be set globally on your system.	**c:\gms**
RTV_HOME\bin (Append this to the PATH environment variable)	RTView subdirectory containing scripts and executables. Set "Locally" with rtv_init scripts.	**c:\gms\bin**

Include lanscan in PATH Environment Variable (HP users only)

If the **hostid** script is not available on your system, HP users must include the location of **lanscan** in their PATH environment variable. The standard location is **/usr/bin**.

RTV_USERPATH

Java options specified in **RTV_USERPATH** will be added to the classpath by all of the RTView run scripts.

Name	Description	Example
RTV_USERPATH	Java options. These will be passed into the java process from the RTView run scripts. This environment variable is optional.	**c:\mycustomjars\myjar.jar**

RTV_JAVAOPTS

Java options specified in **RTV_JAVAOPTS** will be used by all of the RTView run scripts.

Name	Description	Example
RTV_JAVAOPTS	Java options. These will be passed into the java process from the RTView run scripts. This environment variable is optional.	-verbose
	Specify the name of a directory, including path, for configuration files to be saved to and read by the Display Builder, Display Viewer, Display Server, Data Server and Historian. -Dcom.sl.rtview.optionsFileDir=(directory) This option is useful when you want to run multiple RTView applications from the same project directory, but with different configuration files. Let's suppose you want to run both the Display Viewer Application and the Display Server from the same project directory, but for the Display Server you want a direct data connection and for the Display Viewer you want to use the Data Server. Since both applications require an **OPTIONS.ini** file to determine whether to use direct connect or served data, a separate **OPTIONS.ini** file is necessary for each application. To resolve this issue, and avoid having separate project directories for each application, you can create a subdirectory for the Display Viewer **OPTIONS.ini** file and specify that subdirectory in the **RTV_JAVAOPTS** environment variable. Then, as usual, you may run the Display Server and the Display Viewer from your project directory, but the **OPTIONS.ini** file for the Display Viewer will be located in the directory you specified.	-Dcom.sl.rtview.optionsFileDir=MyOptionsFilesDir

If no alternate directory is specified and configuration files are not found in your project directory (i.e. the directory where current application is running), then RTView will search under lib in your installation directory.	
By default, all RTView applications include timestamps in the console and log file output. To disable timestamps, include the following jvm option: -Dcom.sl.rtview.logTime=	-Dcom.sl.rtview.logTime=false

Demo Server, Data Server, Display Server and JSP Example

To use the Demo Server, Data Server, Display Server or "Java Server Pages" example you must also set the following variables:

Name	Description	Example
CATALINA_HOME	Web server installation directory.	C:\Tomcat
JAVA_HOME	JDK1.7.0_51+ installation directory. Only required if you will be using the "RTView Demo Server".	C:\Program Files\Java\jdk1.7.0_51
RTV_DEMOSERVER	The location of the Demo Server in your RTView installation. This variable may be setup in the Windows installer. It is also set to the correct location when you run the **rtv_init** script to initialize a command window (see "Initializing a Command Prompt or Terminal Window" for more information). Only required if you will be using the "RTView Demo Server".	C:\Program Files\ERTV\servers\apache_tomcat-5.5.17_sl

Registration

Application Registration

In order to register for a license, your system environment must be properly setup. If you have not already reviewed the "Setup" section please do so now. Following registration, we strongly suggest that you complete the "Quick Start Tutorial" before getting started.

1. In an initialized command/terminal window (see "Initializing a Command Prompt or Terminal Window"), type:

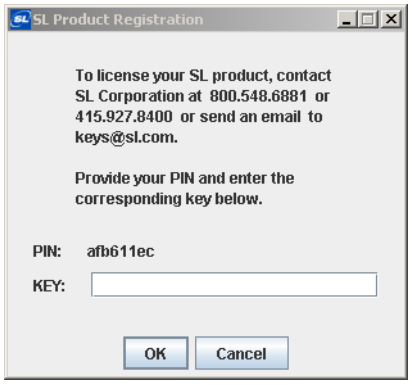
   run_gmsregister

Note: For machines that are headless (i.e. without monitor, keyboard or mouse), use the **–nogui** option with the **run_gmsregister** script (i.e. **run_gmsregister -nogui**).

Registration Dialog

A Registration dialog will open that displays your PIN and prompts you to enter a license key. To request a key, contact SL Corporation at 800.548.6881 or 415.927.8400 or send an email to **keys@sl.com** with the Subject line **RTView**. Copy and paste your PIN into the email and in return you will receive a corresponding numeric key.

Copy and paste the key you receive into the **Registration** dialog.



For future access to your license key a file named **KEYS** will be stored in the **lib** directory, which is located in your installation directory.

Note: Your license key enables you to run RTView applications on a single host.

For licensing systems that do not have a display, use the **–nogui** command line option (**run_gmsregister -nogui**) to print the PIN to the console. Follow the instructions above for getting a license KEY, then use the **-writekey** option (e.g. **run_gmsregister -nogui -writekey:01234567890123456789012345678901**) to enter a key on the command line and have it written to the **KEYS** file.

Note: The **-writekey** option is only valid when used in conjunction with the **-nogui** option.

CHAPTER 3 System Requirements

Please refer to the **README_sysreq.txt** file from your product installation. A copy of this file is also available on the product download page.

CHAPTER 4 RTView Upgrades

As a rule, each release of RTView is designed to be upward compatible. This means that any RTView application constructed with a set of RTView files and configuration parameters should function without modification in subsequent releases. Any exceptions to that rule are detailed below.

This section describes the following:

Note: There are no upgrade steps required when upgrading from version 6.1.0 to the current version or to any version in between.

Upgrading to 6.1.0

Alert Persistence

The alert persistence feature has been enhanced for easier configuration. Customers persisting alert data will need to do the following:

- Add the **Row Update Time** column to the alert persistence database table. The database schemas in **RTV\dbconfig\alert_persist** have been updated to include this column. See "Viewing Alerts" and "Alert Persistence" for more information.

Upgrading to 6.0.1

Alerts / Self Service Alerts

Customers using the **Alert Action Audit Table** and/or the **Self Service Audit Table** will need to do the following:

- Modify table schemas so that the **TIME_STAMP** column uses the correct timestamp type for their database.

  For example, in MySQL the following SQL statement will modify the TIME_STAMP column:

  ```
  ALTER TABLE AUDIT_TABLE ALTER TIME_STAMP DATETIME
  ```

- Update database to contain the **ALERT_NAME** (string) and **ALERT_INDEX** (string) columns.

Alert Persistence

The alert persistence feature has been enhanced for easier configuration. Customers persisting alert data will need to do the following:

- In your existing persistence table, rename the **AlertEngineName** column to **AgentName**, or delete the table and select the **Create Database Table If Not Found** option on the **Alerts>Alert Persistence** tab of the **Application Options** dialog.
- On the **Alerts>Alert Persistence** tab of the **Application Options** dialog, fill in the **Alert Engine Name** with the same value previously used for **Agent Name**.
- If you are only using the Historian to persist alerts, you no longer need to run the Historian.
- Add the **Row Update Time** column to the alert persistence database table. The database schemas in **RTV\dbconfig\alert_persist** have been updated to include this column. See "Viewing Alerts" and "Alert Persistence" for more information.

CMDB

Customers using the CMDB from a previous release need to do the following:

- Create a "Relationship Table".
- For each row in the Container Table and Metric Source Table where a parent value is listed, add a row to the Relationship Table where ID1 is the value in the Parent column and ID2 is the value in the Name column.
- Remove the **Parent** column from the Container Table and Metric Source Table.
- Specify the name of the Relationship Table in the **Application Option**s dialog on the "CMDB Tables Tab".

Upgrading from 5.9.1 to 6.0.0

Alert Persistence

Customers using the Alert Persistence feature in 5.9.1 will need to modify their database table to include a String column named **Alert Index Values** immediately following the Count column. The schemas in **RTV\dbconfig\alert_persist** have been updated to include this column.

Cache Data Source

If upgrading from an RTView 5.9.1 installation in which the Historian's **persistCaches** feature was used to persist the history table of a cache in a database table, then it may be necessary to edit the schema of that database table to remove the **Expired** column (if it exists).

Upgrading from 5.9.0 to 6.0.0

Self Service Alerts

Customers using Self Service Alerts from a previous release need to do the following:

- Modify the Alert Settings Table in your database to add two String columns named **INDEXTYPE** and **ALERTINDEX** immediately following **ALERTNAME** column. The value for all existing rows in your database table for these two columns must be Default. Add a Boolean column named **USEINDEX** after the **ENABLED**. The value for all existing rows in your database table for this column must be true. Change the primary key for this table from **ALERTNAME** to **ALERTNAME** and **ALERTINDEX**.
- Modify the Audit Table in your database to add a String columns named INDEXTYPE immediately following **ALERTNAME** column. The value for all existing rows in your database table for the **INDEXTYPE** and **ALERTINDEX** columns must be Default. Add a Boolean column named **USEINDEX** after the **ENABLED**. The value for all existing rows in your database table for this column must be true.

Self Service Alerts Demo

Customers using the Self Service Alerts Demo from a previous release (either stand-alone or integrated into their application) need to do the following:

- Modify their database tables as detailed above.
- The Alert Administration screen has been enhanced to show the description of the selected alert. It is not required that you add descriptions to your alerts when you upgrade, but if you do not this field will be blank. To add a description to an alert, fill in the description property on the alert definition.
- The Alert Administration has been enhanced to check the role from the RTView login. If the login is disabled, the **admin** role is used, otherwise the logged in role is used. Administrative actions such as adding, modifying and setting alert thresholds are disabled if you are not logged in as **admin**. You may need to modify the roles for your users to include **admin** if they require access to these actions.
- To deploy in the Thin Client and use the built-in help files, you need to include the **selfservicealerts\docs** directory in the .war file for your thin client application and include **selfservicealerts\ssa_displays.html** in the application directory. See the "Customization Options" section for more information on modifying and/or deploying the help files.

An "Alert Action Audit Trail" display has been added to the Self Service Alerts Demo. Upgrading customers have two options:

- Update their Self Service Alerts database to include the new "Alert Action Audit Trail" display as defined in the Self Service Alerts Demo.

  OR

- Disable action auditing on the "Alerts Tab" of the **Application Options** dialog and remove the "Alert Action Audit Trail" display from the application.

CHAPTER 5 Product Overview

RTView provides the ability to access real-time and archived data from multiple data sources and EAI messaging infrastructures. RTView reads data directly from the source, which eliminates the need for intermediate data servers, reduces complexity, and maximizes performance and scalability. Flexible deployment options allow you to embed displays in existing Java applications or enterprise portals as well as distribute displays throughout the enterprise as browser-based images that update automatically.

RTView includes the Display Builder which is used to quickly build customized displays with objects that connect easily to live and archived data. Displays can be deployed via application, rich client browser or thin client browser. RTView supports a variety of built in data sources in addition to custom data sources. See the **Data Sources** section of this documentation for more information.

Note: You may not be licensed to run all RTView data sources.

This section describes the following:

Display Builder

The Display Builder features point-and-click palettes of graphic objects, including meters, labels, tables, graphs and scales. Within the Display Builder it is possible to perform calculations on data (e.g.: sum of all values in a table column) and configure objects to display data and drill down to more detailed displays. Objects can also be configured to execute commands such as sending an email, issuing an SNMP trap or data source specific commands. To minimize the number of displays that need to be configured, a single display can be reused—without any edits—to show data from a number of different sources depending on values passed in when the display is viewed. See "Building Displays" for more information.

Deployment

RTView can be deployed via application, rich client browser or thin client browser, connecting to the backend data either directly or through the Data Server which caches and federates the data. Since RTView is a portable delivery platform, all displays, including graphical elements, data attachments, drill downs, functions, substitutions and security settings, are portable to any deployment option - without re-engineering. This facilitates implementation and rollout since development, testing, and production systems can use different deployment technologies without significant porting costs. If situations change in your organization and you would like to choose a different deployment option, you can readily do so.

SL keeps current on the latest information delivery technologies, such as browser and portal options, scalable data distribution, application servers and security. This ensures that the most suitable option for your enterprise remains deployable with minimal costs. See "Deployment" for more information.

Data Server

The Data Server uses EII and XML technologies to gather, federate and distribute information from disparate data sources based on information currently in demand. It also caches the data so that multiple demands are delivered to any number of clients - without need of subsequent data queries. These important factors greatly enhance processing speed. Because the Data Server can exist behind firewalls, it also greatly simplifies and strengthens the secured delivery of information to clients beyond the firewall. See "Data Server" for more information.

Historian

With the Historian you can archive data to any database using JDBC. The Historian enables the comparison of live data to historical data for trend analysis or comparison to other metrics. See "Historian" for more information.

Alerts

RTView features a real-time alert engine that enables management and operational personnel to monitor the health and status of business operations. The alert engine can monitor conditions and perform automated actions from any available RTView data source. Alert definitions can include thresholds, severity, notification policies and automated actions, such as email, system commands or data source specific commands. The RTView alert engine can load any number of alert definitions, and any number of customized dashboards can be created to view alert status, filter alerts, use alerts as drill down navigation for analysis and corrective action, or to interactively change alert status such as alert acknowledgment. See "Alerts" for more information.

Customization

RTView can be customized to meet your specific needs:

- Graphic objects and palettes can be created using SL-GMS J/Developer™
- Easily extensible user interface supports the creation of custom menu items, toolbar buttons and dialogs
- In addition to packaged data sources, customize to read data directly from any source

Contact SL Technical Support at **800.548.6881** or **415.927.8400** for more information on customizing your RTView package.

CHAPTER 6 Role-based Security

RTView offers role based security that allows you to limit access to displays based on the user's role (see "Roles and Displays" for more information). To use this security feature, you must define the users that can access RTView and assign each one a password and one or more associated roles. The role definitions allow you to specify which displays users can access. User and role definitions also support substitutions. In addition, the user name and password from the RTView login can also be passed into data sources that have the **Use Client Credentials** option enabled. See "Configuration" for more information.

Note: Some data sources do not support this feature. For information on Application Options for your data source, refer to the **Data Sources** section of this documentation.

Role-based security is available in the following RTView components:

- Display Builder
- Configuration Utility
- Display Viewer Application
- Display Server

This feature is intended to reduce the possibility of user information being accessed unintentionally, but does not secure displays against users with malicious intent.

This section includes the following:

Login

By default, login for the following RTView components is disabled:

- Display Builder
- Configuration Utility
- Display Viewer Application
- Display Server

If login is enabled, the default user name and password are:

User Name: **admin**

Password: **admin**

Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role.

Set up Login

RTView supports single sign-on. By securely passing on authenticated user information, single sign-on allows users to login once and gain access to all the resources to which they have permission. See "Command Line Options: Display Builder and Display Viewer" to learn about the parameters necessary (i.e.: rtvuser, rtvpass, rtvrole and rtvsign) to setup the Display Builder and the Display Viewer, and see "Display Server" for a Thin Client Browser deployment. The Display Server also supports single sign-on through the servlet container or custom JavaScript. See "Display Server Single Sign-On" for more information.

Enable Login

1. Open an initialized Windows command window or Unix terminal window (see "Initializing a Command Prompt or Terminal Window" for more information about initializing a command/terminal window), and type:

   run_builder -login

2. Login as admin using default user name and password.

   User Name: **admin**

   Password: **admin**

3. Select **Tools>Options** and click on the **Security** tab to configure and save your login options.

Note: To disable security, login in as admin and deselect **Login Enabled** on the **Security** tab.

Set up Roles

The login validates the user name and password against the defined users. Each user must be assigned at least one role. If the user has multiple roles, a drop down menu of roles will be added to the dialog once the password has been validated. The role definitions control access to displays. In addition, the user name and password from the RTView login can also be passed into data sources that have the Use Client Credentials option enabled. See "Configuration" and "Roles and Displays" for more information.

Note: Some data sources do not support this feature. For information on Application Options for your data source, refer to the "RTView Data Sources" section of this documentation.

Admin Role

A "Security Tab" is added to the **Application Options** dialog if you are logged in with the admin role that allows you to disable the login. This dialog also allows you to configure the location of the user and role definition files. You cannot disable the login from the **Security** tab for the Display Server. This must be configured in the "Display Servlet".

Roles and Displays

If a display is not allowed for a role, the behavior is slightly different according to the situation:

Display Scenario	Behavior
Tree	If the display is in a tree node definition, that node (and any child nodes) will not be added to the tree. There will be no error message. This means that the display for the top node of the tree must always be allowed for all roles.
Tab	If the display is in a tab definition, the tab will not be added to the window and there will be no error message.
GridPanel BorderPanel CardPanel	If the display is in a GridPanel, BorderLayout, or CardPanel, it will not be loaded into the layout and there will be no error message.
File->Open Command Line	If the specified display is not allowed, we will open a blank display and popup an error.
Drill Down	If the display you are drilling down to is not allowed, a dialog will popup with an error and the drill down will not be performed.

Configuration

In order to setup security for your application, you must define your users and roles.

User Definitions

Each user definition must include a password, one or more role definitions and optional substitutions. You may define your users either in an XML file or by implementing a subclass of the **GmsCustomUserManager**. See "Custom User Manager" for more information.

Defining Users in an XML File

By default, RTView will look for user definitions in a file named **users.xml** in the directory where your application started. You may specify another file name or path for your user definition file in the "Security Tab" of the Display Builder **Application Options** dialog.

The user definition file must be an XML file and must start with the following:

```
<?xml version="1.0"?>
<users xmlns="www.sl.com" >
```

The user definition file must end with the following:

```
</users>
```

The following tags are supported:

User	**name**	The user name.
	password	The password for the user
	role	The role(s) for this user. You may assign multiple roles per user. **Note:** The role must have a corresponding role definition. Each user must have at least one role. See "Role Definitions" for more information.
	sub	Set initial substitutions for this user, specifying name and value. In addition, RTView automatically defines the following substitution for each user when the application is started with the login dialog: **$rtvuser** - user's login name Substitutions are optional and must use the following syntax: **<sub name="$sub1" value="value1" />** **<sub name="$sub2" value="value2" />**

Examples

User definition:

```
<?xml version="1.0"?>
<users xmlns="www.sl.com" >
        <user>
                <name>user name</name>
                <password>user password</password>
               <sub name="$sub1" value="value1" />
                <sub name="$sub2" value="value2" />
                <role>user role1</role>
                <role>user role2</role>
        </user>
</users>
```

When creating your users definition file, the passwords can be entered in plain text. You can then re-save your user file with the passwords encrypted by using the -saveusers command line option with the Display Builder, Display Viewer.

Note: This option will only work if you are logged in as the admin role.

Role Definitions

Use role definitions to permit and deny access to displays based on a user's role. Specify each displays you wish to permit or deny in the include and exclude lists (respectively), and optionally specify substitutions. The displays that are on the role definition exclude list are removed when a user assigned to that role logs in.

Note: By default, all displays are excluded. You may define your roles either in an XML file or by implementing a subclass of the **GmsCustomRoleManager**. See "Custom Role Manager" for more information.

Defining Roles in an XML File

By default, RTView will look for role definitions in a file named **roles.xml** in the directory where your application started. You may specify another file name or path for your role definition file in the "Security Tab" of the Display Builder **Application Options** dialog.

The role definition file must be an XML file and must start with the following:

```
<?xml version="1.0"?>
<roles xmlns="www.sl.com" >
```

The role definition file must end with the following:

```
</roles>
```

The following tags are supported.

Role	**name**	The role name.
	displays--include	The name of a display to include for this role. * is supported as a wildcard character, but only one * per display name is allowed. Specify ALL for the display name to include all displays.
	displays--exclude	The name of a display to exclude for this role. * is supported as a wildcard character, but only one * per display name is allowed.
	sub	Set initial substitutions for this role, specifying name and value. In addition, RTView automatically defines the following substitution for each role when the application is started with the login dialog: **$rtvrole** - user's role Substitutions are optional and must use the following syntax: **<sub name="rolesub1" value="value1" />** **<sub name="rolesub2" value="value2" />**

Examples

Role definition:

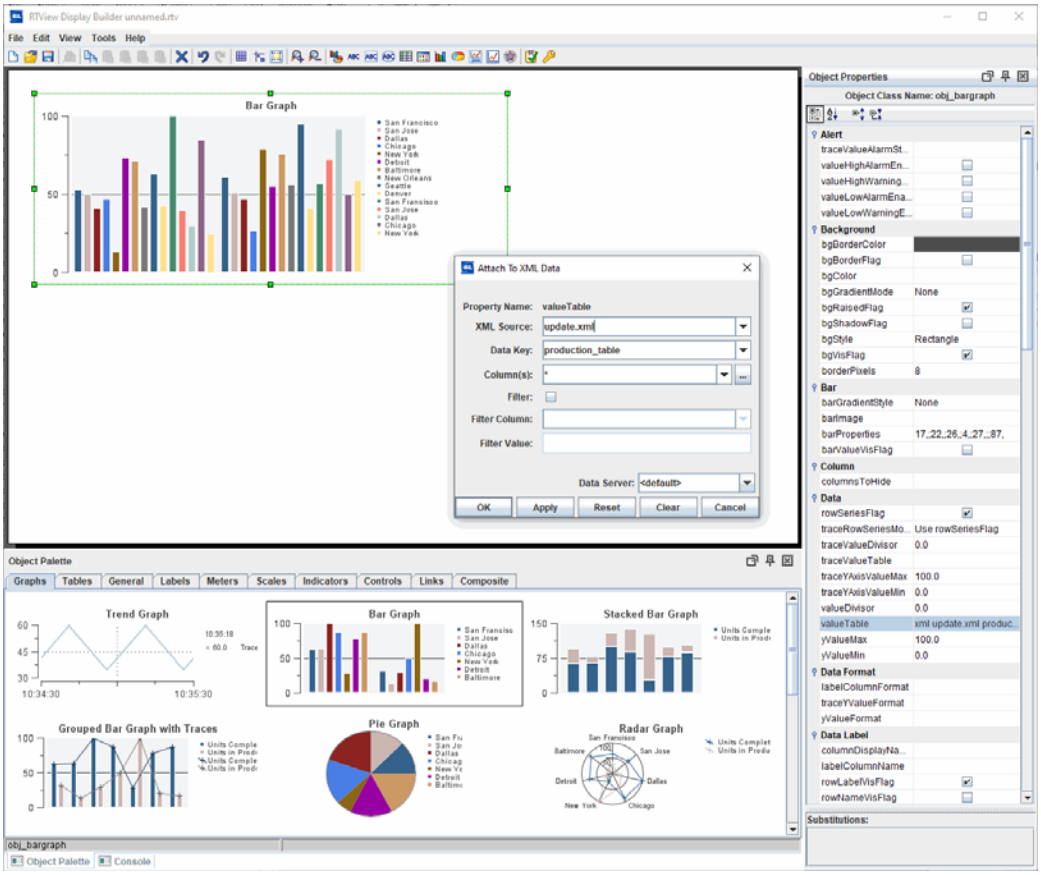
```
<?xml version="1.0"?>
<roles xmlns="www.sl.com" >
        <role>
                <name>operator</name>
                <displays>
                        <include>ALL</include>
                        <exclude>admin*.rtv</exclude>
                        <sub name="rolesub1" value="value1"/>
                        <sub name="rolesub2" value="value2"/>
                </displays>
        </role>
</roles>
```

The displays listed for each role are processed in reverse order, so the last display listed takes precedence over all others. So, in the example above, displays starting with admin are excluded, but all other displays are included. If the lines were reversed, all displays would be included, even those starting with admin.

CHAPTER 7 Building Displays

With the Display Builder you can quickly create flexible, customized dashboards, animated by your data, that provide at-a-glance access to critical business operation data. Using a point-and-click palette, select from a variety of graphic objects and attach them to live and archived "RTView Data Sources".

Note: You may not be licensed to run all RTView data sources. Choose from a wide selection of meters, gauges, tables, graphs and scales. Once you have configured your display, you can run it in the Display Viewer as a stand-alone Java application.



Working Area	Located in the upper-left portion of the Display Builder, the Working Area enables you to quickly build customized displays with objects that connect easily to live and archived data.
Object Properties Window	Located on the right side of the Display Builder, the Object Properties window lists available attributes of the selected object and lets you edit property values.
Object Palette	Located just below the Working Area, the Object Palette features point-and-click access to graphic objects. Refer to the Using the Object Palette section for details.
Console	Located just below the Object Palette, select the Console tab to view error messages and debug information as you develop your display. By default, timestamps are displayed in the Console window. Refer to **Setup>**"RTV_JAVAOPTS" for details on how to disable timestamps.
Toolbar	Located above the Working Area, the Toolbar gives you access to a wide variety of objects and functions.

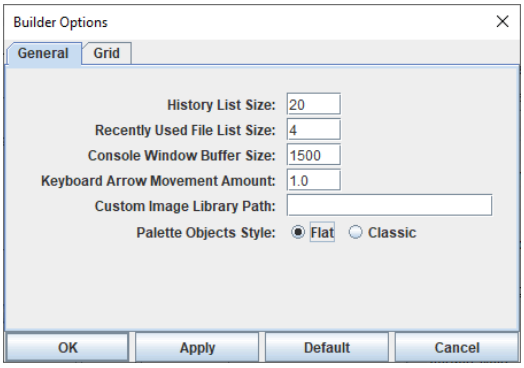
This section includes the following:

Builder Options

You can manage settings for the Display Builder in the **Builder Options** dialog. Settings shown below are default settings. The values set in this dialog are automatically restored on application startup and saved on application exit.

Select **Tools>Builder Options** to modify the following settings:

General Tab



Field Name	Description
History List Size	Specify number of consecutive actions (from 0 to 1000) to store in the History Undo and Redo lists. Default is 20.
Recently Used File List Size	Specify number of recently viewed files (from 0 to 9) to list in the File menu. Default is 4.
Console Window Buffer Size	Specify number of lines of text (from 0 to 5000) to display in the Console window. Default is 1500.
Keyboard Arrow Movement Amount	Specify number of pixels (from 1.0 to 100.0) to move an object when using the keyboard arrows. Default is 1.0.
Custom Image Library Path	Specify the location of your image files to create a custom image library. The custom image library enables you to make your own images available in the **Select Image** dialog. Before entering the path in the **Custom Image Library Path** field, you must place your images in a **.jar** file and add it to the RTV_USERPATH environment variable. **Note:** The images must be in a directory and not at the top level of the .jar file. They can be organized into subdirectories of one top level directory.

After you have prepared the **.jar** file as described above, specify the location of the directory containing your images in the **Custom Image Library Path** field.

For example, suppose you have a .jar file with the following directory structure:

com/mycompany/Images

com/mycompany/Images/Blue Images

com/mycompany/Images/Red Images

com/mycompany/Images/Green Images

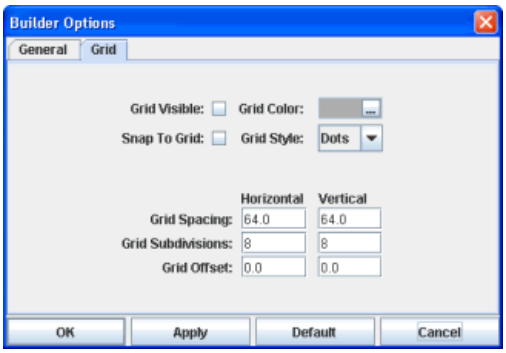
In the **Custom Image Library Path** field you would enter **com/mycompany/Images** to add a directory named Images to the tree in the Select Image dialog. The **Images** directory will have three subdirectories: **Blue Images**, **Red Images**, and **Green Images**.

Note: Only directories containing images will be added to the Select Image dialog.

To access your custom image library, edit any property that allows you to set an image on an object (e.g. **image**, **barImage** and **filterProperties** properties) or select **File>Background Properties** and edit the Image Name field.

Palette Objects Style Choose the Flat or Classic style for object palettes and default background image. This does not effect objects or the background already in a display. Default is Flat.

Grid Tab



Field Name	Description
Grid Visible	Select to display the grid.
Snap To Grid	Select to snap objects to the grid, this controls alignment when moving or resizing an object. It is not necessary for the grid to be visible to utilize this setting.
Grid Color	Select grid color. Click on the ellipses button ... and choose a color from the palette.
Grid Style	Select grid style: Dots or Lines. Default is Dots.
Grid Spacing (Horizontal and Vertical)	Specify number of pixels between major divisions in the grid. Valid values are positive numbers. Default is 64.0.
Grid Subdivisions (Horizontal and Vertical)	Specify number of subdivisions between each major division. Valid values are positive integers. Default value is 8.
Grid Offset (Horizontal and Vertical)	Specify number of pixels to offset grid lines. Values may be negative. Default is 0.0.

The following describes the **Builder Options** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Default	Resets all fields to default values (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied.

Running the Display Builder

In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"), type:

run_builder

By default, the Display Builder does not require a login. Login can be enabled at setup to support role based security. The default user name and password are:

User Name: **admin**

Password: **admin**

Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role.

RTV_JAVAOPTS

Java options specified in "RTV_JAVAOPTS" will be used by the **run_builder** scripts.

Using the Object Palette

This section contains the following:

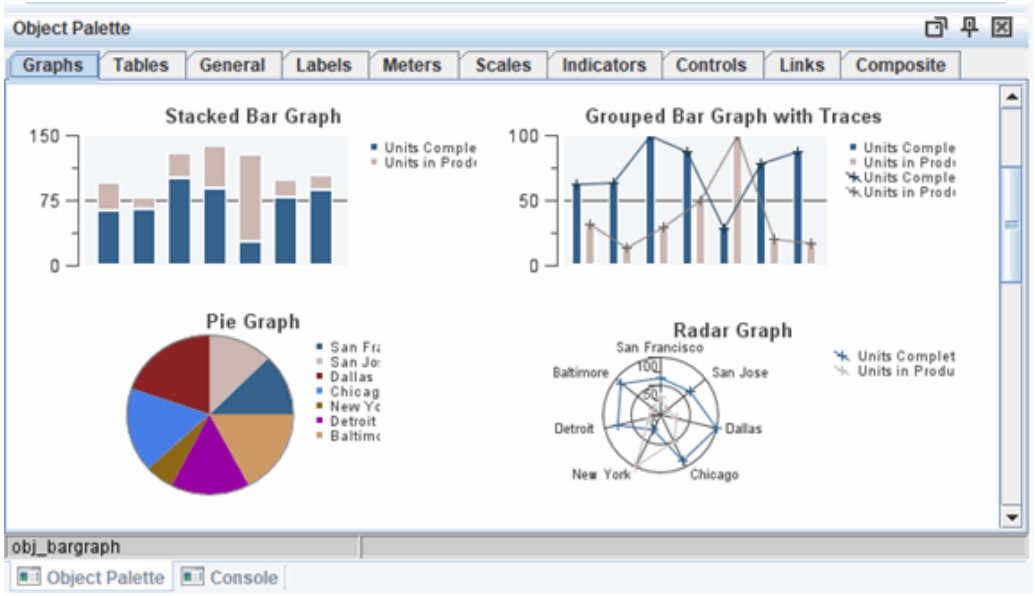
Add/Edit Objects

RTView features a wide variety of objects accessible through the Object Palette or from the Add Objects toolbar at the top of the Display Builder.

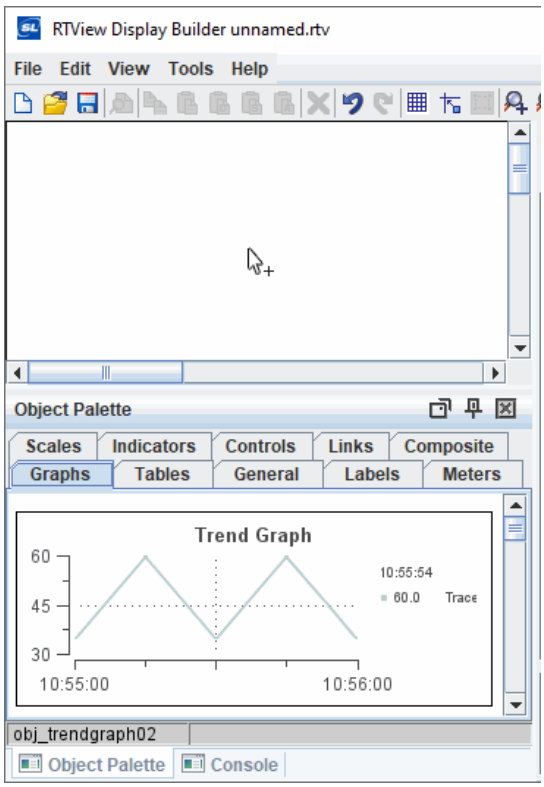
Adding an Object

Open the Object Palette by clicking on the Open Object Palette button  or by selecting **Edit>Add**.

Choose a tab from the Object Palette and click on an object. When an object is selected, a black border outlines the object and the class name of that object appears in the status bar at the bottom left of the palette.



Move the cursor into the Working Area. A + symbol appears next to the cursor, indicating that the application is in Add mode.

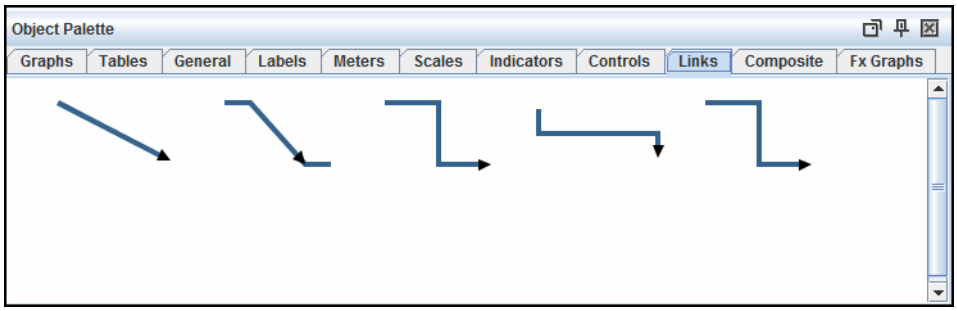


Click in the Working Area to place the object. To cancel Add mode, press the <Esc> key.

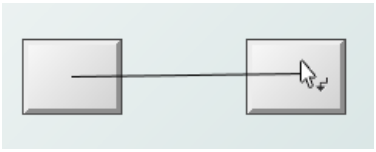
For more detailed information on each type of object, see "Object Descriptions".

Connecting Objects (applies to Links only)

With two objects in your Working Area, choose the Links tab from the Object Palette and click on a link. A rectangle appears on the link when it is selected and the class name of the link appears in the status bar at the bottom left of the palette.



Move the cursor into the Working Area. An arrow appears next to the cursor, indicating that the application is in Connect mode.



Click on the two objects in your display to connect them. To cancel Connect mode, press the **<Esc>** key.

Working with Objects

Set Location

Set the position in pixels of the objects in the display area with the **objX** and **objY** properties.

Move Objects

Click-and-drag to move objects within the Working Area or use the keyboard arrows to nudge objects. Select **Tools>**"Builder Options" to set the **Keyboard Arrow Movement Amount**. Click on the **Grid** button in the toolbar to toggle grid visibility. When the **Snap to Grid** button is selected, objects align to the grid while being dragged. See the **Grid** tab in the **Builder Options** dialog for additional settings.

Select Objects

Hold down **<Ctrl>** or **<Shift>** on your keyboard to select or deselect objects in the Working Area or click-and-drag to select multiple objects.

Note: The first object selected and objects not entirely included in the Select by Extent rectangle will not be selected.

Select by Extent replaces any current selection, unless you hold down **<Ctrl>** or **<Shift>** on your keyboard while performing the operation. You can also click on the **Select by Extent** button in the toolbar to enter Select by Extent mode. Select **Edit>Select All** (or **Ctrl-A**) to select all objects.

Note: When selecting multiple objects, object properties and scaling only apply to the last object selected.

For a list of all objects currently in the Working Area, select **Tools>Object List**. Within the Object List hold down the **<Ctrl>** or **<Shift>** key to select or deselect objects in the list, which will be simultaneously selected in the Working Area.

Copy and Paste Objects

Select one or more objects from the Working Area and click the Copy button  in the toolbar (or right-click on an object). Then click the Paste button  . This will put you in Add or Connect mode depending on the object you have selected. If you are in Add mode, click in the Working Area to paste the object. If you are in Connect mode, click on two objects in your display to paste the link. When copying and pasting nodes with their links, select all nodes connected to the links in order to include them.

Copy and paste are also available via keyboard shortcuts:

- **Ctrl-C** to Copy
- **Ctrl-V** to Paste

Scale Objects

To resize an object, click and drag one of the green selection handles on the object.

Order Objects

Right-click on an object and choose **Order** (or select **Edit>Order**), then select **Move to Front** or **Move to Back** to move the selected object in front of or behind other objects in the Working Area.

Note: Objects are always drawn in front of links.

Align Objects

Hold down the **<Ctrl>** or **<Shift>** key and select at least two objects. Right-click on an object and choose **Align** from the popup menu (or select **Edit>Align**) and one of the following options. The alignment is based on the first object you select.

- **Top** - To align the tops of all objects against the first object selected.
- **Bottom** - To align the bottoms of all objects against the first object selected.
- **Left** - To align the left sides of all objects against the first object selected.
- **Right** - To align the right sides of all objects against the first object selected.
- **Center Horizontal** - To align the objects along their horizontal centers.
- **Center Vertical** - To align the objects along their vertical centers.

Distribute Objects

Hold down the **<Ctrl>** or **<Shift>** key and select at least two objects. Right-click on an object and choose **Distribute** from the popup menu (or, select **Edit>Distribute**) and one of the following options. The alignment is based on the first object you select.

- **Horizontally** - To put equal amounts of horizontal space between the objects without changing their vertical location.
- **Vertically** - To put equal amounts of vertical space between the objects without changing their horizontal location.

Preview Control Objects

Control objects are not active in the Display Builder main editing window. This enables you to setup a control objects without activating their associated action commands. To preview a control object in the Display Builder, save your file and click the Preview button  in the toolbar. Your display opens in a Preview Window that allows you to activate control objects. The Preview button becomes disabled when you edit your display. Save the display again to enable it.

Delete Objects

Click on the object and click the Delete button  in the toolbar or press **<Delete>** on your keyboard to remove an object from the Working Area.

Edit Object Properties

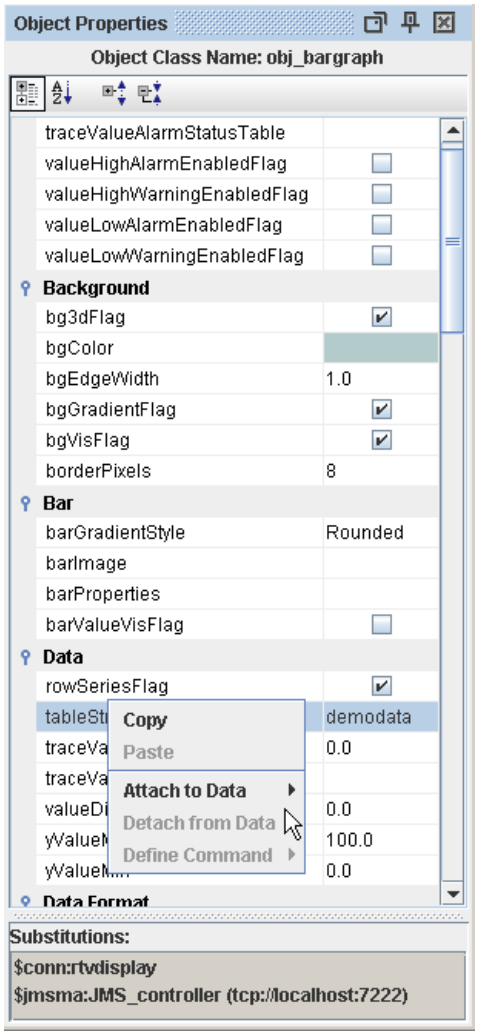
See the "Object Properties" section for information on how to edit objects from the **Object Properties** window.

Undo / Redo

Click the Undo / Redo button  in the toolbar to cancel or reapply up to one thousand consecutive actions. Select **Tools>**"Builder Options" to set the **History List Size**. Default setting is 20 consecutive actions.

Object Properties

RTView features a wide variety of objects. The **Object Properties** window lists attributes of the selected object and lets you edit property values. (See "Object Descriptions" section for descriptions of each object.)



To open the **Object Properties** window, select an object and click the **Object Properties** button on the toolbar. In the **Object Properties** window, you can view and/or edit the property values of the object selected in the Working Area.

Editing Property Values

Property Names listed in the first column cannot be changed. Property Values, listed in the second column, can be set to static values or attached to dynamic data.

- Blue text signifies that a Property Value is static and cannot be attached to a dynamic data source.
- Green text signifies that a Property Value is currently attached to a dynamic data source and therefore it is no longer possible to edit this value from the Object Properties window. See the **Attach to Data** section specific to your data source for information on attaching dynamic data.

To remove a data attachment and resume editing capabilities from the Object Properties window, right-click on the Property Name (e.g., value) and select **Detach from Data** from the popup menu. An object property has been detached from the data source when the Property Name and Value are no longer green.

Copying Property Values

Copying and pasting makes it easy to transfer property values from one object to another.

There are two options for copying object properties:

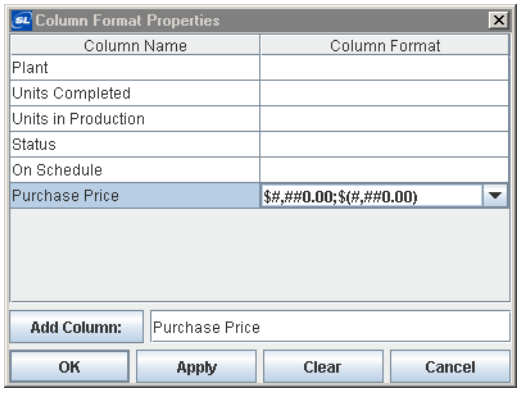
- **Copy All Properties** - To copy all object properties, both static properties and data attachments, select an object and click the Copy button on the toolbar.
- **Copy Single Property** - To copy an individual property from the Object Properties window, right-click on the Property Name and select **Copy**. To copy a property from the Edit Function dialog, right-click in a text field and select **Copy**.

There are four options for pasting object properties:

- **Paste Data Attachments** - To paste only data attachments, select one or more objects and click on the Paste Data Attachments button on the toolbar or use the keyboard shortcut Ctrl+Shift+V. **Note:** Only properties common to both objects will be pasted onto the selected object(s).
- **Paste Static Properties** - To paste only the static properties, select one or more objects and click on the Paste Static Properties button on the toolbar. **Note:** Only properties common to both objects will be pasted onto the selected object(s).
- **Paste All Properties** - To paste all properties, select one or more objects and click on the Paste All Properties button on the toolbar. This will paste all static properties as well as all data attachments. **Note:** Only properties common to both objects will be pasted onto the selected object(s).
- **Paste Single Property** - To paste an individual property into the Object Properties window, right-click on the Property Name and select Paste. The Paste option will only be enabled if the copied attribute can be set on the selected property (e.g., data attachments cannot be pasted onto static properties). To paste a property in the Edit Function dialog, right-click in a text field and select Paste.

Column Format Properties

In the **Object Properties** window, double-click on **columnFormat** in the **Property Name** field to bring up the **Column Format Properties** dialog. In the **Column Format Properties** dialog you can assign numerical and date formats to columns.



Field Name	Description
Column Name	This list is populated based on the table's data attachment. If you have not yet attached the table to data, this list will be empty.
Column Format	Enter or select a format from the drop down menu and press <Enter>. Specify numerical formats using the Java format specification, or with the following shorthand: **$** for US dollar money values, **$$** for US dollar money values with additional formatting, **()** for non-money values, formatted similar to money, or **#** for positive or negative whole values. Specify date formats using the Java date specification.
Add Column	Enter the name of the column and click the **Add Column** button to insert a column into the table.
Clear	Click the **Clear** button to remove all Column Formats listed. **Note:** Text columns containing data where all the values represent numbers are treated as if they are numeric columns, so number formats can be applied.

Image Property

Click in the **image** Property Value field and type the name of the image or select the  button to open the Select Image dialog containing up to three directories:

- Current Directory - Contains images in the current directory and one level of subdirectories.
- Custom Image Library - If you have specified a custom image library, this directory contains those images. See the "Creating a Custom Image Library" section for details.
- Symbol Library - Contains symbolic images (for example, symbols for various types of hardware, shapes, lights, arrows, etc.).

Navigate to the image you want to use and select it. A preview of the image appears in the pane to the right. Click **OK** or **Apply** to set the image on your object. If an image is not listed, enter the name of the file, including the relative path.

To scale your image to the size of the object, check the **imageScaleFlag**. The **visFlag** property controls the visibility of the object. The **transparencyPercent** property controls the transparency of the object.

The sample display file **general_objects.rtv** (located in **demos/tutorials**) features information on working with objects from the General tab.

Creating a Custom Image Library

The custom image library enables you to make your own images available in the **Select Image** dialog. To add your own image library, perform the following steps.

1. Place your images in a **.jar** file and add it to the "RTV_USERPATH" environment variable.

   Note: The images must be in a directory and not at the top level of the **.jar** file. They can be organized into subdirectories of one top level directory.

2. In the Display Builder, select **Tools>Builder Options** and, in the **Custom Image Library Path** field, set the path to the directory containing your **.jar** file.

   For example, suppose you have a .jar file with the following directory structure:

   ```
   com/mycompany/Images
   ```

```
com/mycompany/Images/Blue Images
com/mycompany/Images/Red Images
com/mycompany/Images/Green Images
```

In the **Custom Image Library Path** field you would enter **com/mycompany/Images** to add a directory named **Images** to the tree in the Select Image dialog. The **Images** directory will have three subdirectories: **Blue Images**, **Red Images**, and **Green Images**.

Note: Only directories containing images will be added to the Select Image dialog.

To access your custom image library, edit any property that allows you to set an image on an object (e.g. **image**, **barImage**, and **filterProperties** properties) or select **File>**"Background Properties" and edit the **Image Name** field.

webLabelFlag

In a Display Server deployment when the **webLabelFlag** property is selected, the object will be rendered as an HTML element in the thin client rather than rendered in the image by the Display Server. This allows the Display Server to update the individual object (if necessary) when the display is refreshed, instead of regenerating the entire display image. In some cases, this can improve the performance of the Display Server and thin client.

The **webLabelFlag** property also enables you to select an object's text and copy it to the clipboard.

General objects that support **webLabelFlag** are:

- obj_rect
- obj_rect_il
- obj_rect_ilv
- obj_rect_ilvs
- obj_rect_ilvx_da3
- obj_rect_ilvx_ra4
- obj_circ2d
- obj_circ2d_il
- obj_circ2d_ilv
- obj_circ2d_ilvs
- obj_circ2d_ilvx_da3
- obj_circ2d_ilvx_ra4

Use the following entry to set **webLabelFlag** in an rtview stylesheet (**.rts**) file:

```
rtv-all {
 webLabelFlag: 1;
}
```

Limitations

In the following situations, an object with **webLabelFlag** selected will still be rendered in the image by the Display Server:

- Display is opened in Internet Explorer 8 or older.
- Display Server is run with the -nohtml5 option set on the command line.
- labelTextPosY is set to "Tab Top" (obj_rect*)
- labelTextPosY is set to "Title Top" and bgBorderFlag is set to "true" (obj_rect*)
- obj_rect* with non-html object above it in Z-order which overlaps (e.g. obj_rect* used as a background behind several meter objects)
- Object is inside an object grid, and is not inside a composite object in that grid.

In other cases, an object may have a slightly different appearance in the thin client when it is rendered as HTML:

- bgGradientMode property set to "Diagonal Edge" or "Diagonal Center" is treated as "Vertical Edge" (Only vertical and horizontal gradients are supported on HTML objects in the thin client.)
- bgShadowFlag property is ignored

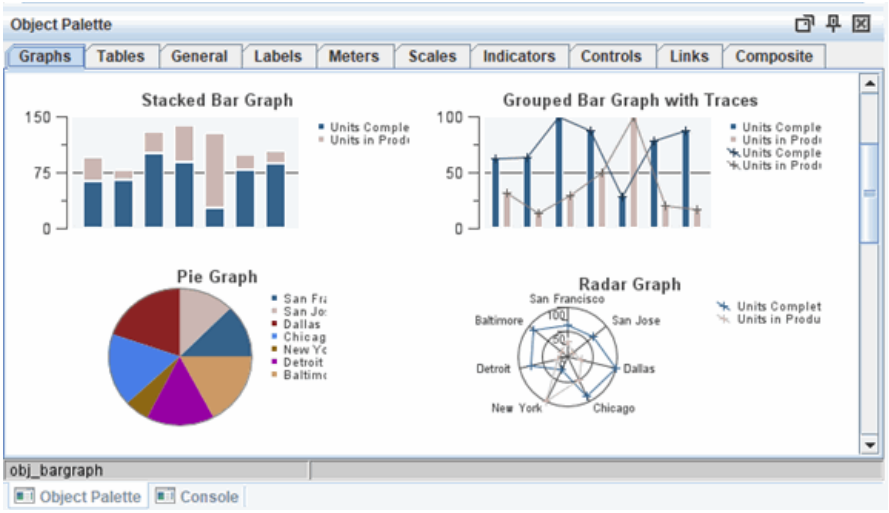
Object Descriptions

RTView objects are categorized in the Object Palette according to the appearance and behavior of each type of object. This section describes how to work with each type of object and assumes you have already reviewed the following: "Add/Edit Objects", "Object Properties", and the **Attach To Data** section for your data source.

To view sample displays with tutorials, product features, and data source specific examples go to the **demos** directory, run the data simulators that apply to your "RTView Data Sources" and start the Display Builder. Open the file **alldisplays.rtv** and double-click on your selection.

Note: To learn how to start the Display Builder and run the data simulators refer to the "Quick Start Tutorial".

Object Palette



Graphs

Graphs are useful for comparing data. Trend graphs are used to display a single variable per trace over time. Bar, pie, radar, and XY graphs are designed to display information returned by a tabular element in your data attachment. See "Graph Objects" for details.

Tables

Tables offer the ability to display large amounts of data. See "Table Objects" for details.

General

The **General** tab contains flexible objects that can be used to display data and images in addition to executing commands either manually or based on thresholds. To display values on these objects, attach the **value** property to numeric data or the **valueString** property to text data, and the **label** property to text data.

If you have a very long **label** or **valueString** and do not want to have the **label**/**valueString** extend past the edges of the rectangle, you can select the **bgClipTextFlag** option in the **Background** category in **Object Properties** to clip the edges of the text so that only that portion within the rectangle displays. **Note:** The **bgVisFlag** must be enabled for **bgClipTextFlag** to work.

General objects that feature the image property can be customized to display your image (**.gif**, **.jpg**, or **.png**) file.

The **styleClass** property allows you to enter the style class name for this object as defined in your "Style Sheets". If not specified, the object class name is used.

Note: The value entered must not contain spaces and cannot start with rtv-.

The "webLabelFlag" property allows select objects to be rendered as HTML elements in a thin client deployment.

Objects labeled **Range Dynamic** and **Discrete Dynamic** on the **General** tab support threshold functionality, allowing you to change the image and color of the object as well as execute a threshold command from the object based on the current value of the **value** property. Both the range dynamic and the discrete dynamic objects have the thresholding functionality turned off by default. See "Threshold Commands" for more information.

Labels

The Labels tab includes several types of labels. All of the labels have at least one of the following properties: **value**, **valueString**, and **label**. The **value** property used to display numeric data, either from a data attachment or static values. Some labels can be designed to change color according to **value** property or feature vu meters to dynamically display that data. The **valueString** and **label** properties are both used to display text data, either from a data attachment or static values. See "Labels Objects" for more information.

Meters

Meters are useful for displaying data that falls within a known range. You can attach your data to the **value** property and you can control the range of data displayed in the meter by setting the **valueMax** and **valueMin** properties. See "Meter Objects" for more information.

The **styleClass** property allows you to enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. See "Creating Style Sheets" for more information.

Note: The value entered must not contain spaces and cannot start with **rtv-**.

Scales

Scales are useful for displaying data that falls within a known range. You can control the range of data displayed in the scale by setting the **valueMax** and **valueMin** properties. Attach your data to the **value** property. All of the scales, except the pie scale, support a variety of axis styles and can be oriented vertically or horizontally. See "Scale Objects" for details.

Indicators

The Object Palette features four types of "Indicator Objects":

- **Discrete** - Supports 3 discrete comparisons.
- **Limits** - Supports 2 high and 2 low thresholds.
- **Multi** - Supports an unlimited number of comparison values.
- **Panel** - Panel of 2 or 3 indicator lights, each supports a discrete comparison.

For all indicators, attach your data to the **value** property and setup comparisons using the properties in the **Alert** category. Indicators that feature the "Image Property" can be customized to display your image (**.gif, .jpg** or **.png**) file.

Note: Objects previously featured on the **Indicators** tab will continue to function in existing displays, but are no longer available in the Object Palette.

Controls

Controls allow you to issue action "Commands" and update variables (see "Add/Edit Variables") that may be used to control other objects in the display. The **visFlag** property controls the visibility of the object. See "Control Objects" for details.

Links

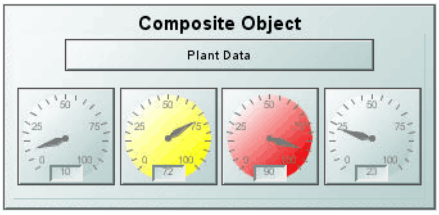
The **Links** tab features five objects. The **Direct** link draws a straight line between the objects it connects. The **Direct Offset** link attaches to objects at a right angle, but draws a diagonal line for the remainder of the link. The **Orthogonal** link draws a line at right angles between the objects it connects. The **Horizontal Square** link draws a link from the right/left side of the parent closest to the child to the right/left side of the child closest to the parent. The **Vertical Square** link draws a link from the top/bottom side of the parent closest to the child to the top/bottom side of the child closest to the parent. See "Link Objects" for details.

Composite

The Composite tab features pre-configured composite objects. The composite object allows you to show a display (.rtv) file within an object. This is useful for creating groups of objects that you want to use multiple times, either directly in your display, or in an object grid. See "Composite Object" for details.

Composite Object

The composite object (class name: obj_composite) allows you to show a display (**.rtv**) file within an object. This is useful for creating groups of objects that you want to use multiple times, either directly in your displays or in the "Object Grid". There are three default composite objects:



Creating Composite Displays

First, create the display that you want to use inside of a composite object. This display is referred to as the composite display. While any display (**.rtv**) file can be used as a composite display, there are some rules about constructing that display:

- Property names for composite objects cannot be used as variable names, in the **Composite** section this only includes **rtvName** and **substitutions**.
- Only public variables that are not mapped to data or substitutions will show up as properties on the composite object.
- Names of all public variables used for tabular data must have Data Type set to Tabular. See "Add/Edit Variables" for more information.
- Composite displays can use any object from the Object Palette, except the composite object.
- Nested composite objects are not supported.

Next, open new display and add a composite object. On the composite object, set the **rtvName** property to the name of the composite display you just created.

Note: The rtvName property should be set before any other properties.

Once you have set **rtvName**, all public unmapped variables in the composite display will show up as properties on the composite object. Like any other property, you can attach those properties to data or set them to static values. See "Add/Edit Variables" for more information.

Note: If you do not want a variable to be exposed as a property, mark it as private.

For example, you can create a display (.rtv) file with two meters that are attached to two public variables: **value1** and **value2**. When you enter the name of that display (.rtv) file in the rtvName property of the composite object, then value1 and value2 will show up in the **Composite** section of the Object Properties list. These properties can be set to static data or attached to data to drive the values in the composite display. When the composite object is used in an "Object Grid", these properties will also show up in the **iconProperties** dialog, which allows you to assign them to either values or data columns.

Note: Composite objects used in Object Grids must have their **bgOpaqueFlag** property selected.

Use the **tabcomposites** property to enable the **Tab** key to switch focus to controls that are inside composite objects. See "Command Line Options: Display Builder and Display Viewer" for more information.

Using Interaction Properties

The specified **drillDownTarget** or **command** for objects in the composite display will take precedence over the **drillDownTarget** or **command** set on the composite object when it is instanced directly in a display. When the composite object is used in an Object Grid, the specified **drillDownTarget** or **command** for objects in the composite display will take precedence over the **drillDownTarget** or **command** set on the Object Grid.

For example, let's say you have a composite display with an object that drills down to display A and you set a **drillDownTarget** on the composite object to display B. When you click on the composite object, if you click directly on the object that drills down to display A, then display A is opened. If you click anywhere else in the composite display, then display B is opened.

A similar thing happens when you use that composite object in an Object Grid. However, in this case, you can't set a **drillDownTarget** on the composite object, you can only set the **drillDownTarget** on the Object Grid. So, let's say you set the Object Grid to drill down to display B. Again when you click directly on the object in the composite display that drills down to A, display A is opened. When you click anywhere else in the composite or in the Object Grid, display B is opened.

Commands

To assign a command, right-click in the **Property Value** field of the **command** property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Once a drill down target has been set, double-click to activate the drill down. Drill down displays can be activated in the same window or opened in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Background Properties

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. This property is only applicable if bgBorderFlag is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle,
bgColor	Select the ... button and choose a color from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the bgStyle selected is 3D Rectangle.
bgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is white. The bgColor property sets the first color in the gradient.
bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge -** Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgOpaqueFlag	When a composite object is used in an "Object Grid", the **bgOpaqueFlag** property must be selected. If deselected, then objects inside a composite that is clipped by the edge of the Object Grid may be drawn outside the Object Grid. In order to avoid showing a border around each composite object in the Object Grid, select **Edit>Model Properties** and deselect the **bgBorderFlag**.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example if **objWidth** is 100 and **objHeight** is 50, then the value of **bgRoundness** cannot exceed 25. If it does, then half the value of **objHeight** (25) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.

bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the object and the border.

Composite Properties

There are two standard properties in the Composite section: **rtvName** and **substitutions**. Once you have set the **rtvName** property, the composite display will show up in your composite object and all of the public variables from the composite display that are not mapped to data or substitutions will be added as Composite properties. See "Add/Edit Variables" for more information.

Property Name	Description
resizeMode	Specify how the composite object is resized. There are two modes: **Size to Display** -The size of the composite object is determined by the size of the display it contains. The composite object cannot be resized. This is the default setting. The dock and anchor properties should not be used in this mode. **Layout** -The composite object can be resized. Objects in the composite display are situated according to their anchor and dock properties. The dock and anchor properties can be set on the composite object so that it resizes during a window resize if the window resize mode is also set to **Layout**. If the window resize mode for the display containing the composite object is set to **Scale**, the composite object performs a scale instead of a layout.
rtvName	Specify the name of the display (**.rtv**) file to show in your object. This display is referred to as the composite display. See "Creating Composite Displays" for details.
substitutions	Specify the substitutions for the composite display. **Note:** The composite display also inherits substitutions from the display that contains the composite object. Substitutions are optional and must use the following syntax: `$subname:subvalue $subname2:subvalue2` If a substitution value contains a single quote, it must be escaped using a / : **`$filter:Plant=/'Dallas/'`** If a substitution value contains a space, it must be enclosed in single quotes. Do not escape these single quotes: **`$subname:subvalue $subname2:'sub value 2'`** A substitution string cannot contain the following: : | . tab space , ; = < > ' " & / \ { } [ ] ( )

Historian Properties

Property Name	Description
historyTableName	Specify name of table in your history database in which to store tabular data. See "Configuring the Historian" for information.
historyTableRowNameFlag	If selected, data from the row name field will be stored in the first column of the table specified in **historyTableName**.

Interaction Properties

Property Name	Description
command	Assign a command. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownTarget	Name of display (.rtv) file targeted as a drill down. See "Drill Down Displays" for information.

Label Properties

Property Name	Description
label	Edit label text directly in the Property Value field or select the ... button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. This property is only applies if **labelTextAlignY** is set to **TabTop**.
labelTextAlignX	Select x-axis position of label text from the drop down menu.

labelTextAlignY	Select y-axis position of label text from the drop down menu. **Outside Top** - Position label well above the background rectangle. **Top** - Position label just above the background rectangle. **Title Top** - Position label along the top line of the background rectangle. **Tab Top** - Position label tab just above the background rectangle. Height and width of the tab is dependent on the height and width of the text. Use the labelMinTabWidth property to specify a minimum tab width. **Inside Top** - Position label inside the top of the background rectangle.
labelTextColor	Select the  button and choose a color from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop down menu.
labelTextHeight	Set the height of the label text in pixels.

Object Properties

Property Name	Description
anchor	Select the  button to display the "Anchor Property Window", which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. An object should not be docked if the Resize Mode is set to **Scale**. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	This property is read only. It shows the height in pixels of the composite object. See "Creating Composite Displays" for more information.
objName	Name given to facilitate object management via the Object List dialog. Select **Tools>Object List**.
objWidth	This property is read only. It shows the width in pixels of the composite object. See "Creating Composite Displays" for more information.
objX	Set the x position of the object.
objY	Set the y position of the object.

styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. See "Creating Style Sheets" for more information. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Sets the transparency of the object. This property only applies to the background of the composite object.
visFlag	Set the visibility of the object.

Control Objects

The **Controls** tab features objects that allow you to issue commands and update variables that may be used to control other objects in the display. The sample display file **controls.rtv** (located in **demos/tutorials**) features information on working with objects from the Controls tab.

The **value**, **valueString**, and **selectedValue** properties allow you to set the value for the controls. The **varToSet** property allows you to update the attached variable with the value from the control.

The **actionCommand** property allows you to set up a command which will be executed from the control. This **actionCommand** can reference the value from the control by using the keyword **$value**. To enable selecting, moving and editing the controls without activating the associated action command, the controls are not active in the main window of the Display Builder.

The **tabIndex** property allows you to define the order in which table and control objects will receive focus when navigated from your keyboard. Initial focus is given to the object with the smallest **tabIndex** value, from there the tabbing order proceeds in ascending order. If multiple objects share the same **tabIndex** value, then initial focus and tabbing order are determined by the alpha-numeric order of the object names. Objects with a **tabIndex** value of 0 are last in the tabbing order.

Note: The **tabIndex** property does not apply to Slider objects or objects that are disabled, invisible, or have a value of less than 0.

To preview a control object in the Display Builder, save your file and click the Preview button  in the toolbar. Your display opens in a Preview Window that allows you to activate control objects. The Preview button becomes disabled when you edit your display. Save your display again to enable it.

Some control object properties cache their colors and therefore do not update when a custom color definition changes. You will need to either to restart RTView or reload the display to see the color change for these objects.

The **Controls** tab features the following objects:

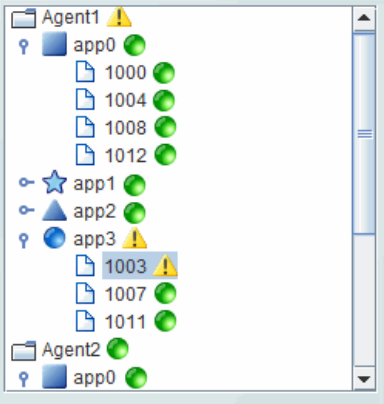
Navigation Controls

Navigation controls allow you to create a rich and compact visual presentation of hierarchical data. Navigation controls are most often used in a multi-panel application for display navigation. Typically, the navigation control is shown in one panel and another panel contains the displays which you drill-down to by selecting items in the navigation control. Navigation controls can also be used in any application where hierarchical data is most effectively displayed using expandable/collapsible nodes.

You can optionally configure status icons for navigation controls, using images of your choice, to visually indicate whether an element is in a critical state, and to also propagate the status of elements to the top of the navigation control (see the tree control, for example, below).

Tree Control

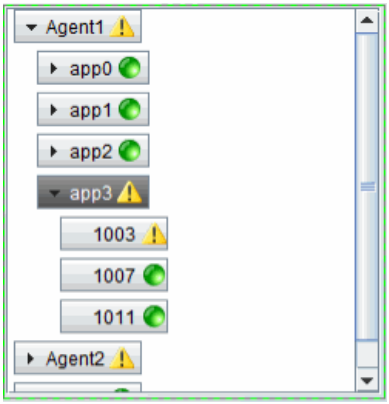
With the tree control (class name: **obj_c1tree**), multiple nodes can expand at a time. The tree control not only allows you to configure status icons and to propagate the status up to parent nodes - you can also configure type icons. Type icons are static images you assign to tree nodes to visually indicate the element type (for example, Production or Sales) or the node depth.



For details, see "Tree Control".

Accordion Control

With the accordion control (class name: **obj_c1accordion**), nodes appear as pushbuttons and a single node expands at a time. The accordion control visually indicates whether an element in the accordion is closed or open. As with the tree control, the accordion control allows you to configure status icons and to propagate the status up to parent nodes. The accordion control does not display type icons (as the tree control can).



For details, see "Accordion Control".

Text Entry Fields

There are three text entry fields to choose from:

- obj_c1textedit: Allows you to enter numeric or text information. Attach your data to **valueString** to update the value shown in this text entry field from your data.
- obj_c1textedit_i: Accepts only numbers without decimal points.
- obj_c1textedit_d: Allows numbers with decimal points.

Attach your data to value to update the values shown in these text entry fields from your data.

Note: To get initial values from an internal RTView variable and base those values on user interaction, you must attach the **value** and **varToSet** properties to the same variable.

The **actionCommand** is executed and the variable attached to **varToSet** is updated with the data you typed into the text entry field when you press <Enter> and, if the **executeOnFocusLostFlag** is selected, when the text entry field loses focus. To disable text entry, deselect the **editableFlag** check box. If the **editableFlag** is off and **enabledFlag** is on, the mouse and arrow keys can still be used to move the cursor. If the **executeOnKeystrokeFlag** check box is selected, then each keystroke that modifies the text field will set the **varToSet** variable and execute the **actionCommand**, if applicable, using the modified text.

Note: In a Thin Client deployment there may be a delay between the keystroke and the response, depending on the speed of your network and server.

The following properties support input validation. Attach **inputValidVarToSet** to a local variable and this variable will be updated when the control executes with a value of **1** if the input is valid or a value of **0** if the input is invalid. Attach **invalidInputMsgVarToSet** to a local variable and this variable will be updated when the control executes with an error message if the input is invalid or an empty string if the input is valid. If validation fails, the **actionCommand** will not be executed. By default, input validation occurs whenever a control executes (i.e. when you press <Enter>). If **executeOnFocusLostFlag** is selected, then the control will also execute whenever the object loses focus. If **executeOnKeyStrokeFlag** is selected, the control will also execute whenever a key is pressed while the object has focus. For numeric validation, set **valueMax** and **valueMin** to specify the maximum or minimum valid values. If left blank, there will be no maximum or minimum value validation. Additionally, the following properties apply for text string validation:

- **characterCase** - Select from the following options: Mixed Case, Upper Case, or Lower Case.
- **minCharacters** - Specify minimum number of characters allowed in the field. If left blank, there will be no validation of minimum number of characters.
- **maxCharacters** - Specify maximum number of characters allowed in the field. If left blank, there will be no validation of maximum number of characters.

To set the **bgColor** and **fgColor** properties of a text entry field, click on the  button and choose a color from the palette. The **bgColor** property sets the color of the object's background area and the **fgColor** property sets the color of the text. When **bgColor** and **fgColor** properties are set to **Default**, then the text entry field will adopt default settings for background and foreground colors. For example, in the Display Viewer the text entry field in your display will adopt a standard Java appearance. When that same display is viewed via Thin Client Browser, the text entry field will conform to that browser's color preferences.

Use the **valueTextAlignX** property to set text alignment: **Left**, **Center**, or **Right**.

Note: The selected alignment can only be applied if the length of the text is less than the length of the text entry field.

Use the **mouseOverText** property to enter a tool tip for this control. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use \n to delimit the lines (e.g. **control\nobject**).

Note: The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

The **styleClass** property allows you to enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. NOTE: The value entered must not contain spaces and cannot start with **rtv-**.

The numeric text entry controls (**obj_c1textedit_i** and **obj_c1textedit_d**) support a validation option for blank entries. To enable this feature, select the **validateBlankValuesFlag** property. If selected and a blank string is entered, the **actionCommand** will not execute and **invalidInputVarToSet** and **invalidInputMsgVarToSet** will indicate an invalid entry.

Set the **executeOnlyIfChangedFlag** property to true to have the control's command execute only when the text in the control is changes. The text control's **varToset** and **valueString** or value property must be attached to the same local variable. By default, **executeOnlyIfChangedFlag** is false. When **false**, the control's command is executed when the control loses focus (if **executeOnFocusLostFlag** = **true** or the control is a text area), or when the user presses the **Enter** key in the control (if the control is not a text area), regardless of whether the text in the control was changed. (This matches the text control behavior in all previous releases.)

Text Area Object

The text area control supports display and entry of multi-line text with optional word wrap. Scrollbars will appear as needed. Attach your data to **valueString** to update the value shown in this text area from your data.

The **actionCommand** is executed and the variable attached to **varToSet** is updated with the data you typed into the text area control when the text area control loses focus. To disable text entry, deselect the **editableFlag** check box. If the **editableFlag** is off and **enabledFlag** is on, the mouse and arrow keys can still be used to move the cursor. If the **executeOnKeystrokeFlag** check box is selected, then each keystroke that modifies the text area will set the **varToSet** variable and execute the **actionCommand**, if applicable, using the modified text.

Note: In a Thin Client deployment there may be a delay between the keystroke and the response, depending on the speed of your network and server.

To set the **bgColor** and **fgColor** properties of a text area object, click on the  button and choose a color from the palette. The **bgColor** property sets the color of the object's background area and the **fgColor** property sets the color of the text. When **bgColor** and **fgColor** properties are set to Default, then the text area control will adopt default settings for background and foreground colors. For example, in the Display Viewer the text area control in your display will adopt a standard Java appearance. When that same display is viewed via Thin Client Browser, the text area control will conform to that browser's color preferences.

Use the **mouseOverText** property to enter a tool tip for this control. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use \n to delimit the lines (e.g. **control\nobject**).

Note: The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

The **styleClass** property allows you to enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used.

Note: The value entered must not contain spaces and cannot start with **rtv-**.

maxCharacters - The maximum number of characters that is submitted to the control's **actionCommand**. The default value is blank (no limit). The limit does not restrict the length of the text in the control, instead the limit is checked before the command is executed and if the limit is exceeded, the command is not executed.

inputValidVarToSet - Attach to a local variable. This variable is updated with a value of 1 if the **maxCharacters** limit is blank or the text is less than or equal to the **maxCharacters** limit, or updated with a value of 0 if the text length exceeds the **maxCharacters** limit.

invalidInputMsgVarToSet - Attach to a local variable. This variable is updated with an error message if the text length exceeds the **maxCharacters** limit, or updated with an empty string if the text length is less than or equal to the **maxCharacters** limit.

In the Display Viewer, pressing the <Tab> key inside a text area control inserts a tab character in the text. To move the keyboard focus away from a text area control, either click outside of the control or press <Ctrl-Tab>.

Button

The **actionCommand** is executed and the variable attached to **varToSet** is updated with the value from the **valueToSet** property when you click on the button.

Use the **hotKey** property to enable keyboard activation on a button. Assign an alphanumeric value (A-Z or 0-9) to be used on the keyboard with the Alt key. The button is activated when the panel it resides in has focus and the user holds down the Alt key and simultaneously selects the assigned value on the keyboard. For example, Alt+B, where B is the assigned key.

When keyboard activation is enabled and the **defaultButtonFlag** is selected, a button can be activated by pressing the <Enter> key. In the display, the activated button will be highlighted with an extra border. While multiple objects in the display can have the **defaultButtonFlag** selected, it will only apply to first one added to the display.

Note: The **defaultButtonFlag** property does not apply to objects that are disabled or invisible. If you are viewing the Display Server in Firefox, another control object must have focus when the <Enter> key is pressed in order for the button to be activated.

To set the **bgColor**, **fgColor**, and **fgDisabledColor** properties of a button, click on the  button and choose a color from the palette. The **bgColor** property sets the color of the object's background area, the **fgColor** property sets the color of the text, and the **fgDisabledColor** property sets the color of the button when the button is disabled. When **bgColor, fgColor**, and **fgDisabledColor** properties are set to **Default**, then the button will adopt default settings for background, foreground, and disabled colors. For example, in the Display Viewer the button in your display will adopt a standard Java appearance. When that same display is viewed via Thin Client Browser, the button will conform to that browser's color preferences.

Use the **mouseOverText** property to enter a tool tip for this control. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use \n to delimit the lines (e.g. control\nobject).

Note: The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

The **styleClass** property allows you to enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used.

Note: The value entered must not contain spaces and cannot start with **rtv-**.

Use the **menuItemIndex**, **menuItemGroup** and **menuItemSubmenu** properties with the button to extend the RTView context menu. For details, see "Extending the Context Menu".

The **image** property enables you to customize a button to display your image (.gif, .jpg or .png) file. Click in the Property Value field and type the name of your image. Or, click on the  button to open the **Select Image** dialog.

A button can display an image, a label, or both. If both the **image** and **label** properties are set, the image will be to the left of the label. It is best to leave several pixels of padding between the image and/or label of a button to avoid wrapping or clipping of text.

The appearance of a button with an image and/or label can vary slightly between the Display Viewer and supported browsers. You may notice the following variations:

- Alignment of the image inside the button: The image may be somewhat closer to the top or bottom edge of the button.
- Vertical alignment (if any) of the image with the button's label: The center of the image may be somewhat lower or higher than the center of the label.
- Appearance of the image and label if they don't fit inside the button: In some cases the label may be clipped with an ellipsis (...) replacing the clipped text, or the label may be clipped with no indication, or the label may wrap and may extend below the button.

Note: In Firefox a button with an image will not depress when it is clicked, however the button's command will still be executed.

Check box

Attach your data to the **value** property to update the value displayed by the check box. **Note:** To get initial values from an internal RTView variable and base those values on user interaction, you must attach the **value** and **varToSet** properties to the same variable.

If the value of your data attachment equals the **valueToSetChecked** property, the check box will be selected. Default values for **valueToSetChecked** is 1 (On) and for **valueToSetUnchecked** is 0 (Off).

The **actionCommand** is executed and the variable attached to **varToSet** is updated when you click on the check box. If clicking on the check box selects it, then **varToSet** will be updated with the value of **valueToSetChecked**. If you click on the check box and it does not become selected, **varToSet** will be updated with the value of **valueToSetUnchecked**.

Use the **mouseOverText** property to enter a tool tip for this control. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use \n to delimit the lines (e.g. **control\nobject**).

Note: The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

The **styleClass** property allows you to enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used.

Note: The value entered must not contain spaces and cannot start with **rtv-**.

Radio Button Group

Use the **radioListValues** property to specify the labels and values for each radio button (e.g., Low,25;Medium,50;High,75), or make a data attachment to a two-column table. When attaching to a table, the first column is used for the radio button labels, the second column is used for radio button values. If only labels are specified, they will also be used as values. Specify which radio button is selected using the **selectedValue** property.

The **orientationMode** property specifies how radio buttons are populated, horizontally (by row) or vertically (by column). Specify the width, in pixels, of the space between radio buttons using the **horizontalGap** and **verticalGap** properties.

The **actionCommand** is executed and the variable attached to **varToSet** is updated with the selected value when you select a previously unselected radio button from the group.

Use the **mouseOverText** property to enter a tool tip for this control. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **control\nobject**). The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

Note: In a Thin Client deployment, multi-line and non-static tool tips are not supported.

The **styleClass** property allows you to enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used.

Note: The value entered must not contain spaces and cannot start with **rtv-**.

Slider

The **valueMin** and **valueMax** properties control the range of values for the slider. Attach your data to the value property to update the value displayed by the slider from a data attachment.

Note: To get initial values from an internal RTView variable and base those values on user interaction, you must attach the value and **varToSet** properties to the same variable.

The **actionCommand** is executed and the variable attached to **varToSet** is updated with the value from the slider when you release the slider thumb after dragging it or when you click on the slider to advance the thumb one unit. Select the **updateWhileAdjustingFlag** to receive updates while dragging the slider thumb.

The **fgColor** property sets the tick mark color. Click on the  button to select a color or select Default to use the best color for your look and feel. The **fgColor** is only applied when the Slider control is enabled, so tick marks on sliders in the Display Builder's Working Area are not visible. Select **Tools>Preview Window** to see the **fgColor** applied.

Note: In the Thin Client deployment, Slider controls do not have tick marks and therefore the **fgColor** property does not apply.

Use the **mouseOverText** property to enter a tool tip for this control. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use \n to delimit the lines (e.g. control\nobject).

Note: The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

The **styleClass** property allows you to enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used.

Note: The value entered must not contain spaces and cannot start with **rtv-**.

The **axisDirection** property allows you to set the axis direction of the scale. Select from the following options:

Bottom to Top - Vertical orientation with min on bottom and max on top.

Left to Right - Horizontal orientation with min on left and max on right.

Top to Bottom - Vertical orientation with min on top and max on bottom.

Right to Left - Horizontal orientation with min on right and max on left.

List Box

Use the **listValues** property to:

- Specify the labels and values for each radio button (e.g. Low,25;Medium,50;High,75),
- Make a data attachment to a single cell table containing a semicolon (;) delimited list of values, or
- Make a data attachment to a two-column table, where the first column is used for the radio button labels, the second column is used for radio button values. NOTE: If only labels are specified, they will also be used as values.

To set the selected list item from a data attachment, attach your data to the **selectedValue** property.

The **actionCommand** is executed and the variable attached to **varToSet** is updated with the **selectedValue** when you chose a previously unselected item from the list box.

To set the **bgColor** and **fgColor** properties of a list box, click on the  button and choose a color from the palette. The **bgColor** property sets the color of the object's background area and the **fgColor** property sets the color of the text. When **bgColor** and **fgColor** properties are set to **Default**, then the list box will adopt default settings for background and foreground colors. For example, in the Display Viewer the list box in your display will adopt a standard Java appearance. When that same display is viewed via Thin Client Browser, the list box will conform to that browser's color preferences.

Use the **mouseOverText** property to enter a tool tip for this control. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use \n to delimit the lines (e.g. control\nobject).

Note: The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

The **styleClass** property allows you to enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used.

Note: The value entered must not contain spaces and cannot start with **rtv-**.

Combo Box

Use the **listValues** property to:

- Specify the labels and values for each radio button (e.g. Low,25;Medium,50;High,75),
- Make a data attachment to a single cell table containing a semicolon (;) delimited list of values, or
- Make a data attachment to a two-column table, where the first column is used for the radio button labels, the second column is used for radio button values. NOTE: If only labels are specified, they will also be used as values.

To set the selected menu item from a data attachment, attach your data to the **selectedValue** property. Check the **textEditEnabledFlag** to allow the user to type a selection in addition to selecting from the drop down menu.

The **actionCommand** is executed and the variable attached to **varToSet** is updated with the **selectedValue** when you choose a previously unselected item from the drop down menu or type a new item into the text entry area of the combo box.

To set the **bgColor** and **fgColor** properties of a combo box, click on the ... button and choose a color from the palette. The **bgColor** property sets the color of the object's background area and the **fgColor** property sets the color of the text. When **bgColor** and **fgColor** properties are set to Default, then the combo box will adopt default settings for background and foreground colors. For example, in the Display Viewer the combo box in your display will adopt a standard Java appearance. When that same display is viewed via Thin Client Browser, the combo box will conform to that browser's color preferences.

Use the **mouseOverText** property to enter a tool tip for this control. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use \n to delimit the lines (e.g. control\nobject).

Note: The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

The styleClass property allows you to enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used.

Note: The value entered must not contain spaces and cannot start with rtv-.

Date Chooser

Use the **selectedValue** property to specify an initial date value. The **maximumDate** and **minimumDate** properties control the selectable dates in the popup calendar. It is recommended that initial date you specify is within the **maximumDate**/**minimumDate** value range so that the user will not have to navigate to a valid date.

The **actionCommand** is executed and the variable attached to **varToSet** is updated with the **selectedValue** when either the user selects an date from the popup calendar or, if the date is typed in, when the user presses <Enter> or clicks out of the field.

The **dateFormat** property controls both the input of a date typed into a field, as well as the output to a local variable or action command. If **dateFormat** is not specified, then a locale dependent format will be used. When users mouse-over the text entry field, a tooltip will display the specified **dateFormat** so they will know which format to enter.

Since the **dateFormat** controls both the input and the output of the Date chooser, information not included in the **dateFormat** will be lost. For example, if the date format is MMMM dd, yyyy (e.g. January 01,2000), and the **timeEntryEnabledFlag** is selected, even if you enter a time it will not be stored since it was not included in the specified **dateFormat**. Or, if the specified **dateFormat** does include the time (e.g. MM/dd/yyyy hh:mm) and the time you entered was PM, it would be stored in the AM since am/pm was not specified in the **dateFormat** and AM is the default.

The **dateFormat** should be specified using format specifiers from the **SimpleDateFormat** class. The **dateFormat** property does not support backslashes (\). In the Display Server, the parsing is not guaranteed unless the date entered is in the same format as the specified **dateFormat**.

Note: In the Display Server, the text entry field is not validated and the following format specifiers are not supported: G, w, W, D, F, E, k, K, S, z, Z.

The **timeEntryEnabledFlag** property controls the visibility of the Time field in the popup calendar. Times entered into this field must be in the following format: hh:mm:ss a. When **timeEntryEnabledFlag** is selected, an OK and Cancel button are added to the popup calendar, so you can select a date and enter a time before closing the popup. If the **timeEntryEnabledFlag** is deselected, the popup calendar closes as soon as you select a date.

The **validColor** property sets the font color to use while editing in the text entry area if the entered value uses the correct format. The default **validColor** is green. The **invalidColor** property is the font color used during and after editing in the text entry area if the entered value does not use the correct format. The default **invalidColor** is red. The **fgColor** property sets the color of the text entry area. The default **fgColor** is black.

Note: The **validColor** and **invalidColor** properties are not supported in the Thin Client deployment.

The styleClass property allows you to enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used.

Note: The value entered must not contain spaces and cannot start with **rtv-**.

Localization:

- The Time field, OK and Cancel buttons are localized by SL, currently only English and Japanese are available.
- In the Display Builder and Display Viewer Application, the month and day names will use the language settings on the client system.
- In the Display Server, the month and day names can be localized by replacing the date.js file in the Display Servlet with a localized version.

Note: The Display Server uses the Datejs date library for parsing and formatting dates and times.

Password Field

The Password Field control object works just like Text Entry Fields except that instead of displaying entered text, it displays asterisks (*).

Tab Control Object

The tab control (obj_c1tabs) has two temporary tabs labeled **A** and **B** when initially drawn on the palette, and also when the user places an instance on a display. The temporary tabs are replaced with the actual tabs when you attach the tab control's **valueTable** property to a data table containing one row for each tab.

The size given to the control in the Builder determines the space available for tabs. The tabs are drawn horizontally with the first tab at the left edge of the control. If the control is not wide enough to show all of the tabs, the tabs wrap vertically. If the control is not tall enough to show all of the tabs, some of the tabs might be clipped or invisible.

Changing the control's **labelTextSize** property changes the size of the tab label text, which also affects the size of the tabs.

The tab control is populated from a data table attached to its **valueTable** property, with one tab created for each row in the table. The following tab control properties are used to map the columns of the data table to each tab:

- **labelColumnName**: The value from this column is used as the tab label. If **labelColumnName** does not specify a column in the **valueTable** or contains an empty string, then the tab for row **N** in the **valueTable** is labeled "**Tab N**". The **labelColumnName** property appears in the Label category.
- **valueColumnName**: The value from this column is assigned to the variable (if any) attached to the tab control's **varToSet** property when the corresponding tab is selected by the user. If **valueColumnName** does not specify a column in the **valueTable**, then the tab's index (**0** through **N** - **1**, for a table with **N** rows) is used as the tab value. The **valueColumnName** property appears in the Data category.

- **imageColumnName**: The value from this column is used to load an icon image, shown to the left of the tab's label. If **imageColumnName** does not specify a column in the **valueTable**, or if the column is empty, or if the image cannot be found, the tab does not contain an icon. An icon affects the size of the tab. The **imageColumnName** property appears in the **Image** category.
- **mouseOverColumnName**: The value from this column is used as the tooltip for each tab. If **mouseOverColumnName** does not specify a column in the **valueTable**, or if the column is empty, the tab does not display a tooltip. The **mouseOverColumnName** property appears in the **Interaction** category.

The tab control supports the **drillDownColumnSubs** property, which can be useful in cases where the tab's command is a drilldown. As with other objects that support **drillDownColumnSubs**, it can be used to set the values for substitutions and local variables from columns in the row of the **valueTable** that corresponds to the selected tab.

The control's **visFlag** property can be used to toggle the control's visibility. Unlike the other control objects, the tab control does not support the **enabledFlag** property so it is always enabled.

The **selectBgColor** and **selectTextColor** fields in the **Interaction** region in **Object Properties** allow you to specify the background and text colors for the selected tab. To set the colors, click in the field in the **Interaction** region, click on the ... button, and select a color from the **Color Chooser** palette that displays. When **selectBgColor** and **selectTextColor** properties are set to **Default**, then the tab will adopt default settings for the background and text colors. Both options are set to **Default** initially.

The **paddingVertical** and **paddingHorizontal** options in the **Label** category in **Object Properties** provide extra padding, in pixels, above and below (**paddingVertical**) and left and right (**paddingHorizontal**) of the label of each tab. These options enable you to make the tabs taller and wider without increasing the font size. The default value for both options is zero.

The **webHoverTabColor** and **webHoverTextColor** options in the **Interaction** category in **Object Properties** define the background color and text color for an unselected tab when the cursor is positioned over the tab. The default values are the same as the normal tab background and text colors. This property only affects the thin client and is ignored in the builder/viewer.

As on other control objects, the predefined substitution named **$value** can be used in the control's command. The value of the selected tab is substituted for **$value** when the command is executed. (See the **valueColumnName** property for a description of how a tab's value is determined).

If a tab control has a command defined and the **commandConfirm** property is checked, the user is asked to confirm the command when a tab is clicked, but the clicked tab becomes the selected tab regardless of the user's response.

Limitations / Differences:

- In the Thin Client, the tab control is not supported in IE version 8 or older and will not appear in displays opened in those versions.
- The tab sizes and appearance may differ somewhat when viewed in the Builder/Viewer versus the Thin Client.
- In the Thin Client, the background color of the selected tab is brighter than the unselected tabs, while in the Builder/Viewer the selected tab is darker than the other tabs. (This difference is intentional, to conform with the standard appearance of tabs in Swing versus a browser.)

webLabelFlag

In a Display Server deployment when the **webLabelFlag** property is selected, the object will be rendered as an HTML element in the thin client rather than rendered in the image by the Display Server. This allows the Display Server to update the individual object (if necessary) when the display is refreshed, instead of regenerating the entire display image. In some cases, this can improve the performance of the Display Server and thin client.

The **webLabelFlag** property also enables you to select an object's text and copy it to the clipboard.

Use the following entry to set **webLabelFlag** in an rtview stylesheet (.rts) file:

```
rtv-all {
 webLabelFlag: 1;
}
```

Limitations

In the following situations, an object with **webLabelFlag** selected will still be rendered in the image by the Display Server:

- Display is opened in Internet Explorer 8 or older.
- Display Server is run with the **-nohtml5** option set on the command line.
- Object is inside an object grid, and is not inside a composite object in that grid.

In other cases, an object may have a slightly different appearance in the thin client when it is rendered as HTML:

- **bgGradientMode** property set to "Diagonal Edge" or "Diagonal Center" is treated as "Vertical Edge" (Only vertical and horizontal gradients are supported on HTML objects in the thin client.)
- **bgShadowFlag** property is ignored

Navigation Control Objects

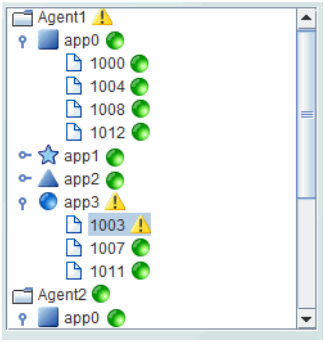
Display your data in navigation controls by attaching it to the **valueTable** property. The input of tabular data determines the content of the navigation control, as well as the appearance of each object in the navigation control. As with other controls, to configure a drill-down, set substitutions, or execute a command when a user clicks an navigation control node, use the **actionCommand** property. As with other table-driven objects, the **drillDownColumnSubs** property can be configured to set substitutions to column values from the row in the **valueTable** that corresponds to the selected navigation control node. If the navigation control is not large enough to display all of the data, scroll bars are automatically added.

You can configure user access control for nodes in navigation controls based on user roles. For example, the specified displays are not visible (nor accessible) to users with the associated role.

There are two types of navigation controls: the tree control (class name: **obj_c1tree**) and the accordion control (class name: **obj_c1accordion**). Both controls allow you to configure status icons and to propagate the status up to parent nodes.

Tree Control

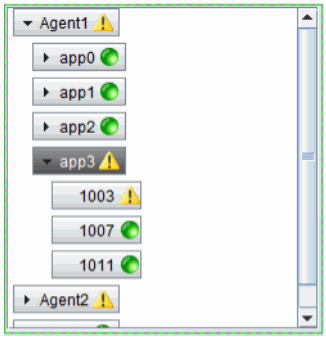
Multiple nodes can expand at a time and you can configure type icons for tree nodes (static images that visually indicate the element type, such as Production or Sales, or the node depth).



To configure, see "Tree Control".

Accordion Control

A single node expands at a time and icons visually indicate whether an element in the accordion is closed, open, or a leaf node.



To configure, see "Accordion Control".

Table Formats for Navigation Controls

There are two table format options for navigation controls, each with their own requirements:

- **Row-Leaf**: This format is intended for use when the **valueTable** property is attached to an indexed table and all leaves in the navigation control are at the same depth. For example, where the navigation control is attached to the current table of an indexed cache.

  Typically, the RTView cache data source is indexed. That is, the cache **indexColumnNames** property specifies the columns to index. If those index columns represent a hierarchy (for example, server, application, process or country, state, city), the cache current table could be used as the **valueTable** property of a control in the **Row-Leaf** format. In other cases, a table in the **Row-Node** format might be required.

  To configure, see **Creating a Row-Leaf Format Table** (below).

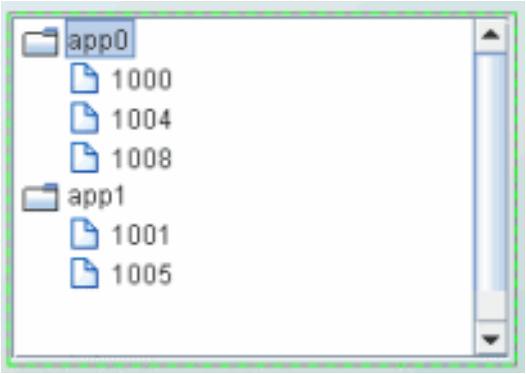
- **Row-Node**: If the **Row-Leaf** format is not suitable for your data, use the **Row-Node** format. This format is intended for use when the **valueTable** property is attached to an un-indexed table. Your data table must also contain a row for each node in the navigation control, including the top-level node (rather than just the leaf nodes, as with the Row-Leaf format), as well as a column for the node and a column for the parent node. The Row-Node format allows each leaf of the navigation control to have a different depth.

To configure, see **Creating a Row-Node Format Table** (below).

Note: If you have an XML file that has the **navtree.xml** format you can convert it into a row-node format. For details, see "navtree.xml For Navigation Control Objects".

The default table format is **Row-Leaf**. The following are examples of the Row-Leaf and Row-Node table formats which both produce the tree control below the table:

Row-Leaf		**Row-Node**	
App Name	**PID**	**Node**	**Parent**
App0	1000	app0	<blank>
App0	1004	1000	app0
App0	1008	1004	app0
App1	1001	1008	app0
App1	1005	app1	<blank>
		1001	app1
		1005	app1



Creating a Row-Leaf Format Table

In the Row-Leaf table format, there is one row in the table for each leaf node in the navigation control. A leaf node is added to the navigation control for each row in the **valueTable**. The path to a leaf node (that is, the ancestor nodes of the leaf) is determined by the values in each of the table columns specified by the **nodeIndexColumnNames** property. When the **valueTable** property is attached to the current table of an indexed RTView cache, the navigation control **nodeIndexColumnNames** property is typically set to the same columns that are specified for the cache index columns.

To illustrate how to create a navigation control using the **Row-Leaf** format, let us say we have a cache named Applications that has two index columns, **App Name** and **PID**, and the current table in that cache has the following rows:

App Name	PID
App0	1000
App0	1004
App0	1008
App1	1001
App1	1005

We set the navigation control **valueTable** property to attach to the cache data, with the following fields specified in the **Attach To Cache Data** dialog:

Cache: **Applications**

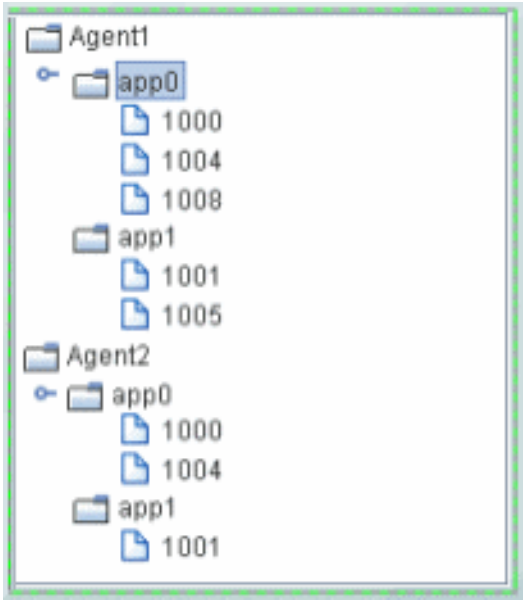
Table: **Current**

Specify **Row-Leaf** format for the **valueTableFormat** property, and **App Name;PID** for the **nodeIndexColumnNames** property. The following figure illustrates the structure of the navigation control. There are two nodes labeled **App0** and **App1**. Node **App0** has three child nodes labeled **1000**, **1004**, **1008**. Node **App1** has two child nodes, labeled **1001** and **1005**.

We add another index column, **AgentName**, to the **Applications** cache, so that the current table now has the following rows:

Agent Name	App Name	PID
Agent1	App0	1000
Agent1	App0	1004
Agent1	App0	1008
Agent1	App1	1001
Agent1	App1	1005
Agent2	App0	1000
Agent2	App0	1004
Agent2	App1	1001

We also update the navigation control **nodeIndexColumnNames** property to **AgentName;App Name;PID**. The following figure illustrates the new structure of the navigation control: two top-level nodes labeled **Agent1** and **Agent2**, each with two child nodes, app0 and app1.



As illustrated above, the label string for a node at depth **N** is taken from the **N**th column in **nodeIndexColumnNames** property. Therefore, the labels for the top-level nodes come from the first column in the **nodeIndexColumnNames** property (**AgentName**), the labels for the second-level nodes come from the second column in **nodeIndexColumnNames** property (**App Name**), and so forth.

To specify node labels from a different set of **valueTable** columns, use the **nodeLabelColumnNames** property. Enter a semicolon-separated list of column names in the **nodeLabelColumnNames** property, one for each level in the navigation control, where the **N**th column name in the list contains the labels for navigation control nodes at depth **N**.

Return to "Tree Control".

Return to "Accordion Control".

Creating a Row-Node Format Table

In the **Row-Node** table format, there is one row in the table for each node in the navigation control, including the top-level node (rather than just the leaf nodes, as with the Row-Leaf format).

Note: For a demo of an tree control that uses a data table in the row-node format, see the **treecon.rtv** display, located in the **RTV_HOME/demos/features** directory. The tree in that display is attached to an XML file (**navtree_rownode.xml**) that defines a navigation tree containing display titles and filenames in the row-node format. To view the demo: in an initialized command window, navigate to the **RTV_HOME/demos/features** directory, and type: **run_viewer -panelconfig:PANELS_treecon.ini**. See "Initializing a Command Prompt or Terminal Window" for more information on initializing a command window.

There are two columns in the table that help define the navigation control structure:

- One of the table columns contains a node ID string and is identified by the **nodeIdColumnName** property. By default, the node ID string is used as the node label in the navigation control. The node ID string must be unique among all nodes with the same parent. Or, if the **uniqueNodeIdFlag** property is checked, each node ID string must be unique in the entire navigation control.
- Another table column contains the ID of the parent node, which is identified by the **parentIdColumnName** property.

To create the same navigation control as with our **Row-Leaf** format example, a table representation of the navigation control using the **Row-Node** format would be as follows:

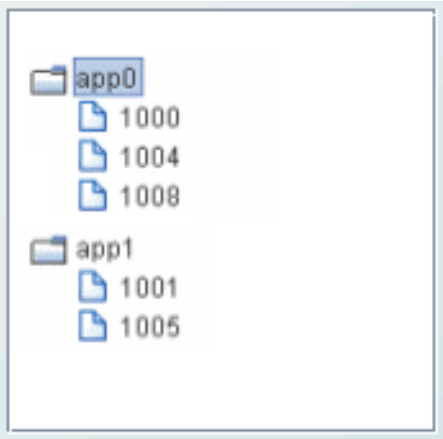
Node	Parent
app0	<blank>
1000	app0
1004	app0
1008	app0
app1	<blank>
1001	app1
1005	app1

The **<blank>** entries represent an empty string, which indicates that nodes **app0** and **app1** have no parent, making them top-level nodes in the navigation control.

We set the navigation control:

- **valueTable** property to attach to the unindexed table
- **valueTableFormat** property to the **Row-Node** format
- **nodeIdColumnName** property to **Node**
- **parentIdColumnName** property to **Parent**

We then have the following navigation control structure. There are two top-level nodes labeled **App0** and **App1**. Node **App0** has three child nodes labeled **1000**, **1004**, **1008**. Node **App1** has two child nodes, labeled **1001** and **1005**.

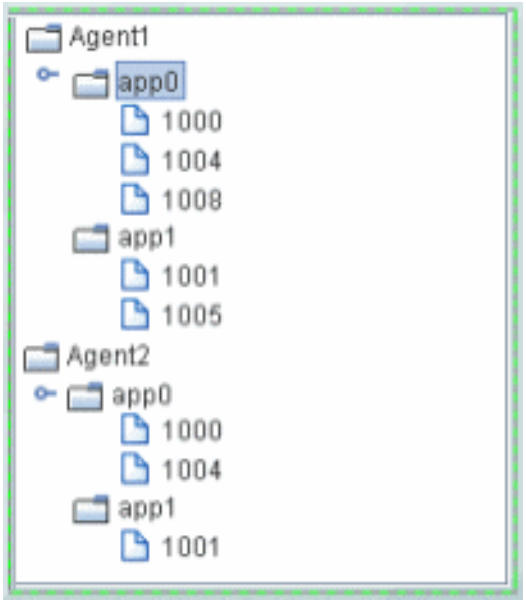


To add another navigation control level for the **AgentName**, as we did with the **Row-Leaf** format example, we modify our table as follows:

Node	Parent
Agent1	<blank>
app0	Agent1
1000	app0
1004	app0
1008	app0
Agent2	<blank>
app0	Agent2
1000	app0
1004	app0
app1	Agent2
1001	app1
1005	app1

We also uncheck the **uniqueNodeIdFlag** property to allow for node IDs that are not unique, such as the **app0** and **1000** nodes in our example which are used in multiple navigation control levels. Because some of the rows are also identical, the order of the table rows is important. For example, this row appears twice in the table: **1000 app0**. In each case the row comes after a unique parent row: first under **app0 Agent1** and then under **app0 Agent2**. The navigation control uses this information to determine where to place the node for **1000** in each case.

The following figure illustrates the new structure of the navigation control. The navigation control now has two top-level nodes labeled **Agent1** and **Agent2**, each with two child nodes, **app0** and **app1**.



By default, the node ID string is used as the node label in the navigation control. If a different column in the table must provide the label, specify the name of that column in the **nodeLabelColumnName** property.

Note that in the **Row-Node** format each branch of the navigation control can have a different depth, while in the **Row-Leaf** format all branches typically have the same depth (the number of columns specified in the **nodeIndexColumnNames** property).

Return to "Tree Control".

Return to "Accordion Control".

Status Icons

Status icons indicate the current state of a node. You can optionally configure status icons by assigning images that are used when an element in the navigation control has a specified value. You can also configure the status in order of priority so that the most critical status is propagated up to ancestor nodes. The status icon shown for an ancestor node corresponds to the current highest status priority of all of its descendants. By default, no status icons appear in navigation controls.

The status icon for a node is determined by the discrete value of a specified column in the **valueTable**. The values can be strings or numbers. The comparison is done for an exact match. If the current status value for a navigation control node does not match any of the values you specify in the **nodeStatusProperties** property, no status icon is displayed for that node.

When a status icon is configured for the tree control, by default the status icon appears on the left of the node label. The value, **Left** or **Right**, is specified in the **nodeStatusIconPos** property. If a node has both a type icon and a status icon, the type icon always appears to the left of the status icon. By default, no status icons appear in the tree.

When a status icon is configured for the accordion control, by default the status icon appears inside the node button, either to the left or right of the button label, according to the value of the **nodeStatusIconPos** property.

Configuring Status Icons

This section describes how to configure status icons for the navigation controls using the **Node Properties** dialog.

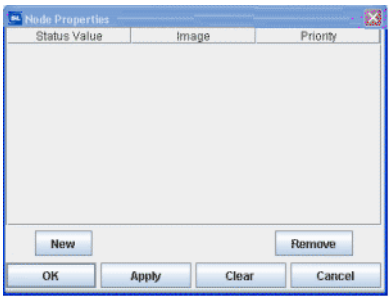
Note: Status icons can also be configured by Attaching a Status Icon To Data (see below).

To configure the Status Icon

To configure the status icon, specify the status table column name in the **nodeStatusColumnName** property, then use the **nodeStatusProperties** property to define a three-column table of data and configure icon behavior.

Note: The **nodeStatusProperties** property is only visible if the **nodeStatusColumnName** property is non-blank.

Select the **nodeStatusProperties** property in the property sheet, then click the ... button to open the **Node Properties** dialog.



To map an image to a node status

Click **New**, then click in the **Status Value** column. A drop-down list appears containing all values in the node status column of the **valueTable** (which you previously specified in the **nodeStatusColumnName** property). Select a value from the drop-down list.

Click the ... button in the **Image** column for the node and choose an icon from the **Select Image** dialog. The icon is used as the status icon for all nodes that match the value selected in the **Status** column.

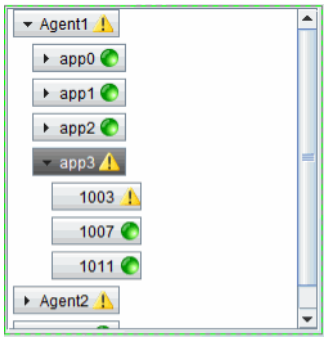
Click the ... button in the **Priority** column for the node and assign a integer value: **0**, **1**, **2**, **3**, **4**, **5**, or a higher number (there is no upper limit on the number), where the largest number is the highest priority and propagated up the navigation control first, and a value of zero (**0**) is not propagated. A value of zero (**0**) can be assigned to multiple nodes (so that they do not propagate up the navigation control). Non-zero values must be assigned only once.

For example, if we set the **nodeStatusColumnName** property to our **Application Status** table column of our **valueTable**, and define the mapping for the **nodeStatusProperties** as follows:

Status Value	Image	Priority
blocked		2
running		1

The values in the **Application Status** column of each row in the **valueTable** is compared to the two values listed in the **Status Value** column (**blocked** and **running**). If the **Application Status** value in one of the rows is blocked, the  status icon is displayed as the status icon for that row. If there is no match (for example, the **Application Status** value in one of the rows is unknown), no status icon is displayed in the navigation control node for that row. Because the  status icon is assigned the highest priority, the  status icon is always propagated up the navigation control first. If none of the rows has a blocked status, the  status icon is propagated up the navigation control.

For example, in the following illustration the priority status of a single node, **app3/1003**, is propagated up to its parent, **app3**, and also to the top-level ancestor, **Agent1**.



For details about creating an application with multiple panels, see "Multiple Display Panels".

Attaching a Status Icon To Data

For convenience, the status icon can be attached to data. Typically, an attachment to a static XML file containing the appropriate tables is used. To attach the status icon to data, use the **nodeStatusProperties** property. The data attachment must be a three-column table. Typically, a static XML file containing the table is used. The first column must contain string values that match values from the column in **valueTable** specified by the **nodeStatusColumnName** property. The second column must be the path to the **.png**, **.gif**, or **.jpg** image. The third column must contain the non-negative integer priority value.

Note: A static XML file is only read once each time you run RTView. If you specify (or modify) an XML source using the **Application Options** dialog, you may specify whether that XML source is static. For details, see "Creating XML Sources".

---

navtree.xml For Navigation Control Objects

This section describes how to populate a navigation control object from an XML data source using the **navtree.xml** format. Using the **navtree.xml** format enables the XML data source to parse the **navtree.xml** content and convert it into a table in the **row-node** format, which is a format accepted by "Navigation Control Objects". A navigation control--the tree control (class name: **obj_c1tree**) or the accordion control (class name: **obj_c1accordion**)--can then be used on an RTView display.

When the XML data source parses the **navtree.xml** file it creates a table and gives it the data key **navtree**. When you use the **Builder Attach to XML Data** dialog, enter **navtree** in the **Data Key** field. The table has one row for each node that is defined in the **navtree.xml** file. The table contains the following columns:

Node ID:	An unique integer assigned to a navigation control object node.
Parent ID:	The ID of the node's parent, or blank if the node is a top-level node.
Label:	The value of the node's **label** attribute.
Display:	The value of the node's **display** attribute.
Subs:	The value of the node's **subs** attribute (a string containing **sub:value** pairs, separated by spaces).

Typically, you attach the **valueTable** property of a navigation control object to the **navtree** table, and the navigation control object's **valueTableFormat** property is set to **Row-Node**. Subsequently, the column names listed above are specified as values for other properties on the navigation control as follows:

```
nodeIdColumnName = Node ID
nodeLabelColumnName = Label
parentIdColumnName = Parent ID
uniqueNodeIdFlag = <checked>
valueColumnName = Display
```

To see an example of an accordion control configured in this manner, use the Builder to examine the display named **accordcon.rtv**, located in the **RTV_HOME/demos/features** directory. The XML file used by the example is **demos/features/navtree.xml**. To see the working example, type the following commands in an RTView command prompt:

cd %RTV_HOME%\demos\features

run_viewer -panelconfig:PANELS_accordcon.ini

In addition to the columns mentioned above, the **navtree** table's **Subs** column can also be used to configure the **drilldownColumnSubs** property of a navigation control. In the "Drill Down Column Substitutions Dialog", find the row for **Column Name = Subs**, and in the **Substitution String** column enter the string **__parse** (there are two underscores in the prefix), that is:

Column Name	Substitution
Subs	__parse

This tells RTView to parse the string found in the **Subs** column for the node selected in the navigation control, which is expected to contain one or more **sub:value** pairs, separated by spaces. For example, if the **Subs** column contains this string:

$sub1:abc $sub2:xyz

and if the navigation control's **drilldownColumnSubs** property is configured as described, a drilldown from the navigation control the **$sub1** and **$sub2** substitutions is assigned the values **abc** and **xyz**, respectively.

Tree Control

This section describes how to create a tree control (class name: **obj_c1tree**).

There are two methods for creating a tree-driven multi-panel application: the static tree navigation panel and the tree control. Use the static tree navigation panel method if you know the specific sources that are to populate the tree and they remain constant for the life of the application. For example, if you know all the displays that compose your application and the static representation of a tree will only be used for navigating those displays, the static tree navigation panel is suitable (and is easier to configure). To configure the static tree navigation panel, add the tree using the **rtvTreePanel** tag to the **PANELS.ini** file. For details about configuring the tree, see "Multiple Display Panels".

Use the tree control method if the number of nodes or leaves, labels or icons of the tree change during the lifetime of the application. Data can be provided that will change the nodes and leaves of the tree and also change the labels, and icon representations on the tree with dynamic data.

When constructing an application with multiple panels using the tree control, one panel displays an RTView file that has instanced the tree control and the other contains the displays which are drilled down to by selecting items on the tree. The following illustrates a two-panel application in which the tree control in the left panel updates the display in the right panel:

To see an example of a multi-panel layout that uses a tree control to display live status information (based on alerts) and perform navigation via drill-down in another panel, see the "Navigation Control Demo".

For details about creating an application with multiple panels, see "Multiple Display Panels".

Creating a Tree Control

To create a tree control:

1. Attach tabular data to the tree control **valueTable** property.

2. Specify the table format for the tree in the **valueTableFormat** property. There are two table format options for navigation controls. The **Row-Leaf** format is intended for use when the **valueTable** property is attached to an indexed table and all leaves in the navigation control are at the same depth. Otherwise, the **Row-Node** format is used. See "Navigation Control Objects" for details.

3. Optionally, configure type icons.

4. Optionally, configure status icons.

Type Icons

Type icons indicate the type of node in the tree. The type icon for each node is determined either by the value of a column in the **valueTable** property, or by the node position in the tree. By default, the type icon appears to the left of the node label. Also by default, a folder image is used for all non-leaf nodes, and a document image is used for all leaf nodes. If a node has a type icon and a status icon, the type icon appears to the left of the status icon.

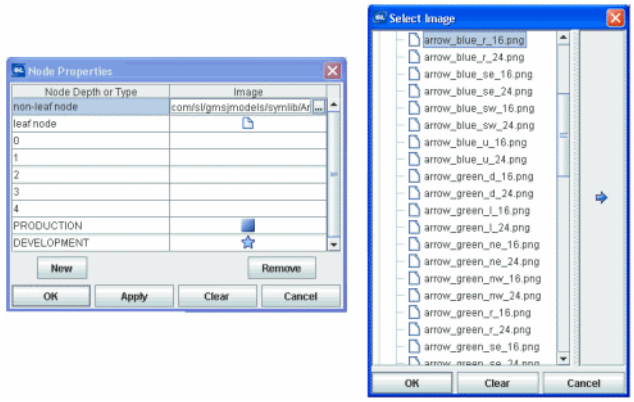
You can optionally assign images for type icons. For example, if you have groups of elements based on geographic location, you could assign an icon for the country, state and city (for example, US, California, San Francisco). Or, if you have groups of elements based on their function, you could assign an icon for each function (Purchases, Operations, Sales, and so forth). You can also assign images for each depth in the tree for a visual indication of where you are while navigating within the tree.

Configuring Type Icons

This section describes how to configure status icons for the navigation controls using the **Node Properties** dialog.

Note: Status icons can also be configured by Attaching a Status Icon To Data.

Use the **nodeTypeProperties** property to define a two-column table of data. Select the **nodeTypeProperties** property in the property sheet, then click the ... button to open the **Node Properties** dialog.



In the **Node Properties** dialog, the **Node Depth or Type** column lists the available nodes. The first two rows, **non-leaf node** and **leaf node**, indicate the default settings: non-leaf nodes in the tree use a folder image and leaf nodes use a document image. To change the default setting, click the ... button in the **Image** column for the node and choose an icon from the **Select Image** dialog.

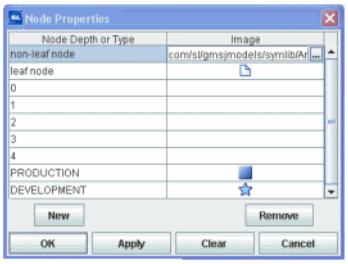
The next five rows, numbered **0 - 4**, represent the node depth or level, zero (0) being the root node. The **Image** column lists the icons being used for each node. By default, the **Image** column for all of those rows is **<blank>**, indicating that the **non-leaf node** and **leaf node** icon assignments are used. Icon assignments override non-leaf node and leaf node assignments.

Mapping an Image to a Node Level

You can assign an image icon based on node level. The icon provides a visual indication of where you are while navigating in the tree. To assign an image to a specific node level in the tree, click the ... button for one of the rows numbered **0 - 4** in the **Image** column and choose an icon from the **Select Image** dialog. Repeat for each node level.

Mapping an Image to a Node Type

You can assign an image icon based on node labels you create that describe the nodes as a group. For example, in the following illustration the **Node Depth** or **Type** column contains the string **PRODUCTION** with the image selected, and the string **DEVELOPMENT** with the image selected.



This means that all nodes whose label matches the **PRODUCTION** string display the image in the tree, and all nodes whose label matches the **DEVELOPMENT** string display the image in the tree.

To assign an image to a specific node type in the tree, assign a column name in the **nodeTypeColumnName** property. Select the **nodeTypeProperties** property in the property sheet, then click on the ... button to open the **Node Properties** dialog. Click **New** to add a custom row to the table. A drop-down list of values for the column assigned to the **nodeTypeColumnName** property appears in the **Node Depth** or **Type** column. Select a column name from the drop-down list. Click the ... button in the **Image** column and choose an icon (to use for all nodes that have that column name in the **valueTable** row that corresponds to the node) from the **Select Image** dialog.

You can also type a string in the **Node Depth** or **Type** column and the **Image** column.

To not use an icon, in the **Node Properties** dialog, select the icon in the **Image** column, then click **Clear**.

Note: The root node is invisible if the **rootNodeLabel** property is blank.

For details about type icon determination, see Logic for determining type icon (below).

For details about creating an application with multiple panels, see "Multiple Display Panels".

Attaching an Icon To Data

For convenience, the type icon can be attached to data. Typically, an attachment to a static XML file containing the appropriate tables is used. To attach the type icon to data, use the **nodeTypeProperties** property. The data attachment must be a two-column table. The first column must contain string values of **_node** (for non-leaf nodes), **_leaf** (for leaf nodes), numeric values for depth, or string values that match the node labels, or the values from the column in **valueTable** specified by the **nodeTypeColumnName** property. The second column must be the path to the **.png**, **.gif**, or **.jpg** image. The default assignments are **_node**, **rtvTreeNode16.png** and **_leaf, rtvTreeLeaf16.png**. The column names are not important.

Note: A static XML file is only read once each time you run RTView. If you specify (or modify) an XML source using the **Application Options** dialog, you may specify whether that XML source is static. For details, see "Creating XML Sources".

Background Properties

Specify how the background is displayed in the tree control.

Property Name	Description
bgBorderFlag	When checked, a one pixel dark border is drawn around the tree. If unchecked, no border is drawn. By default, the flag is checked and the border is visible. **Note:** The border color and width are not configurable.
bgColor	Select the  button and choose a color from the palette to set the background color of the tree control.

Data Properties

Specify how data is displayed in the tree control.

Property Name	Description
nodeIdColumnName	This property is available when the **valueTableFormat** is **Row-Node**. With the **Row-Node** format there are two table columns that define the tree structure: the **nodeIdColumnName** property and the **parentIdColumnName** property specify the two columns. The **nodeIdColumnName** property specifies the table column containing the node ID string. The node ID string must be unique among all nodes with the same parent. Or, if the **uniqueNodeIdFlag** property is checked, each node ID string must be unique in the entire tree. By default, the node ID string is used as the node label in the tree.
nodeIndexColumnNames	This property is available when the **valueTableFormat** is **Row-Leaf**. Specifies the path to a leaf node (that is, the ancestor nodes of the leaf). When the **valueTable** property is attached to the current table of an indexed RTView cache, the **nodeIndexColumnNames** property is typically set to the same columns that are specified for the cache index columns. Enter a semicolon-separated list of column names, where the Nth column name in the list contains the labels for tree nodes at depth N. The labels for top-level nodes are defined by the first column in the **nodeIndexColumnNames** property, the labels for the second-level nodes are defined by the second column, and so forth.

	For example: **AgentName;App Name;PID** The labels for the top-level nodes are defined by the **AgentName** column, the labels for the second-level nodes are defined by the **App Name** column, and labels for the third-level nodes are defined by the PID column. To specify node labels from a different set of **valueTable** columns, use the **nodeLabelColumnNames** property.
nodeLabelColumnName	This property is available when the **valueTableFormat** is **Row-Node**. By default, the node ID string is used as the node label in the tree. Use the **nodeLabelColumnName** property to specify a different **valueTable** column to provide the label.
nodeLabelColumnNames	This property is available when the **valueTableFormat** is **Row-Leaf**. Use the **nodeLabelColumnNames** property to specify a different set of **valueTable** columns to provide node labels. Enter a semicolon-separated list of column names, one for each level in the tree, where the Nth column name in the list contains the labels for tree nodes at depth N.
nodeStatusColumnName	This property applies to the status icon. Specifies the name of the **valueTable** column containing node status values. The column specified populates the **Node Properties** dialog **Status Value** column, in which you map node status values to image icons. The icons are displayed for any node whose value matches the value selected.
nodeTypeColumnName	This property applies to the type icon. Specifies the name of the **valueTable** column containing values to use for mapping icon images to node types in the tree. The column specified populates the list of available values in the **Node Properties** dialog **Node Depth or Type** column, in which you map node types to image icons. The icons are displayed for any node whose value matches the value selected.
parentIdColumnName	This property is available when the **valueTableFormat** is **Row-Node**. With the **Row-Node** format there are two table columns that define the tree structure: the **parentIdColumnName** property and the **nodeIdColumnName** property specify the two columns. The **parentIdColumnName** property specifies the table column containing the parent node ID.
roleCheckColumnName	The name of a string column in the data table containing display the names to be checked against the user's role. The default value is Display. Use this property to show/hide tree nodes based on user role definitions. If a row's value in the specified column is the name of a display to which the user's role is denied access, the tree does not show a node for that row. For details about user roles and display access, see "Role Definitions".
selectedValue	This column specifies the value of the currently selected node in the tree.
selectedValueColumnName	If specified, the values in this column will be compared to the **selectedValue** in order to set the tree selection. If not specified, the **selectedValue** will be compared to the values in the column specified in **valueColumnName** property.
uniqueNodeIdFlag	This property is available when the **valueTableFormat** is **Row-Node**. When enabled, specifies that each node ID string must be unique in the entire tree. When disabled, specifies that each node ID string must be unique among all nodes with the same parent.
valueColumnName	Specifies the name of the column whose value is assigned to the **$value** variable when a node in the tree is selected. If not specified, the label string of the selected node is assigned to the **$value** variable. The **$value** variable is the only substitution that can be used in the **Display Name** field of a drill-down command.

valueTable	Attach your tabular input data to this property. There are two **valueTable** format options you can use to create your table: • **Row-Leaf** format: See "Navigation Control Objects" for details about creating the table. • **Row-Node** format: See Row-Node for details about creating the table. If you have an XML file that has the navtree.xml format you can convert it into a row-node format. For details, see "navtree.xml For Navigation Control Objects". **Note:** As with other table-driven objects, the drillDownColumnSubs property can be configured to set substitutions to column values from the row in the valueTable that corresponds to the selected tree node.
valueTableFormat	Specifies the format of the valueTable: **Row-Leaf** or **Row-Node**.
varToSet	Allows you to update the attached variable with the value from the control.

Interaction Properties

Specify interactions in the tree control.

Property Name	Description
actionCommand	Use the **actionCommand** property to assign a command to the tree. You can configure the tree to open a drill-down display, set substitutions, or execute a command in response to a user click on a tree node. The **actionCommand** can reference the value from the tree by using the keyword **$value**. When the command is executed, the variable attached to **varToSet** is updated with the selected node data. The **drillDownColumnSubs** property can be configured to set substitutions to column values from the row in the **valueTable** that corresponds to the selected tree node. If the **execOnLeafOnlyFlag** property is checked, the tree **actionCommand** property executes only when a leaf node is clicked (a click on a non-leaf node expands only the node). If unchecked, the tree **actionCommand** property executes on all nodes, not just the leaf.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore, if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the Property Value field or select the  button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.

drillDownColumnSubs	Use the **drillDownColumnSubs** property to set substitutions to column values from the row in the **valueTable** that corresponds to the selected tree node. Select the ... button to open the "Drill Down Column Substitutions Dialog" to customize which substitutions are passed into drill-down displays.
drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information.
enabledFlag	If unchecked, the tree nodes are the color gray and do not respond to user input.
execOnLeafOnlyFlag	If checked, the tree **actionCommand** is executed only for leaf nodes, and a click on a non-leaf node only expands the node. Also, the mouseover tooltip only appears for leaf nodes. If unchecked, the tree **actionCommand** property executes on all nodes, and the mouseover tooltip appears for all nodes.
menuItemGroup	Use the **menuItemGroup** property to extend RTView context menu items. For details, see "Extending the Context Menu".
mouseOverFlag	Specifies whether a tooltip appears when the cursor is positioned over a node. The tooltip shows the node path (the node label preceded by the labels of all of its ancestors), the node status (if the nodeStatusColumnName property is specified), and its value (if the valueColumnName property is specified).
rightClickActionFlag	Use the **rightClickActionFlag** property to extend RTView context menu items. For details, see "Extending the Context Menu".
tabIndex	Use the **tabIndex** property to define the order in which the tree receives focus when navigated from your keyboard. Initial focus is given to the object with the smallest **tabIndex** value, from there the tabbing order proceeds in ascending order. If multiple objects share the same **tabIndex** value, initial focus and tabbing order are determined by the alpha-numeric order of the table names. Tables with a **tabIndex** value of 0 are last in the tabbing order. **Note:** The **tabIndex** property does not apply to tables in the Display Server, nor to objects that are disabled, invisible, or have a value of less than 0.

Label Properties

Specify the label in the tree control.

Property Name	Description
labelTextColor	Select the ... button and choose a color from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop-down menu.
labelTextSize	Set the height of the label text in pixels.

Node Properties

Specify the node structure in the tree control.

Property Name	Description
initialExpandDepth	Specify the depth that nodes automatically expand (open) to when the tree is populated. All non-leaf nodes at a depth that is less than or equal to the **initialExpandDepth** property are expanded automatically. By default, the value is **0** (zero). When calculating node depth, the root node has a depth of zero. (All trees have a root node but it is visible only if the **rootNodeLabel** property is set). Because the default value of **initialExpandDepth** is zero, only the root node is expanded automatically. Changing the value of **initialExpandDepth** from a larger value to a smaller value has no effect until the display containing the tree is reopened. **Note:** Use this property with caution on large trees as the automatic expansion of a large number of nodes can cause slow performance.
nodeClosedImage	This property specifies the name of the images to be shown to the left of a closed parent (non-leaf) tree node. This property is blank by default, which indicates that the default closed images should be used.
nodeOpenImage	This property specifies the name of the images to be shown to the left of an open parent (non-leaf) tree node. This property is blank by default, which indicates that the default open images should be used.
nodeSelectColor	This property sets the background color for the selected tree node. If set to **Default**, then the platform or browser default selection color is used.
nodeSelectTextColor	This property sets the text color for the selected tree node. If set to **Default**, then the platform or browser default selection color is used.
nodeStatusIconPos	Specify the status icon position in the tree: **Left** or **Right**. By default, the status icon appears on the left of the node label. If a node has both a type icon and a status icon, the type icon always appears to the left of the status icon. By default, no status icons appear in the tree.
nodeStatusProperties	This property applies to the status icon. Specifies the status icon for a node. By default, no status icon is displayed. Select the  button to open the Node Properties dialog and map images to values, and set the status priority order for propagation up the tree. **Note:** The **nodeStatusProperties** property is visible only if the **nodeStatusColumnName** property is non-blank. You can also use the **nodeStatusProperties** property to attach a status icon to data. The data attachment must be a three-column table. Typically, a static XML file containing the table is used. The first column must contain string values that match values from the column in **valueTable** specified by the **nodeStatusColumnName** property. The second column must be the path to the .png, .gif, or .jpg image. The third column must contain the non-negative integer priority value. **Note:** A static XML file is only read once each time you run RTView. If you specify (or modify) an XML source using the **Application Options** dialog, you may specify whether that XML source is static. For details, see "Creating XML Sources".

nodeTypeProperties	This property applies to the type icon. Specifies the type icon for a node. By default, non-leaf nodes in the tree use a folder image and leaf nodes use a document image. Select the ... button to open the **Node Properties** dialog to map images to nodes. Mapping can be based on the node depth in the tree or the type of node. You can also use the **nodeTypeProperties** property to attach a type icon to data. The data attachment must be a two-column table. Typically, a static XML file containing the table is used. The first column must contain string values of **_node** (for non-leaf nodes), **_leaf** (for leaf nodes), numeric values for depth, or string values that match the node labels, or the values from the column in **valueTable** specified by the **nodeTypeColumnName** property. The second column must be the path to the .png, .gif, or .jpg image. The default assignments are **_node, rtvTreeNode16.png** and **_leaf, rtvTreeLeaf16.png**. The column names are not important. Logic for determining which type icon is used: If the **nodeTypeColumnName** property specifies column **C**, and the value of **C** in the **valueTable** row that corresponds to **N** is **V**, and there is a row in **nodeTypeProperties** that assigns value **V** to image **I1**, **I1** is used as the type icon for **N**. Otherwise: • if the label of node **N** is **XYZ**, and there is a row in the **nodeTypeProperties** property that assigns value **XYZ** to image **I2**, **I2** is used. Otherwise, • if the depth of node **N** is **D**, and there is a row in the **nodeTypeProperties** property that assigns depth **D** to image **I3**, **I2** is used. Otherwise, • if **N** is a leaf, and the leaf node image is **I4**, **I4** is used. If **I4** is blank no type icon appears. Otherwise, • if the non-leaf node image is **I5**, **I5** is used. If **I5** is blank no type icon appears. **Note:** A static XML file is only read once each time you run RTView. If you specify (or modify) an XML source using the **Application Options** dialog, you may specify whether that XML source is static. For details, see "Creating XML Sources".
rootNodeLabel	Specify whether the tree root node is visible. By default, this property is blank and the root node is not visible.

Object Properties

Specify the layout in the tree control.

Property Name	Description
anchor	Select the ... button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.

dock	Specify the docking location of an object in the display. **Note:** An object should not be docked if the **Resize Mode** is set to **Scale**. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
visFlag	Set the visibility of the object.

Unique Behavior

The following describes properties that behave uniquely with the tree control.

valueColumnName: This property specifies the name of the column whose value should be assigned to the **$value** variable when a node in the tree is clicked. If not specified, the label string of the selected node is assigned to **$value**. Note the **$value** is the only substitution that can be used in the **Display Name** field of a drill-down command.

mouseOverFlag: If this property is checked, a tooltip appears when the cursor is positioned over a leaf node. The tooltip shows the node path (the node label preceded by the labels of all of its ancestors), the node status (if the **nodeStatusColumnName** property is specified), and its value (if the **valueColumnName** property is specified).

execOnLeafOnlyFlag: If this property is checked, the tree **actionCommand** property executes only when a leaf node is clicked (a click on a non-leaf node expands only the node). If unchecked, the tree **actionCommand** property executes on all nodes, not just the leaf.

rootNodeLabel: This property specifies the tree root node (of which there is only one). By default, this property is blank and the root node is not visible.

Limitations

- In the Display Viewer, mouseover text is displayed only if the tree has focus.
- If the **valueTableFormat** property for the tree control is **Row-Leaf**, the context menu feature applies to a right-click or a double-click on leaf nodes only. That is, a right-click on a non-leaf node does not add items to the menu and a double-click on a non-leaf node simply expands or collapses the tree node, it does not invoke the double-click action.

In the Thin Client:

- The tree node appearance, such as spacing and fonts, might vary slightly as compared to the Display Viewer, and also may vary slightly between different browsers.
- A tree node cannot expand/collapse by double-clicking on it. The **+ / -** icon must be clicked.
- In Internet Explorer, nodes expand/collapse even if the tree **enabledFlag** property is unchecked. (However, the tree **actionCommand** cannot be invoked).
- In Mozilla Firefox, the horizontal scrollbar might appear and disappear after each mouse click in the tree.
- In iOS Safari (iPad), if the tree **mouseOverFlag** property is checked, a user must click a tree node twice to invoke the tree command. The first click only displays the node mouseover text.
- In iOS Safari (iPad), scrollbars will not appear in a tree control. If the tree contains more nodes than are visible, use a two-finger drag gesture inside the tree area to scroll.
- In iOS Safari, a click on the **+ / -** icon expands/collapses the node as expected. However, if the **execOnLeafOnlyFlag** property is unchecked, the tree command is also executed.
- In Internet Explorer, if you double-click directly on a leaf node the node is selected but the double-click action is NOT invoked. The double-click action is invoked only if you click on the empty space in the tree to the left or the right of the leaf node, or in the small hollow square located to the left of the leaf node.

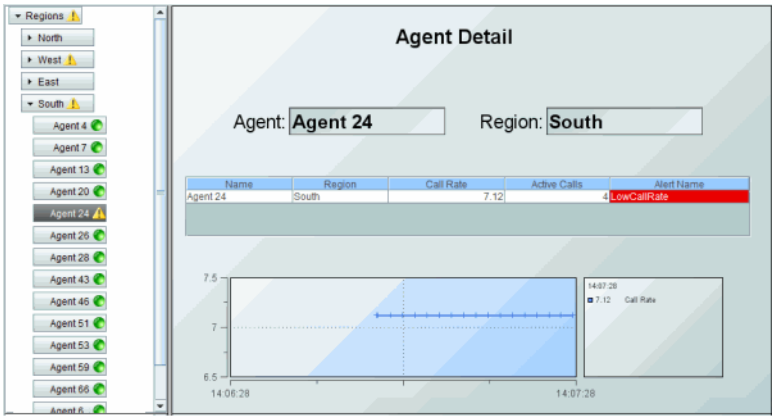
Accordion Control

This section describes how to create an accordion control (class name: **obj_c1accordion**).

There are two methods for creating an accordion-driven multi-panel application: the static accordion navigation panel and the accordion control. Use the static accordion navigation panel method if you know the specific sources that are to populate the accordion and they remain constant for the life of the application. For example, if you know all the displays that compose your application and the static representation of an accordion will only be used for navigating those displays, the static accordion navigation panel is suitable (and is easier to configure). To configure the static accordion navigation panel, add the accordion using the **rtvAccordionPanel** tag to the **PANELS.ini** file. For details about configuring the accordion, see "Multiple Display Panels".

Use the accordion control method if the number of nodes or leaves, labels or icons of the accordion change during the lifetime of the application. Data can be provided that will change the nodes and leaves of the accordion and also change the labels, and icon representations on the accordion with dynamic data.

When constructing an application with multiple panels using the accordion control, one panel displays an RTView file that has instanced the accordion control and the other contains the displays which are drilled down to by selecting items on the accordion. The following illustrates a two-panel application in which the accordion control in the left panel updates the display in the right panel:



To see an example of a multi-panel layout that uses an accordion control to display live status information and perform navigation via drill-down in another panel, see the "Navigation Control Demo".

For details about creating an application with multiple panels, see "Multiple Display Panels".

Creating an Accordion Control

To create an accordion control:

1. Attach tabular data to the accordion control **valueTable** property.
2. Specify the table format for the accordion in the **valueTableFormat** property. There are two table format options for navigation controls. The Row-Leaf format is intended for use when the **valueTable** property is attached to an indexed table and all leaves in the navigation control are at the same depth. Otherwise, the Row-Node format is used. See "Navigation Control Objects" for details.
3. Optionally, configure accordion icons.
4. Optionally, configure status icons.

Accordion Icons

Nodes in an accordion control appear as pushbuttons and a single branch is expanded at a time. Accordion control icons indicate whether an element is closed, open, or a leaf node.

Each node in an accordion is either a parent node or a leaf node. A parent node has one or more child nodes that are visible when the node is expanded (opened) and are invisible when the node is collapsed (closed). When a parent node is expanded, a small down arrow icon is displayed near the left edge of the button. When a parent node is collapsed, a small right arrow icon is displayed.

A leaf node has no children and thus cannot be expanded or collapsed.

Configuring Accordion Icons

The images used for these icons are specified by the **accordionOpenImage** and **accordionClosedImage** properties, respectively. Note that a leaf node never displays either of these icons.

The color of accordion node buttons that are not selected is specified by the **accordionColor** property, the color of the accordion node button that is selected is specified by the **accordionSelectColor** property, the text color for the accordion node that is selected is specified by the **accordionSelectTextColor** property and the width for each accordion button is specified by the **accordionWidth** property.

Background Properties

Specify how the background is displayed in the accordion control.

Property Name	Description
bgBorderFlag	When checked, a one pixel dark border is drawn around the accordion. If unchecked, no border is drawn. By default, the flag is checked and the border is visible. **Note:** The border color and width are not configurable.
bgColor	Select the  button and choose a color from the palette to set the background color of the accordion control.

Data Properties

Specify how data is displayed in the accordion control.

Property Name	Description
nodeIdColumnName	This property is available when the **valueTableFormat** is **Row-Node**. With the **Row-Node** format there are two table columns that define the accordion structure: the **nodeIdColumnName** property and the **parentIdColumnName** property specify the two columns. The **nodeIdColumnName** property specifies the table column containing the node ID string. The node ID string must be unique among all nodes with the same parent. Or, if the **uniqueNodeIdFlag** property is checked, each node ID string must be unique in the entire accordion. By default, the node ID string is used as the node label in the accordion.
nodeIndexColumnNames	This property is available when the **valueTableFormat** is **Row-Leaf**. Specifies the path to a leaf node (that is, the ancestor nodes of the leaf). When the **valueTable** property is attached to the current table of an indexed RTView cache, the **nodeIndexColumnNames** property is typically set to the same columns that are specified for the cache index columns. Enter a semicolon-separated list of column names, where the Nth column name in the list contains the labels for accordion nodes at depth N. The labels for top-level nodes are defined by the first column in the **nodeIndexColumnNames** property, the labels for the second-level nodes are defined by the second column, and so forth. For example: **AgentName;App Name;PID** The labels for the top-level nodes are defined by the **AgentName** column, the labels for the second-level nodes are defined by the **App Name** column, and labels for the third-level nodes are defined by the **PID** column. To specify node labels from a different set of **valueTable** columns, use the **nodeLabelColumnNames** property.
nodeLabelColumnName	This property is available when the **valueTableFormat** is **Row-Node**. By default, the node ID string is used as the node label in the accordion. Use the **nodeLabelColumnName** property to specify a different **valueTablecolumn** to provide the label.

nodeLabelColumnNames	This property is available when the **valueTableFormat** is Row-Leaf. Use the **nodeLabelColumnNames** property to specify a different set of **valueTablecolumns** to provide node labels. Enter a semicolon-separated list of column names, one for each level in the accordion, where the Nth column name in the list contains the labels for accordion nodes at depth N.
nodeStatusColumnName	This property applies to the status icon. Specifies the name of the **valueTable** column containing node status values. The column specified populates the **Node Properties** dialog **Status Value** column, in which you map node status values to image icons. The icons are displayed for any node whose value matches the value selected.
parentIdColumnName	This property is available when the **valueTableFormat** is **Row-Node**. With the Row-Node format there are two table columns that define the accordion structure: the **parentIdColumnName** property and the **nodeIdColumnName** property specify the two columns. The **parentIdColumnName** property specifies the table column containing the parent node ID.
roleCheckColumnName	The name of a string column in the data table containing the display names to be checked against the user's role. The default value is **Display**. Use this property to show/hide accordion nodes based on user role definitions. If a row's value in the specified column is the name of a display to which the user's role is denied access, the accordion does not show a node for that row. For details about user roles and display access, see "Role Definitions".
selectedValue	This column specifies the value of the currently selected node in the accordion.
selectedValueColumnName	If specified, the values in this column will be compared to the **selectedValue** in order to set the accordion selection. If not specified, the **selectedValue** will be compared to the values in the column specified in **valueColumnName** property.
uniqueNodeIdFlag	This property is available when the **valueTableFormat** is **Row-Node**. When enabled, specifies that each node ID string must be unique in the entire accordion. When disabled, specifies that each node ID string must be unique among all nodes with the same parent.
valueColumnName	Specifies the name of the column whose value is assigned to the **$value** variable when a node in the accordion is selected. If not specified, the label string of the selected node is assigned to the **$value** variable. The **$value** variable is the only substitution that can be used in the **Display Name** field of a drill-down command.
valueTable	Attach your tabular input data to this property. There are two **valueTable** format options you can use to create your table: • **Row-Leaf format**: See "Navigation Control Objects" for details about creating the table. • **Row-Node format**: See "Navigation Control Objects" for details about creating the table. If you have an XML file that has the navtree.xml format you can convert it into a row-node format. For details, see "navtree.xml For Navigation Control Objects". **Note:** As with other table-driven objects, the **drillDownColumnSubs** property can be configured to set substitutions to column values from the row in the **valueTable** that corresponds to the selected accordion node.
valueTableFormat	Specifies the format of the **valueTable**: **Row-Leaf** or **Row-Node**.
varToSet	Allows you to update the attached variable with the value from the control.

Interaction Properties

Specify interactions in the accordion control.

Property Name	Description
accordionCollapseFlag	When checked, only one branch of the accordion will be open at a time. If you click a node to open another branch, then the previously open branch is automatically closed. If this flag is unchecked, opening another branch has no affect on other open branches. This flag is checked by default.
actionCommand	Use the **actionCommand** property to assign a command to the accordion. You can configure the accordion to open a drill-down display, set substitutions, or execute a command in response to a user click on an accordion node. The **actionCommand** can reference the value from the accordion by using the keyword $value. When the command is executed, the variable attached to **varToSet** is updated with the selected node data. The **drillDownColumnSubs** property can be configured to set substitutions to column values from the row in the **valueTable** that corresponds to the selected accordion node. If the **execOnLeafOnlyFlag** property is checked, the accordion **actionCommand** property executes only when a leaf node is clicked (a click on a non-leaf node expands only the node). If unchecked, the accordion **actionCommand** property executes on all nodes, not just the leaf.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownColumnSubs	Use the **drillDownColumnSubs** property to set substitutions to column values from the row in the **valueTable** that corresponds to the selected accordion node. Select the ... button to open the "Drill Down Column Substitutions Dialog" to customize which substitutions are passed into drill-down displays.
enabledFlag	If unchecked, the accordion nodes are the color gray and do not respond to user input.

execOnLeafOnlyFlag	If checked, the accordion **actionCommand** is executed only for leaf nodes, and a click on a non-leaf node only expands the node. Also, the mouseover tooltip only appears for leaf nodes. If unchecked, the accordion **actionCommand** property executes on all nodes, and the mouseover tooltip appears for all nodes.
mouseOverFlag	Specifies whether a tooltip appears when the cursor is positioned over a node. The tooltip shows the node path (the node label preceded by the labels of all of its ancestors), the node status (if the **nodeStatusColumnName** property is specified), and its value (if the **valueColumnName** property is specified).
tabIndex	Use the **tabIndex** property to define the order in which the accordion receives focus when navigated from your keyboard. Initial focus is given to the object with the smallest **tabIndex** value, from there the tabbing order proceeds in ascending order. If multiple objects share the same **tabIndex** value, initial focus and tabbing order are determined by the alpha-numeric order of the table names. Tables with a **tabIndex** value of 0 are last in the tabbing order. **Note:** The **tabIndex** property does not apply to tables in the Display Server, nor to objects that are disabled, invisible, or have a value of less than 0.

Label Properties

Specify the label in the accordion control.

Property Name	Description
labelTextColor	Select the  button and choose a color from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop-down menu.
labelTextSize	Set the height of the label text in pixels.

Node Properties

Specify the node structure in the accordion control.

Property Name	Description
accordionClosedImage	Specify the icon that appears to the left of the label for a closed (collapsed) accordion node. The default is a small right arrow.
accordionColor	Specify the color of the accordion node buttons that are not selected.
accordionOpenImage	Specify the icon that appears to the left of the label for an open (expanded) accordion node. The default is a small down arrow.
accordionSelectColor	Specify the color of the accordion node that is selected.
accordionSelectTextColor	Specify the text color for the accordion node that is selected (all unselected nodes use **labelTextColor**).
accordionWidth	Specify the width, in pixels, for each accordion button. If a value of zero (the default) is specified, the width of each node is computed automatically and might be different for each node. Specify a nonzero value if you want all buttons to have the same size, for a more uniform appearance.
nodeStatusIconPos	Specify the status icon position in the accordion: **Left** or **Right**. By default, the status icon appears on the left of the node label. By default, no status icons appear in the accordion.

nodeStatusProperties	This property applies to the status icon. Specifies the status icon for a node. By default, no status icon is displayed. Select the  button to open the **Node Properties** dialog and map images to values, and set the status priority order for propagation up the accordion. The **nodeStatusProperties** property is visible only if the **nodeStatusColumnName** property is non-blank. You can also use the **nodeStatusProperties** property to attach a status icon to data. The data attachment must be a three-column table. Typically, a static XML file containing the table is used. The first column must contain string values that match values from the column in **valueTable** specified by the **nodeStatusColumnName** property. The second column must be the path to the .png, .gif, or .jpg image. The third column must contain the non-negative integer priority value. A static XML file is only read once each time you run RTView. If you specify (or modify) an XML source using the **Application Options** dialog, you may specify whether that XML source is static. For details, see "Creating XML Sources".
rootNodeLabel	Specify whether the accordion root node is visible. By default, this property is blank and the root node is not visible.

Object Properties

Specify the layout in the accordion control.

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. **Note:** An object should not be docked if the **Resize Mode** is set to **Scale**. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.

objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. See "Creating Style Sheets" for more information. **Note:** The value entered must not contain spaces and cannot start with rtv-.
visFlag	Set the visibility of the object.

Unique Behavior

The following describes properties that behave uniquely with the accordion control.

valueColumnName: This property specifies the name of the column whose value should be assigned to the $value variable when a node in the accordion is clicked. If not specified, the label string of the selected node is assigned to $value. Note the $value is the only substitution that can be used in the Display Name field of a drill-down command.

mouseOverFlag: If this property is checked, a tooltip appears when the cursor is positioned over a leaf node. The tooltip shows the node path (the node label preceded by the labels of all of its ancestors), the node status (if the **nodeStatusColumnName** property is specified), and its value (if the **valueColumnName** property is specified).

execOnLeafOnlyFlag: If this property is checked, the accordion **actionCommand** property executes only when a leaf node is clicked (a click on a non-leaf node expands only the node). If unchecked, the accordion **actionCommand** property executes on all nodes, not just the leaf.

rootNodeLabel: This property specifies the accordion root node (of which there is only one). By default, this property is blank and the root node is not visible.

Limitations

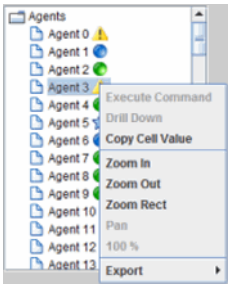
In the Display Viewer, mouseover text is displayed only if the accordion has focus.

In the Thin Client:

- The accordion node appearance, such as spacing and fonts, might vary slightly as compared to the Display Viewer, and also may vary slightly between different browsers.
- In iOS Safari (iPad), if the accordion **mouseOverFlag** property is checked, a user must click an accordion node twice to invoke the accordion command. The first click only displays the node mouseover text.
- In iOS Safari (iPad), scrollbars will not appear in an accordion control. If the accordion contains more nodes than are visible, use a two-finger drag gesture inside the accordion area to scroll.

Extending the Context Menu

This section describes how to extend items in the RTView context menu. The context menu appears when you right-click on an object. The context menu can be extended for a table object (**obj_table02**), heatmap object (**obj_heatmap**), grid object (**obj_objectgrid**) or tree control (**obj_c1tree**). Configure the context menu if you want to click an element on a tabular object (for example, a row in a table, a cell in a heatmap or grid, or a node in a tree control) and then choose one of several possible actions from a menu. The context menu appears when you right-click on an object (illustrated in the following figure).



The extended context menu is not supported in the Thin Client in iOS Safari (iPad/iPhone).

To configure the context menu, you create a button for each of the possible actions (for example, an **Ack** button, an **Unack** button, and an **Own** button) and edit button control properties (**menuItemIndex**, **menuItemGroup**, and **menuItemSubmenu**). You also edit properties on the table, heatmap or grid object (**menuItemGroup** and **rightClickActionFlag**), which are referred to as the target object. The buttons in the **menuItemGroup** define the items that are added to the context menu, when the target object with the same **menuItemGroup** value on the same display is right-clicked. The context menu items are, effectively, shortcuts to the buttons. The buttons are not affected by the properties of the associated target object.

Context menu items can be organized into submenus. The target object can be configured to invoke the first associated menu item when the user double-clicks on the object.

Summary of Steps

1. Verify that the following is your desired outcome: When a user selects an element on a tabular object (for example, a row in a table), several possible actions are available and the user can choose one to execute.

2. Configure the target object to set **drillDownColumnSubs** as necessary to record the selected row. Typically, the **rightClickActionFlag** property is checked, too.

3. Configure buttons on the display to perform each of the possible actions, using the **drillDownColumnSubs** set in step 2.

4. Choose a name for the **menuItemGroup** property and specify that name on the target object and all of the buttons.

5. Set the other **menuItem*** properties on each of the buttons as appropriate.

Example

To illustrate the how to configure the context menu, consider a display containing a table object.

Note: A heatmap object is configured similarly to a table object, and has the same behavior when a cell in the heatmap is right-clicked or double-clicked.

In our example, the table object shows the RTView AlertTable. A list of actions (**Ack**, **Unack**, and **Own**) are associated with each row in the table. The display also contains three buttons labeled **Ack**, **Unack**, and **Own** which execute different alert commands. A single left-click on a row could select the corresponding alert, or be followed by a click on the **Ack** (or **Unack**) button to acknowledge (or unacknowledge) the selected alert, or a click on the **Own** button could set the alert's owner to the current RTView username.

To configure the above described behavior, the table object's **drillDownColumnSubs** property is configured to set a substitution variable, **$alertID**, that is equal to the value of the ID column for the selected row. The table object's command property is configured to drill-down to the current display window. This simply means that each click on a row stores the ID of the selected alert in the **$alertID** substitution variable. Also, the **actionCommand** property of the **Ack** button is configured to execute an **Acknowledge Alert on ID = $alertID**. The **actionCommand** for the **Unack** and **Own** buttons is configured similarly. If necessary, the **enabledFlag** of each button is attached to a function that determines whether the button's action is appropriate for the current selection.

When an extended context menu is configured on the display:

- In addition to clicking on the button to execute an action, each action can also be executed from an item in the RTView context menu (selecting an extended context menu item is equivalent to clicking on the corresponding button.).
- By double-clicking on a row in a table (or a cell in a heatmap or grid), the object's drill-down can be executed or, if the object has no drill-down, the first action in the list can be executed.

The extra menu items appear at the top of the menu. One extra item appears in the menu for each button object (**obj_c1button**) on the display that meets these requirements:

- The button's **visFlag** property is set to 1.
- The button's **menuItemIndex** property has a nonzero value.
- If the **menuItemGroup** property of the currently selected table or heatmap object is set, it matches the **menuItemGroup** property of the button.

If multiple buttons meet the requirements above, they also appear in the context menu, ordered by the **menuItemIndex** property in increasing order.

Continuing with our example, let us assume we want context menu items for the **Ack**, **Unack**, and **Own** actions to appear in the context menu when the alert table object is selected, and in that order. To accomplish this, the **menuItemIndex** property of the **Ack**, **Unack**, and **Own** buttons is set to **1**, **2**, and **3** respectively. Also, to ensure that the menu items only appear when the alert table object is selected, the **menuItemGroup** property of the table object and all three buttons is set to the same string, for example "alert_group".

The extra menu items are added to the context menu by increasing order of their **menuItemIndex** property values. That is, if button X has its **menuItemIndex** set to a value that is less than the **menuItemIndex** of button Y, the item for X is situated above (nearer the top of the menu) the item for Y. Note that **menuItemIndex** can be assigned a value less than or greater than 0 (zero), but a value of 0 (zero) means that no item is added to the menu for that button.

The final required configuration step is to check the **rightClickActionFlag** property on the table object. This accomplishes two things:

1. Ensures that the substitutions for the right-clicked row are set before the menu is opened.

2. Enables the following double-click behavior on a table row:

- The table's **command**, if one is specified, is executed. Typically this sets substitutions on the current display for the selected row.
- The table's drill-down is executed. If the table has no **drilldownTarget**, the first enabled button on the display with the same **menuItemGroup** value is activated.

To summarize, configuring the table and buttons as described in our example, the table object behaves as follows:

- A single left-click or right-click on a row in the alert table executes the table's command, setting the **$alertID** substitution variable to the ID of the alert in that row.
- A right-click on a row in the alert table opens the context menu, in which the **Ack**, **Unack**, and **Own** menu items appear below the **Drill Down** item. (These menu items are either enabled or disabled, according to the state of the corresponding button).
- A double-left click on a row executes the **Ack** command on that row's alert, because the **Ack** button is the first button in the table's **menuItemGroup** property (it has the smallest **menuItemIndex** value in the group, **1** in this example).

Optionally, the extra menu items on the context menu can be organized into one or more submenus. This is done by assigning the same value to the **menuItemSubmenu** property of each button whose item is to appear in the same submenu. For example, to have the **Ack**, **Unack**, and **Own** menu items appear in a submenu (rather than the top-level context menu), we set the **menuItemSubmenu** property of the **Ack**, **Unack**, and **Own** buttons to (for example) "Alerts". This adds an item labeled "Alerts ->" to the top-level of the context menu, and selecting that item opens a cascading submenu with the **Ack**, **Unack**, and **Own** menu items.

If multiple submenus are defined, they are ordered by increasing value of the lowest **menuItemIndex** of the all the buttons in each submenu.

If a grid object is configured to use the context menu and the grid's cells contain a composite display, the behavior is as described above except when a user right-clicks or double-clicks an object in the composite display that has its own **drillDownTarget** or **command** property configured. In that case, the click is processed by the object inside the composite and not by the grid.

Limitations

The extended context menu is not supported in the Thin Client in iOS Safari (iPad/iPhone). All of the new properties (**rightClickActionFlag**, **menuItemIndex**, **menuItemGroup**, **menuItemSubmenu**) are ignored by the Thin Client on iOS Safari, therefore no menu items are added to the context menu and no double-click actions are supported. In particular, if a button is configured on the display so that it is off-screen but has a nonzero **menuItemIndex**, the button's command is not accessible on iOS Safari.

Extra menu items do not appear in the context menu in the main editing panel of the Builder. (They do appear in the Builder's preview window).

The double-click feature on the **obj_table02**, **obj_heatmap**, and **obj_objectgrid** objects does not work in any window in the Builder.

In the Thin Client, a double-click on a table, heatmap, or grid object that performs the first action in the menu group is sometimes ignored instead. This happens if the Display Server is slow to respond to the first click (which might enable the first button in the menu group), so the button is still disabled when the second click occurs. Its most likely to occur if the user double-clicks on several different cells in quick succession.

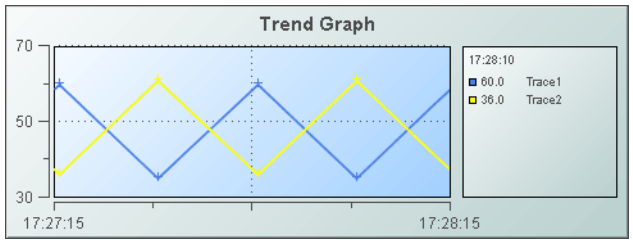
Graph Objects

The Object Palette features several types of graph objects -- trend, bar, pie, radar, and XY graphs-- that are useful for comparing data. Trend graphs are used to display a single variable per trace over time. Bar, pie, radar, and XY graphs are designed to display information returned by a tabular element in your data attachment. Based on your data attachment, substitutions are created that will be passed into drill down displays. See "Drill Down Substitutions" for more information.

Note: Some graph objects (e.g.: the bar graph legend, pie wedges and legend) cache their colors and therefore do not update when a custom color definition changes (see "Custom Colors Tab" for more information). You will need to either to restart RTView or reload the display to see the color change for these objects.

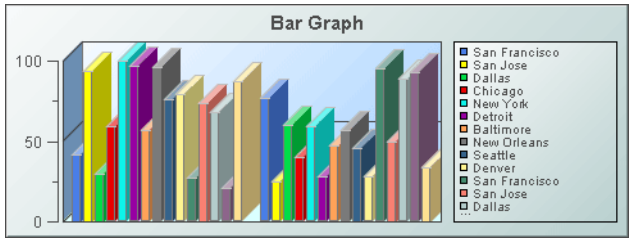
"Trend Graphs"

Trend graphs (class name: **obj_trendgraph02**) are useful for plotting a single value per trace over time and for displaying historical data stored by the "Historian".



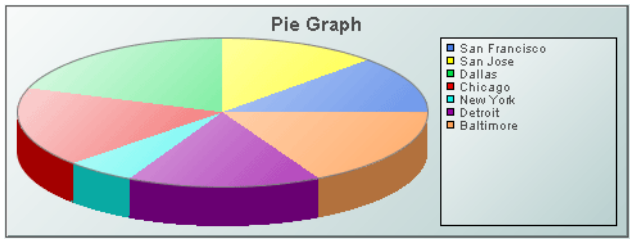
"Bar Graphs"

Bar graphs (class name: **obj_bargraph**) are useful for comparing columns or rows of numeric data from a tabular data element returned by your data attachment.



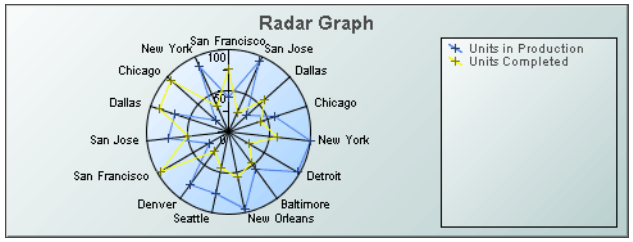
"Pie Graphs"

Pie graphs (class name: **obj_pie**) are useful for comparing values from a single column or a single row of the tabular data element returned by your data attachment.



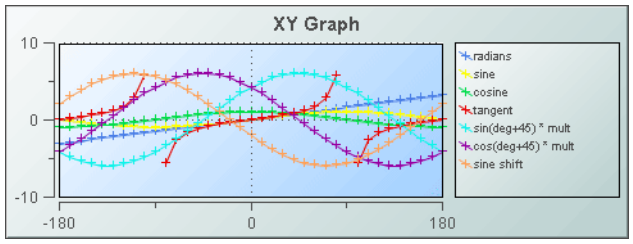
"Radar Graphs"

Radar graphs (class name: **obj_radar**) are useful for comparing columns or rows of numeric data from a tabular data element returned by your data attachment.



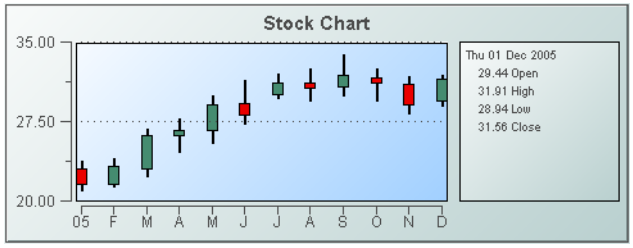
"XY Graphs"

XY graphs (class name: **obj_xygraph**) are useful for comparing pairs of values from a tabular data element returned by your data attachment.



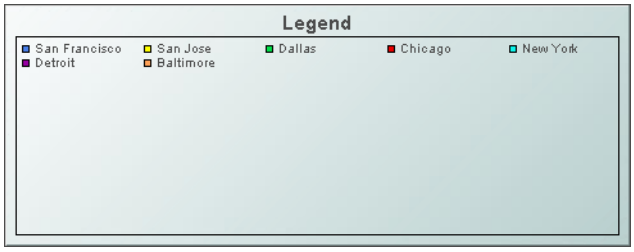
"Stock Chart"

Stock charts (class name: **obj_stockchart**) are useful for displaying live and archived stock data.



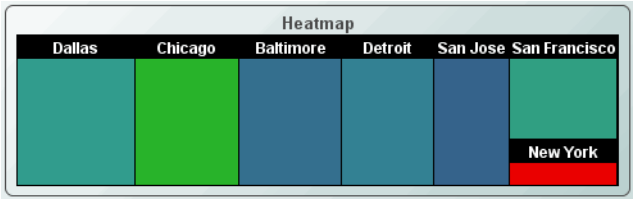
"Legend"

This object (class name: **obj_legend**) is useful for displaying a legend that is too lengthy for the built-in legends of the graph objects. The legend is easily resizable and can be used in conjunction with any object on the Graphs tab, excluding the Heatmap.



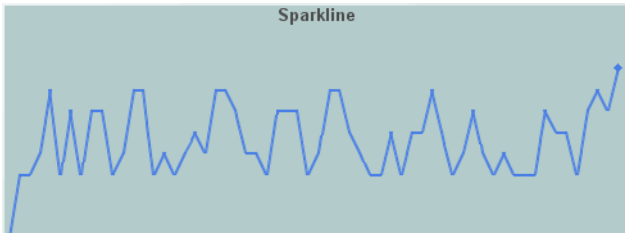
"Heat Maps"

Heatmaps (class name: **obj_heatmap**) display indexed hierarchical data as a set of nested rectangles. A rectangle exists for each index and, when attached to data, each is filled with smaller rectangles representing sub-indexes, known as nodes.



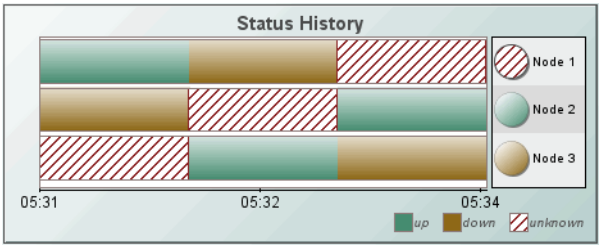
"Sparkline"

Sparkline graphs (class name: **obj_sparkline**) are generally used to present trends and variations in a simple and condensed way.



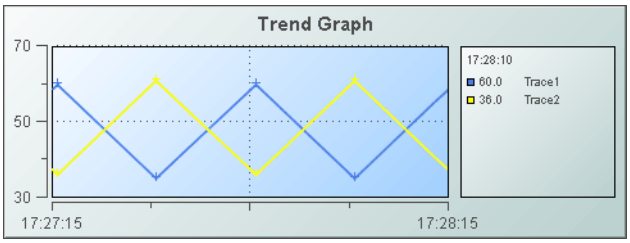
"Status History"

Status History charts (class name: **obj_statushistory**) show discrete status values as horizontal bars drawn against a horizontal time axis. A label appears at the right edge of each bar along with a circular indicator displaying that bar's most recent status.



Trend Graphs

Trend graphs (class name: **obj_trendgraph02**) are generally used for tracing two types of data in one trend graph: real-time, or live data, and archived data. Displaying these two types of traces in conjunction is useful for comparing data trends. The trend graph supports up to ten traces, set the **traceCount** to control the number of traces.



Using Trace* Properties - Trend Graphs

The Display Builder automatically creates a set of **trace* properties** according to the number you specify using the **traceCount** object property. Attach your data to **trace*Value** or **trace*ValueTable**, where ***** is the trace number. For example if you set the **traceCount** to three (3), you will have three traces in your graph (Trace01, Trace02, and Trace03) and a corresponding **trace*Value** (trace1Value, trace2Value, and trace3Value) for each.

All traces in the trend graph have the option to display historical and/or current data:

Historical Data	Current Data	Historical and Current Data
To display historical data, attach to **trace*ValueTable**, where ***** is the trace number, and include two columns in your attachment. The first column must be the time value and the second column the value to plot. Optionally, include a third (string) column in your attachment if you want to display a data label for each corresponding data value. See the Attach to Data section specific to your data source for details on how to use the Select Columns dialog. **Note:** In order to attach historical data to the **trace*ValueTable** property, the **trace*ValueHistoryFlag** must be deselected.	To display current data, attach to **trace*Value**, where ***** is the trace number. When you attach data to the **trace*Value** property, the time displayed on the trend graph is automatically updated each time data is received. Your data attachment can contain either a single point of data or two columns of data. See the **Attach to Data** section specific to your data source for details on how to use the Select Columns dialog. If your attachment contains a single point of data, RTView assigns a time stamp when the graph receives the data. If your attachment contains two columns of data, the first column must be the time value and the second column the value to plot. Optionally, include a third (string) column in your attachment if you want to display a data label for each corresponding data value.	To display both historical and current data in a single trace, attach your current data to **trace*Value** and select one of the following options for your historical data: • Select the **trace*ValueHistoryFlag** so that initial data is loaded from the Historian database • Attach your historical data to **trace*ValueTable**. **Note:** The **trace*ValueHistoryFlag** must be deselected.

Supported formats for the time value column are: **mm/dd/yyyy hh:mm:ss** (e.g., 01/16/2004 12:30:03), **yyyy-mm-dd hh:mm:ss** (e.g., 2004-01-16 12:30:03), and the number of milliseconds since midnight, January 1, 1970 UTC. In order to view all available data, you must set the properties **timeRange** to **-1** and **timeShift** to a negative value. This negative value will be used to round the start and end times for the y-axis. For example, if you specify **-15** for the **timeShift** property, the start and end times for the y-axis will be rounded to the nearest 15 seconds.

Data labels will be displayed (enclosed in parentheses) in the fixed legend and in the popup legend, between the trace value and the trace label. If the **cursorFlag** property is selected, then the data label in the popup legend is for the data value that is directly under or to the left of the cursor.

Alert Properties

To set trace marker colors and styles based on a threshold value, select the corresponding value alarm or value warning flags.

Property Name	Description	
valueHighAlarmEnabledFlag	Select to enable the high alarm threshold and the following related properties:	
	valueHighAlarm	Set the value of the high alarm threshold.
	valueHighAlarmLineVisFlag	Select to display a dotted line at the high alarm threshold. The color of the line is set to the **valueHighAlarmMarkColor**.

	valueHighAlarmMarkColor/ valueHighAlarmMarkStyle	When a trace marker's value is greater than or equal to the **valueHighAlarm** property, the marker will change to the **valueHighAlarmMarkColor** and **valueHighAlarmMarkStyle**.
	valueHighAlarmTraceColor/ valueHighAlarmTraceStyle	When the value of any segment of a trace line is greater than or equal to the **valueHighAlarm** property, that segment of the trace line will change to the **valueHighAlarmTraceColor** and **valueHighAlarmTraceStyle**. **Note:** If **valueHighAlarmTraceStyle** is set to No Line, then **valueHighAlarmTraceColor** will not change.
valueHighWarningEnabledFlag	Select to enable the high warning threshold and the following related properties:	
	valueHighWarning	Set the value of the high warning threshold.
	valueHighWarningLineVisFlag	Select to display a dotted line at the high warning threshold. The color of the line is set to the **valueHighWarningMarkColor**.
	valueHighWarningMarkColor/ valueHighWarningMarkStyle	When a trace marker's value is greater than or equal to the **valueHighWarning** property but less than the **valueHighAlarm** property, the marker will change to the **valueHighWarningMarkColor** and **valueHighWarningMarkStyle**.
	valueHighWarningTraceColor/ valueHighWarningTraceStyle	When the value of any segment of a trace line is greater than or equal to the **valueHighWarning** property but less than the **valueHighAlarm** property, that segment of the trace line will change to the **valueHighWarningTraceColor** and **valueHighWarningTraceStyle**.
valueLowAlarmEnabledFlag	Select to enable the low alarm threshold and the following related properties:	
	valueLowAlarm	Set the value of the low alarm threshold.
	valueLowAlarmLineVisFlag	Select to display a dotted line at the low alarm threshold. The color of the line is set to the **valueLowAlarmMarkColor**.
	valueLowAlarmMarkColor/ valueLowAlarmMarkStyle	When the trace marker's value is less than or equal to the **valueLowAlarm** property, the marker will change to the **valueLowAlarmMarkColor** and **valueLowAlarmMarkStyle**.
	valueLowAlarmTraceColor/ valueLowAlarmTraceStyle	When the value of any segment of a trace line is less than or equal to the **valueLowAlarm** property, that segment of the trace line will change to the **valueLowAlarmTraceColor** and **valueLowAlarmTraceStyle**.
valueLowWarningEnabledFlag	Select to enable the low warning threshold and the following related properties:	

valueLowWarning	Set the value of the low warning threshold.
valueLowWarningLineVisFlag	Select to display a dotted line at the low warning threshold. The color of the line is set to the **valueLowWarningMarkColor**.
valueLowWarningMarkColor/ valueLowWarningMarkStyle	When the trace marker's value is less than or equal to the **valueLowWarning** property, but greater than the **valueLowAlarm** property, the marker will change to the **valueLowWarningMarkColor** and **valueLowWarningMarkStyle**.
valueLowWarningTraceColor/ valueLowWarningTraceStyle	When the value of any segment of a trace line is less than or equal to the **valueLowWarning** property, but greater than the **valueLowAlarm** property, that segment of the trace line will change to the **valueLowWarningTraceColor** and **valueLowWarningTraceStyle**.

Background Properties

Specify how the background is displayed in your trend graph.

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose a color from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is 3D Rectangle.
bgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is white. **Note:** The bgColor property sets the first color in the gradient.
bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.

bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is 100 and **objHeight** is 50, then the value of **bgRoundness** cannot exceed 25. If it does, then half the value of **objHeight** (25) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the graph and the border.

Data Properties

Specify how data is displayed in your graph.

Property Name	Description
historyOnlyFlag	If selected, the graph will plot only data that is applied to the **trace*ValueTable** properties and will ignore the **timeShift** property and any data that is applied to the **trace*Value** properties. This is useful when same graph is used to view Historical data or Historical and Current data by setting substitutions on the display.
maxPointsPerTrace	The default is set to **1000**. The maximum value for this property is **30000**.
yValueMax yValueMin	The **yValueMin** and **yValueMax** properties control the range of the y-axis for this trace if the **yAxisAutoScaleMode** is set to **Off**. Select **On** for the **yAxisAutoScaleMode** to calculate y-axis range according to data values being plotted. To calculate the y-axis range including **yValueMin** and **yValueMax**, select **On - Include Min/Max**. If **yAxisMultiRangeMode** is set to **Multiple Axis** or **Strip Chart**, then use the **trace*YAxisAutoScaleMode**, **trace*YAxisValueMin**, and **trace*YAxisValueMax** properties to control the range of the y-axis.

Data Format Properties

Specify data format in your graph.

Property Name	Description
yValueFormat	Select or enter the numeric format of values displayed in the legend and popup legend. To enter a format, use syntax from the Java **DecimalFormat** class.

Historian Properties

Property Name	Description
historyTableName	Specify name of table in your history database in which to store tabular data. See "Configuring the Historian" for information.
historyTableRowNameFlag	If selected, data from the row name field will be stored in the first column of the table specified in **historyTableName**.

Interaction Properties

Property Name	Description
command	Assign a command to your trend graph. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the Property Value field or select the  button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
cursorColor	Select the  button and choose a color from the palette to set the color of the cursor. **Note:** This property is only available if **cursorFlag** is selected.
cursorFlag	Select to enable the cursor. When the cursor is enabled, point to a location on a trace to see a cursor line at that location and display the time and values of all traces at the cursor line on the legend. Hold down the control key to snap the cursor to the closest data point. Select the **legendPopupFlag** to display the legend along the cursor.

drillDownTarget	Name of display (.rtv) file targeted as a drill down. See "Drill Down Displays" for information. When you double-click on a trace in the graph, the following predefined substitutions will be set on the specified **drillDownTarget**: • **$traceNumber** - number of the trace (1 to 10) that contains the selected point • **$traceLabel** - label of selected trace • **$pointValue** - data value of point • **$pointTimestamp** - timestamp of point • **$pointLabel** - data label (if any) of point • **$pointIndex** - position of point in trace data (0 to maxPointsPerTrace) **Note:** When **drillDownSelectMode** is set to **Anywhere** double-clicking anywhere on the graph will activate the specified **drillDownTarget**, however you must double-click on a trace in the graph to set the substitutions listed above.
drillDownSelectMode	Control how a drill down display is activated. Select one of the following options: **Anywhere** - Activate a drill down display by double-clicking anywhere on the graph. **Element Only** - Enable a drill down display only when you double-click on a trace in the graph.
legendPopupFlag	Select to display the legend along the cursor. **Note:** This property is only available if **cursorFlag** is selected.
scrollbarMode	Select **Never**, **As Needed**, or **Always** from the **scrollbarMode** property to set the behavior of the scroll bar in the graph. **Never** is the default setting. Select **Always** to display a scroll bar at all times or **As Needed** to display the scroll bar when necessitated by zooming in the trace area or when you have more data loaded into the graph than you are displaying in the time range. For example, if **timeRangeOfHistory** is greater than **timeRange**, setting **scrollbarMode** to **As Needed** will enable a scroll bar to view all historical data loaded into the graph.
scrollbarSize	Specify the height of the horizontal scroll bar and the width of the vertical scroll bar, in pixels. The default value is **-1**, which sets the size to the system default.
zoomEnabledFlag	Select to enable zooming within the graph. Click in the graph's trace area and drag the cursor until a desired range is selected. While dragging, a rectangle is drawn to show the zoom area. The rectangle's default color is yellow (this can be changed in the **cursorColor** property). After the zoom is performed, the graph stores up to four zoom operations in queue. To zoom out, press the shift key and click in the graph's trace area. **Note:** The graph is paused after a zoom is performed and returns to updating with live data when the graph is zoomed back out to 100%.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the  button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. This property is only applies if **labelTextAlignY** is set to **TabTop**.
labelTextAlignX	Select x-axis position of label text from the drop down menu.

labelTextAlignY	Select y-axis position of label text from the drop down menu. **Outside Top** - Position label well above the background rectangle. **Top** - Position label just above the background rectangle. **Title Top** - Position label along the top line of the background rectangle. **Tab Top** - Position label tab just above the background rectangle. Height and width of the tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. **Inside Top** - Position label inside the top of the background rectangle.
labelTextColor	Select the ... button and choose a color from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop down menu.
labelTextHeight	Set the height of the label text in pixels.

Legend Properties

Specify the way the legend is displayed in your trend graph. Click and press a trace's entry in the legend to temporarily hide all other traces in the graph.

Property Name	Description
legendBgColor	Select the ... button and choose a color from the palette to set the background color of the legend.
legendBgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is white. **Note:** The **legendBgColor** property sets the first color in the gradient.
legendBgGradientMode	Display a gradient in the legend background. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
legendTextFont	Defines the font used for the text in the legend. The default is SansSerif.
legendTextSize	Defines the font size used for the text in the legend. The default is 9 (pixels).
legendTimeFormat	Set the format for the time displayed in the legend using syntax from the Java SimpleDateFormat class. For example, **MMMM dd, yyyy hh:mm:ss** a would result in the form August 30, 2003 05:32:12 PM. If no format is given, the **timeFormat** will be used.
legendValueMinSpace	Specify the minimum distance between values and labels in the legend.
legendVisFlag	Select to display the legend.

legendWidthPercent	Set the percent of the total width of the object used for the legend.
outlineColor	Select the button and choose a color from the palette to set the color of the one-pixel outline around the legend area, around each color swatch within the legend, and around the plot area. The default value is black (color 7).

Marker Properties

Specify the way markers are displayed in your trend graph.

Property Name	Description
markDefaultSize	Set **markDefaultSize** to specify the size of the markers in pixels.
markScaleMode	Set **markScaleMode** to scale markers according to the order of the data in your data attachment, e.g., the marker for the first data in the attachment is the smallest and the marker for the last data is the largest. Select one of the following from the drop down menu to set the scale mode: **No Scale**, **Scale by Trace**, or **Scale Within Trace**.

Object Properties

Property Name	Description
anchor	Select the button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Plot x-axis position of object.

objY	Plot y-axis position of object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with rtv-.
transparencyPercent	Set transparency of the object. Enter a value between 0 and 100. A value of 0, the default, sets the object to be completely opaque. A value of 100 will render the object completely transparent.
visFlag	Control visibility of the object.

Plot Area Properties

Specify the way the plot area is displayed in your trend graph.

Property Name	Description
gridColor	Select the ... button and choose a color from the palette to set the color of the grid.
traceBgColor	Select the ... button and choose a color from the palette to set the background color of your graph.
traceBgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is white. **Note:** The **traceBgColor** property sets the first color in the gradient.
traceBgGradientMode	Display a gradient in the plot background. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
traceBgImage	Select an image to display in the plot background of your graph. If necessary, the image will be stretched to fit the plot area.

Trace Groups

Specify trace groups displayed in your trend graph. The Display Builder automatically creates a set of **traceGroup* properties** (where ***** is the trace group number) according to the number of trace groups specified in the **traceGroupCount** property. A trace group is a collection of two or more traces. Trace groups are useful for identifying multiple traces sharing the same vertical axis (when **yAxisMultiRangeMode** is set to **Strip Chart** or **Multi-Axis**) or for identifying three traces combined as a banded trace.

Property Name	Description
traceGroupCount	Specify the number of traces in your trace group. The maximum is 5.

traceGroup*TraceNumbers	According to the number of traces specified in the **traceCount** property, enter a comma-separated list of the traces to include in this group (i.e. 1,3,5,6). **Note:** If **traceGroup*BandedFlag** is selected, then the number of traces listed should be 3. If the **yAxisMultiRangeMode** selected is **Strip Chart** or **Multi-Axis**, all the traces in the group will share the same strip/axis. If the **yAxisMultiRangeMode** selected is **Multi-Axis**, the color of the axis for a trace group is determined by the specified **trace*LineColor** of the first visible trace in the group. **Note:** In **Multi-Axis** mode it is recommended that you assign the same **trace*LineColor**/**trace*LineStyle** and **trace*MarkColor**/ **trace*MarkStyle** to all traces included within a group. Once a trace is included in a group, the following properties are no longer available: **trace*YAxisAutoScaleMode**, **trace*YAxisValueMax**, **trace*YAxisValueMin**. To scale the y-axis range of your trace group, use the **yAxisAutoScaleMode**, **yValueMax**, and **yValueMin** properties. **Note:** All traces included in a group must have the **trace*YAxisVisFlag**, **traceVisFlag**, and **trace*YAxisGridVisFlag** properties selected, otherwise the y-axis and/or the y-axis grid will not be visible.
traceGroup*BandedFlag	If selected, the plot area beneath the first trace (the low band trace) and the area above the third trace (the high band trace) will be filled. The second trace (the value trace) will not be filled.

Trace Properties

Specify how traces are displayed in your trend graph.

Property Name	Description	
multiTraceTableFlag	Select to plot multiple traces from a single data table.	
	multiTraceCurrentValueTable	Attach your current data. The first column in the attachment must contain a time value. The remaining columns can either contain numeric or string data. The first numeric value is used as the data point, the first string value is used as the first data point's data label. Within the attachment, columns can be arranged in any order (e.g.: numeric1, numeric2, string1, string2, numeric3, string3, string4, numeric4). Unless a **trace*Label** is already specified, the attached trace* column names will be used as trace labels. **Note:** If **multiTraceHistoryValueTable** is also attached to data, it must contain the same number and type of columns as the specified **multiTraceCurrentValueTable**.

	multiTraceHistoryValueTable	Attach your historical data. The first column in the attachment must contain a time value. The remaining columns can either contain numeric or string data. The first numeric value is used as the data point, the first string value is used as the first data point's data label. Within the attachment, columns can be arranged in any order (e.g.: numeric1, numeric2, string1, string2, numeric3, string3, string4, numeric4). Unless a **trace*Label** is already specified, the attached trace* column names will be used as trace labels. Typically, the attached data table contains initial data points to be plotted (i.e. from a SQL attachment with **Update Mode** set to **On Demand**, or from a Cache attachment with **Update Once** selected.) **Note:** If **multiTraceCurrentValueTable** is also attached to data, it must contain the same number and type of columns as the specified **multiTraceHistoryValueTable**.
	Note: Once **multiTraceTableFlag** is selected, all trace data is expected to be provided via **multiTraceCurrentValueTable** or **multiTraceHistoryValueTable**. Therefore, the **trace*Value** and **trace*ValueTable** properties will no longer be available.	
traceCount	Specify the number of traces in your graph. The maximum is **10**. RTView automatically creates a set of properties (see **Trace* Properties** below) for each trace.	
traceFillStyle	Set **traceFillStyle** to **Solid**, **Transparent**, **Gradient**, or **Transparent Gradient** to fill the area under the trace. Set to None to disable this feature. **None** is the default.	

Trace* Properties

The Display Builder automatically creates a set of **trace* properties** (where ***** is the trace number) according to the number of traces specified in the **traceCount** property. Attach your data to **trace*Value** or **trace*ValueTable**, see the "Using Trace* Properties - Trend Graphs" section for details.

Property Name	Description
trace*Label	Set a label for your trace.
trace*LineColor	Select the  button and choose a color from the palette to set the trace color.
trace*LineStyle	Select the style of the trace from the drop down menu: **No Line**, **Solid**, **Dotted**, **Dashed**, or **Dot Dashed**.
trace*LineThickness	Select the thickness of the trace from the drop down menu: **Thin**, **Medium**, or **Thick**.
trace*MarkColor	Select the  button and choose a color from the palette to set the trace marker color.
trace*MarkStyle	Select the style of the marker used on the trace from the drop down menu: **No Marker**, **Dot**, **+**, *****, **o**, **x**, **Filled Circle**, **Filled Diamond**, **Filled Triangle**, **Filled Square**, or **Filled Star**.

trace*Type	Select the type of trace from the drop down menu: **Line**, **Bar**, or **Event**. **Line** - Line connecting each data point. **Bar** - Vertical bar for each data point, from zero to the data point's value. Use the **trace*LineThickness** property to control the width of the bar. **Note:** If **trace*MarkStyle** is set to any value other than **No Marker**, the mark will be drawn at the end of the Bar trace. **Event** - Small rectangle containing the first character of the corresponding data label for each data point. If no data label exists, then the first character of the specified **trace*Label** is used. Each Event trace is positioned vertically according to the data value for the corresponding data point. The **trace*LineColor** property sets the color of rectangle's edges and the enclosed text character. The **trace*MarkColor** property sets the fill color of the rectangle. If **trace*LineColor** and **trace*MarkColor** are set to the same color, then **traceBgColor** is used to set the fill color of the rectangle. **Note:** If the attached data contains data labels but no data values, then an Event trace will be plotted regardless of the specified **trace*Type** setting and Event trace rectangles will be drawn near the bottom of the trace area.
trace*Value	Attach your current data, either containing a single point of data or two columns of data. If your attachment contains a single point of data, RTView will assign a time stamp when the graph receives the data. If your attachment contains two columns of data, the first column must be the time value and the second column the value to plot. Optionally, include a third (string) column in your attachment to display a data label for each corresponding data value. The data label will be displayed (enclosed in parentheses) in the fixed legend and in the popup legend, between the trace value and the trace label. If the **cursorFlag** property is selected, then the data label in the popup legend is for the data value that is directly under or to the left of the cursor.
trace*ValueAlarmStatus	Use if your trace is being driven by **trace*Value**. To apply an alarm status to a trace, enter an alarm status index which will be applied to any new points plotted. Valid indexes are: 0 = use normal marker color and style, 1 = use low alarm marker color and style, 2 = use low warning marker color and style, 3 = use high warning marker color and style, 4 = use high alarm marker color and style, -1 = determine marker color and style by comparing the value to the enabled alarm thresholds.
trace*ValueAlarmStatus Table	Use if your trace is being driven by **trace*ValueTable**. Attach an alarm table containing status indexes to **trace*ValueAlarmStatusTable** to enable rule based alarm statuses for trace markers. This table must have a time column (formatted like the time value in the **trace*ValueTable**) and a value column where the value column contains alarm status values 0-4. The table must also have the same number of rows as the corresponding **trace*ValueTable**. For each data element in **trace*ValueTable**, the status index at the corresponding position in **trace*valueAlarmStatusTable** will be used to set the alarm status of the marker. Valid indexes are: 0 = use normal marker color and style, 1 = use low alarm marker color and style, 2 = use low warning marker color and style, 3 = use high warning marker color and style, 4 = use high alarm marker color and style, -1 = determine marker color and style by comparing the value to the enabled alarm thresholds. If no data is attached to **trace*ValueAlarmStatusTable**, then the alarm status for a trace marker is determined by comparing the marker's value to the enabled thresholds.
trace*ValueDivisor	Enter a number. All traces will be divided by this value.

trace*ValueHistoryFlag	If selected, RTView will attempt to load initial data from the Historian database for the corresponding trace. **Note:** Only data within the specified **timeRange** will be loaded and the graph will update as live data becomes available. See "Building a Display Using History Data" for more information.
trace*ValueTable	Attach your historical data containing two columns: the first column must be the time value and the second column the value to plot. **Note:** In order to attach historical data to the **trace*ValueTable** property, the **trace*ValueHistoryFlag** must be deselected. Optionally, include a third (string) column in your attachment to display a data label for each corresponding data value. The data label will be displayed (enclosed in parentheses) in the fixed legend and in the popup legend, between the trace value and the trace label. If the **cursorFlag** property is selected, then the data label in the popup legend is for the data value that is directly under or to the left of the cursor.
trace*VisFlag	Select to control trace visibility. Click and hold on a trace's entry in the legend to temporarily hide all other traces in the graph.

Web Chart Properties

Property Name	Description
webChartFlag	Select to enable the web trend chart. When the display is opened in the Display Server, the web trend chart will appear in place of the trend graph.
webChartNavigatorTrace	Specify the number of traces (between 1 and the value of the **traceCount** property). By default, this property is disabled with a value of **0**. The navigator will be displayed at the bottom of the web chart, just below the x-axis, and highlight the time range currently visible in the chart. The highlighted section can be resized or dragged to zoom/pan to the time range of interest. The color of the navigator trace will reflect the color of the corresponding trace as specified by the **trace*LineColor** property.

X-Axis Properties

Specify the way the x axis is displayed in your trend graph.

Property Name	Description
timeFormat	Set the format for the time displayed in the x-axis using syntax from the **Java SimpleDateFormat** class. For example, **MMMM dd, yyyy hh:mm:ss** a would result in the form August 30, 2003 05:32:12 PM. If no format is given, the date and time will not be displayed on the x-axis. Include a new line character ('\n') to display multiple line text in the time axis labels. For example, **MM\dd'\n'hh:mm:ss** would result in the form 08\30 05:32:12.
timeRange	Control the total amount of time, in seconds, plotted on the graph. If you attach data to **trace*ValueTable**, set **timeRange** to **-1** so the time range of the graph is driven by the time range in the data attachment. **Note: timeRange** is ignored if both **timeRangeBegin** and **timeRangeEnd** are set.

timeRangeBegin	Set the start time value of the data to be plotted on the graph. Supported formats are: **mm/dd/yyyy hh:mm:ss** (e.g., 01/16/2004 12:30:03), **yyyy-mm-dd hh:mm:ss** (e.g., 2004-01-16 12:30:03), and the number of milliseconds since midnight, January 1, 1970 UTC. **Note:** If only the time is specified, then today's date will be used.
timeRangeEnd	Set the end time value of the data to be plotted on the graph. Supported formats are: **mm/dd/yyyy hh:mm:ss** (e.g., 01/16/2004 12:30:03), **yyyy-mm-dd hh:mm:ss** (e.g., 2004-01-16 12:30:03), and the number of milliseconds since midnight, January 1, 1970 UTC. **Note:** If only the time is specified, then today's date will be used.
timeRangeOfHistory	Specify how much historical data is loaded into the graph, in seconds. If **timeRangeOfHistory** is set to zero or less (default is **-1**), or if it is less than the value of **timeRange**, then the **timeRange** property determines the amount of historical data to be loaded into the graph.
timeShift	Control the amount of time, in seconds, the graph will shift to the left when the trace has filled the graph and controls the rounding of the start and end times. For example, if the **timeShift** is 15, the start and end times on the graph will be rounded to the nearest 15 second interval. By default, the end of the plot area corresponds with the current time. To only shift the graph when new data is received, set **timeShift** to a negative value and the end of the graph will display the most current data plotted. If you attach data to **trace*ValueTable**, you must set **timeShift** to a negative value. **Note:** The **timeShift** property is ignored if either **timeRangeBegin** or **timeRangeEnd** is set.
xAxisFlag	Select to display the x-axis.
xAxisGridVisFlag	Select to display a grid line for each major division along the x-axis.
xAxisLabelTextHeight	Specify the height in pixels of the x-axis labels.
xAxisMajorDivisions	Specify the number of major divisions on the x-axis.
xAxisMinorDivisions	Specify the number of minor divisions on the x-axis.

Y-Axis Properties

Specify the way the y axis is displayed in your trend graph.

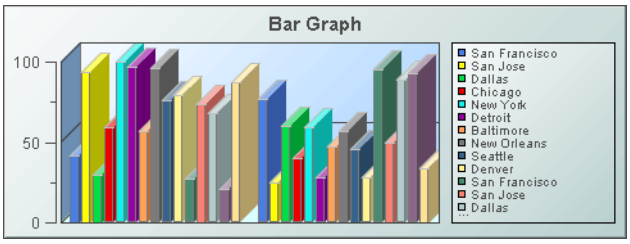
Property Name	Description
yAxisAutoScaleMode	The **yValueMin** and **yValueMax** properties control the range of the y-axis for this trace if the **yAxisAutoScaleMode** is set to Off. Select On for the **yAxisAutoScaleMode** to calculate y-axis range according to data values being plotted. To calculate the y-axis range including **yValueMin** and **yValueMax**, select **On - Include Min/Max**. If **yAxisMultiRangeMode** is set to **Multiple Axis** or **Strip Chart**, then use the **trace*YAxisAutoScaleMode**, **trace*YAxisValueMin**, and **trace*YAxisValueMax** properties to control the range of the y-axis.
yAxisAutoScaleVisDataOnlyFlag	Select to compute the y-axis scale according to the min & max y values of the visible data points only. This means that the y-axis scale may change as the user changes the visible time range by scrolling or zooming. By default, this property is not selected and the y axis is scaled according to the y values of all of data points, visible or not. This property is only available if **webChartFlag** is selected.
yAxisAutoScaleVisTracesOnlyFlag	Select to include only visible traces when calculating the y-axis range. By default all traces in the **traceCount** are used in the auto-scale calculation, including those that are invisible or unattached to data.

yAxisFlag	Select to display the y-axis.
yAxisFormat	Select or enter the numeric format of values displayed on the y-axis. To enter a format, use syntax from the Java DecimalFormat class.
yAxisGridVisFlag	Select to display a grid line for each major division along the y-axis if **yAxisMultiRangeMode** is set to Off or Classic. If **yAxisMultiRangeMode** is set to Multiple Axis or Strip Chart, grid visibility is set by selecting the **trace*YAxisGridVisFlag**.
yAxisLabelTextHeight	Specify the height in pixels of the y-axis labels.
yAxisMajorDivisions	Specify the number of major divisions on the y-axis. This is ignored if **yAxisValueLabels** is set.
yAxisMinLabelWidth	Specify the minimum width in pixels for the y-axis labels. If **yAxisMultiRangeMode** is set to Multiple Axis or Strip Chart, minimum width is set by **trace*YAxisMinLabelWidth**.
yAxisMinorDivisions	Specify the number of minor divisions on the y-axis. This is ignored if **yAxisValueLabels** is set.
yAxisMultiRangeMode	Select one of the following modes: **Off** - All traces are plotted against a single y-axis. **Classic** - One axis per trace, with each trace having its own range. The 1st trace is drawn on the outer left of the graph. The remaining traces are drawn on the inner left of the trace area. **Multiple Axis** - One axis per trace, with each trace having its own range. Use the **trace*YAxisAutoScaleMode**, **trace*YAxisValueMin**, and **trace*YAxisValueMax** properties to control the range of the y-axis. **Strip Chart** - The trace area is divided into strips, one for each trace. Each trace has its own y-axis but all traces share the same x-axis. Use the **trace*YAxisAutoScaleMode**, **trace*YAxisValueMin**, and **trace*YAxisValueMax** properties to control the range of the y-axis. **Stacked** - The y-axis values for multiple traces displayed are "stacked". That is, the value of each successive data point is accumulated and the sum from lower numbered traces are plotted in the highest numbered trace. **Note:** Actual (unstacked) values for each individual trace are shown in the legend. Values for lower numbered traces are not included in the sum if a trace is invisible, has no data, all of its data points are older or newer than the x-axis value, or the y-axis value at the x-axis value is not a number (NaN). If a lower numbered trace has no y-axis value at a particular x-axis data point, but has x-axis values before and after that data point, the y-axis value for the sum in that trace will be interpolated.

yAxisPosition	Select one of the following to set the position of the y-axis on the graph. **Note:** The **yAxisPosition** setting is ignored if **yAxisMultiRangeMode** is set to **Classic**. **Outer Left** - Draw axis to the left of the trace area. **Outer Right** - Draw axis to the right of the trace area. **Outer Mixed** - Draw axis to the left of the trace area for odd numbered traces and to the right for even numbered traces. **Inner Left** - Draw axis on the inside left of the trace area. **Inner Right** - Draw axis on the inside right of the trace area. **Inner Mixed** - Draw axis on the inside left of the trace area for odd numbered traces and on the inside right of the trace area for even numbered traces.
yAxisValueLabels	Set to display a text label or tick mark on the y-axis in place of a numerical value. Include a value with no label to display a tick mark without a label. Use this format: value1=label1,value2,value3=label2 (e.g., 0=Off,1,2=On) If **yAxisMultiRangeMode** is set to **Multiple Axis** or **Strip Chart** mode, the y-axis labels are set using the **trace*YAxisValueLabels** property.

Bar Graphs

Bar graphs (class name: **obj_bargraph**) are useful for comparing columns or rows of numeric data from a tabular data element returned by your data attachment.



Using Data and Data Label Properties

To attach data to your graph, right-click in the Property Value field of the **valueTable** property and select **Attach to Data**. It is possible to graph multiple columns or rows of numeric data. If you include a label column in your data attachment for **valueTable**, it can be used to label either the x-axis or the legend, depending on whether the **rowSeriesFlag** is selected or deselected.

The **rowSeriesFlag** controls how row and column data populate the graph:

If the **rowSeriesFlag** check box is selected, one group of bars will be shown for each numeric column in your data attachment. Within the group for each numeric column, there will be a bar for each row in that column. Column names will be used for the x-axis labels. If your data attachment has a label column and the **rowLabelVisFlag** is selected, data from this column will be used in the legend. If your data attachment does not have a label column, select the **rowNameVisFlag** check box to use row names in the legend. By default, the label column is the first non-numeric text column in your data. Specify a column name in the **labelColumnName** property to set the label column to a specific column.

If the **rowSeriesFlag** check box is not selected, one group of bars will be shown for each row in your data attachment. Within the group for each row, there will be a bar for each column in that row. Column names will appear in the legend. If your data attachment has a label column and the **rowLabelVisFlag** is selected, data from this column will appear on the x-axis. If your data attachment does not have a label column, select the **rowNameVisFlag** check box to use row names on the x-axis. By default, the label column is the first non-numeric text column in your data. Specify a column name in the **labelColumnName** property to set the label column to a specific column.

To add one or more traces to your bar graph, right-click in the Property Value field of the **traceValueTable** property and select **Attach to Data**. If you include a label column in your data attachment for **traceValueTable**, this is used to label either the x-axis or the legend (depending on the **rowSeriesFlag**) if no label is available from the **valueTable** data attachment. By default, the label column is the first non-numeric text column in your data. Specify a column name in the **traceLabelColumnName** property to set the label column to a specific column.

Using Interaction Properties

Commands

To assign a command to your graph, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (.rtv) file can be targeted as a drill down. Based on your data attachment, substitutions are created that will be passed into drill down displays (see "Drill Down Substitutions" for more information). To customize which substitutions will be passed into drill down displays, double-click on **drillDownColumnSubs** in the **Object Properties** window to open the "Drill Down Column Substitutions Dialog". Once a drill down target has been set, double-click on the bar or trace marker in the graph to activate the drill down. Drill down displays can be activated in the same window that contains the graph or open in a separate window. This allows you to build a customizable hierarchy of displays.

Use **drillDownSelectMode** to control how a drill down display is activated. Set to **Anywhere** to activate a drill down display by double-clicking anywhere on the graph. Set to **Element Only** to enable a drill down display only when you double-click on a bar in the graph.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your bar graph. To display a tool tip, point to a bar or marker with your mouse. The tool tip will contain information from your data attachment about that bar or marker. Select the **mouseOverHighlightFlag** to enable bar highlighting.

Alert Properties

To set bar colors, trace marker colors and trace marker styles based on a threshold value, select the corresponding value alarm or value warning flags.

Property Name	Description	
traceValueAlarmStatusTable	Attach an alarm table containing status indexes to **valueAlarmStatusTable** to enable rule based alarm statuses for trace markers. The table attached to **valueAlarmStatusTable** must have the same number of rows and columns as **traceValueTable**. For each data element in **traceValueTable**, the status index at the corresponding position in **valueAlarmStatusTable** will be used to set the alarm status of the marker. Valid indexes are: **0** = use normal marker color and style **1** = use low alarm marker color and style **2** = use low warning marker color and style **3** = use high warning marker color and style **4** = use high alarm marker color and style **-1** = determine marker color and style by comparing the value to the enabled alarm thresholds If no data is attached to **valueAlarmStatusTable**, then the alarm status for a trace marker is determined by comparing the marker's value to the enabled thresholds.	
valueHighAlarmEnabledFlag	Select to enable the high alarm threshold and the following related properties:	
	valueHighAlarm	Set the value of the high alarm threshold.
	valueHighAlarmColor	When a bar's value is greater than or equal to the **valueHighAlarm** property, the color of the bar will change to the **valueHighAlarmColor**.
	valueHighAlarmLineVisFlag	Select to display a dotted line at the high alarm threshold. The color of the line is set to the **valueHighAlarmMarkColor**.
	valueHighAlarmMarkColor/ valueHighAlarmMarkStyle	When a trace marker's value is greater than or equal to the **valueHighAlarm** property, the marker will change to the **valueHighAlarmMarkColor** and **valueHighAlarmMarkStyle**.
valueHighWarningEnabledFlag	Select to enable the high warning threshold and the following related properties:	
	valueHighWarning	Set the value of the high warning threshold.
	valueHighWarningColor	When a bar's value is greater than or equal to the **valueHighWarning**, but less than the **valueHighAlarm**, the color of the bar will change to the **valueHighWarningColor**.
	valueHighWarningLineVisFlag	Select to display a dotted line at the high warning threshold. The color of the line is set to the **valueHighWarningMarkColor**.

	valueHighWarningMarkColor/ valueHighWarningMarkStyle	When a trace marker's value is greater than or equal to the **valueHighWarning** property but less than the **valueHighAlarm** property, the marker will change to the **valueHighWarningMarkColor** and **valueHighWarningMarkStyle**.
valueLowAlarmEnabledFlag	Select to enable the low alarm threshold and the following related properties:	
	valueLowAlarm	Set the value of the low alarm threshold.
	valueLowAlarmColor	When a bar's value is less than or equal to the **valueLowAlarm** property, the color of the bar will change to the **valueLowAlarmColor**.
	valueLowAlarmLineVisFlag	Select to display a dotted line at the low alarm threshold. The color of the line is set to the **valueLowAlarmMarkColor**.
	valueLowAlarmMarkColor/ valueLowAlarmMarkStyle	When the trace marker's value is less than or equal to the **valueLowAlarm** property, the marker will change to the **valueLowAlarmMarkColor** and **valueLowAlarmMarkStyle**.
valueLowWarningEnabledFlag	Select to enable the low warning threshold and the following related properties:	
	valueLowWarning	Set the value of the low warning threshold.
	valueLowWarningColor	When a bar's value is less than the **valueLowWarning** property, but greater than the **valueLowAlarm** property, the bar will change to the **valueLowWarningColor**.
	valueLowWarningLineVisFlag	Select to display a dotted line at the low warning threshold. The color of the line is set to the **valueLowWarningMarkColor**.
	valueLowWarningMarkColor/ valueLowWarningMarkStyle	When the trace marker's value is less than or equal to the **valueLowWarning** property, but greater than the **valueLowAlarm** property, the marker will change to the **valueLowWarningMarkColor** and **valueLowWarningMarkStyle**.

Background Properties

Specify how the background is displayed in your bar graph.

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.

bgBorderFlag	If selected, a border is drawn around the background rectangle
bgColor	Select the ... button and choose a color from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is 3D Rectangle.
bgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is white. **Note:** The bgColor property sets the first color in the gradient.
bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is 3D Rectangle.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is 100 and **objHeight** is 50, then the value of **bgRoundness** cannot exceed 25. If it does, then half the value of **objHeight** (25) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the graph and the border.

Bar Properties

Specify the way the bars are displayed in your graph.

Property Name	Description
barGradientStyle	Select **None**, **Shaded**, or **Rounded** to set the gradient style of the bars. None is the default setting. Select **Shaded** to display bars with a flat gradient or **Rounded** to display bars with a rounded gradient.
barImage	Type the name of the image to display in the bars or, select the  button to open the **Select Image** dialog containing up to three directories: • **Current Directory** - Contains images in the current directory and one level of subdirectories. • **Custom Image Library** - If you have specified a custom image library, this directory contains those images (.gif, .jpg or .png). See Creating a Custom Image Library (below) for details. • **Symbol Library** - Contains symbolic images (for example, symbols for various types of hardware, shapes, lights, arrows, etc.). Navigate to the image you want to use and select it. A preview of the image appears in the pane to the right. Click **OK** or **Apply** to set the image on your object. If an image is not listed, enter the name of the file, including the relative path. **Note:** If necessary, the image will be stretched to fit the bar size. **Creating a Custom Image Library** The custom image library enables you to make your own images available in the **Select Image** dialog. To add your own image library, perform the following steps. 1. Place your images .jar file and add it to the "RTV_USERPATH" environment variable. The images must be in a directory (not in the top level of the jar). They can be organized into subdirectories of one top level directory. 2. In the Display Builder, select **Tools/Builder Options** and, in the **Custom Image Library Path** field, set the path to the directory containing your images .jar file. For example, if you have a jar with this directory structure: com/mycompany/Images com/mycompany/Images/Blue Images com/mycompany/Images/Red Images com/mycompany/Images/Green Images you would enter **com/mycompany/Images**. This adds a directory named **Images** to the tree in the **Select Image** dialog. The **Images** directory will have three subdirectories: **Blue Images**, **Red Images**, and **Green Images**. Only directories containing images are added to the **Select Image** dialog. To access the images, you can edit any property that allows you to set an image on an object (for example, the image, **barImage** and **filterProperties** properties), or edit the **File>Background Properties>Image Name** field. See "Background Properties" for more information.
barProperties	Double-click on **barProperties** to open the "Bar Properties" dialog and specify the color and fill pattern to be used to display each bar in the graph. **Note:** Fill Patterns set in the **Bar Properties** dialog will be ignored unless the **barGradientStyle** property is set to **None**.
barValueVisFlag	Select to display a label containing the value for each bar.

Column Properties

Specify which columns are displayed in your bar graph.

Property Name	Description
columnsToHide	Specify columns from the data attachment to exclude from being used for plotted data or labels. Data from the **labelColumnName** column will be used for labels even if that column name is also specified in the **columnsToHide** property. Columns specified in the **columnsToHide** property can still be used in the **drillDownColumnSubs** property.

Data Properties

Property Name	Description
rowSeriesFlag	Controls how row and column data populate the graph. If **rowSeriesFlag** is selected, one group of bars will be shown for each numeric column in your data attachment. Within the group for each numeric column, there will be a bar for each row in that column. Column names will be used for the x-axis labels. If your data attachment has a label column and the **rowLabelVisFlag** is selected, data from this column will be used in the legend. If your data attachment does not have a label column, select the **rowNameVisFlag** check box to use row names in the legend. By default, the label column is the first non-numeric text column in your data. Specify a column name in the **labelColumnName** property to set the label column to a specific column. If **rowSeriesFlag** is not selected, one group of bars will be shown for each row in your data attachment. Within the group for each row, there will be a bar for each column in that row. Column names will appear in the legend. If your data attachment has a label column and the **rowLabelVisFlag** is selected, data from this column will appear on the x-axis. If your data attachment does not have a label column, select the **rowNameVisFlag** check box to use row names on the x-axis. By default, the label column is the first non-numeric text column in your data. Specify a column name in the **labelColumnName** property to set the label column to a specific column.
traceValueDivisor	Divides trace values by the number entered.
traceValueTable	Attach your data to the **traceValueTable** property to add one or more traces to your bar graph. If you include a label column in data attached to **traceValueTable**, it can be used to label either the x-axis or the legend (depending on the **rowSeriesFlag**) if no label is available from the **valueTable** data attachment. **Note:** By default, the label column is the first non-numeric text column in your data attachment. Enter a column name in the **traceLabelColumnName** property to set a specific label column.
traceYAxisValueMax traceYAxisValueMin	Select the **traceYAxisFlag** to plot the traces against a separate y-axis than the bars. **Note:** The **traceYAxisFlag** property is unavailable if the **drawHorizontalFlag** property is selected. The trace y-axis will be drawn to the right of the plot area. When the **traceYAxisFlag** is selected, the **traceYAxisValueMin** and **traceYAxisValueMax** properties are used to control the range of the trace y-axis if **yAxisAutoScaleMode** is set to **Off** or **On-include Min/Max**. Select or enter the numeric format of trace values displayed on the y-axis with the **traceYAxisFormat** property. To enter a format, use syntax from the **Java DecimalFormat** class.
valueDivisor	Divides bar and y-axis values by the number entered.

valueTable	Attach your data to the **valueTable** property. It is possible to graph multiple columns or rows of numeric data. If you include a label column in data attached to **valueTable**, it can be used to label either the x-axis or the legend, depending on whether the **rowSeriesFlag** is selected or deselected. **Note:** By default, the label column is the first non-numeric text column in your data attachment. Enter a column name in the **labelColumnName** property to set a specific label column.
yValueMax yValueMin	Controls the range of y-axis if the **yAxisAutoScaleMode** is set to **Off**. Select **On** for the **yAxisAutoScaleMode** to calculate the y-axis range according to data values being plotted. To calculate the y-axis range including **yValueMin** and **yValueMax**, select **On - Include Min/Max**.

Data Format Properties

Specify the data format for your bar graph.

Property Name	Description
labelColumnFormat	Select or enter the format of numeric or date labels displayed on the x-axis, in the legend and in tool tips. To enter a format, use syntax from the **Java DecimalFormat** class for numeric labels and syntax from the **Java SimpleDateFormat** class for date labels. To enable tool tips, select the **mouseOverFlag**.
traceYValueFormat	Select or enter the numeric format of bar values displayed in the legend and in tool tips. To enter a format, use syntax from the **Java DecimalFormat** class. To enable tool tips, select the **mouseOverFlag**.
yValueFormat	Select or enter the numeric format of bar values displayed on bars, in the legend and in tool tips To enter a format, use syntax from the **Java DecimalFormat** class. To enable tool tips, select the **mouseOverFlag**.

Data Label Properties

Property Name	Description
columnDisplayNames	Set alternate display names for column names in your bar graph's data. Column names are displayed either along the x-axis or in the legend, depending on whether or not **rowSeriesFlag** is selected.
labelColumnName	Specify the name of a label column in your data attachment. If a **labelColumnName** is not specified, then by default the first non-numeric text column in your data attachment will be used as the label column.
rowLabelVisFlag	If selected, then data from the label column are displayed either along the x-axis or in the legend, depending on whether or not **rowSeriesFlag** is selected. Enter a column name in the **labelColumnName** property to specify a label column in your data attachment.
rowNameVisFlag	If your data attachment does not contain a label column (i.e. a non-numeric text column) select the **rowNameVisFlag** check box to use row names along the x-axis or in the legend, depending on whether or not **rowSeriesFlag** is selected.
traceLabelColumnName	Specify name of label column in your data attachment. If a **traceLabelColumnName** is not specified, then by default the first non-numeric text column in your data attachment will be used as the label column.

Historian Properties

Property Name	Description
historyTableName	Specify name of table in your history database in which to store tabular data. See "Configuring the Historian" for information.
historyTableRowNameFlag	If selected, data from the row name field will be stored in the first column of the table specified in **historyTableName**.

Interaction Properties

Property Name	Description
command	Assign a command to your bar graph. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the Property Value field or select the  button to open the Edit **commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownColumnSubs	Select the  button to open the "Drill Down Column Substitutions Dialog" to customize which substitutions will be passed into drill down displays.
drillDownSelectMode	Control how a drill down display is activated. Select one of the following options: **Anywhere** - Activate a drill down display by double-clicking anywhere on the graph. **Element Only** - Enable a drill down display only when you double-click on a bar in the graph.
drillDownTarget	Name of display (.rtv) file targeted as a drill down. See "Drill Down Displays" for information.
mouseOverFlag	Select to enable tool tips for your bar graph. To display a tool tip, point to a bar or marker with your mouse. The tool tip will contain information from your data attachment about that bar or marker.

mouseOverHighlightFlag	Select to enable bar highlighting. Once highlighting is enabled, point to a bar with your mouse to highlight that bar.
scrollbarMode	Select **Never**, **As Needed**, or **Always** from the **scrollbarMode** property to set the behavior of the x-axis scroll bar in the graph. **Note:** If **drawHorizontalFlag** is selected, the x-axis is vertical. **Never** is the default setting. If set to **Never**, some bars may get clipped. Select **Always** to display a scroll bar at all times. Set to **As Needed** to display the scroll bar when there is not enough space to display all of the bars in the plot area. Each bar uses at least **minSpacePerBar** pixels along the x-axis.
scrollbarSize	Specify the height of the horizontal scroll bar and the width of the vertical scroll bar, in pixels. The default value is **-1**, which sets the size to the system default.

Label Properties

Property Name	Description
label	Edit label text directly in the Property Value field or select the ... button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. This property is only applies if **labelTextAlignY** is set to **Tab Top**.
labelTextAlignX	Select x-axis position of label text from the drop down menu.
labelTextAlignY	Select y-axis position of label text from the drop down menu. **Outside Top** - Position label well above the background rectangle. **Top** - Position label just above the background rectangle. **Title Top** - Position label along the top line of the background rectangle. **Tab Top** - Position label tab just above the background rectangle. Height and width of the tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. **Inside Top** - Position label inside the top of the background rectangle.
labelTextColor	Select the ... button and choose a color from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop down menu.
labelTextHeight	Set the height of the label text in pixels.

Layout Properties

Specify how the background is displayed in your bar graph.

Property Name	Description
barCenterFlag	Select to center the bars in the plot area. If not selected, the bars will be left or top aligned, depending on the **drawHorizontalFlag**. The **barCenterFlag** property is only applicable if **barFitFlag** is not selected.
barFitFlag	Select to stretch the bars to fit the available space in the plot area. If deselected, the **minSpacePerBar** property is used to determine bar width.
draw3dDepth	Specify width (in pixels) for the shadow of the bars.
draw3dFlag	Select to change the display of the bars from 2D to 3D.

drawHorizontalFlag	Select to have the bars in your graph displayed horizontally.
drawStackedFlag	Select to stack each bar group in your graph.
drawWaterfallFlag	Select to stack each bar group in your graph with an offset between bar sections.
horizAxisLabelRotationAngle	Set the amount of rotation of labels on the horizontal axis. Values range from 0 to 90 degrees. A value of 0 causes the bar graph to automatically pick the optimum angle of rotation.
horizAxisMinLabelHeight	Set the minimum amount of space to reserve for labels on the horizontal axis. If axis labels vary over time, this property may be used to reserve a consistent amount of space to prevent overlapping.
minSpaceBetweenBars	Set the minimum amount of space (in pixels) between adjacent bars in the same group.
minSpaceBetweenGroups	Set the minimum amount of space (in pixels) between the last bar in one group and the first bar in the next group.
minSpacePerBar	Set the minimum width (in pixels) for each bar. Default is **1**. **Note:** If **drawHorizontalFlag** is selected, **minSpacePerBar** will set the minimum height (in pixels) for each bar.
vertAxisMinLabelWidth	Specify the minimum width in pixels for the vertical axis labels.

Legend Properties

Specify the way the legend is displayed in your bar graph.

Property Name	Description
legendBgColor	Select the ... button and choose a color from the palette to set the background color of the legend.
legendBgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is white. **Note:** The legendBgColor property sets the first color in the gradient.
legendBgGradientMode	Display a gradient in the legend background. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
legendTextFont	Defines the font used for the text in the legend. The default is SansSerif.
legendTextSize	Defines the font size used for the text in the legend. The default is 10 (pixels).
legendValueVisFlag	Select to display the numerical values of your data in the legend.
legendVisFlag	Select to display the legend.

legendWidthPercent	Set the percent of the total width of the object used for the legend.
outlineColor	Select the button and choose a color from the palette to set the color of the one-pixel outline around the legend area, around each color swatch within the legend, around the plot area and each bar in the graph. The default value is **black** (color 7).

Marker Properties

Property Name	Description
markDefaultSize	Set to specify the size of the markers in pixels.
markScaleMode	Set to scale markers according to the order of the data in your data attachment. For example, the marker for the first data in the attachment is the smallest and the marker for the last data is the largest. Select one of the following options from the drop down menu: **No Scale**, **Scale by Trace**, or **Scale Within Trace**.

Object Properties

Property Name	Description
anchor	Select the button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools> Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Plot x-axis position of object.
objY	Plot y-axis position of object.

styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. See "Creating Style Sheets" for more information. **Note:** The value entered must not contain spaces and cannot start with rtv-.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Plot Area Properties

Property Name	Description
gridBgColor	Select the button and choose a color from the palette to set the color of the grid background.
gridBgGradientColor2	Select the button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **gridBgColor** property sets the first color in the gradient.
gridBgGradientMode	Display a gradient in the grid background. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
gridBgImage	Select an image to display in the grid background of your graph. **Note:** If necessary, the image will be stretched to fit the grid.
gridColor	Select the button and choose a color from the palette to set the color of the grid lines. Default is **white**.

Trace Properties

Specify the appearance of plotting traces in your bar graph. See "Using Data and Data Label Properties" for information on how to add traces to your bar graph.

Property Name	Description
traceFillStyle	Set **traceFillStyle** to **Solid**, **Transparent**, **Gradient**, or **Transparent Gradient** to fill the area under the trace. Set to **None** to disable this feature. **None** is the default.
traceProperties	Specify the line color, line style, line width, marker color and marker style of all traces.
traceShadowFlag	Select to enable trace shadows.

Web Chart Properties

Property Name	Description
webChartFlag	Select to enable the web bar chart. When the display is opened in the Display Server, the web bar chart will appear in place of the standard Bar Chart. There are multiple differences between the RTView bar chart and the web bar chart with respect to the number of available properties and the standard behavior. See "Using Web Bar Chart Properties" for more information.
webChartVisBarGroups	Specifies the maximum number of bar groups that are visible in the plot area. You can use this option as an alternative to the **minSpacePerBar** property to control the size and number of visible bars. Depending on the size of the plot area, scrollbar position, and other factors, one more or one less bar group may sometimes be visible. This property defaults to **0**, which means there is no limit to the number of visible bar groups. See "Using Web Bar Chart Properties" for more information. **Note:** This property is only visible when **webChartFlag** is enabled.

X-Axis Properties

Specify how the x-axis is displayed in your bar graph.

Property Name	Description
xAxisFlag	Select to display the x-axis.

Y-Axis Properties

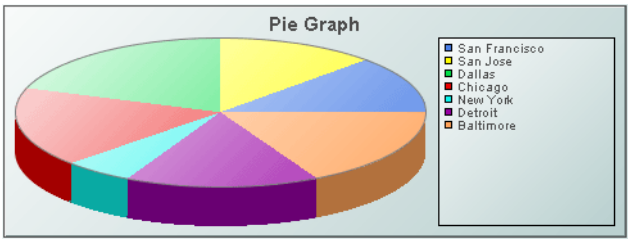
Specify how the y-axis is displayed in your bar graph.

Property Name	Description
traceYAxisFlag	Select the **traceYAxisFlag** to plot the traces against a separate y-axis than the bars. **Note:** The **traceYAxisFlag** property is unavailable if the **drawHorizontalFlag** property is selected. The trace y-axis will be drawn to the right of the plot area. When the **traceYAxisFlag** is selected, the **traceYAxisValueMin** and **traceYAxisValueMax** properties are used to control the range of the trace y-axis if **yAxisAutoScaleMode** is set to **Off** or **On-include Min/Max**.
traceYAxisFormat	Select or enter the numeric format of trace values displayed on the y-axis. To enter a format, use syntax from the **Java DecimalFormat** class.
traceYAxisMajorDivisions	Specify the number of major divisions on the trace y-axis. This option only applies if the **traceYAxisFlag** is on.
traceYAxisMinorDivisions	Specify the number of minor divisions on the trace y-axis. This option only applies if the **traceYAxisFlag** is on.
yAxisAutoScaleMode	The **yValueMin** and **yValueMax** properties control the range of y-axis if the **yAxisAutoScaleMode** is set to **Off**. Select **On** for the **yAxisAutoScaleMode** to calculate the y-axis range according to data values being plotted. To calculate the y-axis range including **yValueMin** and **yValueMax**, select **On - Include Min/Max**.
yAxisFlag	Select to display the y-axis.

yAxisFormat	Select or enter the numeric format of values displayed on the y-axis. To enter a format, use syntax from the **Java DecimalFormat** class.
yAxisGridMode	Controls the alignment of grid lines drawn to the left and right of the bar graph. By default, **yAxisGridMode** is set to **Bar Axis** which aligns grid lines with the left y-axis. Select **Trace Axis** to align with the right y-axis. Select **Bar and Trace Axis** to draw two sets of grid lines, one aligned with the left y-axis and the other with the right y-axis.
yAxisMajorDivisions	Specify the number of major divisions on the y-axis.
yAxisMinorDivisions	Specify the number of minor divisions on the y-axis.

Pie Graphs

The pie graph (class name: obj_pie) is useful for comparing values from a single column or a single row of the tabular data element returned by your data attachment.



Using Data and Data Label Properties

Attach your data to the **valueTable** property. If you include a label column in your data attachment for **valueTable**, it can be used to label the legend if the **rowSeriesFlag** is selected. The **rowSeriesFlag** controls whether row or column data populates the graph:

When the **rowSeriesFlag** check box is not selected, the first numeric column from your data attachment will be used to populate the wedges in the pie. Each wedge will correspond to a row in that column and will display that row's relative value. If your data attachment has a label column, data from that column will be used in the legend.

If the **rowSeriesFlag** check box is selected, the first numeric row from your data attachment will be used to populate the wedges in the pie. Each wedge will correspond to a column in that row and will display that column's relative value. Column names will be used in the legend.

If your data attachment has no label column, select **rowNameVisFlag** to use row names in the legend when the **rowSeriesFlag** is not selected. By default, the label column is the first non-numeric text column in your data. Specify a column name in the **labelColumnName** property to set the label column to a specific column.

Using Interaction Properties

Commands

To assign a command to your graph, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** Property Name. Any display (.rtv) file can be targeted as a drill down. Based on your data attachment, substitutions are created that will be passed into drill down displays. To customize which substitutions will be passed into drill down displays, double-click on **drillDownColumnSubs** in the Object Properties window to open the "Drill Down Column Substitutions Dialog". Once a drill down target has been set, double-click on the pie wedge to activate the drill down. Drill down displays can be activated in the same window that contains the graph or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Use **drillDownSelectMode** to control how a drill down display is activated. Set to Anywhere to activate a drill down display by double-clicking anywhere on the graph. Set to Element Only to enable a drill down display only when you double-click on a wedge in the pie graph.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your pie graph. To display a tool tip, point to a pie wedge with your mouse. The tool tip will contain information from your data attachment about that pie wedge.

Background Properties

Specify how the background is displayed in your pie graph.

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose a color from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is white. **Note:** The bgColor property sets the first color in the gradient.
bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.

bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example if **objWidth** is 100 and **objHeight** is 50, then the value of **bgRoundness** cannot exceed 25. If it does, then half the value of **objHeight** (25) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the graph and the border.

Column Properties

Specify which columns are displayed in your pie graph.

Property Name	Description
columnsToHide	Specify columns from the data attachment to exclude from being used for plotted data or labels. Data from the **labelColumnName** column will be used for labels even if that column name is also specified in the **columnsToHide** property. Columns specified in the **columnsToHide** property can still be used in the **drillDownColumnSubs** property.

Data Properties

Property Name	Description
rowSeriesFlag	Controls how row and column data populate the graph. If **rowSeriesFlag** is selected, the first numeric row from your data attachment will be used to populate the wedges in the pie. Each wedge will correspond to a column in that row and will display that column's relative value. Column names will be used in the legend. If **rowSeriesFlag** is not selected, the first numeric column from your data attachment will be used to populate the wedges in the pie. Each wedge will correspond to a row in that column and will display that row's relative value. If your data attachment has a label column, data from that column will be used in the legend.
valueTable	Attach your data to the **valueTable** property. It is possible to graph multiple columns or rows of numeric data. If you include a label column in data attached to **valueTable**, it can be used to label either the x-axis or the legend, depending on whether the **rowSeriesFlag** is selected or deselected. By default, the label column is the first non-numeric text column in your data attachment. Enter a column name in the **labelColumnName** property to set a specific label column.

Data Format Properties

Specify the format to display values in your pie graph.

Property Name	Description
labelColumnFormat	Select or enter the format of numeric or date labels displayed in the legend and in tool tips. To enter a format, use syntax from the **Java DecimalFormat** class for numeric labels and syntax from the **Java SimpleDateFormat** class for date labels. To enable tool tips, select the **mouseOverFlag**.
valueFormat	Select or enter the numeric format of wedge values displayed on wedges, in the legend and in tool tips. To enter a format, use syntax from the **Java DecimalFormat** class. To enable tool tips, select the **mouseOverFlag**.

Data Label Properties

Property Name	Description
columnDisplayNames	Set alternate display names for column names in your pie graph's data. Column names are displayed either along the x-axis or in the legend, depending on whether or not **rowSeriesFlag** is selected.
labelColumnName	Specify the name of a label column in your data attachment. If a **labelColumnName** is not specified, then by default the first non-numeric text column in your data attachment will be used as the label column.
rowLabelVisFlag	If selected, then data from the label column are displayed either along the x-axis or in the legend, depending on whether or not **rowSeriesFlag** is selected. Enter a column name in the **labelColumnName** property to specify a label column in your data attachment.
rowNameVisFlag	If your data attachment does not contain a label column (i.e. a non-numeric text column) select the **rowNameVisFlag** check box to use row names along the x-axis or in the legend, depending on whether or not **rowSeriesFlag** is selected.

Historian Properties

Property Name	Description
historyTableName	Specify name of table in your history database in which to store tabular data. See "Configuring the Historian" for information.
historyTableRowNameFlag	If selected, data from the row name field will be stored in the first column of the table specified in **historyTableName**.

Interaction Properties

Property Name	Description
command	Assign a command to your pie graph. See "Define/Execute Command" for information.

commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the Property Value field or select the ... button to open the Edit **commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownColumnSubs	Select the ... button to open the "Drill Down Column Substitutions Dialog" to customize which substitutions will be passed into drill down displays.
drillDownSelectMode	Control how a drill down display is activated. Select one of the following options: **Anywhere** - Activate a drill down display by double-clicking anywhere on the graph. **Element Only** - Enable a drill down display only when you double-click on a wedge in the pie graph.
drillDownTarget	Name of display (.rtv) file targeted as a drill down. See "Drill Down Displays" for information.
mouseOverFlag	Select to enable tool tips for your pie graph. To display a tool tip, point to a pie wedge with your mouse. The tool tip will contain information from your data attachment about that pie wedge.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. This property only applies if **labelTextAlignY** is set to **TabTop**.
labelTextAlignX	Select x-axis position of label text from the drop down menu.

labelTextAlignY	Select y-axis position of label text from the drop down menu. **Outside Top** - Position label well above the background rectangle. **Top** - Position label just above the background rectangle. **Title** - Top Position label along the top line of the background rectangle. **Tab Top** - Position label tab just above the background rectangle. Height and width of the tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. **Inside Top** - Position label inside the top of the background rectangle.
labelTextColor	Select the ... button and choose a color from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop down menu.
labelTextHeight	Set the height of the label text in pixels.

Legend Properties

Specify the way the legend is displayed in your pie graph.

Property Name	Description
legendBgColor	Select the ... button and choose a background color for the legend.
legendBgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **legendBgColor** property sets the first color in the gradient.
legendBgGradientMode	Display a gradient in the legend background. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
legendPercentVisFlag	Select to display the value percentages of your data in the legend.
legendTextFont	Defines the font used for the text in the legend. The default is SansSerif.
legendTextSize	Defines the font size used for the text in the legend. The default is 10 (pixels).
legendValueVisFlag	Select to display the numerical values of your data in the legend.
legendVisFlag	Select to display the legend.
legendWidthPercent	Set the percent of the total width of the object used for the legend.
outlineColor	Select the ... button and choose a color from the palette to set the color of the one-pixel outline around the legend area and around each color swatch within the legend. The default value is **black** (color 7).

Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Plot x-axis position of object.
objY	Plot y-axis position of object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with rtv-.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

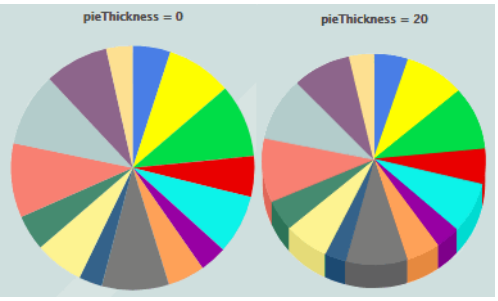
Web Chart Properties

Select this option to enable the web pie chart instead of the standard pie chart.

Property Name	Description
webChartFlag	Select to enable the web pie chart. When the display is opened in the Display Server, the web pie chart will appear in place of the RTView Pie Chart. There are multiple differences between the RTView pie chart and the web pie chart with respect to the number of available properties and the standard behavior. See "Using Web Pie Chart Properties" for more information.

Wedge Properties

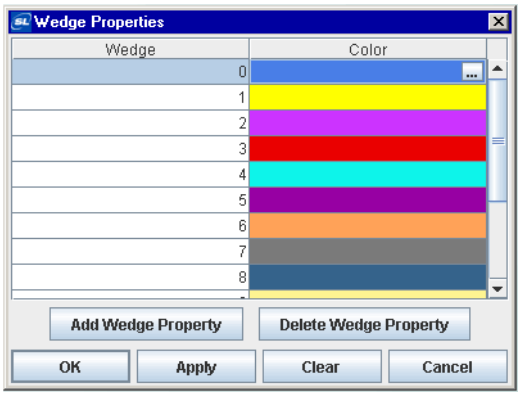
Specify the way wedges are displayed in your pie graph.

Property Name	Description
pieThickness	Sets the thickness (height) of the pie, in pixels. If set to **0**, the pie is flat. If set to any number greater than 0, then the bottom edge of the pie is tilted toward the viewer by the defined number of pixels. 
wedgeGradientFlag	Select the box to enable the gradient effect. All wedges in the pie are affected by the **wedgeGradientFlag** property.
wedgeProperties	Specify the color of wedges in the pie graph. See "Wedge Properties Dialog" for more information.

Wedge Properties Dialog

In the **Object Properties** window, double-click on **wedgeProperties** in the **Property Name** field to bring up the **Wedge Properties** dialog. You can use the **Wedge Properties** dialog to assign the color to each wedge in a pie graph.

Note: Before assigning attributes to wedges in your pie graph, it is recommended that you first attach the pie graph to data.



Field Name	Description
Wedge	Each wedge from the pie graph is listed. The **Color** column lists the current settings for the wedges.
Color	Click the ellipsis button in the **Color** column and choose a color from the palette. Close the **Color Chooser** window.
Add Wedge Property	Click to add a wedge entry field. The data for the wedge does not have to be available yet. You may consider adding and assigning attributes to more wedges than your data currently needs for when you have more data to show. **Note:** Before assigning attributes to wedges in your pie graph, it is recommended that you first attach the pie graph to data.
Delete Wedge Property	Removes the last wedge entry field from the **Wedge Properties** dialog.

The following describes the **Wedge Properties** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Trace Properties

In the **Object Properties** window, double-click on **traceProperties** in the **Property Name** field to bring up the **Trace Properties** dialog. In the **Trace Properties** dialog you can assign attributes to each plotting trace in your graph.

Note: Before assigning attributes to traces, it is recommended that you attach the graph to data.

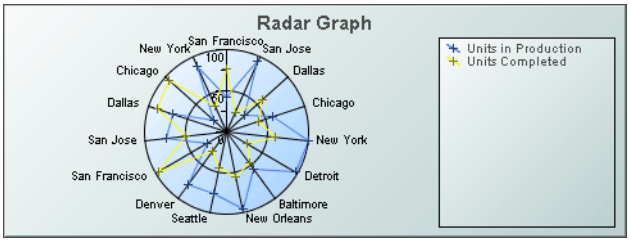
Field Name	Description
Trace	One field for each trace that is currently in the graph. Current settings for each trace are shown.
Line Color	Click the ellipsis button in the **Color** column and choose a color from the palette. Close the **Color Chooser** window.
Line Style	Click the ellipsis button in the **Line Style** column and choose a style from the drop down menu. Choose either **No Line**, **Solid**, **Dotted**, **Dashed**, or **Dot Dashed**.
Line Width	Click the ellipsis button in the **Line Width** column and choose a size from the drop down menu. Choose either **Thin**, **Medium**, or **Thick**.
Marker Color	Click the ellipsis button in the **Marker Color** column and choose a color from the palette. Close the **Color Chooser** window.
Marker Style	Click the ellipsis button in the **Marker Style** column and choose a style from the drop down menu. Choose either **No Marker**, **Dot**, **+**, *****, **o**, **x**, **Filled Circle**, **Filled Diamond**, **Filled Triangle**, **Filled Square**, or **Filled Star**.
Add Trace Property	Click to add a trace property field. The data for the trace does not have to be available yet. You may consider adding and assigning attributes to more traces than your data currently needs for when you have more data to show. It is not necessary to set properties for each trace you currently or subsequently have. This is optional and can be done after additional data is displayed in a subsequent new trace.
Delete Trace Property	Removes the last trace property field from the **Trace Properties** dialog.

The following describes the **Trace Properties** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Radar Graphs

Radar graphs (class name: **obj_rada**r) are useful for comparing columns or rows of numeric data from a tabular data element returned by your data attachment. The radar graph contains a value axis and a radial axis. The value axis is the vertical axis that displays the maximum and minimum values for the graph. The value axis grid lines are all of the rings except for the outer ring, which is the radial axis.



Using Data and Data Label Properties

To attach data to your graph, right-click in the Property Value field of the **valueTable** property and select **Attach to Data**. If you include a label column in your data attachment, this can be used to label either the radial axis or the legend depending on whether the **rowSeriesFlag** check box is selected or deselected.

The **rowSeriesFlag** controls how data populate the graph:

If the **rowSeriesFlag** is selected, the graph will display one trace per row of data and one radial axis per numeric column of data.

If you included a label column in your data attachment, select **rowLabelVisFlag** and the values from that column will be used as legend labels. By default, the label column is the first non-numeric text column in your data. To set the label column to a specific column, specify a column name using the **labelColumnName** property.

If your data attachment does not have a label column, select the **rowNameVisFlag** to use row names in the legend.

If the **rowSeriesFlag** is not selected, the graph will display one radial axis per row of data and one trace per column of numeric data.

If you included a label column in your data attachment, select **rowLabelVisFlag** and the values from that column will be used as radial axis labels. By default, the label column is the first non-numeric text column in your data. To set the label column to a specific column, specify a column name using the **labelColumnName** property.

If your data attachment does not have a label column, select the **rowNameVisFlag** to use row names on the radial axis.

Using Interaction Properties

Commands

To assign a command to your graph, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Based on your data attachment, substitutions are created that will be passed into drill down displays. To customize which substitutions will be passed into drill down displays, double-click on **drillDownColumnSubs** in the **Object Properties** window to open the "Drill Down Column Substitutions Dialog". Once a drill down target has been set, double-click on the trace marker in the graph to activate the drill down. Drill down displays can be activated in the same window that contains the graph or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Substitutions" for more information.

Use **drillDownSelectMode** to control how a drill down display is activated. Set to **Anywhere** to activate a drill down display by double-clicking anywhere on the graph. Set to **Element Only** to enable a drill down display only when you double-click on a marker in the graph.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your graph. To display a tool tip, point to a trace marker with your mouse. The tool tip will contain information from your data attachment about that marker.

Alert Properties

To set trace marker colors and styles based on a threshold value, select the corresponding value alarm or value warning flags.

Property Name	Description
valueAlarmStatusTable	Attach an alarm table containing status indexes to **valueAlarmStatusTable** to enable rule based alarm statuses for trace markers. The table attached to **valueAlarmStatusTable** must have the same number of rows and columns as **valueTable**. For each data element in **valueTable**, the status index at the corresponding position in **valueAlarmStatusTable** will be used to set the alarm status of the marker. Valid indexes are: **0** = use normal marker color and style, **1** = use low alarm marker color and style, **2** = use low warning marker color and style, 3 = use high warning marker color and style, **4** = use high alarm marker color and style, **-1** = determine marker color and style by comparing the value to the enabled alarm thresholds. If no data is attached to **valueAlarmStatusTable**, then the alarm status for a trace marker is determined by comparing the marker's value to the enabled thresholds.
valueHighAlarmEnabledFlag	Select to enable the high alarm threshold.
valueHighAlarmLineVisFlag	Select to display a dotted line at the high alarm threshold. The color of the line is set to the **valueHighAlarmMarkColor**.
valueHighAlarmMarkColor/ valueHighAlarmMarkStyle	When a trace marker's value is greater than or equal to the **valueHighAlarm** property, the marker will change to the **valueHighAlarmMarkColor** and **valueHighAlarmMarkStyle**.
valueHighWarningEnabledFlag	Select to enable the high warning threshold.
valueHighWarningLineVisFlag	Select to display a dotted line at the high warning threshold. The color of the line is set to the **valueHighWarningMarkColor**.
valueHighWarningMarkColor/ valueHighWarningMarkStyle	When a trace marker's value is greater than or equal to the **valueHighWarning** property but less than the **valueHighAlarm** property, the marker will change to the **valueHighWarningMarkColor** and **valueHighWarningMarkStyle**.

valueLowAlarmEnabledFlag	Select to enable the low alarm threshold.
valueLowAlarmLineVisFlag	Select to display a dotted line at the low alarm threshold. The color of the line is set to the **valueLowAlarmMarkColor**.
valueLowAlarmMarkColor/ valueLowAlarmMarkStyle	When the trace marker's value is less than or equal to the **valueLowAlarm** property, the marker will change to the **valueLowAlarmMarkColor** and **valueLowAlarmMarkStyle**.
valueLowWarningEnabledFlag	Select to enable the low warning threshold.
valueLowWarningLineVisFlag	Select to display a dotted line at the low warning threshold. The color of the line is set to the **valueLowWarningMarkColor**.
valueLowWarningMarkColor/ valueLowWarningMarkStyle	When the trace marker's value is less than or equal to the **valueLowWarning** property, but greater than the **valueLowAlarm** property, the marker will change to the **valueLowWarningMarkColor** and **valueLowWarningMarkStyle**.

Background Properties

Specify how the background is displayed in your radar graph.

Property Name	Description
bgBorderColor	Select the ... button and choose from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **bgColor** property sets the first color in the gradient.
bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.

bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is 100 and **objHeight** is 50, then the value of **bgRoundness** cannot exceed 25. If it does, then half the value of **objHeight** (25) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the graph and the border.

Column Properties

Specify which columns are displayed in your radar graph.

Property Name	Description
columnsToHide	Specify columns from the data attachment to exclude from being used for plotted data or labels. Data from the **labelColumnName** column will be used for labels even if that column name is also specified in the **columnsToHide** property. Columns specified in the **columnsToHide** property can still be used in the **drillDownColumnSubs** property.

Data Properties

Property Name	Description
rowSeriesFlag	Controls how data populate the graph: If the **rowSeriesFlag** is selected, the graph will display one trace per row of data and one radial axis per numeric column of data. If you included a label column in your data attachment, select **rowLabelVisFlag** and the values from that column will be used as legend labels. By default, the label column is the first non-numeric text column in your data. To set the label column to a specific column, specify a column name using the **labelColumnName** property. If your data attachment does not have a label column, select the **rowNameVisFlag** to use row names in the legend. If the **rowSeriesFlag** is not selected, the graph will display one radial axis per row of data and one trace per column of numeric data. If you included a label column in your data attachment, select **rowLabelVisFlag** and the values from that column will be used as radial axis labels. By default, the label column is the first non-numeric text column in your data. To set the label column to a specific column, specify a column name using the **labelColumnName** property. If your data attachment does not have a label column, select the **rowNameVisFlag** to use row names on the radial axis.
valueDivisor	Divides values by the number entered.

valueMin valueMax	The **valueMin** and **valueMax** properties control the range of the value axis if the **valueAxisAutoScaleMode** is set to **Off**. Select **On** for the **valueAxisAutoScaleMode** to calculate the value axis range according to data values being plotted. To calculate the value axis range including **valueMin** and **valueMax**, select **On - Include Min/Max**.
valueTable	Attach your tabular data. If you include a label column in your data attachment, this can be used to label either the radial axis or the legend depending on whether **rowSeriesFlag** is selected or deselected.

Data Format Properties

Property Name	Description
labelColumnFormat	Select or enter the format of numeric or date labels displayed on the radial axis, in the legend and in tool tips. To enter a format, use syntax from the **Java DecimalFormat** class for numeric labels and syntax from the **Java SimpleDateFormat** class for date labels. To enable tool tips, select the **mouseOverFlag**.
valueFormat	Select or enter the numeric format of trace values displayed in tool tips. To enter a format, use syntax from the **Java DecimalFormat** class. To enable tool tips, select the **mouseOverFlag**.

Data Label Properties

Property Name	Description
columnDisplayNames	Set alternate display names for column names in your radar graph's data. Column names are displayed either along the x-axis or in the legend, depending on whether or not the **rowSeriesFlag** is selected.
labelColumnName	Specify a column in your data attachment to be the label column. If left blank, the first non-numeric text column in your data will be used. **Note:** If you include a label column in your data attachment, this can be used to label the legend depending on whether **rowSeriesFlag** is selected or deselected.
rowLabelVisFlag	Control the visibility of labels that appear in the legend or on the radial axis, depending on whether **rowSeriesFlag** is selected or deselected.
rowNameVisFlag	Select to use row names from your data attachment to label either the radial axis or the legend, depending on whether **rowSeriesFlag** is selected or deselected. **Note:** This is useful if your data attachment does not have a label column.

Historian Properties

Property Name	Description
historyTableName	Specify name of table in your history database in which to store tabular data. See "Configuring the Historian" for information.
historyTableRowNameFlag	If selected, data from the row name field will be stored in the first column of the table specified in **historyTableName**.

Interaction Properties

Property Name	Description
command	Assign a command to your graph. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownColumnSubs	Select the ... button to open the "Drill Down Column Substitutions Dialog" to customize which substitutions will be passed into drill down displays.
drillDownSelectMode	Control how a drill down display is activated. Select one of the following options: **Anywhere** - Activate a drill down display by double-clicking anywhere on the graph. **Element Only** - Enable a drill down display only when you double-click on a marker in the graph.
drillDownTarget	Name of display (.rtv) file targeted as a drill down. See "Drill Down Displays" for information.
mouseOverFlag	Select to enable tool tips for your radar graph. To display a tool tip, point to a trace marker with your mouse. The tool tip will contain information from your data attachment about that marker.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. This property is only applies if **labelTextAlignY** is set to **Tab Top**.
labelTextAlignX	Select x-axis position of label text from the drop down menu.

labelTextAlignY	Select y-axis position of label text from the drop down menu. **Outside Top** - Position label well above the background rectangle. **Top** - Position label just above the background rectangle. **Title Top** - Position label along the top line of the background rectangle. **Tab Top** - Position label tab just above the background rectangle. Height and width of the tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. **Inside Top** - Position label inside the top of the background rectangle.
labelTextColor	Select the ... button and choose from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop down menu.
labelTextHeight	Set the height of the label text in pixels.

Legend Properties

Specify the way the legend is displayed in your radar graph.

Property Name	Description
legendBgColor	Select the ... button and choose from the palette to set the background color of the legend.
legendBgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **legendBgColor** property sets the first color in the gradient.
legendBgGradientMode	Display a gradient in the legend background. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
legendVisFlag	Select to display the legend.
legendWidthPercent	Set the percent of the total width of the object used for the legend.
outlineColor	Select the ... button and choose a color from the palette to set the color of the one-pixel outline around the legend area and around each color swatch within the legend. The default value is **black** (color 7).

Marker Properties

Property Name	Description

markDefaultSize	Specify (in pixels) the size of the markers.
markScaleMode	Scale markers according to the order of the data in your data attachment (i.e. The marker for the first data in the attachment is the smallest and the marker for the last data is the largest.) Select from the following options: **No Scale**, **Scale by Trace**, **Scale Within Trace**.

Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Plot x-axis position of object.
objY	Plot y-axis position of object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Plot Area Properties

Specify how the plot area is displayed in your radar graph.

Property Name	Description
gridColor	Select the button and choose from the palette to set the color of the grid lines.
plotBgColor	Select the button and choose from the palette to set the background color of the plot area.
plotBgGradientColor2	Select the button and choose the second color in the gradient. Default is **white**. **Note:** The **plotBgColor** property sets the first color in the gradient.
plotBgGradientMode	Display a gradient in the plot background. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
plotBgImage	Select an image to display in the plot background of your graph. If necessary, the image will be stretched to fit the plot area.

Radial Axis Properties

Specify how the radial axis is displayed in your radar graph. The radar graph contains a value axis and a radial axis. The value axis is the vertical axis that displays the maximum and minimum values for the graph. The value axis grid lines are all of the rings except for the outer ring, which is the radial axis. The radial axis grid lines are the spokes.

Property Name	Description
radialAxisColor	Select the button and choose from the palette to set the color of the radial axis and radial axis labels.
radialAxisLabelVisFlag	Select to display the radial axis labels.
radialAxisLineStyle	Select the line style of the radial axis from the drop down menu: **No Line**, **Solid**, **Dotted**, **Dashed**, or **Dot Dashed**.
radialAxisMinLabelWidth	Specify the minimum width in pixels for the radial axis labels.
radialAxisVisFlag	Select to display the radial axis.
radialGridLineStyle	Select the line style of the radial grid lines from the drop down menu: **No Line**, **Solid**, **Dotted**, **Dashed**, or **Dot Dashed**.
radialGridVisFlag	Select to display the radial grid lines.

Trace Properties

Specify how traces are displayed in your radar graph.

Property Name	Description
traceFillStyle	Specify the style to fill the area under the trace. Select from the following options: **Solid**, **Transparent**, **Gradient**, or **Transparent Gradient**. Default is **None**.
traceProperties	Specify the line color, line style, line width, marker color, marker style, and fill style of all traces.

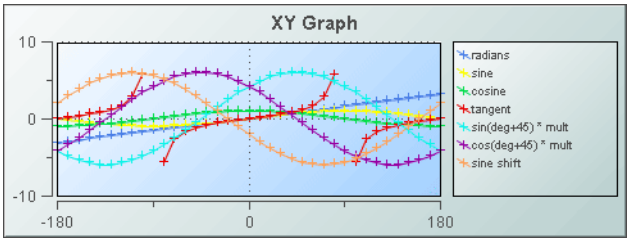
Value Axis Properties

Specify how the value axis is displayed in your radar graph. The radar graph contains a value axis and a radial axis. The value axis is the vertical axis that displays the maximum and minimum values for the graph. The value axis grid lines are all of the rings except for the outer ring, which is the radial axis. The radial axis grid lines are the spokes.

Property Name	Description
valueAxisAutoScaleMode	The **valueMin** and **valueMax** properties control the range of the value axis if the **valueAxisAutoScaleMode** is set to **Off**. Select **On** for the **valueAxisAutoScaleMode** to calculate the value axis range according to data values being plotted. To calculate the value axis range including **valueMin** and **valueMax**, select **On - Include Min/Max**.
valueAxisColor	Select the  button and choose from the palette to set the color of the value axis and value axis labels.
valueAxisFlag	Select to display the value axis.
valueAxisFormat	Select or enter the numeric format of trace values displayed on the value axis. To enter a format, use syntax from the **Java DecimalFormat** class.
valueAxisLineStyle	Select the line style of the value axis from the drop down menu: **No Line**, **Solid**, **Dotted**, **Dashed**, or **Dot Dashed**.
valueAxisMajorDivisions	Specify the number of major divisions on the value axis.
valueAxisMinorDivisions	Specify the number of minor divisions on the value axis.
valueGridLineStyle	Select the line style of the grid line from the drop down menu: **No Line**, **Solid**, **Dotted**, **Dashed**, or **Dot Dashed**.
valueGridVisFlag	Select to display the grid lines.

XY Graphs

XY graphs (**obj_xygraph**) are useful for comparing pairs of values from a tabular data element returned by your data attachment. XY graphs plot quantitative data on both axes.



Using Data and Data Label Properties

To attach data to your graph, right-click in the **Property Value** field of the **valueTable** property and select **Attach to Data**. Your data attachment must contain numeric data for both the x and y axes. If you include a label column in your data attachment, this can be used to label the legend depending on whether the **rowSeriesFlag** check box is selected or deselected.

The **rowSeriesFlag** controls how data populate the graph:

If the **rowSeriesFlag** check box is selected, your first row of data is used for the x-axis. The remainder will be plotted on the y-axis, with one trace per row of data containing one point for each column.

If you included a label column in your data attachment, select **rowLabelVisFlag** and values from that column will be used as legend labels. By default, the label column is the first non-numeric text column in your data. To set the label column to a specific column, specify a column name using the **labelColumnName** property.

If your data attachment does not have a label column, select the **rowNameVisFlag** check box to use row names in the legend.

If the **rowSeriesFlag** check box is not selected, your first column of data is used for the x-axis. The remainder will be plotted on the y-axis, with one trace per column of data containing one point per row.

If you included a label column in your data attachment, select **rowLabelVisFlag** and values from that column will be used as x-axis labels. Column names will appear in the legend.

Using Interaction Properties

Commands

To assign a command to your graph, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (.rtv) file can be targeted as a drill down. Based on your data attachment, substitutions are created that will be passed into drill down displays. To customize which substitutions will be passed into drill down displays, double-click on **drillDownColumnSubs** in the Object Properties window to open the "Drill Down Column Substitutions Dialog". Once a drill down target has been set, double-click on the trace marker to activate the drill down. Drill down displays can be activated in the same window that contains the graph or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Use **drillDownSelectMode** to control how a drill down display is activated. Set to Anywhere to activate a drill down display by double-clicking anywhere on the graph. Set to Element Only to enable a drill down display only when you double-click on a marker in the graph.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your XY graph. To display a tool tip, point to a trace marker with your mouse. The tool tip will contain information from your data attachment about that trace marker.

Alert Properties

To set trace marker colors and styles based on a threshold value, select the corresponding value alarm or value warning flags.

Property Name	Description
valueAlarmStatusTable	Attach an alarm table containing status indexes to **valueAlarmStatusTable** to enable rule based alarm statuses for trace markers. The table attached to **valueAlarmStatusTable** must have the same number of rows and columns as **valueTable**. For each data element in **valueTable**, the status index at the corresponding position in **valueAlarmStatusTable** is used to set the alarm status of the marker. Valid indexes are: **0** = use normal marker color and style, **1** = use low alarm marker color and style, **2** = use low warning marker color and style, **3** = use high warning marker color and style, **4** = use high alarm marker color and style, **-1** = determine marker color and style by comparing the value to the enabled alarm thresholds. If no data is attached to **valueAlarmStatusTable**, the alarm status for a trace marker is determined by comparing the marker's value to the enabled thresholds.
valueHighAlarmEnabledFlag	Select to enable the high alarm threshold.
valueHighAlarmLineVisFlag	Select to display a dotted line at the high alarm threshold. The color of the line is set to the **valueHighAlarmMarkColor**.
valueHighAlarmMarkColor/ valueHighAlarmMarkStyle	When a trace marker's value is greater than or equal to the **valueHighAlarm** property, the marker will change to the **valueHighAlarmMarkColor** and **valueHighAlarmMarkStyle**.
valueHighAlarmTraceColor/ valueHighAlarmTraceStyle	When the value of any segment of a trace line is greater than or equal to the **valueHighAlarm** property, that segment of the trace line will change to the **valueHighAlarmTraceColor** and **valueHighAlarmTraceStyle**. **Note:** If **valueHighAlarmTraceStyle** is set to **No Line**, then **valueHighAlarmTraceColor** will not change.
valueHighWarningEnabledFlag	Select to enable the high warning threshold.
valueHighWarningLineVisFlag	Select to display a dotted line at the high warning threshold. The color of the line is set to the **valueHighWarningMarkColor**.
valueHighWarningMarkColor/ valueHighWarningMarkStyle	When a trace marker's value is greater than or equal to the **valueHighWarning** property but less than the **valueHighAlarm** property, the marker will change to the **valueHighWarningMarkColor** and **valueHighWarningMarkStyle**.
valueHighWarningTraceColor/ valueHighWarningTraceStyle	When the value of any segment of a trace line is greater than or equal to the **valueHighWarning** property but less than the **valueHighAlarm** property, that segment of the trace line will change to the **valueHighWarningTraceColor** and **valueHighWarningTraceStyle**.

valueLowAlarmEnabledFlag	Select to enable the low alarm threshold.
valueLowAlarmLineVisFlag	Select to display a dotted line at the low alarm threshold. The color of the line is set to the **valueLowAlarmMarkColor**.
valueLowAlarmMarkColor/ valueLowAlarmMarkStyle	When the trace marker's value is less than or equal to the **valueLowAlarm** property, the marker will change to the **valueLowAlarmMarkColor** and **valueLowAlarmMarkStyle**.
valueLowAlarmTraceColor/ valueLowAlarmTraceStyle	When the value of any segment of a trace line is less than or equal to the **valueLowAlarm** property, that segment of the trace line will change to the **valueLowAlarmTraceColor** and **valueLowAlarmTraceStyle**.
valueLowWarningEnabledFlag	Select to enable the low warning threshold.
valueLowWarningLineVisFlag	Select to display a dotted line at the low warning threshold. The color of the line is set to the **valueLowWarningMarkColor**.
valueLowWarningMarkColor/ valueLowWarningMarkStyle	When the trace marker's value is less than or equal to the **valueLowWarning** property, but greater than the **valueLowAlarm** property, the marker will change to the **valueLowWarningMarkColor** and **valueLowWarningMarkStyle**.
valueLowWarningTraceColor/ valueLowWarningTraceStyle	When the value of any segment of a trace line is less than or equal to the **valueLowWarning** property, but greater than the **valueLowAlarm** property, that segment of the trace line will change to the **valueLowWarningTraceColor** and **valueLowWarningTraceStyle**.

Background Properties

Specify how the background is displayed in your XY graph.

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose a color from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is 3D Rectangle.
bgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **bgColor** property sets the first color in the gradient.

bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is Round Rectangle. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the graph and the border.

Column Properties

Specify which columns are displayed in your XY graph.

Property Name	Description
columnsToHide	Specify columns from the data attachment to exclude from being used for plotted data or labels. Data from the **labelColumnName** column will be used for labels even if that column name is also specified in the **columnsToHide** property. Columns specified in the **columnsToHide** property can still be used in the **drillDownColumnSubs** property.

Data Properties

Property Name	Description
rowSeriesFlag	Controls how data populate the graph: • If the **rowSeriesFlag** check box is selected, your first row of data is used for the x-axis. The remainder will be plotted on the y-axis, with one trace per row of data containing one point for each column. • If you included a label column in your data attachment, select **rowLabelVisFlag** and values from that column will be used as legend labels. By default, the label column is the first non-numeric text column in your data. To set the label column to a specific column, specify a column name using the **labelColumnName** property. • If your data attachment does not have a label column, select the **rowNameVisFlag** check box to use row names in the legend. • If the **rowSeriesFlag** check box is not selected, your first column of data is used for the x-axis. The remainder will be plotted on the y-axis, with one trace per column of data containing one point per row. • If you included a label column in your data attachment, select **rowLabelVisFlag** and values from that column will be used as x-axis labels. Column names will appear in the legend.
valueTable	Attach your tabular data. Your data attachment must contain numeric data for both the x and y axes. If you include a label column in your data attachment, this can be used to label the legend depending on whether **rowSeriesFlag** is selected or deselected.
xValueDivisor	The x-axis values are divided by the value entered.
xValueMin xValueMax	The **xValueMin** and **xValueMax** properties control the range of x-axis if the **xAxisAutoScaleMode** is set to Off. Select On for the **xAxisAutoScaleMode** to calculate x-axis range according to data values being plotted. To calculate the x-axis range including **xValueMin** and **xValueMax**, select **On - Include Min/Max**.
yValueDivisor	The y-axis values are divided by the value entered.
yValueMin yValueMax	The **yValueMin** and **yValueMax** properties control the range of y-axis if the **yAxisAutoScaleMode** is set to **Off**. Select **On** for the **yAxisAutoScaleMode** to calculate y-axis range according to data values being plotted. To calculate the y-axis range including **yValueMin** and **yValueMax**, select **On - Include Min/Max**.

Data Format Properties

Property Name	Description
labelColumnFormat	Select or enter the format of numeric or date labels displayed in the legend and popup legend. To enter a format, use syntax from the Java DecimalFormat class for numeric labels and syntax from the Java SimpleDateFormat class for date labels.
xValueFormat yValueFormat	Select or enter the numeric format of trace values displayed in the legend and popup legend. To enter a format, use syntax from the Java DecimalFormat class.

Data Label Properties

Property Name	Description
columnDisplayNames	Set alternate display names for column names in your XY graph's data. Column names are displayed either along the x-axis or in the legend, depending on whether or not the **rowSeriesFlag** is selected.
labelColumnName	Specify a column in your data attachment to be the label column. If left blank, the first non-numeric text column in your data will be used. If you include a label column in your data attachment, this can be used to label the x-axis or the legend depending on whether **rowSeriesFlag** is selected or deselected.

Historian Properties

Property Name	Description
historyTableName	Specify name of table in your history database in which to store tabular data. See Configuring the Historian for information.
historyTableRowNameFlag	If selected, data from the row name field will be stored in the first column of the table specified in historyTableName.

Interaction Properties

Property Name	Description
command	Assign a command to your graph. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the **Command Confirmation** dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
cursorColor	Select the ... button and choose a color from the palette. The default is **yellow**. **Note:** This property is only available if **cursorFlag** is selected.

cursorFlag	Select to enable the cursor. When the cursor is enabled, point to a location on a trace to see a cursor line at that location and display the time and values of all traces at the cursor line on the legend. Hold down the Control key to snap the cursor to the closest data point. Select the **legendPopupFlag** to display the legend along the cursor.
drillDownColumnSubs	Select the  button to open the "Drill Down Column Substitutions Dialog" to customize which substitutions will be passed into drill down displays.
drillDownSelectMode	Control how a drill down display is activated. Select one of the following options: **Anywhere** - Activate a drill down display by double-clicking anywhere on the graph. **Element Only** - Enable a drill down display only when you double-click on a trace marker in the graph.
drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information.
legendPopupFlag	Select to display the legend along the cursor. **Note:** This property is only available if **cursorFlag** is selected.
mouseOverFlag	Select to enable tool tips for your graph. To display a tool tip, point to a trace marker with your mouse. The tool tip will contain information from your data attachment about that trace marker.
outlineColor	Select the  button and choose a color from the palette to set the color of the one-pixel outline around the legend area, around each color swatch within the legend, and around the plot area. The default value is **black** (color 7).
scrollbarMode	Set whether and when the scroll bar appears in the graph. There are three options: **Never** - This is the default. **Always** - Displays a scroll bar at all times. **As Needed** - Displays the scroll bar when necessitated by zooming in the trace area, or when you have x or y values that are outside of the min/max range.
scrollbarSize	Specify (in pixels) the height of the horizontal scroll bar and the width of the vertical scroll bar. The default value is **-1**, which sets the size to the system default.
zoomEnabledFlag	Select to enable zooming within the graph. Click in the graph's trace area and drag the cursor until a desired range is selected. While dragging, a yellow rectangle is drawn to show the zoom area. After the zoom is performed, the graph stores up to four zoom operations in queue. To zoom out, hold the Shift key and click in the graph's trace area.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the  button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. This property is only applies if **labelTextAlignY** is set to **Tab Top**.
labelTextAlignX	Select x-axis position of label text from the drop down menu.

labelTextAlignY	Select y-axis position of label text from the drop down menu. **Outside Top** - Position label well above the background rectangle. **Top** - Position label just above the background rectangle. **Title Top** - Position label along the top line of the background rectangle. **Tab Top** - Position label tab just above the background rectangle. Height and width of the tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. **Inside Top** - Position label inside the top of the background rectangle.
labelTextColor	Select the ... button and choose a color from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop down menu.
labelTextHeight	Set the height of the label text in pixels.

Legend Properties

Property Name	Description
legendBgColor	Select the ... button and choose a color from the palette to set the background color of the legend.
legendBgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **legendBgColor** property sets the first color in the gradient.
legendBgGradientMode	Display a gradient in the legend background. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
legendValueMinSpace	Specify the minimum distance between values and labels in the legend.
legendVisFlag	Select to display the legend.
legendValueVisFlag	Select to display the numerical values of your data in the legend.
legendWidthPercent	Set the percent of the total width of the object used for the legend.

Marker Properties

Property Name	Description
markDefaultSize	Specify (in pixels) the size of the markers.

markScaleMode	Scale markers according to the order of the data in your data attachment (i.e. The marker for the first data in the attachment is the smallest and the marker for the last data is the largest.) Select from the following options: **No Scale**, **Scale by Trace**, or **Scale Within Trace**.

Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Plot x-axis position of object.
objY	Plot y-axis position of object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Plot Area Properties

Property Name	Description
gridBgColor	Select the ... button and choose a color from the palette to set the color of the grid background.
gridBgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **gridBgColor** property sets the first color in the gradient.
gridBgGradientMode	Display a gradient in the grid background. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
gridBgImage	Select an image to display in the grid background of your graph. **Note:** If necessary, the image will be stretched to fit the grid.
gridColor	Select the ... button and choose from the palette to set the color of the grid lines. Default is **white**.

Trace Properties

Property Name	Description
traceFillStyle	Specify the style to fill the area under the trace. Select from the following options: **Solid**, **Transparent**, **Gradient**, or **Transparent Gradient**. Default is **None**.
traceProperties	Specify the line color, line style, line width, marker color, marker style and fill style of all traces.

X-Axis Properties

Specify how the x axis is displayed on your XY graph.

Property Name	Description
xAxisAutoScaleMode	The **xValueMin** and **xValueMax** properties control the range of x-axis if the **xAxisAutoScaleMode** is set to **Off**. Select **On** for the **xAxisAutoScaleMode** to calculate x-axis range according to data values being plotted. To calculate the x-axis range including **xValueMin** and **xValueMax**, select **On - Include Min/Max**.
xAxisAutoScaleRoundFlag	Select the **xAxisAutoScaleRoundFlag** to round values on the x-axis.

xAxisFlag	Select to display the x-axis.
xAxisFormat	Select or enter the numeric format of values displayed on the x-axis. To enter a format, use syntax from the **Java DecimalFormat** class.
xAxisLabel	Specify the x-axis label.
xAxisLabelTextHeight	Specify the height in pixels of the x-axis labels.
xAxisMajorDivisions	Specify the number of major divisions on the x-axis.
xAxisMinorDivisions	Specify the number of minor divisions on the x- axis.
xAxisReverseFlag	Select to reverse the order of the x-axis values and plot values decreasing from left to right.
xValueSortFlag	Select to sort data from lowest to highest x values.

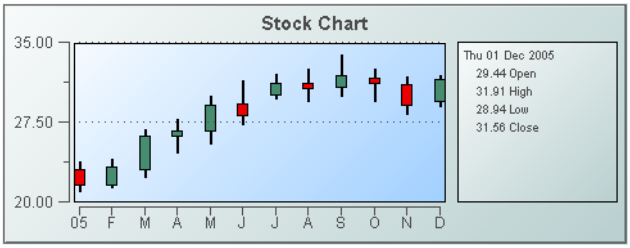
Y-Axis Properties

Specify how the y axis is displayed on your XY graph.

Property Name	Description
yAxisAutoScaleMode	The **yValueMin** and **yValueMax** properties control the range of y-axis if the **yAxisAutoScaleMode** is set to **Off**. Select **On** for the **yAxisAutoScaleMode** to calculate y-axis range according to data values being plotted. To calculate the y-axis range including **yValueMin** and **yValueMax**, select **On - Include Min/Max**.
yAxisFormat	Select or enter the numeric format of values displayed on the y-axis. To enter a format, use syntax from the **Java DecimalFormat** class.
yAxisFlag	Select to display the y-axis.
yAxisLabel	Specify the y-axis label.
yAxisLabelTextHeight	Specify the height in pixels of the y-axis labels.
yAxisMajorDivisions	Specify the number of major divisions on the y-axis.
yAxisMinLabelWidth	Specify the minimum width in pixels for the y-axis labels.
yAxisMinorDivisions	Specify the number of minor divisions on the y-axis.
yAxisMultiRangeFlag	Select to enable one axis per trace with each trace having its own range.

Stock Chart

Stock charts (class name: **obj_stockchart**) are useful for displaying live and archived stock data. Overlays can be used to compare this data against other stock, overall activity on the stock index, or to show periodic events such as stock splits and earnings announcements. The chart supports up to nineteen overlays.



Using Price Trace Properties

To attach tabular data to your chart, right-click in the **Property Value** field of the **priceTraceCurrentTable** property or the **priceTraceHistoryTable** property and select **Attach to Data**.

Historical Data	Current Data
To display historical data, attach to **priceTraceHistoryTable**.	To display current data, attach to **priceTraceCurrentTable**.
The attached historical data table must contain the following five columns (as described in the table below) in this specific order: Date, Open, High, Low, Close.	The attached data table must contain the following five columns (as described in the table below) in this specific order: Date, Open, High, Low, Close.
Note: See the **Attach to Data** section specific to your data source for details on how to use the **Select Columns** dialog.	**Note:** See the **Attach to Data** section specific to your data source for details on how to use the **Select Columns** dialog.
The **priceTraceHistoryTable** property is used for viewing and analyzing archived data (data generated before the current moment). The table in your data attachment should contain multiple rows, each corresponding to a point on the graph.	The **priceTraceCurrentTable** property is used for viewing live data and would most likely be used in conjunction with the **priceTraceHistoryTable**. The table in your data attachment should contain a single row that corresponds to and continually updates the last point on the graph.

Column Name	Description
Date	Supported formats for time column are: **mm/dd/yyyy hh:mm:ss** (01/16/2004 12:30:03) **yyyy-mm-dd hh:mm:ss** (2004-01-16 12:30:03) The number of milliseconds since midnight, January 1, 1970 UTC
Open	Value of stock price at first market open for defined time period.
High	High value of stock price for defined time period.
Low	Low value of stock price for defined time period.
Close	Value of stock price at last market close for defined time period.

Using Trace* Properties

The Display Builder automatically creates a set of **overlay*** properties according to the number you specify using the **overlayCount** object property. For example, if you set the **overlayCount** to three (3), you will have three overlay traces in your graph (Trace 01, Trace 02 and Trace 03) and a corresponding set of **overlay*** properties for each.

The **overlay*CurrentTable** and **overlay*HistoryTable** properties are used in conjunction with the **priceTraceHistoryTable** or **priceTraceCurrentTable** properties to compare data (e.g. to compare the activity of several stocks). Tabular data attached to **overlay*Table** properties must have the following two columns: a **Date** column (as described in the table above) and a **Value** column (numeric). The **overlay*HistoryTable** property should contain multiple rows, each corresponding to a point on the graph.

Using Interaction Properties

Commands

To assign a command to your graph, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Based on your data attachment, substitutions are created that will be passed into drill down displays. To customize which substitutions will be passed into drill down displays, double-click on **drillDownColumnSubs** in the **Object Properties** window to open the "Drill Down Column Substitutions Dialog". Once a drill down target has been set, double-click on the bar or trace marker in the chart to activate the drill down. Drill down displays can be activated in the same window that contains the chart or open in a separate window. This allows you to build a customizable hierarchy of displays.

Use **drillDownSelectMode** to control how a drill down display is activated. Set to **Anywhere** to activate a drill down display by double-clicking anywhere on the chart. Set to **Element Only** to enable a drill down display only when you double-click on a bar in the chart.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your stock chart. To display a tool tip, point to a bar or marker with the mouse. The tool tip will contain information from your data attachment about that bar or marker.

Background Properties

Specify how the background is displayed in the stock chart.

Property Name	Description
bgBorderColor	Select the ... button and choose from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. Default is **white**. **Note:** The **bgColor** property sets the first color in the gradient.

bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. Vertical Center Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is Round Rectangle. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (25) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the chart and the border.

Data Properties

Property Name	Description
yValueMax yValueMin	Controls the range of y-axis if the **yAxisAutoScaleMode** is set to **Off**. Select **On** for the **yAxisAutoScaleMode** to calculate the y-axis range according to data values being plotted. To calculate the y-axis range including **yValueMin** and **yValueMax**, select **On - Include Min/Max**.

Data Format Properties

Specify data format in your stock chart.

Property Name	Description
yValueFormat	Select or enter the numeric format of values displayed in the legend and popup legend. To enter a format, use syntax from the **Java DecimalFormat** class.

Historian Properties

Property Name	Description
historyTableName	Specify name of table in your history database in which to store tabular data. See "Configuring the Historian" for information.
historyTableRowNameFlag	If selected, data from the row name field will be stored in the first column of the table specified in **historyTableName**.

Interaction Properties

Property Name	Description
command	Assign a command to your graph. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
cursorColor	Select the ... button and choose from the palette. The default is **yellow**. **Note:** This property is only available if **cursorFlag** is selected.
cursorFlag	Select to enable the cursor. When the cursor is enabled, select the chart and point to a location on a trace to see a cursor line at that location and display the time and values of all traces at the cursor line on the legend. Select the **legendPopupFlag** to display the legend along the cursor.
drillDownColumnSubs	Select the ... button to open the "Drill Down Column Substitutions Dialog" to customize which substitutions will be passed into drill down displays.
drillDownSelectMode	Control how a drill down display is activated. Select one of the following options: **Anywhere** - Activate a drill down display by double-clicking anywhere on the graph. **Element Only** - Enable a drill down display only when you double-click on a bar in the graph.

drillDownTarget	Name of display (.rtv) file targeted as a drill down. See "Drill Down Displays" for information.
legendPopupFlag	Select to display the legend along the cursor. **Note:** This property is only applicable if **cursorFlag** is selected.
mouseOverFlag	Select to enable tool tips for your stock chart. To display a tool tip, select the chart and point to a bar or marker with the mouse. The tool tip will contain information from your data attachment about that bar or marker.
scrollbarMode	Set whether and when the scroll bar appears in the chart. There are three options: **Never** - This is the default. **Always** - Displays a scroll bar at all times. **As Needed** - Displays the scroll bar when necessitated by zooming in the trace area, or when more data is loaded into the chart than is displayed in the time range. For example, if the time range of the data in your data attachment is greater than **timeRange**, setting **scrollbarMode** to **As Needed** will enable a scroll bar to view all data loaded into the chart.
scrollbarSize	Specify (in pixels) the height of the horizontal scroll bar and the width of the vertical scroll bar. The default value is **-1**, which sets the size to the system default.
zoomEnabledFlag	Select to enable zooming within the chart. Click in the chart's trace area and drag the cursor until a desired range is selected. While dragging, a yellow rectangle is drawn to show the zoom area. After the zoom is performed, the chart stores up to four zoom operations in queue. To zoom out, hold the Shift key and click in the chart's trace area.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the  button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. This property is only applies if **labelTextAlignY** is set to **Tab Top**.
labelTextAlignX	Select x-axis position of label text from the drop down menu.
labelTextAlignY	Select y-axis position of label text from the drop down menu. **Outside Top** - Position label well above the background rectangle. **Top** - Position label just above the background rectangle. **Title Top** - Position label along the top line of the background rectangle. **Tab Top** - Position label tab just above the background rectangle. Height and width of the tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. **Inside Top** - Position label inside the top of the background rectangle.
labelTextColor	Select the  button and choose from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop down menu.
labelTextHeight	Set the height of the label text in pixels.

Legend Properties

Specify the way the legend is displayed in the stock chart.

Property Name	Description
legendBgColor	Select the button and choose from the palette to set the background color of the legend.
legendBgGradientColor2	Select the button and choose a color for the second color in the gradient. Default is **white**. **Note:** The legendBgColor property sets the first color in the gradient.
legendBgGradientMode	Display a gradient in the legend background. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
legendValueMinSpace	Specify the minimum distance between values and labels in the legend.
legendValueVisFlag	Select to display the numerical values of your data in the legend. If the **cursorFlag** is enabled, the numerical values are always shown in the legend.
legendVisFlag	Select to display the legend.
legendWidthPercent	Set the percent of the total width of the object used for the legend.
outlineColor	Select the button and choose a color from the palette to set the color of the one-pixel outline around the legend area, around each color swatch within the legend, and around the plot area. The default value is **black** (color 7).

Object Properties

Property Name	Description
anchor	Select the button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. **Note:** If an object has the dock property set, the anchor property will be ignored.

dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Plot x-axis position of object.
objY	Plot y-axis position of object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Plot Area Properties

Specify how the plot area is displayed in your stock chart.

Property Name	Description
gridBgColor	Select the ... button and choose from the palette to set the background color of the plot area.
gridBgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **gridBgColor** property sets the first color in the gradient.

gridBgGradientMode	Display a gradient in the plot background. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
gridBgImage	Select an image to display in the plot background of your graph. If necessary, the image will be stretched to fit the plot area.
gridColor	Select the ... button and choose from the palette to set the color of the grid line.

Price Trace Properties

Property Name	Description
priceTraceBarGainColor	Set the color to indicate that a stock price value at market close is greater than value at market open. Select the ... button and choose from the palette. The default is **green**. **Note:** This property does not apply if the selected **priceTraceType** is Line.
priceTraceBarLossColor	Set the color to indicate that a stock price value at market close is less than value at market open. Select the ... button and choose from the palette. Default is **red**. **Note:** This property does not apply if the selected **priceTraceType** is Line.
priceTraceCurrentTable	Attach your tabular data. The attached data table must contain the following five columns (**Date**, **Open**, **High**, **Low**, **Close**) as described in the "Using Price Trace Properties" section. **Note:** The **priceTraceCurrentTable** is used for viewing live data and would most likely be used in conjunction with the **priceTraceHistoryTable**.
priceTraceFillStyle	Select the appearance of the price trace from the drop down menu: **Solid**, **Transparent**, **Gradient**, or **Transparent Gradient**. Set to **None**, the default, to disable. The effects change based on the selected **priceTraceType**: **Line** - Fills from the line to the bottom of the graph with the color and effect you choose. **Bar** - Has no effect. **OHLC** - Has no effect. **Candlestick** - Fills the Candlestick icon with the color and effect you choose.
priceTraceHistoryTable	Attach your historical tabular data. The attached data table must contain the following five columns (**Date**, **Open**, **High**, **Low**, **Close**) as described in the "Using Price Trace Properties" section.
priceTraceLabel	Enter a label for the price trace line. Appears in legend and tool tip enabled by **mouseOverFlag**.

priceTraceLineColor	Select the ... button and choose from the palette to set the price trace line color. **Note:** This property does not apply if the selected **priceTraceType** is OHLC or Bar.
priceTraceLineStyle	Select the style of the price trace line from the drop down menu: **No Line**, **Solid**, **Dotted**, **Dashed**, or **Dot Dashed**. **Note:** This property does not apply if the selected **priceTraceType** is OHLC or Bar.
priceTraceLineThickness	Select the thickness of the price trace line from the drop down menu: **Thin**, **Medium**, or **Thick**. **Note:** This property does not apply if the selected **priceTraceType** is OHLC or Bar.
priceTraceType	Select the appearance of the price trace: **Line** - A line graph showing closing price values. **Bar** - A bar graph showing closing price values. **OHLC** -  - A bar extending from the low to high price data points. A left flange indicates the opening price and the right flange indicates the closing price. The **priceTraceBarLossColor** and **priceTraceBarGainColor** properties show whether the stock closed at a higher or lower price than the opening price. **Candlestick** -  - A bar extending from the opening to closing price data points. The "wicks" on either end show the high and low for the day. The **priceTraceBarLossColor** and **priceTraceBarGainColor** properties show whether the stock closed at a higher or lower price than the opening price.
priceTraceVisFlag	Select to control price trace visibility.

Trace Properties

Property Name	Description
overlayCount	Set the number of overlay traces you want to display. Maximum is **19**. RTView automatically creates a set of properties (see "Trace * Properties") for each overlay.
overlayFillStyle	Select the fill style of the price trace from the drop down menu. Sets the appearance of the **overlay*Type**. Set to **Solid**, **Transparent**, **Gradient**, or **Transparent Gradient**. Set to **None**, the default, to disable. The effects change based on the **overlay*Type**: **Line** - Fills from the line to the bottom of the graph with the color and effect you choose. **Bar** - Has no effect. **Event**- Has no effect.

Trace * Properties

Number of overlay traces displayed depend on the specified **overlayCount**.

Property Name	Description
overlay*CurrentTable	Attach your tabular data. The attached data table must have a **Date** column (as described in the "Using Trace* Properties" section) and a **Value** (numeric) column.

overlay*HistoryTable	Attach your historical tabular data. The attached data table must have a **Date** column (as described in the "Using Trace* Properties" section) and a **Value** (numeric) column.
overlay*Label	Enter a label for the overlay line. Appears in legend and tool tip enabled by **mouseOverFlag**.
overlay*LineColor	Select the  button and choose from the palette to set the overlay line color.
overlay*LineStyle	Select the style of the overlay line: **No Line** **Solid** **Dotted** **Dashed** **Dot Dashed** **Note:** This property does not apply if the selected overlay*Type is Bar or Event.
overlay*LineThickness	Select the thickness of the overlay line: **No Line** **Solid,** **Dotted,** **Dashed** **Dot Dashed.** **Note:** This property does not apply if the selected **overlay*Type** is **Bar** or **Event**.
overlay*Type	Select the appearance of the overlay trace from the drop down menu: **Line** - A line graph. **Bar** - A bar graph. **Event** - A series of markers representing company events such as stock splits, company merges, etc. The first letter of the **overlay*Label** is the letter that appears in each event marker.
overlay*VisFlag	Select to control overlay trace visibility.

X-Axis Properties

Specify the way the x axis is displayed and set time values to control how much data is plotted in the stock chart.

Property Name	Description
timeFormat	Set the format for the time displayed in the x-axis using syntax from the **Java SimpleDateFormat** class. For example, **MMMM dd, yyyy hh:mm:ss** a would result in the form August 30, 2003 05:32:12 PM. If no format is given, the date and time will not be displayed on the x-axis. Include a new line character ('\n') to display multiple line text in the time axis labels. For example, **MM\dd'\n'hh:mm:ss** would result in the form 08\30 05:32:12. If left blank, the axis is labeled with a default format based on the range. **Note:** This property is only used when the **timeRangeMode** is **Continuous**.

timeRange	Sets the total amount of time, in seconds, plotted on the chart. If **timeRange** is set to **-1**, the time range is determined by the first and last **timestamp** found in the **priceTraceHistoryTable** and **priceTraceCurrentTable**. If both tables are empty, the chart uses the first and last **timestamp** of the first overlay trace that has a non-empty **overlay*HistoryTable** and/or **overlay*CurrentTable**. **Note: timeRange** is ignored if both **timeRangeBegin** and **timeRangeEnd** are set.
timeRangeBegin	Set the start time value of the data to be plotted on the chart. Supported formats are: **mm/dd/yyyy hh:mm:ss** (e.g., 01/16/2004 12:30:03) **yyyy-mm-dd hh:mm:ss** (e.g., 2004-01-16 12:30:03) The number of milliseconds since midnight, January 1, 1970 UTC **Note:** If only the time is specified, then today's date will be used.
timeRangeEnd	Set the end time value of the data to be plotted on the chart. Supported formats are: **mm/dd/yyyy hh:mm:ss** (e.g., 01/16/2004 12:30:03) **yyyy-mm-dd hh:mm:ss** (e.g., 2004-01-16 12:30:03) The number of milliseconds since midnight, January 1, 1970 UTC **Note:** If only the time is specified, then today's date will be used.
timeRangeMode	Select the **timeRangeMode** from the drop down menu. Sets the interval between trace data points. **timeRangeMode** also determines the type of label to be used in a graph. For example, the label used on the x-axis could be the date or the time depending on the time interval specified for the data to be displayed. There are eight modes: **Auto** - Selects the setting that best matches the time intervals in the price trace data table. **Intra-Day** - Time intervals are less than one day: hourly, every 15 minutes, etc. **Daily** - Time intervals are days. **Weekly** - Time intervals are weeks. **Monthly** - Time intervals are months. **Quarterly** - Time intervals are quarters. **Yearly** - Time intervals are annual. **Continuous** - Plots each point using the corresponding timestamp from the data table. This data can vary in time intervals. **Note:** If the price trace data is more granular than the time interval specified in your data attachment, the price trace data will be aggregated to match the **timeRangeMode**.
xAxisFlag	Select to display the x-axis.
xAxisLabel	Specify label to display below the x-axis.
xAxisLabelTextHeight	Specify the height in pixels of the x-axis labels.
xAxisMajorDivisions	Specify the number of major divisions on the x-axis.
xAxisMinorDivisions	Specify the number of minor divisions on the x-axis. **Note:** This property applies when the **timeRangeMode** property is set to Continuous.

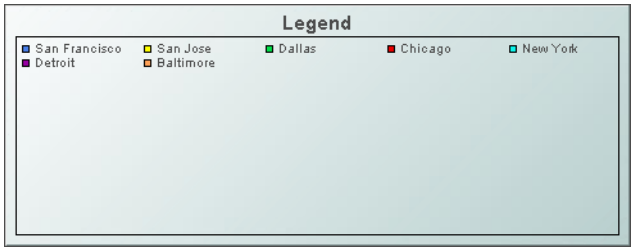
Y-Axis Properties

Specify the way the x and y axes are displayed in the stock chart.

Property Name	Description
tradeDayBegin	Defines the daily start time of the trading day. This property is only used with intraday data (time intervals less than one day: hourly, every 15 minutes, etc.). The default value is **09:30**.
tradeDayEnd	Defines the daily end time of the trading day. This property is only used with intraday data (time intervals less than one day: hourly, every 15 minutes, etc.). The default value is **16:00**.
tradeDayEndLabelFlag	Select to show the last data point of a day and the first data point of the next day (which are equal values) with a unique point on the chart. Otherwise, they are shown together at one point on the chart. This property is only used with intraday data. The default is disabled.
yAxisAutoScaleMode	Select how the y-axis range is calculated from the drop down menu: **Off** - The **yValueMin** and **yValueMax** properties control the range of y-axis. **On** - The **yAxisAutoScaleMode** calculates the y-axis range according to data values being plotted. **On - Include Min/Max** - The **yAxisAutoScaleMode** calculates the y-axis range including **yValueMin** and **yValueMax**.
yAxisFlag	Select to display the y-axis.
yAxisFormat	Select or enter the numeric format of values displayed on the y-axis. To enter a format, use syntax from the **Java DecimalForma**t class.
yAxisLabel	Specify label to display to the left of the y-axis.
yAxisLabelTextHeight	Specify the height of the y-axis labels in pixels.
yAxisMajorDivisions	Specify the number of major divisions on the y-axis.
yAxisMinLabelWidth	Specify the minimum width of the y-axis labels in pixels.
yAxisMinorDivisions	Specify the number of minor divisions on the y-axis.
yAxisMultiRangeFlag	Select to have one axis per trace, with each trace having its own range. The first trace is drawn on the outer left of the graph. The remaining traces are drawn on the inner left of the trace area. Otherwise, all traces are plotted against a single y-axis. The default is enabled.
yAxisPercentFlag	Select to show the percent changed from the first data point instead of values for the y-axis.

Legend

The Legend object (class name: **obj_legend**) is useful for displaying a legend that is too lengthy for the built-in legends of graph objects. The Legend can be used in conjunction with any object on the Graphs tab, excluding the Heatmap, by entering the **objName** property of the graph object in the Legend's **graphName** property. The Legend reflects information from the graph object to which it is connected. All formatting should be setup using the properties of the connected graph object.



Using Interaction Properties

Commands

To assign a command, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Based on your data attachment, substitutions are created that will be passed into drill down displays. Once a drill down target has been set, double-click to activate the drill down. Drill down displays can be activated in the same window that contains the legend or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Background Properties

Specify how the background is displayed in your legend.

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose a color from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **bgColor** property sets the first color in the gradient.

bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (25) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: Rectangle Select to display a background rectangle. 3D Rectangle Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. Round Rectangle Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the legend and the border.

Data Properties

Property Name	Description
graphName	Enter the **objName** property of the graph object to which you want to attach a legend.

Historian Properties

Property Name	Description
historyTableName	Specify name of table in your history database in which to store tabular data. See "Configuring the Historian" for information.
historyTableRowNameFlag	If selected, data from the row name field will be stored in the first column of the table specified in **historyTableName**.

Interaction Properties

Property Name	Description
command	Assign a command to your legend. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab.This property is only applies if **labelTextAlignY** is set to Tab Top.
labelTextAlignX	Select x-axis position of label text from the drop down menu.
labelTextAlignY	Select y-axis position of label text from the drop down menu. **Outside Top** - Position label well above the background rectangle. **Top** - Position label just above the background rectangle. **Title Top** - Position label along the top line of the background rectangle. **Tab Top** - Position label tab just above the background rectangle. Height and width of the tab is dependent on the height and width of the text. Use the labelMinTabWidth property to specify a minimum tab width. **Inside Top** - Position label inside the top of the background rectangle.
labelTextColor	Select the ... button and choose a color from the palette to set the label text color.

labelTextFont	Select font style of label text from the drop down menu.
labelTextHeight	Set the height of the label text in pixels.

Legend Properties

Property Name	Description
legendBgColor	Select the ... button and choose a background color for the legend.
legendBgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **legendBgColor** property sets the first color in the gradient.
legendBgGradientMode	Display a gradient in the legend background. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
legendTextColor	Select the ... button and choose from the palette to set the legend text color.
legendTextFont	Select font style of legend text from the drop down menu.
legendTextHeight	Set the height (in pixels) of the legend text.
legendValueMinSpace	Specify the minimum distance between values and labels in the legend.
legendValueVisFlag	Select to display the numerical values of your data in the legend.
outlineColor	Select the ... button and choose a color from the palette to set the color of the one-pixel outline around the legend area and around each color swatch within the legend. The default value is black (color 7).

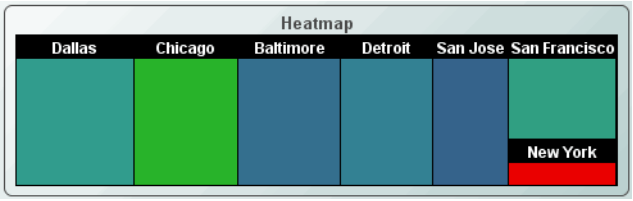
Object Properties

Property Name	Description
anchor	Select the ... button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.

dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Plot x-axis position of object.
objY	Plot y-axis position of object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Heat Maps

Heat maps (class name: **obj_heatmap**) display indexed hierarchical data as a set of nested rectangles. A rectangle exists for each index and, when attached to data, each is filled with smaller rectangles representing sub-indexes, known as nodes. The size and color of a node's rectangle reflects its value, relative to the total of all values for that index. Heat maps are useful because the color and size of the nodes allow you to see patterns that would be difficult to spot in other ways. They also make efficient use of space and therefore can legibly display a large amount of data.



Using Data Properties

To attach data to the heat map, right-click in the Property Value field of the **valueTable** property and select **Attach to Data**. Tabular data attached to the **valueTable** property must contain one or more index columns and at least one data column. The heat map will display one level of nodes for each index column specified. Use the **nodeIndexColumnNames** property to specify column names. The first non-index numeric data column is used to control the size of the node. The second non-index numeric data column is used to control the color of the node. If only one data column is specified, it will control both node size and color.

Data attached to **valueTable** are aggregated by unique index value.

Note: Negative aggregated values are treated as 0. By default, both size and color data is subtotaled. Alternately, you can specify aggregation types using the **colorValueGroupType** and **sizeValueGroupType** properties.

Using Interaction Properties

Commands

To assign a command, right-click in the **Property Value** field of the command property and select **Define Command**. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

Since data in a heat map is aggregated, the value shown in a node might not be the same as the value passed down to a drill down display. For example, suppose your heat map is attached to a table where the index column is Plant and the size column is Units Completed. If you have two rows where the Plant is San Francisco, then the node size is based on the total of the Units Completed values for both rows. However when you drill down, the drill down value for Units Completed will be the value in the first row in the table where the Plant is San Francisco.

To specify a drill down display, double click on the **drillDownTarget** property. Any display (.rtv) file can be targeted as a drill down. Based on your data attachment, substitutions are created that are passed into drill down displays. To customize which substitutions are passed into drill down displays, double-click on **drillDownColumnSubs** in the **Object Properties** window to open the "Drill Down Column Substitutions Dialog". Once a drill down target has been set, double-click to activate the drill down. Drill down displays can be activated in the same window that contains the heat map or open in a separate window; this allows you to build a customizable hierarchy of displays.

Use **drillDownSelectMode** to control how a drill down display is activated. Set to **Anywhere** to activate a drill down display by double-clicking anywhere on the map. Set to **Element Only** to enable a drill down display only when you double-click on a node in map. See "Drill Down Displays" for more information.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your heat map. To display a tool tip, point to a node with your mouse. The tool tip will contain information from your data attachment about that node.

Use the **mouseOverAdditionalColumns** property to select which columns to include in tool tips and, optionally, specify a date format (or other numeric format) and value divisor (for numeric columns) for each column displayed. In the tool tip, the name and value for each selected column is displayed. If the **mouseOverDefaultColumnsFlag** is selected, then columns you include are inserted following the default columns in the tool tip. If specified, **columnDisplayNames** are applied to the columns you selected to include.

Background Properties

Specify how the background is displayed in your heat map.

Property Name	Description
bgBorderColor	Select the ... button and choose from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **bgColor** property sets the first color in the gradient.
bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.

bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the graph and the border.

Data Properties

Property Name	Description
colorValueAutoScaleMode	Nodes are colored according to the value in the color column of the **valueTable** data attachment. The colors range from **minColor** to **maxColor** (around the color wheel) according to their relative value. If **colorValueAutoScaleMode** is **On** the color range will auto-scale according to the values in the data. If **Off**, use **colorValueMin** and **colorValueMax** to control the color range. If **On - Include Min/Max**, include both **colorValueMin** and **colorValueMax** along with the values from the data to control the color range. **Note:** If the **linearColorMappingFlag** is selected, then nodes will display a gradient from **minColor** to **maxColor**.
colorValueDivisor	Divides color value by the number entered. **Note:** If **colorValueDivisor** is specified, it will also be applied to the **colorValueMin** and **colorValueMax** properties.
colorValueGroupType	Select the group type to use for color data. Valid choices are: **sum**, **average**, **count**, **min**, and **max**.
colorValueMax	Set the maximum value for the color data. This value is not applicable if **colorValueAutoScaleMode** is **On**.
colorValueMin	Set the minimum value for the color data. This value is not applicable if **colorValueAutoScaleMode** is **On**.
nodeIndexColumnNames	Specify a semicolon (;) delimited list of index column names. If not specified, the first text column in the table attached to **valueTable** is used as the index column and the first two numeric columns are used as data columns.
sizeValueDivisor	Divides color value by the size entered.
sizeValueGroupType	Select the group type to use for size data. Valid choices are: **sum**, **average**, **count**, **min**, and **max**.
valueTable	Tabular data attached to the **valueTable** property must contain one or more index columns and at least one data column. The heat map will display one level of nodes for each index column specified. Use the **nodeIndexColumnNames** property to specify column names. The first non-index numeric data column is used to control the size of the node. The second non-index numeric data column is used to control the color of the node. If only one data column is specified, it will control both node size and the color. Data attached to **valueTable** are aggregated by unique index value. **Note:** Negative aggregated values are treated as 0. By default, both size and color data is subtotaled. Alternately, you can specify aggregation types using the **colorValueGroupType** and **sizeValueGroupType** properties.

Data Format Properties

Specify the data format for your heat map.

Property Name	Description
colorValueFormat	Select or enter the numeric format of the color value displayed in tool tips. To enter a format, use syntax from the **Java DecimalFormat** class. To enable tool tips, select the **mouseOverFlag**.
linearColorMappingFlag	If selected, the color of nodes will display a gradient from **minColor** to **maxColor**. If deselected, the color of nodes will range from **minColor** and **maxColor** around the color wheel.
maxColor	Select the ... button and choose from the palette to set the maximum color value. Node colors will range from the **minColor** to the **maxColor** according to their color values. **Note:** Use the **linearColorMappingFlag** to display a gradient from **minColor** to **maxColor**.
minColor	Select the ... button and choose from the palette to set the minimum color value. Node colors will range from the **minColor** to the **maxColor** according to their color values. **Note:** Use the **linearColorMappingFlag** to display a gradient from **minColor** to **maxColor**.
sizeValueFormat	Select or enter the numeric format of the size value displayed in tool tips. To enter a format, use syntax from the **Java DecimalFormat** class. To enable tool tips, select the **mouseOverFlag**.

Data Label Properties

Property Name	Description
columnDisplayNames	Set alternate display names for column names in your heatmap data. Column names are displayed in tool tips.

Historian Properties

Property Name	Description
historyTableName	Specify name of table in your history database in which to store tabular data. See "Configuring the Historian"for information.
historyTableRowNameFlag	Select to store data from the row name field in the first column of the table specified in **historyTableName**.

Interaction Properties

Property Name	Description
command	Assign a command to your heat map. See "Define/Execute Command" for information.

commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property is used.
drillDownColumnSubs	Select the ... button to open the "Drill Down Column Substitutions Dialog" to customize which substitutions are passed into drill down displays.
drillDownSelectMode	Control how a drill down display is activated. Select one of the following options: **Anywhere** - Activate a drill down display by double-clicking anywhere on the map. **Element Only** - Enable a drill down display only when you double-click on a node in the map.
drillDownTarget	Name of display (.rtv) file targeted as a drill down. See "Drill Down Displays" for information. **Note:** Heat maps containing large data sets may run slowly on the Display Server if a **drillDownTarget** is specified.
menuItemGroup	Use the **menuItemGroup** property to extend RTView context menu items. For details, see "Extending the Context Menu".
mouseOverAdditionalColumns	Select the ... button to open a dialog to select which columns to include in tool tips and, optionally, specify a date format (or other numeric format) and value divisor (for numeric columns) for each column displayed. In the tool tip, the name and value for each selected column is displayed. If the **mouseOverDefaultColumnsFlag** is selected, then columns you include are inserted following the default columns in the tool tip. If specified, **columnDisplayNames** are applied to the columns you selected to include.
mouseOverDefaultColumnsFlag	Select to include column names and values from **valueTable** (for index column(s) and data columns) in tool tips. If **columnDisplayNames** are specified, they will be applied to all column names.
mouseOverFlag	Select to enable tool tips for your heat map. To display a tool tip, point to a node with your mouse. The tool tip will contain information from your data attachment about that node. **Note:** Heat maps containing large data sets may run slowly on the Display Server if **mouseOverFlag** is selected.
rightClickActionFlag	Use the **rightClickActionFlag** property to extend RTView context menu items. For details, see "Extending the Context Menu".

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. This property is only applies if **labelTextAlignY** is set to **Tab Top**.
labelTextAlignX	Select x-axis position of label text from the drop down menu.
labelTextAlignY	Select y-axis position of label text from the drop down menu. **Outside Top** - Position label well above the background rectangle. **Top** - Position label just above the background rectangle. **Title Top** - Position label along the top line of the background rectangle. **Tab Top** - Position label tab just above the background rectangle. Height and width of the tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. **Inside Top** - Position label inside the top of the background rectangle.
labelTextColor	Select the ... button and choose from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop down menu.
labelTextHeight	Set the height of the label text in pixels.

Layout Properties

Specify how the background is displayed in your heat map.

Property Name	Description
adjustSizeForLabelFlag	Select to compress the ratio between the smaller nodes and larger nodes so that the size of smaller nodes is increased to accommodate labels. This property only applies to nodes that display labels.
layoutStyle	Select from the following layout styles: **Squarified** - Nodes are more square in shape and ordered according to the size of the value from the top-left to the bottom-right. **Strip** - Nodes are more square in shape and ordered according to the order of the rows in the **valueTable**. **Slice Horizontal** - Nodes are short and wide and ordered according to the order of the rows in the **valueTable**. **Slice Vertical** - Nodes are tall and narrow and ordered according to the order of the rows in the **valueTable**. **Slice Best** - Nodes are laid out either like Slice Horizontal or Slice Vertical based on what fits best in the available space. **Slice Alternate Horizontal** - The layout alternates between Slice Horizontal and Slice Vertical based on the node depth. The top level nodes use Slice Horizontal. **Slice Alternate Vertical** - The layout alternates between Slice Horizontal and Slice Vertical based on the node depth. The top level nodes use Slice Vertical.

Node Properties

Property Name	Description
nodeBgBorderHighlightMode	Select from the following options: **None** - No highlight. **Thin** - Single pixel highlight. **Thick** - Two pixel thick highlight with the inner pixel a shade lighter than the outer pixel. **Note:** This property is ignored if **nodeBgBorderSize** is set to 0 or 1.
nodeBgBorderNestDepth	Specify the number of nest levels to display node borders. If set to 0, then no borders are displayed. If set to -1, then borders are displayed on all levels.
nodeBgBorderSize	Specify (in pixels) the size of the border between nodes. If set to -1, the deepest nested level of nodes has a one pixel border and the border increases by two pixels for each level of nesting.
nodeBgColor	Select the ... button and choose from the palette to set the background color for the nodes.
nodeLabelNestDepth	Specify the number of nest levels to display node labels. If set to 0, then no labels are displayed.
nodeLabelTextColor	Select the ... button and choose from the palette to set the text color for the node labels.
nodeLabelTextFont	Select the font to use for the node labels.
nodeLabelTextHeight	Specify the text height for the node labels.
nodeLabelVisFlag	Select to display labels on the nodes. **Note:** This property is ignored if the **nodeLabelNestDepth** is set to 0.
nodeLabelVisIfEmtpyFlag	Select to display the background for empty node labels.

Object Properties

Property Name	Description
anchor	Select the ... button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.

dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Sets the transparency of the object. This property only applies to the background of the composite object.
visFlag	Set the visibility of the object.

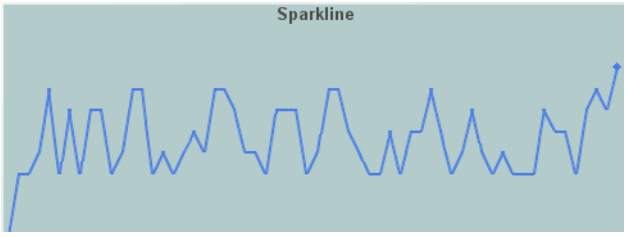
Quality Properties

Property Name	Description
valueQuality	Specify a value to compare to settings for the **valueQualityLostData** and **valueQualityNoData** properties. If the specified **valueQuality** matches, the selected corresponding **valueDataQuality*Color** will be applied to all nodes in the heat map. **Note:** The **valueQuality** property is ignored if the **valueQualityEnabledFlag** is deselected.
valueQualityColumnName	Specify a column in the **valueTable** to compare, per row, to settings for the **valueQualityLostData** and **valueQualityNoData** properties. If values in the specified **valueQualityColumnName** matches, the selected corresponding **valueQuality*Color** will be selectively applied to each node in the heat map. If the **valueTable** contains multiple rows for a single index, the highest data quality value is used. **Note:** The **valueQualityColumnName** property is ignored if the **valueQualityEnabledFlag** is deselected.
valueQualityEnabledFlag	If selected, nodes are colored based on data quality.
valueQualityLostData	Enter the lost data value.

valueQualityLostDataColor	Select the ... button and choose from the palette to set the node color if the value matches the specified **valueQualityLostData**.
valueQualityNoData	Enter the no data value.
valueQualityNoDataColor	Select the ... button and choose from the palette to set the node color if the value matches the specified **valueQualityNoData**.

Sparkline

Sparkline charts (class name: obj_sparkline) are generally used to present trends and variations in a simple and condensed way. As the name implies there is a line associated with data, but no background or axis. It is possible to add labels at the beginning and ending points of the line, which then can be toggled on and off.



Using Data Properties

To attach data to your sparkline chart, right-click in the **Property Value** field of the value or **valueTable** property and select **Attach to Data**. Attach scalar data to the value property and tabular data to the **valueTable** property. Tabular data attached to the **valueTable** property should have two columns: the first must contain numeric values or time stamps (x-axis values) and the second column should contain the corresponding (y-axis) numeric values.

Alert Properties

To set marker colors and styles based on a threshold value, select the corresponding value alarm flags.

Property Name	Description	
valueHighAlarmEnabledFlag	Select to enable the high alarm threshold and the following related properties:	
	valueHighAlarm	Set the value of the high alarm threshold.
	valueHighAlarmLineVisFlag	Select to display a dotted line at the high alarm threshold. The color of the line is set to the **valueHighAlarmMarkColor**.
	valueHighAlarmMarkColor/ valueHighAlarmMarkStyle	When a trace marker's value is greater than or equal to the **valueHighAlarm** property, the marker will change to the **valueHighAlarmMarkColor** and **valueHighAlarmMarkStyle**.

	valueHighAlarmTraceColor/ valueHighAlarmTraceStyle	When the value of any segment of a trace line is greater than or equal to the **valueHighAlarm** property, that segment of the trace line will change to the **valueHighAlarmTraceColor** and **valueHighAlarmTraceStyle**. **Note:** If **valueHighAlarmTraceStyle** is set to No Line, then **valueHighAlarmTraceColor** will not change.
valueLowAlarmEnabledFlag	Select to enable the low alarm threshold and the following related properties:	
	valueLowAlarm	Set the value of the low alarm threshold.
	valueLowAlarmLineVisFlag	Select to display a dotted line at the low alarm threshold. The color of the line is set to the **valueLowAlarmMarkColor**.
	valueLowAlarmMarkColor/ valueLowAlarmMarkStyle	When the trace marker's value is less than or equal to the **valueLowAlarm** property, the marker will change to the **valueLowAlarmMarkColor** and **valueLowAlarmMarkStyle**.
	valueLowAlarmTraceColor/ valueLowAlarmTraceStyle	When the value of any segment of a trace line is less than or equal to the **valueLowAlarm** property, that segment of the trace line will change to the **valueLowAlarmTraceColor** and **valueLowAlarmTraceStyle**.

Background Properties

Specify how the background is displayed in your chart.

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose a color from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is 3D Rectangle.
bgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **bgColor** property sets the first color in the gradient.

bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (25) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the chart and the border.

Data Properties

Specify how data is displayed in your chart.

Property Name	Description
maxPointsPerTrace	The default is set to **1000**. The maximum value for this property is **30000**.
value	Attach your scalar data to the value property.
valueDivisor	Divides y-axis values by the number entered.

valueTable	Attach your tabular data to the **valueTable** property. Tabular data attached must have two columns: the first must contain numeric values or time stamps (x-axis values) and the second column should contain the corresponding (y-axis) numeric values.
yValueMax yValueMin	Controls the range of y-axis if the **yAxisAutoScaleMode** is set to **Off**. Select On for the **yAxisAutoScaleMode** to calculate the y-axis range according to data values being plotted. To calculate the y-axis range including **yValueMin** and **yValueMax**, select **On - Include Min/Max**.

Data Format Properties

Specify data format in your chart.

Property Name	Description
yValueFormat	Select or enter the numeric format of values displayed in the legend and popup legend. To enter a format, use syntax from the **Java DecimalFormat** class.

Historian Properties

Property Name	Description
historyTableName	Specify name of table in your history database in which to store tabular data. See "Configuring the Historian" for information.
historyTableRowNameFlag	If selected, data from the row name field will be stored in the first column of the table specified in **historyTableName**.

Interaction Properties

Property Name	Description
command	Assign a command to your chart. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.

commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
cursorColor	Select the ... button and choose a color from the palette to set the color of the cursor. **Note:** This property is only available if **cursorFlag** is selected.
cursorFlag	Select to enable the cursor. When the cursor is enabled, point to a location on a trace to see a cursor line at that location and display the time and values of all traces at the cursor line on the legend. Hold down the control key to snap the cursor to the closest data point. Select the **legendPopupFlag** to display the legend along the cursor.
drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information.
legendPopupFlag	Select to display the legend along the cursor. **Note:** This property is only available if **cursorFlag** is selected.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. This property is only applies if **labelTextAlignY** is set to **Tab Top**.
labelTextAlignX	Select x-axis position of label text from the drop down menu.
labelTextAlignY	Select y-axis position of label text from the drop down menu. **Outside Top** - Position label well above the background rectangle. **Top** - Position label just above the background rectangle. **Title Top** - Position label along the top line of the background rectangle. **Tab Top** - Position label tab just above the background rectangle. Height and width of the tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. **Inside Top** - Position label inside the top of the background rectangle.
labelTextColor	Select the ... button and choose a color from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop down menu.
labelTextHeight	Set the height of the label text in pixels.

Legend Properties

Specify the way the legend is displayed in your chart.

Property Name	Description
legendBgColor	Select the ... button and choose a color from the palette to set the background color of the legend.
legendBgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **legendBgColor** property sets the first color in the gradient.

legendBgGradientMode	Display a gradient in the legend background. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
legendTimeFormat	Set the format for the time displayed in the legend using syntax from the **Java SimpleDateFormat** class. For example, **MMMM dd, yyyy hh:mm:ss** a would result in the form August 30, 2003 05:32:12 PM. If no format is given, the **timeFormat** will be used.
legendValueMinSpace	Specify the minimum distance between values and labels in the legend.
legendVisFlag	Select to display the legend.
legendWidthPercent	Set the percent of the total width of the object used for the legend.

Marker Properties

Specify the way markers are displayed in your chart.

Property Name	Description
markDefaultSize	Set **markDefaultSize** to specify the size of the markers in pixels.

Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.

dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Plot x-axis position of object.
objY	Plot y-axis position of object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Plot Area Properties

Specify the way the plot area is displayed in your chart.

Property Name	Description
traceBgColor	Select the ... button and choose a color from the palette to set the background color of your chart.

Trace Properties

Specify how traces are displayed in your chart.

Property Name	Description
traceLabel	Set a label for your trace.
traceLineColor	Select the ... button and choose a color from the palette to set the trace line color.
traceLineStyle	Select the style of the line used to display the trace from the drop down menu: **No Line**, **Solid**, **Dotted**, **Dashed**, or **Dot Dashed**.

traceLineThickness	Select the thickness of the line used to display the trace from the drop down menu: **Thin**, **Medium**, or **Thick**.
traceMarkColor	Select the  button and choose a color from the palette to set the trace marker color.
traceMarkStyle	Select the style of the marker used on the trace from the drop down menu: **No Marker**, **Dot**, **+**, *****, **o**, **x**, **Filled Circle**, **Filled Diamond**, **Filled Triangle**, **Filled Square**, or **Filled Star**.
valueHistoryFlag	If you enable this property, RTView will attempt to load initial data from the Historian database for the corresponding trace. **Note:** Only data within the specified **timeRange** will be loaded and the chart will update as live data becomes available.

X-Axis Properties

Specify the way the x axis is displayed in your chart.

Property Name	Description
timeRange	Control the total amount of time, in seconds, plotted on the chart. If you attach data to **trace*ValueTable**, set **timeRange** to **-1** so the time range of the chart is driven by the time range in the data attachment. **Note: timeRange** is ignored if both **timeRangeBegin** and **timeRangeEnd** are set.
timeRangeBegin	Set the start time value of the data to be plotted on the chart. Supported formats are: **mm/dd/yyyy hh:mm:ss** (e.g., 01/16/2004 12:30:03), **yyyy-mm-dd hh:mm:ss** (e.g., 2004-01-16 12:30:03), and the number of milliseconds since midnight, January 1, 1970 UTC. **Note:** If only the time is specified, then today's date will be used.
timeRangeEnd	Set the end time value of the data to be plotted on the chart. Supported formats are: **mm/dd/yyyy hh:mm:ss** (e.g., 01/16/2004 12:30:03), **yyyy-mm-dd hh:mm:ss** (e.g., 2004-01-16 12:30:03), and the number of milliseconds since midnight, January 1, 1970 UTC. **Note:** If only the time is specified, then today's date will be used.
timeRangeOfHistory	Specify how much historical data is loaded into the chart, in seconds. If **timeRangeOfHistory** is set to zero or less (default is **-1**), or if it is less than the value of **timeRange**, then the **timeRange** property determines the amount of historical data to be loaded into the chart.

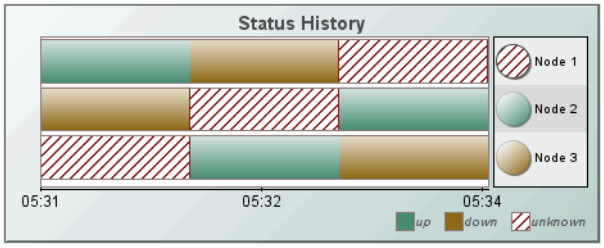
Y-Axis Properties

Specify the way the y axis is displayed in your chart.

Property Name	Description
yAxisAutoScaleMode	The **yValueMin** and **yValueMax** properties control the range of the y-axis for this trace if the **yAxisAutoScaleMode** is set to Off. Select On for the **yAxisAutoScaleMode** to calculate y-axis range according to data values being plotted. To calculate the y-axis range including **yValueMin** and **yValueMax**, select **On - Include Min/Max**. If **yAxisMultiRangeMode** is set to **Multiple Axis** or **Strip Chart**, then use the **trace*YAxisAutoScaleMode**, **trace*YAxisValueMin**, and **trace*YAxisValueMax** properties to control the range of the y-axis.

Status History

Status History charts (class name: **obj_statushistory**) show discrete status values as horizontal bars drawn against a horizontal time axis. A label appears at the right edge of each bar along with a circular indicator displaying that bar's most recent status. If there are more horizontal bars of data than can be drawn in the allocated area, vertical scroll bars will appear to the right. If there is more data than can be displayed in the current time frame, a horizontal scroll bar will appear at the bottom of the chart.



Using Data Properties

To attach tabular data to your chart, right-click in the **Property Value** field of the **valueCurrentTable** property or the **valueHistoryTable** property and select **Attach to Data**.

Historical Data	Current Data
To display historical data, attach to **valueHistoryTable**.	To display current data, attach to **valueCurrentTable**.
The attached historical data table should contain a **timestamp** column, a status (value) column and one or more index columns. The **timestamp** column should be the first column in the data table.	The attached data table should contain at least two columns: a status (value) column and one or more index columns. If it does not contain a **timestamp** column, then the current time will be assumed for each row in the table.
A horizontal bar is displayed in the chart for each unique combination of index column values. The value of the status column is used to determine the color and fill pattern of the bar segment at the corresponding time interval.	A horizontal bar is displayed for each unique combination of index column values. The value of the status column is used to determine the color and fill pattern of the bar segment at the corresponding time interval.
When data is attached to **valueHistoryTable**, any previously plotted data is removed and the entire chart is replotted.	Data attached to **valueCurrentTable** is assumed to be the most recent data and other previously plotted data is appended to it.
Typically valueHistoryTable is attached to the history table of a Cache object. See "Caches" for more information.	Typically **valueCurrentTable** is attached to the current table of a Cache object. See "Caches" for more information.

For example, consider a data table that holds time-based status information for two nodes on a network. It has three columns: **time_stamp**, **node_name**, and **status**. The **time_stamp** column contains the **timestamp** for the row. The **node_name** string is unique for each node so it is used as the index column for the table. The status column contains either up, down, or unknown.

Here is a portion of the contents of that data table:

```
12:00, Node01, up
12:00, Node02, up
12:01, Node01, up
12:01, Node02, down
12:02, Node01, up
```

```
12:02, Node02, up
```

This table could be attached to the **valueHistoryTable** property. The **barProperties** property would be configured to assign a color and fill pattern that corresponds to the status column (e.g. **up** = solid green and **down** = crosshatch red. The chart would display two horizontal bars, one for Node01 and another for Node02, and each bar would contain a segment for each time interval. For example if the chart was observed at 12:03, each bar would contain two segments: one for the 12:00 - 12:01 interval and another for the 12:01 - 12:02 interval. The color and fill pattern of the segment would correspond to the status at the start of that interval, as determined by the **barProperties** settings. Note that no segment would be displayed for the interval starting at 12:02, since there is no end point for that interval in the data table. Instead, the status value for the last data value is always used to control the color and fill pattern of the circular indicator that appears at the right end of each bar.

Note: : Supported formats for the **timestamp** column are: **mm/dd/yyyy hh:mm:ss** (e.g., 01/16/2004 12:30:03), **yyyy-mm-dd hh:mm:ss** (e.g., 2004-01-16 12:30:03), and the number of milliseconds since midnight, January 1, 1970 UTC. In order to view all available data, you must set the properties **timeRange** to **-1** and **timeShift** to a negative value. This negative value will be used to round the start and end times for the y-axis. For example, if you specify -15 for the **timeShift** property, the start and end times for the y-axis will be rounded to the nearest 15 seconds.

Using Interaction Properties

Commands

To assign a command to your chart, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Based on your data attachment, substitutions are created that will be passed into drill down displays. To customize which substitutions will be passed into drill down displays, double-click on **drillDownColumnSubs** in the **Object Properties** window to open the "Drill Down Column Substitutions Dialog". Once a drill down target has been set, double-click on a bar segment or a bar label in the chart to activate the drill down. If a bar segment is clicked, substitutions will be set to match the data row that corresponds to the left edge of the segment. If a bar label is clicked, the substitutions will be set to match the most recent data row for that bar.

Drill down displays can be activated in the same window that contains the chart or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Right-Click Options

Select the **rightClickActionFlag** property so that a right-click by the user executes actions (a drill down or execute command) normally performed only with a left-click, before the right-click popup menu is shown. This is useful when you have a command string configured to set substitutions in the current window, and the drill down target is configured to drill down using those substitutions. In that case, the command is executed when you left-click and when you right-click so the substitutions are guaranteed to be set before you select **Drill Down** from the popup menu. This option should not be used if the left click is configured to drill down to another display.

Note: This property is ignored in the Thin Client on iOS Safari (iPad/iPhone).

Use the **menuItemGroup** and **rightClickActionFlag** properties to extend RTView context menu items. For details, see "Extending the Context Menu".

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your chart. To display a tool tip, point to a bar segment or bar label with your mouse. When you point to a bar segment, the tool tip will show information about the data row that corresponds to the left edge of the segment. When you point to a bar label, the tool tip will show information about the most recent data row for that bar.

Note: Use the **descriptionColumnName** property to identify a column in the attached **value*Table** that contains additional tool tip information for each row.

Background Properties

Specify how the background is displayed in your chart.

Property Name	Description
bgBorderColor	Select the ... button and choose from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **bgColor** property sets the first color in the gradient.

bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is 3D Rectangle.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (25) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle. **bgStyle** - Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the chart and the border.

Bar Properties

Specify the way the bars are displayed in your chart.

Property Name	Description
barCellBorderFlag	If selected, vertical lines are drawn between each segment of each bar corresponding to the location of data points that mark the start and end of each segment. Depending on whether data is updated at regular intervals, these lines may or may not be evenly spaced. **Note:** Use the **gridColor** property to set the color of these lines.

barGradientStyle	Set the type of gradient to display when filling a bar. Select from the following options: **Shaded** - Display bars with a flat gradient **Rounded** - Display bars with a rounded gradient **None** - No gradient displayed. **Note:** The **barGradientStyle** property is ignored for bars with Fill Patterns patterns specified using **barProperties**.
barHeightFitFlag	If selected, each bar will be stretched vertically to fill additional space if the chart's plot area is taller than necessary to accommodate minimum height of each bar as determined by the larger of two properties: **barHeightMin** and **barLabelTextHeight**.
barHeightMin	Set the height of the bar in pixels. **Note:** Minimum height of each bar is determined by the larger of two properties: **barHeightMin** and **barLabelTextHeight**. If the chart's plot area is not tall enough to display each bar at its minimum height, then a vertical scrollbar will appear to show remaining bars. If the chart's plot area is taller than needed and if the **barFitFlag** property is checked, then each bar will be stretched vertically to fill the additional space.
barHeightSpacing	Set the vertical space, in pixels, between the bottom of each bar and the top of the bar below it. **Note:** The value must be an even number, greater than or equal to 2.
barLabelBgColor	Select the ... button and choose from the palette to set the background color of the label for the top (first) bar and every other bar below it.
barLabelBgColor2	Select the ... button and choose from the palette to set the background color of the label for the second bar from the top and every other bar below it.
barLabelMaxLength	Maximum length, in pixels, for each bar label. If the label is longer than the specified length it will be clipped. Enter a value of 0 to indicate no maximum length. **Note:** If a value is specified for both **barLabelMaxLength** and **barLabelMaxLengthPct**, then the smaller of the two values will be used.
barLabelMaxLengthPct	Maximum length for each bar label, specified as a percentage of the plot area width. If the label is longer than the specified length it will be clipped. Default is **0** to indicate no maximum length. **Note:** If a value is specified for both **barLabelMaxLength** and **barLabelMaxLengthPct**, then the smaller of the two values will be used.
barLabelTextColor	Select the ... button and choose from the palette to set the bar label text color.
barLabelTextFont	Select font style of bar label text from the drop down menu.
barLabelTextHeight	Set the height of the bar label text in pixels. **Note:** Minimum height of each bar is determined by the larger of two properties: **barLabelTextHeight** and **barHeightMin**. If the chart's plot area is not tall enough to display each bar at its minimum height, then a vertical scrollbar will appear to show remaining bars. If the chart's plot area is taller than needed and if the **barFitFlag** property is checked, then each bar will be stretched vertically to fill the additional space.
barOutlineFlag	If selected, each bar will be outlined in the specified **gridColor**.
barProperties	Double-click on **barProperties** to specify the color and fill pattern to correspond with each value in the chart. See "Bar Properties - Status History" for more information.

Data Properties

Property Name	Description
descriptionColumnName	Name of the column in the attached **value*Table** that contains additional tool tip information for each row. Within the specified column, use \n to denote a new line. **Note:** The **mouseOverFlag** must be selected to enable tool tips.
indexColumnNames	Enter a semicolon (;) separated list of column names from the attached **value*Table** to be used as index columns. This list will be used to uniquely identify which rows should be plotted on the same horizontal bar. If left blank, the second column in your data attachment will be used.
labelColumnName	Name of the column in the attached **value*Table** that contains labels for each bar. If left blank, the first non-numeric text column in your data attachment will be used.
maxNumberofRows	Maximum number of rows (historical plus current data) that the chart will maintain. Older rows will be discarded as necessary, which may cause bar segments to be removed from the left edge of the chart.
timestampColumnName	Enter name of the column in the attached **value*Table** that contains the **timestamp** for each row. If left blank, RTView will look for a column named **time_stamp** in your data attachment. If no column named **time_stamp** is found and the first column in your data attachment contains **timestamp** data then it will be used. **Note:** If no **timestamp** column can be located, then the current time will be used.
valueColumnName	Enter name of the column in the attached **value*Table** containing the status value used to determine the color and fill pattern of each bar segment. This column can contain a string, number or boolean value. If left is blank, the third column (if any) in the your data attachment will be used. **Note:** This column will be compared to the **barProperties** to find the corresponding color and fill pattern. See "Bar Properties - Status History" for more information.
valueCurrentTable	Right-click in the **Property Value** field and select **Attach to Data**. The attached data table should contain a status (value) column and one or more index columns. Refer to the "Using Data Properties" section for more information. Typically, **valueCurrentTable** will be attached to the current table of a Cache object. **Note:** By default, the label column is the first non-numeric text column in your data attachment. Enter a column name in the **labelColumnName** property to set a specific label column.
valueHistoryTable	Attach historical data table containing a **timestamp** column, one or more index columns, and a status (value) column. Refer to the "Using Data Properties" section for more information.
valueQualityBadValuesList	Sets a color and pattern for plotting when data quality is bad. Use this property in conjunction with **valueQualityColumnName**. The **valueQualityBadValuesList** property can be set to a string containing value,label pairs with each pair separated by semicolons. This is used to assign a label to a numeric bad quality value, for example "**-1,no data;0,stale data**". The default value for **valueQualityBadValuesList** is blank, which means that the feature is disabled.

valueQualityColumnName

Sets a color and pattern for plotting when data quality is bad. Use this property in conjunction with **valueQualityBadValuesList**.

The **valueQualityColumnName** property can be set to the name of the table column that contains a value indicating the data quality for each row. The column can contain string, integer, or boolean values.

The default value for **valueQualityColumnName** is blank, which means that the feature is disabled. In this mode, the color and pattern for each bar is determined by getting the value for that row from the column specified by **valueColumnName** and looking up that value in the chart's **barProperties**. (This is the behavior in all Core releases prior to 6.6).

If **valueQualityColumnName** is not blank, then for each row (**R**), the row's value to be plotted is determined as follows:

- let **Q** = **value** of quality column for row **R**, and **V** = **value** of value column for row **R**
- if **Q** is blank, then the quality is considered good and **V** is used at the row value, as normal.
- else if the **valueQualityBadValuesList** property is blank, then **Q** is used as the row value.
- else if **valueQualityBadValuesList** contains an entry "**Q**,**X**" then **X** is used as the row value.
- else the quality is considered good (since no match was found in **valueQualityBadValuesList**) and V is used at the row value.

Then the row value is looked up in **barProperties**, as usual, to determine the color and fill pattern. That value is also shown in the mouseover text.

For example, consider the following data table:

Plant Status Quality

----- ------ -------

A online 1

B offline 1

C online -1

D offline 0

... where **1** = **data OK**, **-1** = **no data**, and **0** = **stale data**.

Next, consider a status history chart with these properties:

indexColumnNames = Plant

valueColumnName = Status

valueQualityColumnName = Quality

valueQualityBadValuesList = -1,no Data;0,stale data

barProperties =

online : green

offline : blue

no data : red

stale data : orange

Note that **valueQualityBadValuesList** has no entry for **quality = 1**, because that is the "good" quality value.

With that configuration, when the data table shown above is applied to the chart, it will plot a segment for each Plant as follows:

A green (since Status = online and Quality = 1 / OK)

B blue (since Status = online and Quality = 1 / OK)

C red (since Quality = -1 / no data)

Historian Properties

Property Name	Description
historyTableName	Specify name of table in your history database in which to store tabular data. See **History Tables>Tabular Data** ("Configuring the Historian") for information.
historyTableRowNameFlag	If selected, data from the row name field will be stored in the first column of the table specified in **historyTableName**.

Interaction Properties

Property Name	Description
command	Assign a command to your chart. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownColumnSubs	Select the ... button to open the "Drill Down Column Substitutions Dialog" to customize which substitutions will be passed into drill down displays.
drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information.
menuItemGroup	Use the **menuItemGroup** and **rightClickActionFlag** properties to extend RTView context menu items. For details, see "Extending the Context Menu".

mouseOverFlag	Select to enable tool tips for your chart. To display a tool tip, point to a bar or bar label with your mouse. When you point to a bar, the tool tip will show information about the data row that corresponds to the left edge of the segment. When you point to a bar label, the tool tip will show information about the most recent data row for that bar. **Note:** Use the **descriptionColumnName** property to identify a column in the attached **value*Table** that contains additional tool tip information for each row.
rightClickActionFlag	Select to enable right-click actions (e.g.: drill down or execute command) normally performed with a left-click. For details, see "Using Interaction Properties". **Note:** This option should not be used if the left click is configured to drill down to another display.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. This property is only applies if **labelTextAlignY** is set to **Tab Top**.
labelTextAlignX	Select x-axis position of label text from the drop down menu.
labelTextAlignY	Select y-axis position of label text from the drop down menu. **Outside Top** - Position label well above the background rectangle. **Top** - Position label just above the background rectangle. **Title Top** - Position label along the top line of the background rectangle. **Tab Top** - Position label tab just above the background rectangle. Height and width of the tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. **Inside Top** - Position label inside the top of the background rectangle.
labelTextColor	Select the ... button and choose from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop down menu.
labelTextHeight	Set the height of the label text in pixels.

Legend Properties

Specify the way the legend is displayed in your chart.

Property Name	Description
legendTextFont	Select font style of legend text from the drop down menu.
legendTextHeight	Set the height of the legend text in pixels.
legendVisFlag	Select to display the legend.

Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Plot x-axis position of object.
objY	Plot y-axis position of object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Plot Area Properties

Property Name	Description
gridBgColor	Select the  button and choose from the palette to set the color of the grid background.

gridColor	Select the ... button and choose from the palette to set the color of the grid lines. Default is **white**.
gridFlag	Select to display the grid.
gridStripedFlag	If selected, then the alternating background color of each label (as determined by **barLabelBgColor** and **barLabelBgColor2**) is extended into the plot area background. Otherwise, the plot area background color is determined by **gridBgColor**.

Sort Properties

Property Name	Description
sortAscendingFlag	If selected, bars are sorted alphabetically. Otherwise, bars are sorted in reverse alphabetical order.
sortByLabelsFlag	Deselect to sort bars by a column name from the data attachment. Otherwise, if selected, bars are sorted by label.
sortColumnName	Enter the name of a column in the attached **value*Table**. Bars are sorted by comparing the last (most recent) row from the specified **sortColumnName** for each bar. If left blank, bars are left in their natural (unsorted) order. The natural order of the bars is the order in which the first row for each bar was encountered in the attached **value*Table**. **Note:** If **sortColumnName** if left blank, the **sortAscendingFlag** property has no effect.

X-Axis Properties

Specify how the x-axis is displayed in your chart.

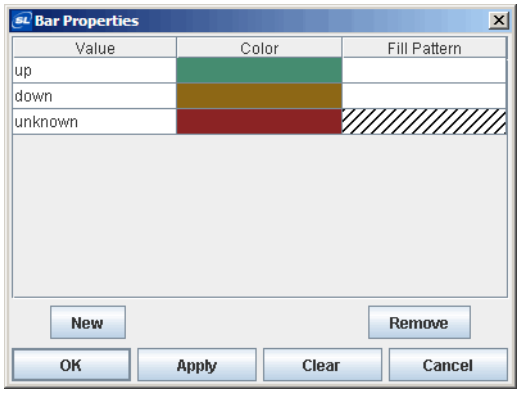
Property Name	Description
timeFormat	Set the format for the time displayed in the x-axis using syntax from the **Java SimpleDateFormat** class. For example, **MMMM dd, yyyy hh:mm:ss** a would result in the form August 30, 2003 05:32:12 PM. If no format is given, the date and time will not be displayed on the x-axis. Include a new line character ('\n') to display multiple line text in the time axis labels. For example, **MM\dd'\n'hh:mm:ss** would result in the form 08\30 05:32:12.
timeRange	Control the total amount of time, in seconds, plotted on the graph. If you attach data to **valueCurrentTable**, set **timeRange** to **-1** so the time range of the chart is driven by the time range in the data attachment.
timeShift	Control the amount of time, in seconds, the graph will shift to the left when the trace has filled the graph and controls the rounding of the start and end times. For example, if the **timeShift** is 15, the start and end times on the graph will be rounded to the nearest 15 second interval. By default, the end of the plot area corresponds with the current time. To only shift the graph when new data is received, set **timeShift** to a negative value and the end of the graph will display the most current data plotted. If you attach data to **currentValueTable**, you must set **timeShift** to a negative value.
xAxisColor	Select the ... button and choose from the palette to set the color of the x-axis. Default is **black**.
xAxisFlag	Select to display the x-axis.
xAxisRotationAngle	Angle by which the x-axis label text will be rotated.

xAxisLabelTextFont	Select font style of x-axis label text from the drop down menu.
xAxisLabelTextHeight	Set the height of the x-axis label text in pixels.
xAxisMajorDivisions	Specify the number of major divisions on the x-axis.

Bar Properties - Status History

In the **Object Properties** window, double-click on **barProperties** in the **Property Name** field to bring up the **Bar Properties** dialog to assign a Color and Fill Pattern to each value in your status history chart.

Note: Before assigning attributes to values in your status history chart, it is recommended that you first attach the chart to data.



Field Name	Description
Value	There is one entry for each value that is currently displaying data in the status history chart. The **Color** and **Fill Pattern** columns list the current settings for each value.
Color	Select the button in the **Color** column and choose from the palette to set the corresponding color of the value. Close the **Color Chooser** window.
Fill Pattern	Select the button in the **Fill Pattern** column and choose from the palette to set the corresponding fill pattern of the value. Close the **Fill Pattern** window.
New	Click to add a new Value. A value of * can be entered to specify a default **Color** and **Fill Pattern** for any rows in your data attachment that don't have a matching value. **Note:** This does not add a value to your data attachment, it allows you to set properties for values that will display data that is not yet available. This is useful if the data attachment is not available when you setup your chart or if values returned by your data attachment vary.
Remove	Removes selected Value entry from the **Bar Properties** dialog.

The following describes the **Bar Properties** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Clear	Clears all fields. Detaches object from data source (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied.

Indicator Objects

The Object Palette features four types of indicators:

"Indicator Objects - Discrete"
Supports 3 discrete comparisons.

"Indicator Objects - Limits"
Supports 2 high and 2 low thresholds.

"Indicator Objects - Multi"
Supports an unlimited number of comparison values. For each comparison, you can specify whether the value property must be equal, not equal, greater than or less than the comparison value.

"Indicator Objects - Panel"
Panel of 2 or 3 indicator lights. Each indicator light supports a discrete comparison.

Image Property

Labels objects that feature the **image** property can be customized to display your image (.**gif**, .**jpg**, or .**png**) file. Click in the **image Property Value** field and type the name of the image. Or, select the ... button to open the **Select Image** dialog containing up to three directories:

- **Current Directory** - Contains images in the current directory and one level of subdirectories.
- **Custom Image Library** - If you have specified a custom image library, this directory contains those images. See Creating a Custom Image Library for details.
- **Symbol Library** - Contains symbolic images (for example, symbols for various types of hardware, shapes, lights, arrows, etc.).

Navigate to the image you want to use and select it. A preview of the image appears in the pane to the right. Click **OK** or **Apply** to set the image on your object. If an image is not listed, enter the name of the file, including the relative path.

To scale your image to the size of the object, check the **imageScaleFlag**. The **visFlag** property controls the visibility of the object. The **transparencyPercent** property controls the transparency of the object.

Note: The sample display file **general_objects.rtv** (located in **demos/tutorials**) features information on working with objects from the General tab.

Creating a Custom Image Library

The custom image library enables you to make your own images available in the Select Image dialog. To add your own image library, perform the following steps.

1. Place your images **.jar** file and add it to the RTV_USERPATH environment variable. The images must be in a directory (not in the top level of the jar). They can be organized into subdirectories of one top level directory.

2. In the Display Builder, select **Tools/Builder Options** and, in the **Custom Image Library Path** field, set the path to the directory containing your images .jar file.

For example, if you have a jar with this directory structure:

com/mycompany/Images

com/mycompany/Images/Blue Images

com/mycompany/Images/Red Images

com/mycompany/Images/Green Images

With this directory structure, you would enter **com/mycompany/Images**. This adds a directory named **Images** to the tree in the Select Image dialog. The **Images** directory will have three subdirectories: **Blue Images**, **Red Images**, and **Green Images**. Only directories containing images are added to the Select Image dialog.

To access the images, you can edit any property that allows you to set an image on an object (for example, the **image**, **barImage**, and **filterProperties** properties), or edit the File>Background Properties>Image Name field.

Indicator Objects - Discrete

The Object Palette features three types of discrete indicators:

obj_ind_discrete	This is an indicator that allows you to set up to three comparison values (**valueLowAlert**, **valueMediumAlert**, and **valueHighAlert**). When the value equals one of the enabled comparisons, the color of the background and the background image will change to the color and image specified for that comparison.
2 Alert State Indicator	This is **obj_ind_discrete** configured to indicate two discrete alert states: **valueHighAlert = 2** and **valueLowAlert = 1**. The **valueQualityEnabledFlag** is selected with **valueQualityNoData = 1** and **valueQualityLostData = 2**. This object uses the image property instead of background colors. The label is positioned to the right of the indicator.
Boolean Indicator	This is **obj_ind_discrete** configured to show one discrete alert state: **valueHighAlert = 0**. The **valueQualityEnabledFlag** is selected with **valueQualityNoData = 1** and **valueQualityLostData = 2**. This object uses the image property instead of background colors. The label is shown to the right of the indicator.

Using Interaction Properties

Commands

To assign a command to your indicator, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Once a drill down target has been set, double-click on the indicator to activate the drill down. Drill down displays can be activated in the same window that contains the indicator or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your indicator. To display a tool tip, point to the indicator with your mouse. The tool tip will contain information from your data attachment.

Note: The property **iconVisFlag** must be selected in order to display tool tips.

Alert Properties

Property Name	Description	
valueHighAlertEnabledFlag	Select to enable the **valueHighAlert** properties.	
	valueHighAlert	Enter the value that the value property must equal for the **valueHighAlert** properties to be applied. This may be a number or text.

	valueHighAlertColor	Select the ... button and choose a color from the palette to use for the background if value equals **valueHighAlert**.
	valueHighAlertGradientColor2	Select the ... button and choose a color from the palette to use for the second gradient color of the background if value equals **valueHighAlert**.
	valueHighAlertImage	Select the image to display if value equals **valueHighAlert**.
valueLowAlertEnabledFlag	Select to enable the **valueLowAlert** properties.	
	valueLowAlert	Enter the value that the value property must equal for the **valueLowAlert** properties to be applied. This may be a number or text.
	valueLowAlertColor	Select the ... button and choose a color from the palette to use for the background if value equals **valueLowAlert**.
	valueLowAlertGradientColor2	Select the ... button and choose a color from the palette to use for the second gradient color of the background if value equals **valueLowAlert**.
	valueLowAlertImage	Select the image to display if value equals **valueLowAlert**.
valueMediumAlertEnabledFlag	Select to enable the **valueMediumAlert** properties.	
	valueMediumAlert	Enter the value that the value property must equal for the **valueMediumAlert** properties to be applied. This may be a number or text.
	valueMediumAlertColor	Select the ... button and choose a color from the palette to use for the background if value equals **valueMediumAlert**.
	valueMediumAlertGradientColor2	Select the ... button and choose a color from the palette to use for the second gradient color of the background if value equals **valueMediumAlert**.
	valueMediumAlertImage	Select the image to display if value equals **valueMediumAlert**.

Background Properties

Property Name	Description
bgBorderColor	Select the ... button and choose from the palette to set the color of the edge on the background. This property is only applicable if **bgBorderFlag** is selected.

bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgClipTextFlag	If selected, then the label and/or its **valueString** (whichever is drawn within the background rectangle) will be clipped by the object's background rectangle. You might use this option if you have a very long **label** or **valueString** and do not want to have the label/valuestring extend past the edges of the rectangle. **Note:** You must enable **bgVisFlag** for this option to work. For example, with **bgClipTextFlag** disabled and **bgVisFlag** enabled (and **labelTextPosX** set to **Inside Left**), your object and long text would look like this:  With both **bgClipTextFlag** and **bgVisFlag** enabled (and **labelTextPosX** set to **Inside Left**), the same object and long text would look like this:  If you want to view the full text/valueString, you can copy the full text/valueString into the **Interaction**> **mouseOverText** field to display the full string when hovering over the object. For example:  **Note:** By default, this flag is unchecked and is only applicable when **bgStyle** is set to **0** (rectangle), **1** (3D Rectangle), or **2** (Round Rectangle).
bgColor	Select the ... button and choose from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is 3D Rectangle.
bgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The **bgColor** property sets the first color in the gradient. **Note:** This property will be ignored if **bgGradientMode** is set to None.
bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.

bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (25) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners. **Circle** - Select to display a background circle. **Diamond** - Select to display a background diamond.
bgVisFlag	Select to display the background.

Data Properties

Property Name	Description
value	Attach your data. This value is compared to the enabled alert comparisons. This can be a number or a string.
valueDivisor	If specified and the value is a number, this divisor is applied to the value.

Data Format Properties

Property Name	Description
valueFormat	Select or enter the numeric format of value displayed in the value label. To enter a format, use syntax from the **Java DecimalFormat** class. This is only used if value is a number.
valueTextColor	Select the  button and choose from the palette to set the color of the value label.
valueTextFont	Select the font to use for the value label.
valueTextHeight	Specify the height for the value label.
valueTextPosX	Set x-axis position of value label. Select from the following options: • **Left** - Position outside the left side of the background. • **Inside Left** - Position inside the left side of the background. • **Center** - Position in the center of the background. • **Inside Right** - Position inside the right side of the background. • **Right** - Position outside the right side of the background.

valueTextPosY	Set y-axis position of value label. Select from the following options: • **Outside Top** - Position well above the background. • **Top** - Position just above the background. • **Inside Top** - Position inside the top of the background. • **Center** - Position in the center of the background. • **Inside Bottom** - Position inside the bottom of the background. • **Bottom** - Position just below the background. • **Outside Bottom** - Position well below the background.
valueVisFlag	Control visibility of the value label and related properties.

Image Properties

Property Name	Description
image	Specify an image to use in the background. This image will be used unless a comparison with an image is active. In that case, the comparison's image will be used instead.
imageAnimatedFlag	When selected, this property allows animated .gif images in the indicator object. If the object's label string overlaps the animated image, you can modify the **labelTextPosY** and/or **labelTextPosX** properties to reposition the label. **Note:** This feature is only supported in the Thin Client. In the builder/viewer, the animation will only update once every two seconds or after a mouse event.
imageScaleFlag	If selected, scale the image to the size of the background. Otherwise, the image is drawn at its original size.

Interaction Properties

Property Name	Description
command	Assign a command to your indicator. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.

commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information. **Note:** When **drillDownSelectMode** is set to **Anywhere,** double-clicking anywhere on the scale will activate the specified **drillDownTarget**; however, you must double-click on a trace in the scale to set the substitutions listed above.
mouseOverText	Enter a tool tip for this indicator. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nscale**). **Note:** The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelTextColor	Select the ... button and choose from the palette to set the color of the label text.
labelTextFont	Select the font style of the label text from drop down menu.
labelTextHeight	Set the height (in pixels) of the label text.
labelTextPosX	Set x-axis position of label text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.
labelTextPosY	Set y-axis position of label text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.
labelVisFlag	Control visibility of the label and related properties.

Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Value Quality Properties

Property Name	Description
valueQualityEnabledFlag	Select to enable the **valueQuality** properties:

valueQuality	Attach this value to data that indicates the quality of the data attached to value. This value must be a number.
valueQualityLostData	Enter a number that **valueQuality** will equal if the data has been lost.
valueQualityLostDataColor	Select the ... button and choose a color from the palette to use for the background if **valueQuality** equals **valueQualityLostData**.
valueQualityLostDataGradientColor2	Select the ... button and choose a color from the palette to use for the second color of the background gradient if **valueQuality** equals **valueQualityLostData**.
valueQualityLostDataImage	Select the image to display if **valueQuality** equals **valueQualityLostData**.
valueQualityNoData	Enter a number that **valueQuality** will equal if no data is available.
valueQualityNoDataColor	Select the ... button and choose a color from the palette to use for the background if **valueQuality** equals **valueQualityLostData**.
valueQualityNoDataGradientColor2	Select the ... button and choose a color from the palette to use for the second color of the background gradient if **valueQuality** equals **valueQualityNoData**.
valueQualityNoDataImage	Select the image to display if **valueQuality** equals **valueQualityLostData**.

Indicator Objects - Limits

The Object Palette features three types of limits indicators:

obj_ind_limits	This is an indicator that allows you to set up to 4 comparison thresholds. When the value is greater than one of the high thresholds or less than one of the low thresholds, the color of the background and the background image will change to the color and image specified for that threshold.

High Value Indicator	This is **obj_ind_limits** configured to show two high thresholds (**valueHighAlarm = 95**, **valueHighWarning = 75**). The **valueQualityEnabledFlag** is selected with **valueQualityNoData = 1** and **valueQualityLostData = 2**. This object uses images instead of background colors. The label is shown to the right of the indicator, and the value is shown to the left.
Low Value Indicator	This is **obj_ind_limits** configured to show two low thresholds (**valueLowAlarm = 5**, **valueLowWarning = 15**. The **valueQualityEnabledFlag** is selected with **valueQualityNoData = 1** and **valueQualityLostData = 2**. This object uses images instead of background colors. The label is shown to the right of the indicator and the value is shown to the left.

Using Interaction Properties

Commands

To assign a command to your indicator, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Once a drill down target has been set, double-click on the indicator to activate the drill down. Drill down displays can be activated in the same window that contains the indicator or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your indicator. To display a tool tip, point to the indicator with your mouse. The tool tip will contain information from your data attachment.

Note: The property **iconVisFlag** must be selected in order to display tool tips.

Alert Properties

Property Name	Description	
valueHighAlarmEnabledFlag	Select to enable **valueHighAlarm** and related properties:	
	valueHighAlarm	Enter the value that the value property must be greater than for the **valueHighAlarm** properties to be applied. This must be a number.
	valueHighAlarmColor	Select the  button and choose a color from the palette to use for the background if value is greater than **valueHighAlarm**.

	valueHighAlertGradientColor2	Select the ... button and choose a color from the palette to use for the second gradient color of the background if value is greater than **valueHighAlarm**.
	valueHighAlarmImage	Select the image to display if value is greater than **valueHighAlarm**.
valueHighWarningEnabledFlag	Select to enable valueHighWarning and related properties:	
	valueHighWarning	Enter the value that the value property must be greater than for the **valueHighWarning** properties to be applied. This must be a number.
	valueHighWarningColor	Select the ... button and choose a color from the palette to use for the background if value is greater than **valueHighWarning**.
	valueHighWarningGradientColor2	Select the ... button and choose a color from the palette to use for the second gradient color of the background if value is greater than **valueHighWarning**.
	valueHighWarningImage	Select the image to display if value is greater than **valueHighWarning**.
valueLowAlarmEnabledFlag	Select to enable **valueLowAlarm** and related properties:	
	valueLowAlarm	Enter the value that the value property must be less than for the **valueLowAlarm** properties to be applied. This must be a number.
	valueLowAlarmColor	Select the ... button and choose a color from the palette to use for the background if value is less than **valueLowAlarm**.
	valueLowAlarmGradientColor2	Select the ... button and choose a color from the palette to use for the second gradient color of the background if value is less than **valueLowAlarm**.
	valueLowAlarmImage	Select the image to display if value is less than **valueLowAlarm**.
valueLowWarningEnabledFlag	Select to enable **valueLowWarning** and related properties:	
	valueLowWarning	Enter the value that the value property must be less than for the **valueLowWarning** properties to be applied. This must be a number.

valueLowWarningColor	Select the ... button and choose a color from the palette to use for the background if value is less than **valueLowWarning**.
valueLowWarningGradientColor2	Select the ... button and choose a color from the palette to use for the second gradient color of the background if value is less than **valueLowWarning**.
valueLowWarningImage	Select the image to display if value is less than **valueLowWarning**.

Background Properties

Property Name	Description
bgBorderColor	Select the ... button and choose from the palette to set the color of the edge on the background. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgClipTextFlag	If selected, then the label and/or its **valueString** (whichever is drawn within the background rectangle) will be clipped by the object's background rectangle. You might use this option if you have a very long **label** or **valueString** and do not want to have the label/valuestring extend past the edges of the rectangle. **Note:** You must enable **bgVisFlag** for this option to work. For example, with **bgClipTextFlag** disabled and **bgVisFlag** enabled (and **labelTextPosX** set to **Inside Left**), your object and long text would look like this:  With both **bgClipTextFlag** and **bgVisFlag** enabled (and **labelTextPosX** set to **Inside Left**), the same object and long text would look like this:  If you want to view the full text/valueString, you can copy the full text/valueString into the **Interaction**> **mouseOverText** field to display the full string when hovering over the object. For example:  **Note:** By default, this flag is unchecked and is only applicable when **bgStyle** is set to **0** (rectangle), **1** (3D Rectangle), or **2** (Round Rectangle).
bgColor	Select the ... button and choose from the palette to set the background color.

bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the  button and choose from the palette to set the second color in the gradient. The **bgColor** property sets the first color in the gradient. **Note:** This property will be ignored if **bgGradientMode** is set to None.
bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners. **Circle** - Select to display a background circle. **Diamond** - Select to display a background diamond.
bgVisFlag	Select to display the background.

Data Properties

Property Name	Description
value	Attach your data. This value is compared to the enabled alert thresholds. This must be a number.

valueDivisor	If specified, this divisor is applied to the value.
valueEqualTriggersThresholdFlag	If selected, the indicator changes color (or image) when the value is greater than or equal to the high thresholds (**valueHighAlarm** or **valueHighWarning**) or less than or equal to the low thresholds (**valueLowAlarm** or **valueLowWarning**). If deselected, the indicator only changes when the value is greater than the high thresholds or less than the low thresholds.

Data Format Properties

Property Name	Description
valueFormat	Select or enter the numeric format of value displayed in the value label. To enter a format, use syntax from the **Java DecimalFormat** class.
valueTextColor	Select the  button and choose from the palette to set the color of the value label.
valueTextFont	Select the font to use for the value label.
valueTextHeight	Specify the height for the value label.
valueTextPosX	Set x-axis position of value label. Select from the following options: • **Left** - Position outside the left side of the background. • **Inside Left** - Position inside the left side of the background. • **Center** - Position in the center of the background. • **Inside Right** - Position inside the right side of the background. • **Right** - Position outside the right side of the background.
valueTextPosY	Set y-axis position of value label. Select from the following options: • **Outside Top** - Position well above the background. • **Top** - Position just above the background. • **Inside Top** - Position inside the top of the background. • **Center** - Position in the center of the background. • **Inside Bottom** - Position inside the bottom of the background. • **Bottom** - Position just below the background. • **Outside Bottom** - Position well below the background.
valueVisFlag	Control visibility of the value label and related properties.

Image Properties

Property Name	Description
image	Specify an image to use in the background. This image will be used unless a comparison with an image is active. In that case, the comparison's image will be used instead.

imageAnimatedFlag	When selected, this property allows animated .gif images in the indicator object. If the object's label string overlaps the animated image, you can modify the **labelTextPosY** and/or **labelTextPosX** properties to reposition the label. **Note:** This feature is only supported in the Thin Client. In the builder/viewer, the animation will only update once every two seconds or after a mouse event.
imageScaleFlag	If selected, scale the image to the size of the background. Otherwise, the image is drawn at its original size.

Interaction Properties

Property Name	Description
command	Assign a command to your indicator. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays"for information. **Note:** When **drillDownSelectMode** is set to **Anywhere** double-clicking anywhere on the scale will activate the specified **drillDownTarget**, however you must double-click on a trace in the scale to set the substitutions listed above.
mouseOverText	Enter a tool tip for this indicator. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nscale**). **Note:** The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelTextColor	Select the ... button and choose from the palette to set the color of the label text.
labelTextFont	Select the font style of the label text from drop down menu.
labelTextHeight	Set the height (in pixels) of the label text.
labelTextPosX	Set x-axis position of label text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.
labelTextPosY	Set y-axis position of label text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.
labelVisFlag	Control visibility of the label and related properties.

Object Properties

Property Name	Description
anchor	Select the ... button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.

dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Value Quality Properties

Property Name	Description	
valueQualityEnabledFlag	Select to enable the valueQuality properties:	
	valueQuality	Attach this value to data that indicates the quality of the data attached to value. This value must be a number.
	valueQualityLostData	Enter a number that **valueQuality** will equal if the data has been lost.
	valueQualityLostDataColor	Select the ... button and choose a color from the palette to use for the background if **valueQuality** equals **valueQualityLostData**.
	valueQualityLostDataGradientColor2	Select the ... button and choose a color from the palette to use for the second color of the background gradient if **valueQuality** equals **valueQualityLostData**.

valueQualityLostDataImage	Select the image to display if **valueQuality** equals **valueQualityLostData**.
valueQualityNoData	Enter a number that **valueQuality** will equal if no data is available.
valueQualityNoDataColor	Select the ... button and choose a color from the palette to use for the background if **valueQuality** equals **valueQualityLostData**.
valueQualityNoDataGradientColor2	Select the ... button and choose a color from the palette to use for the second color of the background gradient if **valueQuality** equals **valueQualityNoData**.
valueQualityNoDataImage	Select the image to display if **valueQuality** equals **valueQualityLostData**.

Indicator Objects - Multi

The object **obj_ind_multi** is an indicator that allows you to enable an unlimited number of comparison values (Alert State 01, Alert State 02, etc.). When the value property meets one of the enabled Alert State conditions, then the background color and background image will change to the specified **alertState*** color and image.

Note: When the **value** meets the conditions of more than one Alert State, then the properties of the highest Alert State are used (e.g. AlertState 02 is higher than AlertState 01).

Using Interaction Properties

Commands

To assign a command to your indicator, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Once a drill down target has been set, double-click on the indicator to activate the drill down. Drill down displays can be activated in the same window that contains the indicator or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Tool Tips

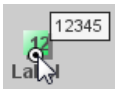
Select the **mouseOverFlag** to enable tool tips for your indicator. To display a tool tip, point to the indicator with your mouse. The tool tip will contain information from your data attachment.

Note: The property **iconVisFlag** must be selected in order to display tool tips.

Alert Properties

Property Name	Description	
numAlertStates	Specify the number of Alert State comparisons to enable (Alert State 01, Alert State 02, etc.). The following **alertState** properties will be listed for each:	
	alertStateN	Enter a value (numeric or text) to compare to the value property for the **alertStateN** properties to be applied.
	alertStateNColor	Select the ... button and choose a color from the palette to use for the background if value and **alertStateN** meet **alertStateNCondition**.
	alertStateNCondition	Comparison condition to use for this alert state. Select from the following: **Equal** - The value must equal **alertStateN** for alertStateN properties to be applied **Not Equal** - The value must not equal **alertStateN** for **alertStateN** properties to be applied **Greater Than** - The value must be greater than **alertStateN** for **alertStateN** properties to be applied. **Note:** This option is only supported if value and **alertStateN** are both numeric values. **Less Than** - The value must be less than **alertStateN** for **alertStateN** properties to be applied. **Note:** This option is only supported if value and **alertStateN** are both numeric values. **Greater Than or Equal To** - The value must be greater than or equal to **alertStateN** for **alertStateN** properties to be applied. **Note:** This option is only supported if value and **alertStateN** are both numeric values. **Less Than or Equal To** - The value must be less than or equal to **alertStateN** for **alertStateN** properties to be applied. **Note:** This option is only supported if value and **alertStateN** are both numeric values.
	alertStateNGradientColor2	Select the ... button and choose a color from the palette to use for the second gradient color of the background if value and **alertStateN** meet **alertStateNCondition**.
	alertStateNImage	Select the image to display if value and **alertStateN** meet **alertStateNCondition**.

Background Properties

Property Name	Description
bgBorderColor	Select the ... button and choose from the palette to set the color of the edge on the background. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgClipTextFlag	If selected, then the label and/or its **valueString** (whichever is drawn within the background rectangle) will be clipped by the object's background rectangle. You might use this option if you have a very long **label** or **valueString** and do not want to have the label/valuestring extend past the edges of the rectangle. **Note:** You must enable **bgVisFlag** for this option to work. For example, with **bgClipTextFlag** disabled and **bgVisFlag** enabled (and **labelTextPosX** set to **Inside Left**), your object and long text would look like this:  With both **bgClipTextFlag** and **bgVisFlag** enabled (and **labelTextPosX** set to **Inside Left**), the same object and long text would look like this:  If you want to view the full text/valueString, you can copy the full text/valueString into the **Interaction**> **mouseOverText** field to display the full string when hovering over the object. For example:  **Note:** By default, this flag is unchecked and is only applicable when **bgStyle** is set to **0** (rectangle), **1** (3D Rectangle), or **2** (Round Rectangle).
bgColor	Select the ... button and choose from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The **bgColor** property sets the first color in the gradient. **Note:** This property will be ignored if **bgGradientMode** is set to **None**.

bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners. **Circle** - Select to display a background circle. **Diamond** - Select to display a background diamond.
bgVisFlag	Select to display the background and related properties.

Data Properties

Property Name	Description
value	Attach your data (numeric or string). This data is compared to all specified **alertState*** values.
valueDivisor	If the specified **value** is numeric, then this divisor is applied.

Data Format Properties

Property Name	Description
valueFormat	Select or enter the numeric format of value displayed in the value label. To enter a format, use syntax from the **Java DecimalFormat** class. This is only used if value is a number.

valueTextColor	Select the ... button and choose from the palette to set the color of the value label.
valueTextFont	Select the font to use for the value label.
valueTextHeight	Specify the height for the value label.
valueTextPosX	Set x-axis position of value label. Select from the following options: • **Left** - Position outside the left side of the background. • **Inside Left** - Position inside the left side of the background. • **Center** - Position in the center of the background. • **Inside Right** - Position inside the right side of the background. • **Right** - Position outside the right side of the background.
valueTextPosY	Set y-axis position of value label. Select from the following options: • **Outside Top** - Position well above the background. • **Top** - Position just above the background. • **Inside Top** - Position inside the top of the background. • **Center** - Position in the center of the background. • **Inside Bottom** - Position inside the bottom of the background. • **Bottom** - Position just below the background. • **Outside Bottom** - Position well below the background.
valueVisFlag	Control visibility of the value label and related properties.

Image Properties

Property Name	Description
image	Specify an image to use in the background. This image will be used unless a comparison with an image is active. In that case, the comparison's image will be used instead.
imageAnimatedFlag	When selected, this property allows animated .gif images in the indicator object. If the object's label string overlaps the animated image, you can modify the **labelTextPosY** and/or **labelTextPosX** properties to reposition the label. **Note:** This feature is only supported in the Thin Client. In the builder/viewer, the animation will only update once every two seconds or after a mouse event.
imageScaleFlag	If selected, scale the image to the size of the background. Otherwise, the image is drawn at its original size.

Interaction Properties

Property Name	Description
command	Assign a command to your indicator. See "Define/Execute Command" for information.

commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information. **Note:** When **drillDownSelectMode** is set to Anywhere double-clicking anywhere on the scale will activate the specified **drillDownTarget**, however you must double-click on a trace in the scale to set the substitutions listed above.
mouseOverText	Enter a tool tip for this indicator. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nscale**). **Note:** The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelTextColor	Select the ... button and choose from the palette to set the color of the label text.
labelTextFont	Select the font style of the label text from drop down menu.
labelTextHeight	Set the height (in pixels) of the label text.
labelTextPosX	Set x-axis position of label text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.

labelTextPosY	Set y-axis position of label text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.
labelVisFlag	Control visibility of the label and related properties.

Object Properties

Property Name	Description
anchor	Select the ... button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.

styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Value Quality Properties

Property Name	Description	
valueQualityEnabledFlag	Select to enable the **valueQuality** properties:	
	valueQuality	Attach this value to data that indicates the quality of the data attached to value. This value must be a number.
	valueQualityLostData	Enter a number that **valueQuality** will equal if the data has been lost.
	valueQualityLostDataColor	Select the  button and choose a color from the palette to use for the background if **valueQuality** equals **valueQualityLostData**.
	valueQualityLostDataGradientColor2	Select the  button and choose a color from the palette to use for the second color of the background gradient if **valueQuality** equals **valueQualityLostData**.
	valueQualityLostDataImage	Select the image to display if **valueQuality** equals **valueQualityLostData**.
	valueQualityNoData	Enter a number that **valueQuality** will equal if no data is available.
	valueQualityNoDataColor	Select the  button and choose a color from the palette to use for the background if **valueQuality** equals **valueQualityLostData**.
	valueQualityNoDataGradientColor2	Select the  button and choose a color from the palette to use for the second color of the background gradient if **valueQuality** equals **valueQualityNoData**.
	valueQualityNoDataImage	Select the image to display if **valueQuality** equals **valueQualityLostData**.

Indicator Objects - Panel

The Object Palette features two panel indicators:

obj_ind_panel	This is an indicator that allows you to set up to 3 comparison values (indicator1, indicator2 and indicator3). When the value equals one of the enabled comparisons, the color of the corresponding indicator in the panel will change to the color and image specified for that comparison.

Using Interaction Properties

Commands

To assign a command to your indicator, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Once a drill down target has been set, double-click on the indicator to activate the drill down. Drill down displays can be activated in the same window that contains the indicator or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your indicator. To display a tool tip, point to the indicator with your mouse. The tool tip will contain information from your data attachment.

Note: The property **iconVisFlag** must be selected in order to display tool tips.

Alert Properties

Property Name	Description
indicator1	Enter the value that the value property must equal for the indicator1 properties to be applied. This may be a number or text.
indicator1Color	Select the ... button and choose a color from the palette to use for the indicator if value equals **indicator1**.
indicator1GradientColor2	Select the ... button and choose a color from the palette to use for the second gradient color of the indicator if value equals **indicator1**.
indicator1VisFlag	Controls visibility of **indicator1** and related object properties.
indicator2	Enter the value that the value property must equal for the **indicator2** properties to be applied. This may be a number or text.

indicator2Color	Select the ... button and choose a color from the palette to use for the indicator if value equals **indicator2**.
indicator2GradientColor2	Select the ... button and choose a color from the palette to use for the second gradient color of the indicator if value equals **indicator2**.
indicator2VisFlag	Controls visibility of **indicator2** and related object properties.
indicator3	Enter the value that the value property must equal for the **indicator3** properties to be applied. This may be a number or text.
indicator3Color	Select the ... button and choose a color from the palette to use for the indicator if value equals **indicator3**.
indicator3GradientColor2	Select the ... button and choose a color from the palette to use for the second gradient color of the indicator if value equals **indicator3**.
indicator3VisFlag	Controls visibility of **indicator3** and related object properties.

Background Properties

Property Name	Description
bgBorderColor	Select the ... button and choose from the palette to set the color of the edge on the background. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The **bgColor** property sets the first color in the gradient. **Note:** This property will be ignored if **bgGradientMode** is set to None.
bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.

bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background.
borderPixels	The number of pixels between the edge of the background and the indicators.

Data Properties

Property Name	Description
value	Attach your data. This value is compared to the enabled indicators. This can be a number or a string.
valueDivisor	If specified and the value is a number, this divisor is applied to the **value**.

Interaction Properties

Property Name	Description
command	Assign a command to your indicator. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.

commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownTarget	Name of display (.rtv) file targeted as a drill down. See "Drill Down Displays" for information. **Note:** When **drillDownSelectMode** is set to **Anywhere** double-clicking anywhere on the scale will activate the specified **drillDownTarget**, however you must double-click on a trace in the scale to set the substitutions listed above.
mouseOverText	Enter a tool tip for this indicator. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nscale**). **Note:** The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelTextColor	Select the ... button and choose from the palette to set the color of the label text.
labelTextFont	Select the font style of the label text from drop down menu.
labelTextHeight	Set the height (in pixels) of the label text.
labelTextPosX	Set x-axis position of label text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.
labelTextPosY	Set y-axis position of label text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.
labelVisFlag	Control visibility of the label.

Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Value Quality Properties

Property Name	Description
valueQualityEnabledFlag	Select to enable the **valueQuality** properties:

valueQuality	Attach this value to data that indicates the quality of the data attached to value. This value must be a number.
valueQualityLostData	Enter a number that **valueQuality** will equal if the data has been lost.
valueQualityLostDataColor	Select the ... button and choose a color from the palette to use for the background if **valueQuality** equals **valueQualityLostData**.
valueQualityLostDataGradientColor2	Select the ... button and choose a color from the palette to use for the second color of the background gradient if **valueQuality** equals **valueQualityLostData**.
valueQualityNoData	Enter a number that **valueQuality** will equal if no data is available.
valueQualityNoDataColor	Select the ... button and choose a color from the palette to use for the background if **valueQuality** equals **valueQualityLostData**.
valueQualityNoDataGradientColor2	Select the ... button and choose a color from the palette to use for the second color of the background gradient if **valueQuality** equals **valueQualityNoData**.

Labels Objects

"Value Labels"

The value and label properties are used, respectively, to display numeric and text data, either from a data attachment or static values.

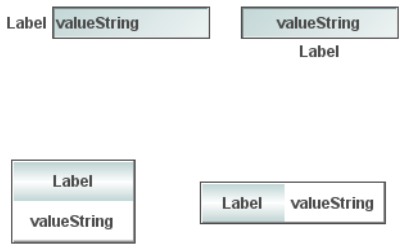
Value Labels that feature Alert properties can be designed to change color according to numeric values from your data attachment.

"Vu Labels"

Similar to Value Labels, Vu Labels contain the value and label properties as well as featuring vu meters to display numeric values from your data attachment.

"ValueString Labels"

The valueString and label properties are both used to display text data, either from a data attachment or static values.



"Text Only Label" / "ValueString Only Label"

The Text Only Label can be rotated at any angle and is limited to a static value.

The ValueString Only Label can display text data, either from a data attachment or a static value.

Image Property

Labels objects that feature the **image** property can be customized to display your image (.**gif**, .**jpg**, or .**png**) file. Click in the **image Property Value** field and type the name of the image. Or, select the ... button to open the **Select Image** dialog containing up to three directories:

- **Current Directory** - Contains images in the current directory and one level of subdirectories.
- **Custom Image Library** - If you have specified a custom image library, this directory contains those images. See Creating a Custom Image Library for details.
- **Symbol Library** - Contains symbolic images (for example, symbols for various types of hardware, shapes, lights, arrows, etc.).

Navigate to the image you want to use and select it. A preview of the image appears in the pane to the right. Click **OK** or **Apply** to set the image on your object. If an image is not listed, enter the name of the file, including the relative path.

To scale your image to the size of the object, check the **imageScaleFlag**. The **visFlag** property controls the visibility of the object. The **transparencyPercent** property controls the transparency of the object.

Note: The sample display file **general_objects.rtv** (located in **demos/tutorials**) features information on working with objects from the General tab.

Creating a Custom Image Library

The custom image library enables you to make your own images available in the Select Image dialog. To add your own image library, perform the following steps.

1. Place your images .**jar** file and add it to the RTV_USERPATH environment variable. The images must be in a directory (not in the top level of the jar). They can be organized into subdirectories of one top level directory.

2. In the Display Builder, select **Tools/Builder Options** and, in the **Custom Image Library Path** field, set the path to the directory containing your images .jar file.

For example, if you have a jar with this directory structure:

com/mycompany/Images

com/mycompany/Images/Blue Images

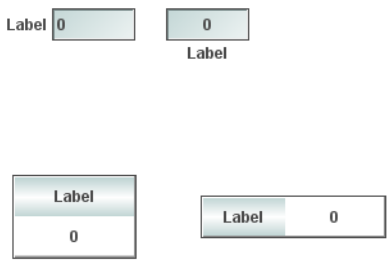
com/mycompany/Images/Red Images

com/mycompany/Images/Green Images

With this directory structure, you would enter **com/mycompany/Images**. This adds a directory named **Images** to the tree in the Select Image dialog. The **Images** directory will have three subdirectories: **Blue Images**, **Red Images**, and **Green Images**. Only directories containing images are added to the Select Image dialog.

To access the images, you can edit any property that allows you to set an image on an object (for example, the **image**, **barImage**, and **filterProperties** properties), or edit the File>Background Properties>Image Name field.

Value Labels



Using Interaction Properties

Commands

To assign a command to your label, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Once a drill down target has been set, double-click on the label to activate the drill down. Drill down displays can be activated in the same window that contains the label or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your label. To display a tool tip, point to the label with your mouse. The tool tip will contain information from your data attachment.

Alert Properties

Note: Not all Value Labels feature Alert Properties.

Property Name	Description	
valueHighAlarmEnabledFlag	Select to enable the high alarm and set the following related properties:	
	valueHighAlarm	Specify the minimum value for the High Alarm.
	valueHighAlarmColor	Select the ... button and choose from the palette to set the color of the High Alarm.
valueHighWarningEnabledFlag	Select to enable the high alarm and set the following related properties:	
	valueHighWarning	Specify the minimum value for the High Warning.
	valueHighWarningColor	Select the ... button and choose from the palette to set the color of High Warning.

Background Properties

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. **Note:** This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.

bgClipTextFlag

If selected, then the label and/or its **valueString** (whichever is drawn within the background rectangle) will be clipped by the object's background rectangle. You might use this option if you have a very long **label** or **valueString** and do not want to have the label/valuestring extend past the edges of the rectangle.

Note: You must enable **bgVisFlag** for this option to work.

For example, with **bgClipTextFlag** disabled and **bgVisFlag** enabled (and **labelTextPosX** set to **Inside Left**), your object and long text would look like this:

With both **bgClipTextFlag** and **bgVisFlag** enabled (and **labelTextPosX** set to **Inside Left**), the same object and long text would look like this:

If you want to view the full text/valueString, you can copy the full text/valueString into the **Interaction**> **mouseOverText** field to display the full string when hovering over the object. For example:

Note: By default, this flag is unchecked and is only applicable for **obj_rect_il**, **obj_rect_ilv**, and **obj_rect_ilvs**.

bgColor

Select the ... button and choose a color from the palette to set the background color.

bgEdgeWidth

Set the width of the 3D edge on the background rectangle.

Note: This property is only available if the **bgStyle** selected is 3D Rectangle.

bgGradientColor2

Select the ... button and choose from the palette to set the second color in the gradient. The **bgColor** property sets the first color in the gradient.

Note: This property will be ignored if **bgGradientMode** is set to None.

bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. **Note:** This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Controls visibility of background. **Note:** This property only applies to **obj_rect_ilv**.

Data Properties

Property Name	Description
value	Attach your data.
valueDivisor	If specified, this divisor is applied to the value.

Data Format Properties

Note: Not all Data Format Properties apply to all Value Labels.

Property Name	Description
valueFormat	Select or enter the numeric format of value displayed in the value text. To enter a format, use syntax from the **Java Decimal Format** class.
valueTextAlignX	Set the position of value text on the x-axis. Select from the following options: • **Left** • **Center** • **Right**
valueTextAlignY	Set the position of value text on the y-axis. Select from the following options: • **Inside Top** • **Center** • **Inside Bottom**
valueTextColor	Select the  button and choose from the palette to set the color of the value text.
valueTextFont	Select the font to use for the value text.
valueTextHeight	Specify (in pixels) the height for the value text.
valueTextPosX	Set x-axis position of value text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.
valueTextPosY	Set y-axis position of value text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.
valueVisFlag	Control visibility of the value.

Image Properties

Property Name	Description
image	Enter the name of an image or click on the  button to open the **Select Image** dialog. See "Labels Objects" for more information. **Note:** This property only applies to **obj_rect_ilv**.

imageAnimatedFlag	When selected, this property allows animated .gif images in the label object. If the object's label string overlaps the animated image, you can modify the **labelTextPosY** and/or **labelTextPosX** properties to reposition the label. **Note:** This property only applies to the Thin Client and only applies to **obj_rect_il**. In the builder/viewer, the animation will only update once every two seconds or after a mouse event.
imageScaleFlag	Select to scale your image to the size of the object. **Note:** This property only applies to **obj_rect_ilv**.

Interaction Properties

Property Name	Description
command	Assign a command to your vu label. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownTarget	Name of display (.rtv) file targeted as a drill down. See "Drill Down Displays" for information.
mouseOverText	Enter a tool tip for this label. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nlabel**). **Note:** The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

Label Properties

Note: Not all Label Properties apply to all Value Labels.

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelBgColor	Select the ... button and choose a color from the palette to set the background color.
labelBgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The **bgColor** property sets the first color in the gradient. **Note:** This property will be ignored if **bgGradientMode** is set to **None**.
labelBgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
labelMinTabWidth	Specify minimum width of the label tab. **Note:** This property only applies if **labelTextPosY** is set to **Tab Top**.
labelTextAlignX	Set the position of label text on the x-axis. Select from the following options: • **Left** • **Center** • **Right**
labelTextAlignY	Set the position of label text on the y-axis. Select from the following options: • **Inside Top** • **Center** • **Inside Bottom**
labelTextColor	Select the ... button and choose from the palette to set the color of the label text.
labelTextFont	Select the font style of the label text from drop down menu.
labelTextHeight	Set the height (in pixels) of the label text.
labelTextPosX	Set x-axis position of label text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.

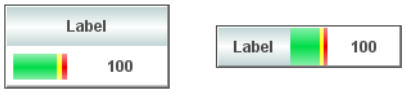
labelTextPosY	Set y-axis position of label text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Tab Top** - Position tab just above the background rectangle. **Note:** Height and width of the label tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. • **Title Top** - Position along the top line of the background rectangle. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.
labelVisFlag	Control visibility of the label.
labelWidth	Set the width (in pixels) of the background area of the label text. **Note:** This property only applies to **obj_label11**.
labelWordWrapMode	Text will be wrapped to fit the width of the object depending on the mode selected. Default is **None**. **Space** - Whenever possible, only add line breaks at whitespaces. If a single word is longer than the object is wide, the line break will be added to the word so that the text fits within the width of the object. **Space and Punctuation** - Whenever possible, only add line breaks at whitespaces and punctuation characters. Supported punctuation characters are: comma(,), period(.), semi-colon(;), colon (:), hyphen(-), question mark(?), asterisk(*), ampersand(&), greater than(<), less than(>), backslash(\), forward slash(/), pipe(|), plus(+), exclamation point(!), and at(@). If a single word is longer than the object is wide and none of these characters is in the word, the line break will be added to the word so that the text fits within the width of the object. **Note:** The **labelWordWrapMode** property is ignored, and text will not be wrapped, if: • **labelTextPosX** is set to **Outside Right** or **Outside Left** • **labelTextPosY** is set to **Tab Top** or **Title Top**.

Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.

dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Vu Labels



Using Interaction Properties

Commands

To assign a command to your label, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Once a drill down target has been set, double-click on the label to activate the drill down. Drill down displays can be activated in the same window that contains the label or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your label. To display a tool tip, point to the label with your mouse. The tool tip will contain information from your data attachment.

Alert Properties

Property Name	Description
valueHighAlarm	Specify the minimum value for the High Alarm range.
valueHighAlarmColor	Select the ... button and choose from the palette to set the color of the High Alarm.
valueHighWarning	Specify the minimum value for the High Warning range.
valueHighWarningColor	Select the ... button and choose from the palette to set the color of High Warning.
valueNoAlarmColor	Select the ... button and choose from the palette to set the color of the No Alarm.

Background Properties

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. **Note:** This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose a color from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. **Note:** This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The bgColor property sets the first color in the gradient. **Note:** This property will be ignored if **bgGradientMode** is set to None.

bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. **Note:** This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.

Data Properties

Property Name	Description
value	Attach your data.
valueDivisor	If specified, this divisor is applied to the **value**, **valueMin**, and **valueMax**.
valueMax	Set the maximum range for the label. This value must be larger than **valueMin**.
valueMin	Set the minimum range for the label. This value must be smaller than **valueMax**.

Data Format Properties

Property Name	Description
valueFormat	Select or enter the numeric format of value displayed in the value text. To enter a format, use syntax from the **Java DecimalFormat** class.
valueTextAlignX	Set the position of value text on the x-axis. Select from the following options: • **Left** • **Center** • **Right**
valueTextAlignY	Set the position of value text on the y-axis. Select from the following options: • **Inside Top** • **Center** • **Inside Bottom**
valueTextColor	Select the  button and choose from the palette to set the color of the value text.
valueTextFont	Select the font to use for the value text.
valueTextHeight	Specify (in pixels) the height for the value text.
vuMeterMinWidth	Set (in pixels) the minimum width of the vu meter.

Interaction Properties

Property Name	Description
command	Assign a command to your vu label. See **Building Displays>**"Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the  button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.

drillDownTarget	Name of display (.rtv) file targeted as a drill down. See "Drill Down Displays" for information.
mouseOverText	Enter a tool tip for this label. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nlabel**). **Note:** The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelBgColor	Select the ... button and choose a color from the palette to set the background color.
labelBgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The **bgColor** property sets the first color in the gradient. **Note:** This property will be ignored if **bgGradientMode** is set to **None**.
labelBgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
labelTextAlignX	Set the position of label text on the x-axis. Select from the following options: • **Left** • **Center** • **Right**
labelTextAlignY	Set the position of label text on the y-axis. Select from the following options: • **Inside Top** • **Center** • **Inside Bottom**
labelTextColor	Select the ... button and choose from the palette to set the color of the label text.
labelTextFont	Select the font style of the label text from drop down menu.
labelTextHeight	Set the height (in pixels) of the label text.
labelWidth	Set the width (in pixels) of the background area of the label text. **Note:** This property only applies to **obj_vulabel02**.

Object Properties

Property Name	Description
anchor	Select the ... button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: None - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with rtv-.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

ValueString Labels

Using Interaction Properties

Commands

To assign a command to your label, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Once a drill down target has been set, double-click on the label to activate the drill down. Drill down displays can be activated in the same window that contains the label or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your label. To display a tool tip, point to the label with your mouse. The tool tip will contain information from your data attachment.

Background Properties

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. **Note:** This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.

bgClipTextFlag

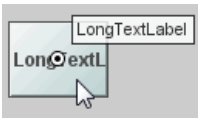
If selected, then the label and/or its **valueString** (whichever is drawn within the background rectangle) will be clipped by the object's background rectangle. You might use this option if you have a very long **label** or **valueString** and do not want to have the label/valuestring extend past the edges of the rectangle.

Note: You must enable **bgVisFlag** for this option to work.

For example, with **bgClipTextFlag** disabled and **bgVisFlag** enabled (and **labelTextPosX** set to **Inside Left**), your object and long text would look like this:

With both **bgClipTextFlag** and **bgVisFlag** enabled (and **labelTextPosX** set to **Inside Left**), the same object and long text would look like this:

If you want to view the full text/valueString, you can copy the full text/valueString into the **Interaction**> **mouseOverText** field to display the full string when hovering over the object. For example:



Note: By default, this flag is unchecked and is only applicable for **obj_rect_il**, **obj_rect_ilv**, and **obj_rect_ilvs**.

bgColor

Select the ... button and choose a color from the palette to set the background color.

bgEdgeWidth

Set the width of the 3D edge on the background rectangle.

Note: This property is only available if the **bgStyle** selected is **3D Rectangle**.

bgGradientColor2

Select the ... button and choose from the palette to set the second color in the gradient. The **bgColor** property sets the first color in the gradient.

Note: This property will be ignored if **bgGradientMode** is set to None.

bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. **Note:** This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Controls visibility of background. **Note:** This property only applies to **obj_rect_ilvs**.

Data Properties

Property Name	Description
valueString	Attach your data.

Data Format Properties

Note: Not all Data Format Properties apply to all Value String Labels.

Property Name	Description
valueFormat	Select or enter the numeric format of value displayed in the value text. To enter a format, use syntax from the **Java DecimalFormat** class. **Note:** This property only applies to **obj_rect_ilvs**.
valueTextAlignX	Set the position of value text on the x-axis. Select from the following options: • **Left** • **Center** • **Right**
valueTextAlignY	Set the position of value text on the y-axis. Select from the following options: • **Inside Top** • **Center** • **Inside Bottom**
valueTextColor	Select the ... button and choose from the palette to set the color of the value text.
valueTextFont	Select the font to use for the value text.
valueTextHeight	Specify (in pixels) the height for the value text.
valueTextPosX	Set x-axis position of value text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.
valueTextPosY	Set y-axis position of value text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.

valueVisFlag	Control visibility of the value text.
valueWordWrapMode	Value text will be wrapped to fit the width of the object depending on the mode selected. Default is **None**. **Space** - Whenever possible, only add line breaks at whitespaces. If a single word is longer than the object is wide, the line break will be added to the word so that the text fits within the width of the object. **Space and Punctuation** - Whenever possible, only add line breaks at whitespaces and punctuation characters. Supported punctuation characters are: comma(,), period(.), semi-colon(;), colon (:), hyphen(-), question mark(?), asterisk(*), ampersand(&), greater than(<), less than(>), backslash(\), forward slash(/), pipe(|), plus(+), exclamation point(!), and at(@). If a single word is longer than the object is wide and none of these characters is in the word, the line break will be added to the word so that the text fits within the width of the object. **Note:** The **valueWordWrapMode** property is ignored, and text will not be wrapped, if **valueTextPosX** is set to **Left** or **Right**.

Image Properties

Property Name	Description
image	Enter the name of an image or click on the ... button to open the **Select Image** dialog. See "Labels Objects" for more information. **Note:** This property only applies to **obj_rect_ilvs**.
imageScaleFlag	Select to scale your image to the size of the object. **Note:** This property only applies to **obj_rect_ilvs**.

Interaction Properties

Property Name	Description
command	Assign a command to your **vu** label. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.

commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information.
mouseOverText	Enter a tool tip for this label. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nlabel**). **Note:** The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

Label Properties

Note: Not all Label Properties apply to all Value String Labels.

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelBgColor	Select the ... button and choose a color from the palette to set the background color.
labelBgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The **bgColor** property sets the first color in the gradient. **Note:** This property will be ignored if **bgGradientMode** is set to **None**.
labelBgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
labelMinTabWidth	Specify minimum width of the label tab. **Note:** This property only applies if **labelTextPosY** is set to **Tab Top**.
labelTextAlignX	Set the position of label text on the x-axis. Select from the following options: • **Left** • **Center** • **Right**

labelTextAlignY	Set the position of label text on the y-axis. Select from the following options: • **Inside Top** • **Center** • **Inside Bottom**
labelTextColor	Select the  button and choose from the palette to set the color of the label text.
labelTextFont	Select the font style of the label text from drop down menu.
labelTextHeight	Set the height (in pixels) of the label text.
labelTextPosX	Set x-axis position of label text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.
labelTextPosY	Set y-axis position of label text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Tab Top** - Position tab just above the background rectangle. **Note:** Height and width of the label tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. • **Title Top** - Position along the top line of the background rectangle. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.
labelVisFlag	Control visibility of the label.

labelWidth	Set the width (in pixels) of the background area of the label text. **Note:** This property only applies to **obj_label11s**.
labelWordWrapMode	Text will be wrapped to fit the width of the object depending on the mode selected. Default is **None**. **Space** - Whenever possible, only add line breaks at whitespaces. If a single word is longer than the object is wide, the line break will be added to the word so that the text fits within the width of the object. **Space and Punctuation** - Whenever possible, only add line breaks at whitespaces and punctuation characters. Supported punctuation characters are: comma(,), period(.), semi-colon(;), colon (:), hyphen(-), question mark(?), asterisk(*), ampersand(&), greater than(<), less than(>), backslash(\), forward slash(/), pipe(|), plus(+), exclamation point(!), and at(@). If a single word is longer than the object is wide and none of these characters is in the word, the line break will be added to the word so that the text fits within the width of the object. **Note:** The **labelWordWrapMode** property is ignored, and text will not be wrapped, if: • **labelTextPosX** is set to **Outside Right** or **Outside Left** • **labelTextPosY** is set to **Tab** or **Title**.

Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.

objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Text Only Label

Using Interaction Properties

Commands

To assign a command to your label, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Once a drill down target has been set, double-click on the label to activate the drill down. Drill down displays can be activated in the same window that contains the label or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Interaction Properties

Property Name	Description
command	Assign a command to your **vu** label. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.

commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays"for information.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelTextAlignX	Set the position of label text on the x-axis. Select from the following options: • Left • Center • Right
labelTextAlignY	Set the position of label text on the y-axis. Select from the following options: • Inside Top • Center • Inside Bottom
labelTextColor	Select the ... button and choose from the palette to set the color of the label text.
labelTextFont	Select the font style of the label text from drop down menu.
labelTextHeight	Set the height (in pixels) of the label text.
rotationAngle	Set the angle at which the label text will be rotated.

Object Properties

Property Name	Description
anchor	Select the ... button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information.

isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools> Object List**.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. See "Style Sheets" for more information. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

ValueString Only Label

Using Interaction Properties

Commands

To assign a command to your label, right-click in the **Property Value** field of the command property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Once a drill down target has been set, double-click on the label to activate the drill down. Drill down displays can be activated in the same window that contains the label or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your label. To display a tool tip, point to the label with your mouse. The tool tip will contain information from your data attachment.

Background Properties

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. **Note:** This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgClipTextFlag	If selected, then the label and/or its **valueString** (whichever is drawn within the background rectangle) will be clipped by the object's background rectangle. You might use this option if you have a very long **label** or **valueString** and do not want to have the label/valuestring extend past the edges of the rectangle. **Note:** You must enable **bgVisFlag** for this option to work. For example, with **bgClipTextFlag** disabled and **bgVisFlag** enabled (and **labelTextPosX** set to **Inside Left**), your object and long text would look like this:  With both **bgClipTextFlag** and **bgVisFlag** enabled (and **labelTextPosX** set to **Inside Left**), the same object and long text would look like this:  If you want to view the full text/valueString, you can copy the full text/valueString into the **Interaction**> **mouseOverText** field to display the full string when hovering over the object. For example:  **Note:** This property is only applicable for **obj_rect_ilvs**.
bgColor	Select the ... button and choose a color from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. **Note:** This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The **bgColor** property sets the first color in the gradient. **Note:** This property will be ignored if **bgGradientMode** is set to **None**.

bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. **Note:** This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Controls visibility of background.

Image Properties

Property Name	Description
image	Enter the name of an image or click on the  button to open the **Select Image** dialog.
imageScaleFlag	Select to scale your image to the size of the object.

Interaction Properties

Property Name	Description
command	Assign a command to your vu label. See "Define/Execute Command" for information.

commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information.
mouseOverText	Enter a tool tip for this label. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nlabel**). **Note:** The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. **Note:** This property only applies if **labelTextPosY** is set to **Tab Top**.
labelTextColor	Select the ... button and choose from the palette to set the color of the label text.
labelTextFont	Select the font style of the label text from drop down menu.
labelTextHeight	Set the height (in pixels) of the label text.
labelTextPosX	Set x-axis position of label text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.

labelTextPosY	Set y-axis position of label text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Tab Top** - Position tab just above the background rectangle. **Note:** Height and width of the label tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. • **Title Top** - Position along the top line of the background rectangle. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.
labelVisFlag	Control visibility of the label.
labelWordWrapMode	Text will be wrapped to fit the width of the object depending on the mode selected. Default is **None**. **Space** - Whenever possible, only add line breaks at whitespaces. If a single word is longer than the object is wide, the line break will be added to the word so that the text fits within the width of the object. **Space and Punctuation** - Whenever possible, only add line breaks at whitespaces and punctuation characters. Supported punctuation characters are: comma(,), period(.), semi-colon(;), colon (:), hyphen(-), question mark(?), asterisk(*), ampersand(&), greater than(<), less than(>), backslash(\), forward slash(/), pipe(|), plus(+), exclamation point(!), and at(@). If a single word is longer than the object is wide and none of these characters is in the word, the line break will be added to the word so that the text fits within the width of the object. **Note:** The **labelWordWrapMode** property is ignored, and text will not be wrapped, if: • **labelTextPosX** is set to **Outside Right** or **Outside Left** • **labelTextPosY** is set to **Tab Top** or **Title Top**.

Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.

dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Meter Objects

There are eight available Meter objects, each of which allow you to view your incoming data in a slightly different format. You can attach your data to the **value** property and you can control the range of data displayed in the meter by setting the **valueMax** and **valueMin** properties. You can also configure your warning and alarm levels along with associated colors and labels so that you can easily view when there are issues with the incoming data attached to the meter.

Using Interaction Properties

Commands

To assign a command to your link, right-click in the **Property Value** field of the **command** property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Once a drill down target has been set, double-click on the link to activate the drill down. Drill down displays can be activated in the same window that contains the link or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your link. To display a tool tip, point to the link with your mouse. The tool tip will contain information from your data attachment.

Note: The properties defined below are combined into one inclusive list that includes all properties from all of the meters. Some of the meters have all of the properties, and some meters only have a portion of the properties listed in this section.

Alert Properties

Property Name	Description
valueHighAlarm	Specify the value that the input value must exceed to activate the high alarm.
valueHighAlarmColor	Select the  button and choose a color for the high alarm.
valueHighWarning	Specify the value that the input value must exceed to activate the high warning.
valueHighWarningColor	Select the  button and choose a color for the high warning.
valueLowAlarm	Specify the value that the input value must go below to activate the low alarm.
valueLowAlarmColor	Select the  button and choose a color for the low alarm.
valueNoAlarmColor	Select the  button and choose a color for when there is no alarm.

Background Properties

Specify how the background is displayed in your meter.

Property Name	Description
bg3dFlag	Control visibility of the bevel on the background.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the  button and choose a color from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle.
bgGradientFlag	Select to display a gradient in the background.
bgRaisedFlag	Reverses the direction of the gradient.
bgVisFlag	Select this check box to display the background rectangle.

Data Properties

Property Name	Description
value	Right click and select **Attach to Data** to attach data to your meter.
valueDivisor	If specified, this divisor is applied to the **value**, **valueMin**, and **valueMax**.
valueMax	Set the maximum range for the meter. This value must be larger than **valueMin**.
valueMin	Set the minimum range for the meter. This value must be smaller than **valueMax**.
value2	Right click and select **Attach to Data** to attach your data to the second meter. **Note:** This option is only available for **obj_dualmeter02**.
value2Divisor	If specified, this divisor is applied to the **value2**, **value2Min**, and **value2Max**. **Note:** This option is only available for **obj_dualmeter02**.
value2Max	Set the maximum range for the scale. This value must be larger than **value2Min**. **Note:** This option is only available for **obj_dualmeter02**.
value2Min	Set the minimum range for the scale. This value must be smaller than **value2Max**. **Note:** This option is only available for **obj_dualmeter02**.

Data Format Properties

Property Name	Description
valueFormat	Select from the drop down list or enter the numeric format of values displayed on the meter and in tool tips. To enter a format, use syntax from the Java **DecimalFormat** class. **Note:** To enable tool tips, select the **mouseOverFlag**.
valueTextBgColor	Select the  button and choose the color for the background used in the value/text box. **Note:** This option is only available for **obj_meter21**.
valueTextBgGradient	Select to display a gradient in the background of the text display. **Note:** This option is only available for **obj_meter21**.
valueTextColor	Select the  button and choose the color of the value text.
valueTextFont	Select the font to use for the value text.
valueTextHeight	Specify the height for the value text.
valueVisFlag	Control visibility of the value text.

Foreground Properties

Property Name	Description
fgColor	Set the color of the space within the meter. Select the ... button and choose a color from the palette.
fgEdgeColor	Set the color of the line surrounding the meter. Select the ... button and choose a color from the palette.
fgEdgeColor2	Set the color of the line surrounding the meter. Select the ... button and choose a color from the palette. **Note:** This option is only available for **obj_meter20**.
fgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **fgColor** property sets the first color in the gradient. This option is only available for **obj_meter20**.
fgGradientFlag	Select to display a gradient in the background of the meter. **Note:** When checked for **obj_meter20**, the **fgEdgeColor**, **fgEdgeColor2**, and **fgGradientColor2** properties display.

Interaction Properties

Property Name	Description
command	Assign a command to your meter. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.

drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information.
mouseOverFlag	Enter a tool tip for this meter. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nlink)**.

Needle Properties

Property Name	Description
needleColor	Select the ... button and choose the color of the needle.
needle2Color	Select the ... button and choose the color of the second needle. **Note:** This option is only available for **obj_dualmeter**.
needle2Text	Enter text for the second needle directly in this field or select the ... button to open the **Edit needle2Text** dialog.
needle2VisFlag	Select this option to display the second needle in the meter.
needleText	Enter text for the first needle directly in this field or select the ... button to open the **Edit needleText** dialog.

Object Properties

Property Name	Description
anchor	Select the ... button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools > Object List**.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Tick Marks Properties

The properties listed in this section are only available for some of the meters.

Property Name	Description
tickColor	Select the ... button and choose the color of the tick marks.
tickLabelTextColor	Select the ... button and choose the color of the tick mark label.
tickLabelTextFont	Select the font you want to use for the tick mark's labels.
tickLabelTextHeight	Sets the height (in pixels) for the tick mark's labels.
tickLabelVisFlag	Controls whether or not the labels for the tick marks display in the meter.

Link Objects

The Object Palette features five links, each offers a different way to display the connection between two linked objects. These five links share the same class name (**link_basic**) and their properties are identical. Once two objects are linked in a display, you may switch between these five link types by modifying the **linkPathType** property.

Direct

Draws a straight line between the objects it connects.

Direct Offset

Attaches to the object at a right angle, but draws a diagonal line for the remainder of the link.

Orthogonal

Draws a line at right angles between the objects it connects.

Vertical Square

Draws a line from the top/bottom side of the parent closest to the child to the right/left side of the child closest to the parent. The link line has two 90 degree angles unless the parent and child are aligned vertically, in which case the link will be straight.

Note: The link line will be automatically clipped when the top/bottom side of the parent and child nodes overlap.

Horizontal Square

Draws a line from the right/left side of the parent closest to the child to the right/left side of the child closest to the parent. The link line has two 90 degree angles unless the parent and child are aligned horizontally, in which case the link will be straight.

Note: The link line will be automatically clipped when the right/left side of the parent and child nodes overlap.

Using Interaction Properties

Commands

To assign a command to your link, right-click in the **Property Value** field of the **command** property and select **Define Command**. Commands can be set up to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold. See "Define/Execute Command" for information on how to set up commands.

Drill Down Displays

To specify a drill down display, double click on the **drillDownTarget** property. Any display (**.rtv**) file can be targeted as a drill down. Once a drill down target has been set, double-click on the link to activate the drill down. Drill down displays can be activated in the same window that contains the link or open in a separate window. This allows you to build a customizable hierarchy of displays. See "Drill Down Displays" for more information.

Tool Tips

Select the **mouseOverFlag** to enable tool tips for your link. To display a tool tip, point to the link with your mouse. The tool tip will contain information from your data attachment.

Note: The property **iconVisFlag** must be selected in order to display tool tips.

Alert Properties

Property Name	Description
valueAlertMode	**Discrete Alerts** Color of the link will change when the **value** property equals the specified **value*Alert**.

valueHighAlert -- Specify the value that the input value must be equal to for the high alert to execute.

valueHighAlertColor -- Select the  button and choose a color for the high alert.

valueLowAlert -- Specify the value that the input value must be equal to for the low alert to execute.

valueLowAlertColor -- Select the  button and choose a color for the low alert.

valueMediumAlert -- Specify the value that the input value must be equal to for the medium alert to execute.

valueMediumAlertColor -- Select the  button and choose a color for the medium alert.

valueNoAlert -- Specify the value that the input value must be equal to for the no alert to execute.

valueNoAlertColor -- Select the  button and choose a color for no alert.

Range Alerts

Color of the link will change when the **value** property is greater than the **valueHighAlarm** or **valueHighWarning** or less than the **valueLowAlarm** or **valueLowWarning**.

valueHighAlarm -- Specify the value that the input value must exceed to activate the high alarm.

valueHighAlarmColor -- Select the  button and choose a color for the high alarm.

valueHighWarning -- Specify the value that the input value must exceed to activate the high warning.

valueHighWarningColor -- Select the  button and choose a color for the high warning.

valueLowAlarm -- Specify the value that the input value must go below to activate the low alarm.

valueLowAlarmColor -- Select the  button and choose a color for the low alarm.

valueLowWarning Specify the value that the input value must go below to activate the low warning.

valueLowWarningColor -- Select the  button and choose a color for the low warning.

No Alerts

Disable all alert and alarm properties.

Arrow Properties

Property Name	Description
arrow1Color	Select the button and choose a color for the source arrow.
arrow1VisFlag	Controls the visibility of the source arrow.
arrow2Color	Select the button and choose a color for the target arrow.
arrow2VisFlag	Controls the visibility of the target arrow.

Data Properties

Property Name	Description
value	Data value used to display the value text and to evaluate any alert definitions.
valueDivisor	If specified, this divisor is applied to the value.

Data Format Properties

Property Name	Description
valueFormat	Select or enter the numeric format of values displayed on the link and in tool tips. To enter a format, use syntax from the Java **DecimalFormat** class. **Note:** To enable tool tips, select the **mouseOverFlag**.
valueTextColor	Select the button and choose the color of the value text.
valueTextFont	Select the font to use for the value text.
valueTextHeight	Specify the height for the value text.
valueTextPosX	Set x-axis position of value text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.
valueTextPosY	Set the y-axis position of value text . Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.
valueVisFlag	Control visibility of the value text. **Note:** This property is only applicable if **iconVisFlag** is selected.

Icon Properties

Property Name	Description
iconBgBorderColor	Select the ... button and choose a color for the border of the background rectangle. This property is only applicable if **iconBgBorderFlag** is selected.
iconBgBorderFlag	If selected, a border is drawn around the background rectangle.
iconBgColor	Select the ... button and choose a fill color for the background rectangle.
iconBgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **iconBgStyle** selected is 3D Rectangle.
iconBgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is white. **Note:** The **iconBgColor** property sets the first color in the gradient.
iconBgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** -- No gradient **Diagonal Edge** -- Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** -- Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** -- Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** -- Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** -- Gradient is drawn vertically from the left to the right of the object. **Vertical Center** -- Gradient is drawn vertically from the center to the left and right of the object.
iconBgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **iconBgStyle** selected is **3D Rectangle**.
iconBgRoundness	Set the arc length of the rounded corners. This property is only available if the **iconBgStyle** selected is **Round Rectangle**. **Note:** The value of **iconBgRoundness** cannot exceed half the value of the **iconWidth** or the **iconHeight**. If **iconBgRoundness** does exceed that value, then half (of the smaller of the two values) of **iconWidth** or **iconHeight** will be used instead. For example if **iconWidth** is 100 and **iconHeight** is 50, then the value of **iconBgRoundness** cannot exceed 25. If it does, then half the value of **iconHeight** (25) will be used instead.
iconBgShadowFlag	Select to display a drop shadow on the background rectangle.
iconBgStyle	Select from the following options: **Rectangle** -- Select to display a background rectangle. **3D Rectangle** -- Select to display a 3D edge on the background rectangle. If selected, use **iconBgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** -- Select to display a background rectangle with rounded edges. If selected, use **iconBgRoundness** to set the arc length of the rounded corners. **Circle** -- Select to display a background circle.
iconBgVisFlag	Select to display a background rectangle. **Note:** This property is only applicable if **iconVisFlag** is selected.

iconConstantSizeFlag	If selected, when the display is zoomed the icon size will remain constant.
iconHeight	Set height of the icon in pixels.
iconVisFlag	Control the visibility of an icon displaying the value of the link.
iconWidth	Set width of the object in pixels.

Interaction Properties

Property Name	Description
command	Assign a command to your link. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the  button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
connectEdgeFlag	If the specified **linkPathType** is Direct, then select the **connectEdgeFlag** to attach the link to the center of the side of the target object that is closest to the source object (instead of connecting to the center of the target object). **Note:** Orthogonal and Direct Offset links always attach to the center of the side closest to the source object.
drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information.
mouseOverFlag	Enter a tool tip for this link. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nlink)**. **Note:** The property **iconVisFlag** must be selected in order to display tool tips.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the  button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. **Note:** This property is only applies if **labelTextPosY** is set to **TabTop**.
labelTextColor	Select the  button and choose the color of the label text.
labelTextFont	Select the font to use for the label text.
labelTextHeight	Specify the height for the label text.
labelTextPosX	Set x-axis position of label text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.
labelTextPosY	Set y-axis position of label text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Title Top** - Position along the top line of the background rectangle. • **Tab Top** - Position tab just above the background rectangle. **Note:** Height and width of the label tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.
labelVisFlag	Control visibility of the label. **Note:** This property is only applicable if **iconVisFlag** is selected.

Link Line Properties

Property Name	Description
linkColor	Select the  button and choose a color for the link line.

linkCrossBarSpacing	Determines the positioning of the perpendicular line between the parent and child nodes. If set to **-1** (the default), the perpendicular line between the parent and child nodes is positioned in the center of the space between the parent and child nodes. If set to any other value, the perpendicular line is positioned the specified number of pixels from the parent node. For example, if value is set to 5, then the perpendicular line is placed 5 pixels from the parent node. If the value specified is greater than the space between the parent and child nodes, then the perpendicular line is drawn in the center of the available space between the parent and child. **Note:** This property is only applied when using the **Horizontal Square** or **Vertical Square** link types.
linkPathType	Defaults the name of the selected link type. Select from the following options to modify the link type: • **Direct** - Draws a straight line between the objects it connects. • **Direct Offset** - Attaches to objects at a right angle, but draws a diagonal line for the remainder of the link. • **Orthogonal** - Draws a line at right angles between the objects it connects. • **Vertical Square** - Draws a line from the top/bottom side of the parent closest to the child to the right/left side of the child closest to the parent. • **Horizontal Square** - Draws a line from the right/left side of the parent closest to the child to the right/left side of the child closest to the parent.
linkSize	Set (in pixels) the thickness of the link line.

Object Properties

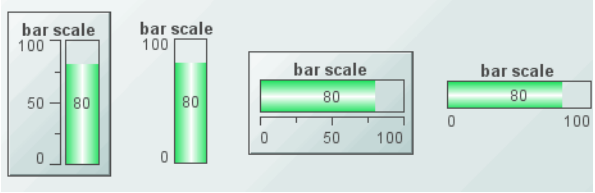
Property Name	Description
objName	Name given to facilitate object management via the Object List dialog. Select **Tools > Object List**.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. See "Style Sheets" for more information. The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the link. Enter a value between 0 and 100. A value of 0, the default, sets the link to be completely opaque. A value of 100 will render the link completely transparent.
visFlag	Control visibility of the link line.

Scale Objects

The Object Palette features several types of scales: bar, indicator, vu, pie, and bullet.

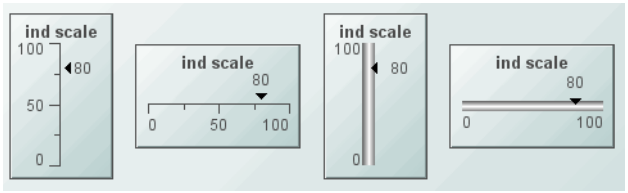
"Bar Scales"

Bar scales (class name: **obj_barscale**) display the value using the fill percent of the bar.



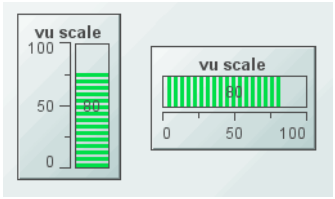
"Indicator Scales"

Indicator scales (class name: **obj_indscale**) display the value using the position of the indicator against an axis.



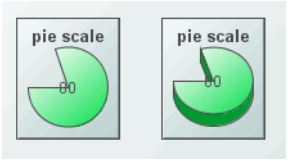
"Vu Scales"

Vu scales (class name: **obj_vuscale**) display the value using the fill percent of the bar. The bar is drawn striped instead of solid.



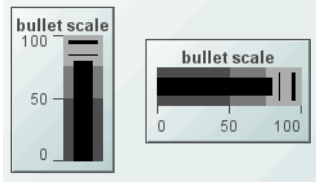
"Pie Scales"

Pie scales (class name: **obj_piescale**) display the value using the fill percent of the pie.

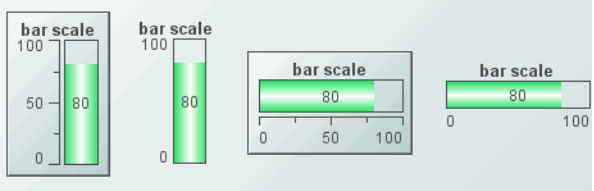


"Bullet Scales"

Bullet scales (class name: **obj_bulletscale**) display the value using the fill percent of the bar. (The bar look and functionality are based on Steven Few's bullet graph).



Bar Scales



Alert Properties

Property Name	Description
thresholdColorBarFlag	If selected, the portion of the bar that exceeds a threshold will be painted in the threshold color. This property is only used if either the **valueHighAlarmEnabledFlag** or the **valueHighWarningEnabledFlag** is selected.
thresholdLineThickness	Select **Thin**, **Medium**, or **Thick** to set the thickness of the threshold lines. This property is only used if the **thresholdLineVisFlag** is selected.
thresholdLineVisFlag	Select to draw a line across the bar at the level of each enabled threshold. This line will be the same color as the threshold color. This property is only used if the **valueHighAlarmEnabledFlag** or **valueHighWarningEnabledFlag** is selected.
valueHighAlarmEnabledFlag	Select to enable the high alarm range and the following related properties: **valueHighAlarm** - Specify the minimum value for the high alarm range. This value must be equal to or greater than **valueMin** and equal to or less than **valueMax**. **Note:** If the **valueHighWarningEnabledFlag** is selected, this value must also be greater than **valueHighWarning**. If the **valueHighAlarm** is greater than **valueMax** or less than **valueMin**, the **valueHighAlarm** will be disabled and the scale will draw without that threshold. An error will print to the console. **valueHighAlarmColor** - Specify the color for the high alarm range.
valueHighWarningEnabledFlag	Select to enable the high warning range and the following related properties: **valueHighWarning** - Specify the minimum value for the high warning range. This value must be equal to or greater than **valueMin** and equal to or less than **valueMax**. **Note:** If the **valueHighAlarmEnabledFlag** is selected, this value must also be less than **valueHighAlarm**. If the **valueHighWarning** is greater than **valueMax** or less than **valueMin**, the **valueHighWarning** will be disabled and the scale will draw without that threshold. An error will print to the console. **valueHighWarningColor** - Specify the color for the high warning range.

Axis Properties

Property Name	Description
axisColor	Select the ... button and choose from the palette to set the color of the axis line and tick marks.
axisDirection	Select the direction of the axis: • **Bottom to Top** - Vertical axis with **valueMin** at the bottom and **valueMax** at the top. • **Left to Right** - Horizontal axis with **valueMin** at the left and **valueMax** at the right. • **Top to Bottom** - Vertical axis with **valueMax** at the bottom and **valueMin** at the top. • **Right to Left** - Horizontal axis with **valueMin** at the right and **valueMax** at the left
axisMinLabelWidth	Specify the minimum width (in pixels) for the axis labels.
axisFormat	Select or enter the numeric format of values displayed on the axis. **Note:** To enter a format, use syntax from the **Java DecimalFormat** class.
axisLineThickness	Select **Thin**, **Medium**, or **Thick** to specify the thickness of the axis line and tick marks. **Note:** This property only applies if the specified **axisStyle** is any of the Classic styles or Ticks and Labels.
axisMajorDivisions	Specify the number of major divisions on the axis.
axisMinorDivisions	Specify the number of minor divisions on the axis.
axisOnTopOrRightFlag	If selected, a horizontal axis will be positioned above the scale or a vertical axis will be positioned to the right of the scale.
axisStyle	Select the axis style from the drop down menu: • **Classic with Labels** - Classic style axis with an axis line, tick marks and labels at the major divisions. • **Classic without Labels** - Classic style axis with an axis line and tick marks but without labels. • **Classic with End Labels** - Classic style axis with an axis line and tick mark. Maximum and minimum value labels will be positioned at the ends of the scale. • **Ticks and Labels** - Axis with only tick marks and labels. • **Bar with Labels** - Axis with labels at the major divisions. • **Bar without Labels** - Bar style axis without labels. • **Bar with End Labels** - Bar style axis with maximum and minimum value labels at the ends of the scale. • **Labels Only** - Show labels at the major divisions, but show no axis lines or tick marks. • **End Labels Only** - Show minimum and maximum value labels at the ends of the scale.
axisTextColor	Select the ... button and choose from the palette to set the color of the axis labels.
axisTextFont	Select a font to use the for the axis labels.
axisTextHeight	Specify a size for the axis labels.

axisVisFlag	Control visibility of the axis.
centerMinMaxAxisLabelsFlag	If selected, minimum and maximum value labels are centered on the axis tick marks. If not selected, these axis labels will be justified so they don't extend past the end of the axis. **Note:** This property only applies if the specified **axisStyle** displays labels next to tick marks.

Background Properties

Property Name	Description
bgBorderColor	Select the ... button and choose from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The **bgColor** property sets the first color in the gradient. **Note:** This property will be ignored if **bgGradientMode** is set to None.
bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.

bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the scale and the border.

Bar Properties

Property Name	Description
barBgColor	Select the ... button and choose from the palette to set the color of the bar background. **Note:** This property only used applies if **barBgOpaqueFlag** is selected.
barBgOpaqueFlag	If selected, the bar background is filled in. Otherwise, it is transparent.
barBorderColor	Select the ... button and choose from the palette to set the color of the border of the bar. **Note:** This property only applies if **barBgBorderFlag** is selected.
barBorderFlag	If selected, a border is displayed around the bar.
barColor	Select the ... button and choose from the palette to set the color of the bar.
barGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The **barColor** property sets the first color in the gradient. **Note:** This property will be ignored if **barGradientMode** is set to **None**.
barGradientMode	Display a gradient in the bar background. Select from the following options: • **None** - No gradient. • **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner. • **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners. • **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom. • **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom. • **Vertical Edge** - Gradient is drawn vertically from the left to the right. • **Vertical Center** - Gradient is drawn vertically from the center to the left and right.
barRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the selected **barStyle** is **3D Rectangle**.

barRoundness	Set the arc length of the rounded corners. **Note:** This property is only applies if the selected **barStyle** is **Round Rectangle**.
barStyle	Choose one of the following three options from the drop down menu: • **Rectangle** - Display the bar as a rectangle. • **3D Rectangle** - Display a 3D edge on the bar. • **Round Rectangle** - Display the bar with rounded edges. If selected, use the **bgRoundness** property to set the arc length of the rounded corners.

Data Properties

Property Name	Description
value	Attach your data.
valueDivisor	If specified, this divisor is applied to the **value**, **valueMin**, and **valueMax**.
valueMax	Set the maximum range for the scale. This value must be larger than **valueMin**.
valueMin	Set the minimum range for the scale. This value must be smaller than **valueMax**.

Data Format Properties

Property Name	Description
valueFormat	Select or enter the numeric format of value displayed in the value label. To enter a format, use syntax from the **Java DecimalFormat** class.
valueTextColor	Select the ... button and choose from the palette to set the color of the value label.
valueTextFont	Select the font to use for the value label.
valueTextHeight	Specify the height for the value label.
valueTextPosition	Set the position of value label. Select from the following options: • **Outside Max** - Position outside the maximum value. • **Inside Max** - Position inside the maximum value. • **Outside Current** - Position outside the current value. • **Inside Current** - Position inside the current value. • **Center** - Position at the center of the scale. • **Inside Min** - Position inside the minimum value. • **Outside Min** - Position outside the minimum value.
valueVisFlag	Control visibility of the value label.

Interaction Properties

Property Name	Description
command	Assign a command to your bar scale. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information. When you double-click on a trace in the scale, the following predefined substitutions will be set on the specified **drillDownTarget**: • **$traceNumber** - number of the trace (1 to 10) that contains the selected point • **$traceLabel** - label of selected trace • **$pointValue** - data value of point • **$pointTimestamp** - timestamp of point • **$pointLabel** - data label (if any) of point • **$pointIndex** - position of point in trace data (0 to **maxPointsPerTrace**) **Note:** When **drillDownSelectMode** is set to **Anywhere,** double-clicking anywhere on the scale will activate the specified **drillDownTarget**; however, you must double-click on a trace in the scale to set the substitutions listed above.
mouseOverText	Enter a tool tip for this scale. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nscale**). **Note:** The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

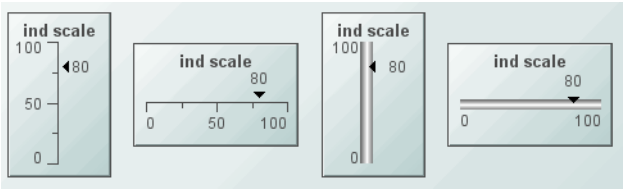
Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the  button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. **Note:** This property is only applies if **labelTextPosY** is set to **Tab Top**.
labelTextColor	Select the  button and choose from the palette to set the color of the label text.
labelTextFont	Select the font style of the label text from drop down menu.
labelTextHeight	Set the height (in pixels) of the label text.
labelTextPosX	Set x-axis position of label text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.
labelTextPosY	Set y-axis position of label text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Title Top** - Position along the top line of the background rectangle. • **Tab Top** - Position tab just above the background rectangle. **Note:** Height and width of the label tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.
labelVisFlag	Control visibility of the label.

Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Indicator Scales



Alert Properties

Property Name	Description
thresholdColorIndicatorFlag	If selected, the indicator will display the threshold color when it exceeds that threshold. This property only applies if either the **valueHighAlarmEnabledFlag** or the **valueHighWarningEnabledFlag** is selected.
thresholdLineThickness	Select **Thin**, **Medium**, or **Thick** to set the thickness of the threshold lines. This property is only used if the **thresholdLineVisFlag** is selected.
thresholdLineVisFlag	Select to draw a line across the bar at the level of each enabled threshold. This line will be the same color as the threshold color. This property is only used if the **valueHighAlarmEnabledFlag** or **valueHighWarningEnabledFlag** is selected.
valueHighAlarmEnabledFlag	Select to enable the high alarm range and the following related properties: **valueHighAlarm** -Specify the minimum value for the high alarm range. This value must be equal to or greater than **valueMin** and equal to or less than **valueMax**. **Note:** If the **valueHighWarningEnabledFlag** is selected, this value must also be greater than **valueHighWarning**. If the **valueHighAlarm** is greater than **valueMax** or less than **valueMin**, the **valueHighAlarm** will be disabled and the scale will draw without that threshold. An error will print to the console. **valueHighAlarmColor** - Specify the color for the high alarm range.
valueHighWarningEnabledFlag	Select to enable the high warning range and the following related properties: **valueHighWarning** - Specify the minimum value for the high warning range. This value must be equal to or greater than **valueMin** and equal to or less than **valueMax**. **Note:** If the **valueHighAlarmEnabledFlag** is selected, this value must also be less than **valueHighAlarm**. If the **valueHighWarning** is greater than **valueMax** or less than **valueMin**, the **valueHighWarning** will be disabled and the scale will draw without that threshold. An error will print to the console. **valueHighWarningColor** - Specify the color for the high warning range.

Axis Properties

Property Name	Description
axisColor	Select the ... button and choose from the palette to set the color of the axis line and tick marks.
axisDirection	Select the direction of the axis: • **Bottom to Top** - Vertical axis with **valueMin** at the bottom and **valueMax** at the top. • **Left to Right** - Horizontal axis with **valueMin** at the left and **valueMax** at the right. • **Top to Bottom** - Vertical axis with **valueMax** at the bottom and **valueMin** at the top. • **Right to Left** - Horizontal axis with **valueMin** at the right and **valueMax** at the left
axisMinLabelWidth	Specify the minimum width (in pixels) for the axis labels.
axisFormat	Select or enter the numeric format of values displayed on the axis. **Note:** To enter a format, use syntax from the **Java DecimalFormat** class.
axisLineThickness	Select **Thin**, **Medium**, or **Thick** to specify the thickness of the axis line and tick marks. **Note:** This property only applies if the specified **axisStyle** is any of the Classic styles or Ticks and Labels.
axisMajorDivisions	Specify the number of major divisions on the axis.
axisMinorDivisions	Specify the number of minor divisions on the axis.
axisOnTopOrRightFlag	If selected, a horizontal axis will be positioned above the scale or a vertical axis will be positioned to the right of the scale.
axisStyle	Select the axis style from the drop down menu: • **Classic with Labels** - Classic style axis with an axis line, tick marks and labels at the major divisions. • **Classic without Labels** - Classic style axis with an axis line and tick marks but without labels. • **Classic with End Labels** - Classic style axis with an axis line and tick mark. Maximum and minimum value labels will be positioned at the ends of the scale. • **Ticks and Labels** - Axis with only tick marks and labels. • **Bar with Labels** - Axis with labels at the major divisions. • **Bar without Labels** - Bar style axis without labels. • **Bar with End Labels** - Bar style axis with maximum and minimum value labels at the ends of the scale. • **Labels Only** - Show labels at the major divisions, but show no axis lines or tick marks. • **End Labels Only** - Show minimum and maximum value labels at the ends of the scale.
axisTextColor	Select the ... button and choose from the palette to set the color of the axis labels.
axisTextFont	Select a font to use the for the axis labels.
axisTextHeight	Specify a size for the axis labels.

axisVisFlag	Control visibility of the axis.
centerMinMaxAxisLabelsFlag	If selected, minimum and maximum value labels are centered on the axis tick marks. If not selected, these axis labels will be justified so they don't extend past the end of the axis. **Note:** This property only applies if the specified **axisStyle** displays labels next to tick marks.

Background Properties

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose a color from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The **bgColor** property sets the first color in the gradient. **Note:** This property will be ignored if **bgGradientMode** is set to **None**.
bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.

bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the scale and the border.

Data Properties

Property Name	Description
value	Attach your data.
valueDivisor	If specified, this divisor is applied to the **value**, **valueMin**, and **valueMax**.
valueMax	Set the maximum range for the scale. This value must be larger than **valueMin**.
valueMin	Set the minimum range for the scale. This value must be smaller than **valueMax**.

Data Format Properties

Property Name	Description
valueFormat	Select or enter the numeric format of value displayed in the value label. To enter a format, use syntax from the **Java DecimalFormat** class.
valueTextColor	Select the ... button and choose from the palette to set the color of the value label.
valueTextFont	Select the font to use for the value label.
valueTextHeight	Specify the height for the value label.
valueTextPosition	Set the position of value label. Select from the following options: • **Outside Max** - Position outside the maximum value on the axis. • **Center** - Position to the side of the indicator. • **Outside Min** - Position outside the minimum value on the axis.
valueVisFlag	Control visibility of the value label.

Indicator Properties

Property Name	Description
indicatorColor	Select the ... button and choose from the palette to set the color of the indicator.
indicatorOnAxisFlag	If selected, draw the indicator on the axis. Otherwise, it is drawn to the side of the axis.

indicatorSize	Set the size of the indicator: **Small**, **Medium**, or **Large**.
indicatorStyle	Set the style of the indicator: **Triangle**, **Circle**, or **Line**.

Interaction Properties

Property Name	Description
command	Assign a command to your indicator scale. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information. When you double-click on a trace in the scale, the following predefined substitutions will be set on the specified **drillDownTarget**: • **$traceNumber** - number of the trace (1 to 10) that contains the selected point • **$traceLabel** - label of selected trace • **$pointValue** - data value of point • **$pointTimestamp** - timestamp of point • **$pointLabel** - data label (if any) of point • **$pointIndex** - position of point in trace data (0 to **maxPointsPerTrace**) **Note:** When **drillDownSelectMode** is set to Anywhere double-clicking anywhere on the scale will activate the specified **drillDownTarget**, however you must double-click on a trace in the scale to set the substitutions listed above.
mouseOverText	Enter a tool tip for this scale. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nscale**). **Note:** The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

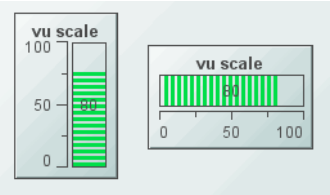
Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the ... button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. **Note:** This property is only applies if **labelTextPosY** is set to **Tab Top**.
labelTextColor	Select the ... button and choose from the palette to set the color of the label text.
labelTextFont	Select the font style of the label text from drop down menu.
labelTextHeight	Set the height (in pixels) of the label text.
labelTextPosX	Set x-axis position of label text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.
labelTextPosY	Set y-axis position of label text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Title Top** - Position along the top line of the background rectangle. • **Tab Top** - Position tab just above the background rectangle. **Note:** Height and width of the label tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle.\ • **Outside Bottom** - Position well below the background rectangle.
labelVisFlag	Control visibility of the label.

Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Vu Scales



Alert Properties

Property Name	Description
thresholdColorBarFlag	If selected, the portion of the bar that exceeds a threshold will be painted in the threshold color. This property is only used if either the **valueHighAlarmEnabledFlag** or the **valueHighWarningEnabledFlag** is selected.
thresholdLineVisFlag	Select to draw thresholds behind the active bar in paler versions of the threshold colors.
valueHighAlarmEnabledFlag	Select to enable the high alarm range and the following related properties: **valueHighAlarm** - Specify the minimum value for the high alarm range. This value must be equal to or greater than **valueMin** and equal to or less than **valueMax**. **Note:** If the **valueHighWarningEnabledFlag** is selected, this value must also be greater than **valueHighWarning**. If the **valueHighAlarm** is greater than **valueMax** or less than **valueMin**, the **valueHighAlarm** will be disabled and the scale will draw without that threshold. An error will print to the console. **valueHighAlarmColor** - Specify the color for the high alarm range.
valueHighWarningEnabledFlag	Select to enable the high warning range and the following related properties: **valueHighWarning** - Specify the minimum value for the high warning range. This value must be equal to or greater than **valueMin** and equal to or less than **valueMax**. **Note:** If the **valueHighAlarmEnabledFlag** is selected, this value must also be less than **valueHighAlarm**. If the **valueHighWarning** is greater than **valueMax** or less than **valueMin**, the **valueHighWarning** will be disabled and the scale will draw without that threshold. An error will print to the console. **valueHighWarningColor** - Specify the color for the high warning range.

Axis Properties

Property Name	Description
axisColor	Select the ... button and choose from the palette to set the color of the axis line and tick marks.
axisDirection	Select the direction of the axis: • **Bottom to Top** - Vertical axis with **valueMin** at the bottom and **valueMax** at the top. • **Left to Right** - Horizontal axis with **valueMin** at the left and **valueMax** at the right. • **Top to Bottom** - Vertical axis with **valueMax** at the bottom and **valueMin** at the top. • **Right to Left** - Horizontal axis with **valueMin** at the right and **valueMax** at the left
axisMinLabelWidth	Specify the minimum width (in pixels) for the axis labels.
axisFormat	Select or enter the numeric format of values displayed on the axis. **Note:** To enter a format, use syntax from the **Java DecimalFormat** class.
axisLineThickness	Select **Thin**, **Medium**, or **Thick** to specify the thickness of the axis line and tick marks. **Note:** This property only applies if the specified **axisStyle** is any of the Classic styles or Ticks and Labels.
axisMajorDivisions	Specify the number of major divisions on the axis.
axisMinorDivisions	Specify the number of minor divisions on the axis.
axisOnTopOrRightFlag	If selected, a horizontal axis will be positioned above the scale or a vertical axis will be positioned to the right of the scale.
axisStyle	Select the axis style from the drop down menu: • **Classic with Labels** - Classic style axis with an axis line, tick marks and labels at the major divisions. • **Classic without Labels** - Classic style axis with an axis line and tick marks but without labels. • **Classic with End Labels** - Classic style axis with an axis line and tick mark. Maximum and minimum value labels will be positioned at the ends of the scale. • **Ticks and Labels** - Axis with only tick marks and labels. • **Bar with Labels** - Axis with labels at the major divisions. • **Bar without Labels** - Bar style axis without labels. • **Bar with End Labels** - Bar style axis with maximum and minimum value labels at the ends of the scale. • **Labels Only** - Show labels at the major divisions, but show no axis lines or tick marks. • **End Labels Only** - Show minimum and maximum value labels at the ends of the scale.
axisTextColor	Select the ... button and choose from the palette to set the color of the axis labels.
axisTextFont	Select a font to use the for the axis labels.
axisTextHeight	Specify a size for the axis labels.

axisVisFlag	Control visibility of the axis.
centerMinMaxAxisLabelsFlag	If selected, minimum and maximum value labels are centered on the axis tick marks. If not selected, these axis labels will be justified so they don't extend past the end of the axis. **Note:** This property only applies if the specified **axisStyle** displays labels next to tick marks.

Background Properties

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose a color from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The **bgColor** property sets the first color in the gradient. **Note:** This property will be ignored if **bgGradientMode** is set to None.
bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.

bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the scale and the border.

Bar Properties

Property Name	Description
barBgColor	Select the ... button and choose from the palette to set the color of the bar background. **Note:** This property only used applies if **barBgOpaqueFlag** is selected.
barBgOpaqueFlag	If selected, the bar background is filled in. Otherwise, it is transparent.
barBorderColor	Select the ... button and choose from the palette to set the color of the border of the bar. **Note:** This property only applies if **barBgBorderFlag** is selected.
barBorderFlag	If selected, a border is displayed around the bar.
barColor	Select the ... button and choose from the palette to set the color of the bar.
barInsets	Specify the insets (in pixels) between the edge of the bar and the bar lines.

Data Properties

Property Name	Description
value	Attach your data.
valueDivisor	If specified, this divisor is applied to the value, **valueMin** and **valueMax**.
valueMax	Set the maximum range for the scale. This value must be larger than **valueMin**.
valueMin	Set the minimum range for the scale. This value must be smaller than **valueMax**.

Data Format Properties

Property Name	Description
valueFormat	Select or enter the numeric format of value displayed in the value label. To enter a format, use syntax from the **Java DecimalFormat** class.
valueTextColor	Select the ... button and choose from the palette to set the color of the value label.
valueTextFont	Select the font to use for the value label.

valueTextHeight	Specify the height for the value label.
valueTextPosition	Set the position of value label. Select from the following options: • **Outside Max** - Position outside the maximum value. • **Inside Max** - Position inside the maximum value. • **Outside Current** - Position outside the current value. • **Inside Current** - Position inside the current value. • **Center** - Position at the center of the scale. • **Inside Min** - Position inside the minimum value. • **Outside Min** - Position outside the minimum value.
valueVisFlag	Control visibility of the value label.

Interaction Properties

Property Name	Description
command	Assign a command to your **vu** scale. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.

drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information. When you double-click on a trace in the scale, the following predefined substitutions will be set on the specified **drillDownTarget**: • **$traceNumber** - number of the trace (1 to 10) that contains the selected point • **$traceLabel** - label of selected trace • **$pointValue** - data value of point • **$pointTimestamp** - timestamp of point • **$pointLabel** - data label (if any) of point • **$pointIndex** - position of point in trace data (0 to **maxPointsPerTrace**) **Note:** When **drillDownSelectMode** is set to **Anywhere** double-clicking anywhere on the scale will activate the specified **drillDownTarget**, however you must double-click on a trace in the scale to set the substitutions listed above.
mouseOverText	Enter a tool tip for this scale. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nscale**). **Note:** The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the  button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. **Note:** This property is only applies if **labelTextPosY** is set to **Tab Top**.
labelTextColor	Select the  button and choose from the palette to set the color of the label text.
labelTextFont	Select the font style of the label text from drop down menu.
labelTextHeight	Set the height (in pixels) of the label text.
labelTextPosX	Set x-axis position of label text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.

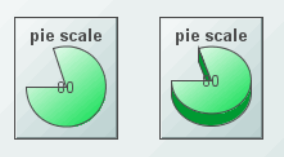
labelTextPosY	Set y-axis position of label text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Title Top** - Position along the top line of the background rectangle. • **Tab Top** - Position tab just above the background rectangle. **Note:** Height and width of the label tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.
labelVisFlag	Control visibility of the label.

Object Properties

Property Name	Description
anchor	Select the ... button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.

objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Pie Scales



Alert Properties

Property Name	Description
thresholdColorPieFlag	If selected, a portion of the pie will display the threshold color when it exceeds that threshold. This property only applies if either the **valueHighAlarmEnabledFlag** or the **valueHighWarningEnabledFlag** is selected.
thresholdLineThickness	Select **Thin**, **Medium**, or **Thick** to set the thickness of the threshold lines. This property is only used if the **thresholdLineVisFlag** is selected.
thresholdLineVisFlag	Select to draw a line across the bar at the level of each enabled threshold. This line will be the same color as the threshold color. This property is only used if the **valueHighAlarmEnabledFlag** or **valueHighWarningEnabledFlag** is selected.

valueHighAlarmEnabledFlag	Select to enable the high alarm range and the following related properties: **valueHighAlarm** - Specify the minimum value for the high alarm range. This value must be equal to or greater than **valueMin** and equal to or less than **valueMax**. **Note:** If the **valueHighWarningEnabledFlag** is selected, this value must also be greater than **valueHighWarning**. If the **valueHighAlarm** is greater than **valueMax** or less than **valueMin**, the **valueHighAlarm** will be disabled and the scale will draw without that threshold. An error will print to the console. **valueHighAlarmColor** - Specify the color for the high alarm range.
valueHighWarningEnabledFlag	Select to enable the high warning range and the following related properties: **valueHighWarning** - Specify the minimum value for the high warning range. This value must be equal to or greater than **valueMin** and equal to or less than **valueMax**. **Note:** If the **valueHighAlarmEnabledFlag** is selected, this value must also be less than **valueHighAlarm**. If the **valueHighWarning** is greater than **valueMax** or less than **valueMin**, the **valueHighWarning** will be disabled and the scale will draw without that threshold. An error will print to the console. **valueHighWarningColor** - Specify the color for the high warning range.

Background Properties

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose a color from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The **bgColor** property sets the first color in the gradient. **Note:** This property will be ignored if **bgGradientMode** is set to **None**.

bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the scale and the border.

Data Properties

Property Name	Description
value	Attach your data.
valueDivisor	If specified, this divisor is applied to the **value**, **valueMin**, and **valueMax**.
valueMax	Set the maximum range for the scale. This value must be larger than **valueMin**.
valueMin	Set the minimum range for the scale. This value must be smaller than **valueMax**.

Data Format Properties

Property Name	Description
valueFormat	Select or enter the numeric format of value displayed in the value label. To enter a format, use syntax from the **Java DecimalFormat** class.
valueTextColor	Select the ... button and choose from the palette to set the color of the value label.
valueTextFont	Select the font to use for the value label.
valueTextHeight	Specify the height for the value label.
valueTextPosition	Set the position of value label. Select from the following options: • **Outside Max** - Position outside the maximum value. • **Inside Max** - Position inside the maximum value. • **Outside Current** - Position outside the current value. • **Inside Current** - Position inside the current value. • **Center** - Position at the center of the scale. • **Inside Min** - Position inside the minimum value. • **Outside Min** - Position outside the minimum value.
valueVisFlag	Control visibility of the value label.

Interaction Properties

Property Name	Description
command	Assign a command to your pie scale. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.

drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See "Drill Down Displays" for information. When you double-click on a trace in the scale, the following predefined substitutions will be set on the specified **drillDownTarget**: • **$traceNumber** - number of the trace (1 to 10) that contains the selected point • **$traceLabel** - label of selected trace • **$pointValue** - data value of point • **$pointTimestamp** - timestamp of point • **$pointLabel** - data label (if any) of point • **$pointIndex** - position of point in trace data (0 to **maxPointsPerTrace**) **Note:** When **drillDownSelectMode** is set to **Anywhere** double-clicking anywhere on the scale will activate the specified **drillDownTarget**, however you must double-click on a trace in the scale to set the substitutions listed above.
mouseOverText	Enter a tool tip for this scale. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nscale**). **Note:** The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the  button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. **Note:** This property is only applies if **labelTextPosY** is set to **Tab Top**.
labelTextColor	Select the  button and choose from the palette to set the color of the label text.
labelTextFont	Select the font style of the label text from drop down menu.
labelTextHeight	Set the height (in pixels) of the label text.
labelTextPosX	Set x-axis position of label text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.

labelTextPosY	Set y-axis position of label text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Title Top** - Position along the top line of the background rectangle. • **Tab Top** - Position tab just above the background rectangle. **Note:** Height and width of the label tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.
labelVisFlag	Control visibility of the label.

Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.

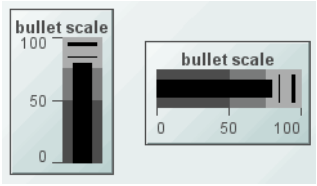
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Pie Properties

Property Name	Description
clockWiseFlag	If selected, the pie is drawn from the start angle in a clockwise direction. Otherwise, it is drawn counter-clockwise.
pie3dThickness	Specify the thickness of the 3D edge of the pie. If the value is **0**, the pie will be drawn 2D.
pieBgColor	Select the  button and choose from the palette to set the color of the background color of the pie. **Note:** This property only applies if **pieBgOpaqueFlag** is selected.
pieBgOpaqueFlag	If selected, the pie background is filled in. Otherwise, it is transparent.
pieBorderColor	Select the border color for the pie. **Note:** This property only applies if **pieBorderFlag** is selected.
pieBorderFlag	If selected, a border is displayed around the edge of the pie.
pieColor	Select the  button and choose from the palette to set the color of the pie.
pieGradientColor2	Select the  button and choose from the palette to set the second color in the gradient. The **pieColor** property sets the first color in the gradient. **Note:** This property will be ignored if **pieGradientMode** is set to **None**.
pieGradientMode	Display a gradient in the pie. Select from the following options: • **None** - No gradient. • **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner. • **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners. • **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom. • **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom. • **Vertical Edge** - Gradient is drawn vertically from the left to the right. • **Vertical Center** - Gradient is drawn vertically from the center to the left and right.

pieRaisedFlag	Reverses the direction of the gradient.
startAngle	Set the start angle of the pie. Start angles range from **0** to **360**, with **0** pointing straight to the right and increasing clockwise around the circle. Default is **180** (pointing straight to the left).

Bullet Scales



Alert Properties

Property Name	Description
compValue1	Set the value of the first comparison line. This value must be equal to or greater than **valueMin** and equal to or less than **valueMax**. If **compValue1** is greater than **valueMax** or less than **valueMin**, that threshold will be disabled and the scale will draw without that threshold. An error will print to the console. This property is only used if the **compValue1EnabledFlag** is selected.
compValue1EnabledFlag	Enable the first comparison line.
compValue1LineColor	Set the color of the first comparison line.
compValue1LineThickness	Select **Thin**, **Medium**, or **Thick** to set the thickness of the first comparison line.
compValue2	Set the value of the second comparison line. This value must be equal to or greater than **valueMin** and equal to or less than **valueMax**. If **compValue2** is greater than **valueMax** or less than **valueMin**, that threshold will be disabled and the scale will draw without that threshold. An error will print to the console. This property is only used if the **compValue2EnabledFlag** is selected.
compValue2EnabledFlag	Enable the second comparison line.
compValue2LineColor	Set the color of the second comparison line.
compValue2LineThickness	Select **Thin**, **Medium**, or **Thick** to set the thickness of the second comparison line.
valueHighAlarmEnabledFlag	Select to enable the high alarm range and the following related properties: **valueHighAlarm** - Specify the minimum value for the high alarm range. This value must be equal to or greater than **valueMin** and equal to or less than **valueMax**. **Note:** If the **valueHighWarningEnabledFlag** is selected, this value must also be greater than **valueHighWarning**. If the **valueHighAlarm** is greater than **valueMax** or less than **valueMin**, the **valueHighAlarm** will be disabled and the scale will draw without that threshold. An error will print to the console. **valueHighAlarmColor** - Specify the color for the high alarm range.

valueHighWarningEnabledFlag	Select to enable the high warning range and the following related properties: **valueHighWarning** - Specify the minimum value for the high warning range. This value must be equal to or greater than **valueMin** and equal to or less than **valueMax**. **Note:** If the **valueHighAlarmEnabledFlag** is selected, this value must also be less than **valueHighAlarm**. If the **valueHighWarning** is greater than **valueMax** or less than **valueMin**, the **valueHighWarning** will be disabled and the scale will draw without that threshold. An error will print to the console. **valueHighWarningColor** - Specify the color for the high warning range.
valueNoAlarmColor	Specify the color for the no alarm range.

Axis Properties

Property Name	Description
axisColor	Select the ... button and choose from the palette to set the color of the axis line and tick marks.
axisDirection	Select the direction of the axis: • **Bottom to Top** - Vertical axis with **valueMin** at the bottom and **valueMax** at the top. • **Left to Right** - Horizontal axis with **valueMin** at the left and **valueMax** at the right. • **Top to Bottom** - Vertical axis with **valueMax** at the bottom and **valueMin** at the top. • **Right to Left** - Horizontal axis with **valueMin** at the right and **valueMax** at the left
axisMinLabelWidth	Specify the minimum width (in pixels) for the axis labels.
axisFormat	Select or enter the numeric format of values displayed on the axis. **Note:** To enter a format, use syntax from the **Java DecimalFormat** class.
axisLineThickness	Select **Thin**, **Medium**, or **Thick** to specify the thickness of the axis line and tick marks. **Note:** This property only applies if the specified **axisStyle** is any of the Classic styles or Ticks and Labels.
axisMajorDivisions	Specify the number of major divisions on the axis.
axisMinorDivisions	Specify the number of minor divisions on the axis.
axisOnTopOrRightFlag	If selected, a horizontal axis will be positioned above the scale or a vertical axis will be positioned to the right of the scale.

axisStyle	Select the axis style from the drop down menu: • **Classic with Labels** - Classic style axis with an axis line, tick marks and labels at the major divisions. • **Classic without Labels** - Classic style axis with an axis line and tick marks but without labels. • **Classic with End Labels** - Classic style axis with an axis line and tick mark. Maximum and minimum value labels will be positioned at the ends of the scale. • **Ticks and Labels** - Axis with only tick marks and labels. • **Bar with Labels** - Axis with labels at the major divisions. • **Bar without Labels** - Bar style axis without labels. • **Bar with End Labels** - Bar style axis with maximum and minimum value labels at the ends of the scale. • **Labels Only** - Show labels at the major divisions, but show no axis lines or tick marks. • **End Labels Only** - Show minimum and maximum value labels at the ends of the scale.
axisTextColor	Select the ... button and choose from the palette to set the color of the axis labels.
axisTextFont	Select a font to use the for the axis labels.
axisTextHeight	Specify a size for the axis labels.
axisVisFlag	Control visibility of the axis.
centerMinMaxAxisLabelsFlag	If selected, minimum and maximum value labels are centered on the axis tick marks. If not selected, these axis labels will be justified so they don't extend past the end of the axis. **Note:** This property only applies if the specified **axisStyle** displays labels next to tick marks.

Background Properties

Specify how the background is displayed in your bullet scale.

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose a color from the palette to set the background color.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the bgStyle selected is 3D Rectangle.
bgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The first color in each gradient is determined by **value*Color** properties specified in the **Alert Properties** section. **Note:** This property will be ignored if **bgGradientMode** is set to **None**.

bgGradientMode	Display a separate gradient in each of the three ranges of the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the scale and the border.

Bar Properties

Property Name	Description
barBgBorderColor	Select the ... button and choose from the palette to set the color of the bar background. **Note:** This property only applies if **barBgBorderFlag** is selected.
barBgBorderFlag	If selected, border is displayed around the bar background.
barBgGradientColor2	Select the ... button and choose from the palette to set the second color in the gradient. The **barColor** property sets the first color in the gradient. **Note:** This property will be ignored if **barBgGradientMode** is set to **None**.

barBgGradientMode	Display a gradient in the bar background. Select from the following options: • **None** - No gradient. • **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner. • **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners. • **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom. • **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom. • **Vertical Edge** - Gradient is drawn vertically from the left to the right. • **Vertical Center** - Gradient is drawn vertically from the center to the left and right.
barBgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the selected **barStyle** is **3D Rectangle**.
barColor	Select the  button and choose from the palette to set the color of the bar.
barRoundness	Set the arc length of the rounded corners. **Note:** This property is only applies if the selected **barStyle** is **Round Rectangle**.
barStyle	Choose one of the following three options from the drop down menu: • **Rectangle** - Display the bar as a rectangle. • **3D Rectangle** - Display a 3D edge on the bar. • **Round Rectangle** - Display the bar with rounded edges. If selected, use the **bgRoundness** property to set the arc length of the rounded corners.
barInsets	Specify the insets (in pixels) between the edge of the bar and the bar lines.

Data Properties

Specify how data is displayed in your scale.

Property Name	Description
value	Attach your data.
valueDivisor	If specified, this divisor is applied to the value, **valueMin** and **valueMax**.
valueMax	Set the maximum range for the scale. This value must be larger than **valueMin**.
valueMin	Set the minimum range for the scale. This value must be smaller than **valueMax**.

Data Format Properties

Specify data format in your scale.

Property Name	Description
valueFormat	Select or enter the numeric format of value displayed in the value label. To enter a format, use syntax from the **Java DecimalFormat** class.
valueTextColor	Select the ... button and choose from the palette to set the color of the value label.
valueTextFont	Select the font to use for the value label.
valueTextHeight	Specify the height for the value label.
valueTextPosition	Set the position of value label. Select from the following options: • **Outside Max** - Position outside the maximum value. • **Inside Max** - Position inside the maximum value. • **Outside Current** - Position outside the current value. • **Inside Current** - Position inside the current value. • **Center** - Position at the center of the scale. • **Inside Min** - Position inside the minimum value. • **Outside Min** - Position outside the minimum value.
valueVisFlag	Control visibility of the value label.

Interaction Properties

Property Name	Description
command	Assign a command to your bullet scale. See "Define/Execute Command" for information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.

drillDownTarget	Name of display (**.rtv**) file targeted as a drill down. See **Building Displays>Drill Down Displays** for information. When you double-click on a trace in the scale, the following predefined substitutions will be set on the specified **drillDownTarget**: • **$traceNumber** - number of the trace (1 to 10) that contains the selected point • **$traceLabel** - label of selected trace • **$pointValue** - data value of point • **$pointTimestamp** - timestamp of point • **$pointLabel** - data label (if any) of point • **$pointIndex** - position of point in trace data (0 to **maxPointsPerTrace**) **Note:** When **drillDownSelectMode** is set to **Anywhere** double-clicking anywhere on the scale will activate the specified **drillDownTarget**, however you must double-click on a trace in the scale to set the substitutions listed above.
mouseOverText	Enter a tool tip for this scale. To display the tool tip, move your mouse over the object. To enter a multi-line tool tip, use **\n** to delimit the lines (e.g. **my\nscale**). **Note:** The object must be visible (i.e. **visFlag** property is selected), in order for the tool tip to be visible.

Label Properties

Property Name	Description
label	Edit label text directly in the **Property Value** field or select the  button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. **Note:** This property is only applies if **labelTextPosY** is set to **Tab Top**.
labelTextColor	Select the  button and choose from the palette to set the color of the label text.
labelTextFont	Select the font style of the label text from drop down menu.
labelTextHeight	Set the height (in pixels) of the label text.
labelTextPosX	Set x-axis position of label text. Select from the following options: • **Left** - Position outside the left side of the background rectangle. • **Inside Left** - Position inside the left side of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Right** - Position inside the right side of the background rectangle. • **Right** - Position outside the right side of the background rectangle.

labelTextPosY	Set y-axis position of label text. Select from the following options: • **Outside Top** - Position well above the background rectangle. • **Top** - Position just above the background rectangle. • **Title Top** - Position along the top line of the background rectangle. • **Tab Top** - Position tab just above the background rectangle. **Note:** Height and width of the label tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. • **Inside Top** - Position inside the top of the background rectangle. • **Center** - Position in the center of the background rectangle. • **Inside Bottom** - Position inside the bottom of the background rectangle. • **Bottom** - Position just below the background rectangle. • **Outside Bottom** - Position well below the background rectangle.
labelVisFlag	Control visibility of the label.

Object Properties

Property Name	Description
anchor	Select the ... button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.

objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with rtv-.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

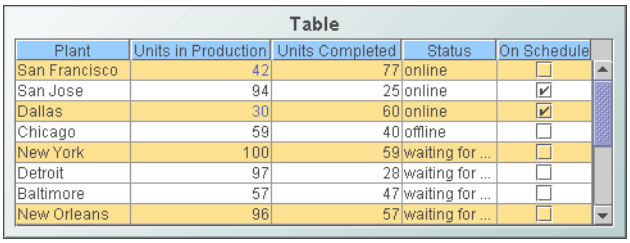
Table Objects

The Object Palette features three types of table objects--standard table, rotated table and object grid--that offer the ability to display large amounts of data. While most objects are driven by a single variable, table objects are driven by one or two dimensional arrays of data. Table objects are designed to display all the information returned by a tabular element in your data attachment. The necessary number of columns and rows are automatically created. Preset substitutions are created based on your data attachment as well, which are passed into drill down displays. See "Drill Down Substitutions" for more information.

Note: The **Table with Background** and **Table with Row Labels** objects are standard tables with some of the object properties set by default.

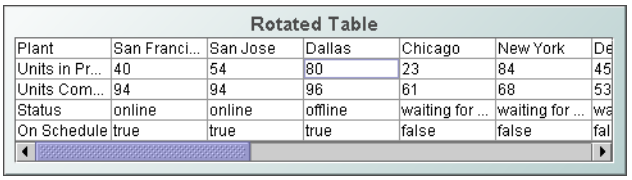
"Standard Table"

The standard table (class name: **obj_table02**) displays the data returned in a series of rows and columns, with the row names and column headers defined by the data attachment.



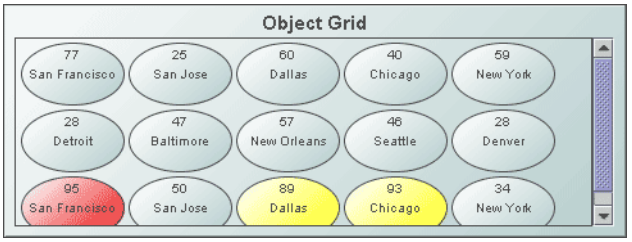
"Rotated Table"

The rotated table (class name: **obj_table03**) is a standard table in which the rows and columns have been flipped. Row names are assigned the name of the column displayed in that row.



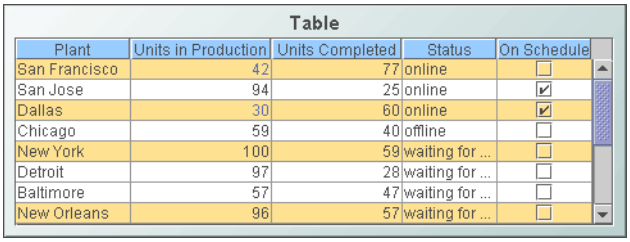
"Object Grid"

The object grid (class name: **obj_objectgrid**) displays returned data as multiple objects, called icon objects, in a scrollable grid. Each icon object in the grid is analogous to a row of data in the standard table. Icon objects display the data using graphic attributes (e.g., color, dimension, orientation, etc.).



Standard Table

Display your data in a standard table (class name: **obj_table02**) by attaching it to the **valueTable** property. In a standard table, data returned is displayed in a series of rows and columns. Row names and column headers depend on your data attachment. If the table is not large enough to display all of the data, scroll bars are automatically added.



Alert Properties

Use the value of individual columns, rows, and cells to set font color, background color, row visibility, or replace a cell value with an image.

Property Name	Description
filterProperties	Double-click on **filterProperties**, or select the ... button to open the **Filter Properties** dialog. See "Table Filters" for more information.

Background Properties

Specify how the background is displayed in a table.

Property Name	Description
bgBorderColor	Select the ... button and choose from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.

bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose from the palette to set the background color of the table.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **bgColor** property sets the first color in the gradient.
bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle. Selecting this check box also enables additional background properties.
borderPixels	Set the amount of space (in pixels) between the table and the border.
tableBgColor	Select the ... button and choose from the palette to set the color of empty space in the table area.

Cell Properties

Property Name	Description
cellBgColor	Select the ... button and choose from the palette to set the background color of the cells.
cellBgStripeContrast	Set the contrast level for alternating row colors. **Note:** This property is only applicable if **cellBgStripedFlag** is selected.
cellBgStripedFlag	Select to display alternating row colors.
cellTextColor	Select the ... button and choose from the palette to set the color of the cell text.
cellTextFont	Select a font for the cell text.
cellTextSize	Specify (in pixels) the size of the cell text or enter **-1** to use the default size.

Column Properties

Property Name	Description
autoResizeFlag	If the **autoResizeFlag** check box is selected, table columns are automatically resized so that at least a portion of each column is visible within the current table width.
columnAlignment	Click the ellipsis button ... in the **columnAlignment** property to display the **Column Alignment Properties** window, which allows you to specify the alignment of a column. Select the alignment from the drop down menu in the **Alignment** column: **Default**, **Left**, **Center**, or **Right**. The default setting depends on the type of column.
columnFormat	Click the ellipsis button ... in the **columnFormat** property to display the "Column Format Properties" window, which allows you to specify numerical or date formats for columns. Select the format from the drop down menu in the **Column Format** column.

columnFormatDate	Click the ellipsis button ... in the **columnFormatDate** property to open the **Edit columnFormatDate** window, which allows you to specify the format for all time and date columns in the table that do not have the **columnFormat** property specified.
	As with any object property, an rtview stylesheet entry can be used to set **columnFormatDate** globally on all obj_table02 instances. For example:
	`obj_table02 {`
	`columnFormatDate: "yyyy-MMM-dd HH:mm:ss"`
	`}`
	By default the **columnFormatDate** property's value is blank.

columnProperties	Click the ellipsis button ... in the **columnProperties** property to display the **Column Width Properties** window, in which you can specify the width (in pixels) of the columns in the table.

columnsToHide	Click the ellipsis button  in the **columnsToHide** property to display the **Columns to Hide** window, in which you can select the check box under **Hide Column** for the columns in the table that you would like to hide.
indexColumns	Click the ellipsis button  in the **indexColumns** property to display the **IndexColumns** window, in which you can specify the name of one or more columns that contain data that uniquely identify a table row. Once defined, any user-selected (highlighted) rows in the table will remain highlighted even if the order or number of rows in the table changes after an update to the reference table.

Column Header Properties

Property Name	Description
columnHeaderBgColor	Select the  button and choose from the palette to set the background color of the column header.
columnHeaderTextColor	Select the  button and choose from the palette to set the color of the column header text.
columnHeaderTextFont	Select a font for the column header.
columnHeaderTextSize	Specify (in pixels) the size of the cell header or enter **-1** to use the default size.

Data Properties

Property Name	Description
insertNewRowsAtTopFlag	When this check box is selected, any new rows added to the source table automatically display at the top of the table upon update.
insertNewRowsFlag	When this check box is selected, new rows added to the source table are automatically added to the table.
maxNumberOfRows	This property defines the maximum number of rows that will display in the table.
rowLabelMode	When set to **0**, row names will not appear in the table.
valueTable	Right-click in this property and select **Attach To Data > "Source"** (**XML**, for example) to select the table from which you want to pull in data. See "Attach to Data" for more information.

Data Format Properties

Property Name	Description
removeLineBreaksFlag	Use this property to remove line breaks within text in table cells. This option does not remove line breaks from drill-down values. For example, if your **drillDownColumnSubs** property targets a substitution from a cell that contains a line break, the drill-down value will contain the line break regardless of the **removeLineBreaksFlag** property setting. **Note:** In the Thin Client, the drill-down value will contain the line break regardless of the **removeLineBreaksFlag** property setting.

Data Label Properties

Property Name	Description
columnDisplayNames	Click the ellipsis button  in the **columnDisplayNames** property to display the **Column Display Name Properties** window, which allows you to define an alternate name for the columns.
textForNaN	Click the ellipsis button  in the **textForNaN** property to display the **Edit textForNaN** window, which allows you to define an alternate text string to be used when **NaN** displays as a value in the table.

Grid Properties

Property Name	Description
gridHorizontalVisFlag	Select this property's check box to display horizontal lines in the table.
gridVerticalVisFlag	Select this property's check box to display vertical lines in the table.

Historian Properties

The cache persistence feature is typically used instead of the following properties to configure the historian. See the **-persistCaches:true** command line option in "Command Line Options: Historian" for more information.

Property Name	Description
historyTableName	Specify name of table in your history database in which to store tabular data. See "Configuring the Historian" for information.
historyTableRowNameFlag	If selected, data from the row name field will be stored in the first column of the table specified in **historyTableName**.

Interaction Properties

Property Name	Description
clearSelection	Use the **clearSelection** property to enable a user to clear multiple selected rows in a table. Typically, the table's **clearSelection** property is attached to a local variable and the local variable is set by a control object on the same display. For example, to add a "Clear Selection" button to a display containing a table that supports multiple row selection, you could configure as follows: **1.** Add a local variable named **$clearSelection** with an initial value of 0. **2.** Create a button object, with these property values: **label : "Clear Selection"** **valueToSet : 1** **varToSet : <attached to the $clearSelection variable>** **3.** Set the table object's properties: **clearSelection : <attached to the $clearSelection variable>** **4.** If the table object has a drilldown command or a **drillDownTarget** configured, add this to its **Drill Down Substitutions**: **$clearSelection : 0** (This step is required to reset the value of the **$clearSelection** variable to zero when the user selects a row in the table.) When the property is set to a value of 1, all rows in the table are deselected. When the property is set to a value of 2, all rows are deselected and either the table's action or its **drillDownTarget** is invoked.
columnResizeEnabledFlag	If the **columnResizeEnabledFlag** is selected, table columns can be resized by dragging the vertical separators between the column headers. The **columnResizeEnabledFlag** cannot be toggled in the Display Server.
command	Double-click on **command** or click the ellipsis button  to display the Define System Command window. See "Define System Command" for more information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.

commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the ... button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownColumnSubs	Double-click on **drillDownColumnSubs** or click the ellipsis button ... to customize which substitutions will be passed into drill down displays. See "Drill Down Column Substitutions Dialog" and "Drill Down Displays" for more information.
drillDownSelectMode	Use **drillDownSelectMode** to control how a drill down display is activated. The standard table (class name: **obj_table02**) supports three modes for **drillDownSelectMode**: • Set to **Anywhere** (**0**) to activate a drill down display by double-clicking anywhere on the table. • Set to **Element Only** (**1**) to enable a drill down display only when you double-click on a cell in a table. Note that when set to **Element Only**, a drill down display can not be activated by double-clicking in the row label column, since this column does not correspond to an actual data column in the tabular data. • Set to **Element Only + Auto Update** (**2**) to enable a drill down display (**Element Only** mode) and execute its command automatically, without a user click, when the row selection changes after a data update. This mode requires that the table's **command** property be configured to perform a drill down to the current display in the current window, and the table's **indexColumns** property be set to a non-empty string. This mode is useful in cases where the substitutions set by the table's drill down command need to be updated if some of the selected rows in the table are removed by a subsequent data update.
drillDownTarget	Double-click on the **drillDownTarget** property to specify a drill down target for the table. Based on your data attachment, preset substitutions are created automatically and passed into your drill down display.
editDataEnabledFlag	To enable editing, you must set a variable and attach the table to data. The **editDataEnabledFlag** allows you to control whether a table can be edited. In the **Object Properties** window when you select the **editDataEnabledFlag**, the **editDataLocalVarName** property will appear. Attach data to the valueTable property to populate the table with initial data in order to enable editing. The table will not display data updates while in Edit mode, however once you save or cancel table edits current data for the valueTable attachment will be displayed.
editDataLocalVarName	This field works in conjunction with **editDataEnabledFlag** and only displays when **editDataEnabledFlag** is enabled. Select the ... button to open the **Edit editDataLocalVarName** dialog and enter the name of the variable you would like to update with changes made to the editable table. This will allow you to identify the table in the Custom Message Handler as well as attach other objects in this display to your edited data. Note: The property **editDataLocalVarName** cannot be changed while the table is in Edit mode.
menuItemGroup	Use the **menuItemGroup** and **rightClickActionFlag** properties to extend RTView context menu items. For details, see "Extending the Context Menu".
multiSelectFlag	Use the **multiSelectFlag** property to enable the selection of multiple rows. When the user selects multiple rows and drills down, the drill down substitution values contain a semi colon delimited list of values, one value for each row that can be used with most data sources in the **Filter Value** field of the **Attach To Data** dialog.

rightClickActionFlag	Use the **rightClickActionFlag** property so that a right-click by the user executes actions (a drill down or execute command) normally performed only with a left-click, before the right-click popup menu is shown. This is useful when you have a command string configured to set substitutions in the current window, and the drill down target is configured to drill down using those substitutions. In that case, the command is executed when you left-click and when you right-click so the substitutions are guaranteed to be set before you select **Drill Down** from the popup menu. This option should not be used if the left click is configured to drill down to another display. This property is ignored in the Thin Client on iOS Safari (iPad/iPhone).
rowHighlightEnabledFlag	Selecting this check box allows an entire row to be highlighted when selected, instead of just a single focused cell. The highlighting of the focused row takes precedence over any filter colors that may be applied to a row.
scrollToSelectionFlag	If this property is checked, the table will scroll if necessary to keep the first (topmost) selected row visible after a sort or a data update.
scrollbarMode	Select **Never**, **As Needed**, or **Always** from the **scrollbarMode** property to set the behavior of the scroll bar in the table. **Note:** The thin client ignores the setting in this property and always behaves as though it is set to **As Needed**.
tabIndex	The **tabIndex** property allows you to define the order in which table and control objects will receive focus when navigated from your keyboard. Initial focus is given to the object with the smallest **tabIndex** value, from there the tabbing order proceeds in ascending order. If multiple objects share the same **tabIndex** value, then initial focus and tabbing order are determined by the alpha-numeric order of the table names. Tables with a **tabIndex** value of 0 are last in the tabbing order. The **tabIndex** property does not apply to tables in the Display Server or to objects that are disabled, invisible, or have a value of less than 0.

Label Properties

Property Name	Description
label	Edit label text directly in the Property Value field or select the  button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. This property is only applies if **labelTextAlignY** is set to **TabTop**.
labelTextAlignX	Select x-axis position of label text from the drop down menu. You can select from **Left**, **Center**, or **Right**.
labelTextAlignY	Select y-axis position of label text from the drop down menu. **Outside Top** - Position label well above the outside border of the table. **Top** - Position label just above the outside border of the table. **Title Top** - Position label in the top line/border of the table. **Tab Top** - Position label tab just above the table. Height and width of the tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. **Inside Top** - Position label inside the top border of the table.
labelTextColor	Select the  button and choose a color from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop down menu.
labelTextHeight	Set the height of the label text in pixels.

Object Properties

Property Name	Description
anchor	Select the ... button to display the "Anchor Property Window", which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window".
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with rtv-.
visFlag	Control visibility of the object.

Row Header Properties

Property Name	Description
rowHeaderBgColor	Select the ... button and choose a color from the palette to set the row header background color.

rowHeaderEnabledFlag	Select the **rowHeaderEnabledFlag** property to lock the first column in a table. The **rowHeaderEnabledFlag** cannot be toggled in the Display Server. When enabled, you can set the appearance of the row header column using the **rowHeaderBgColor**, **rowHeaderTextColor**, **rowHeaderTextFont**, and **rowHeaderTextSize** properties.
rowHeaderFilterColorsEnabled Flag	The **rowHeaderFilterColorsEnabledFlag** property sets whether the row filter colors are applied to the row header. Row visibility filters are applied to the row header regardless of the **rowHeaderFilterColorsEnabledFlag** setting. Resize the row header column by clicking the ellipsis button  in the **columnProperties** property. The row header column cannot be hidden using the **columnsToHide** property and a warning message is displayed if this is attempted.
rowHeaderTextColor	Select the  button and choose a color from the palette to set the row header text color.
rowHeaderTextFont	Sets the font of the text for the row header column.
rowHeaderTextSize	Sets the size of the text for the row header column.

Sort Properties

Property Name	Description
showSortIconFlag	Displays the "sort" icon  in the column header specified in the **sortColumnName** property. To sort a table by a specific column, click on the column header. To reverse the sort order, click on the column header again.  To clear the sort, click on the empty column header at the top left of the table or delete the Property Value of **sortColumnName** in the **Object Properties** window. **Note:** Sorting is disabled if the sort icon is not visible in the column header.
sortAscendingFlag	When selected, data in the column specified in **sortColumnName** is listed in ascending order. When deselected, data in the column is listed in descending order.
sortColumnName	Specifies the column in which you want to sort data. The **showSortIconFlag** and **sortAscendingFlag** properties work in conjunction with this property.

Web Grid Properties

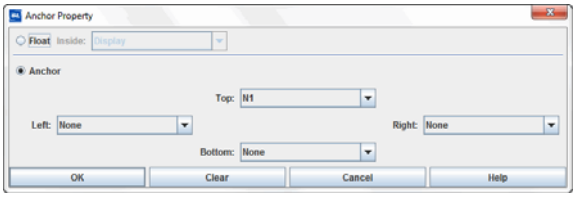
Property Name	Description
webGridFlag	Selecting this check box provides for an advanced HTML implementation of **obj_table02** that provides enhanced filtering (for the Thin Client only). See "Web Grid" for more information.
webGridHoverColor	See "Web Grid" for more information.
webGridRowsPerPage	See "Web Grid" for more information.
webTreeAggregateColumn	See "Web Grid" for more information.
webTreeGridFlag	Selecting this check box provides the ability to enable the first column of the table to behave like a "tree," thus allowing table rows to be expanded/ collapsed in groups according to a defined hierarchy (for the Thin Client only). See "Web Grid" for more information.
webTreeLabelColumn	See "Web Grid" for more information.

Anchor Property Window

The **Anchor Property** window allows you to anchor to a location within a display, or to an anchor object within the display.

Note: If an object has the dock property set, the anchor property will be ignored.

The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode.



By default, the **Anchor Property** window defaults to **Float** in the display. If you select the **Anchor** toggle, the **Top**, **Left**, **Right**, and **Bottom** options become enabled. For each option (**Top**, **Left**, **Right**, and **Bottom**), you can select from the following:

None	Object not anchored to this position. This is the default.
Display	The object is anchored to that area of the display. For example, if you select the **Left** drop down list and select **Display**, the object will be anchored to the left side of the display. If you select **Display** for opposite sides (**Left** and **Right** or **Top** and **Bottom**), the object will be stretched to fill the available space between the two.
<object_name>	The drop down list will also contain any other objects within the display that are assigned as anchors (those objects with the **isAnchorObject** property checked). If you select an anchor object, that side of the object will be anchored to the center of the specified anchor object. When the display resizes, the number of pixels between the specified side of the object and the center of the anchor object remain constant.

Note: Since an anchor object is referenced by name when anchoring to an object or floating inside an anchor object, you must not have multiple anchor objects with the same name.

Docking Objects

Docking an object in the display allows you to define where the object will be placed in the display if the dimensions of the display are modified (either by editing "Background Properties" or by resizing the window in Layout mode). When the dimensions are modified, the defined properties (**objX**, **objY**, **objWidth**, and **objHeight**) of docked objects will automatically adapt to match the new size of the display.

When multiple objects are docked to the same side of the display, the first object is docked against the side of the display, the next object is docked against the edge of the first object, and so on.

When objects are docked to multiple sides of the display, the order in which objects were added to the display controls docking position. For example, let's say the first object added to the display is docked at the **Top** and the second object is docked at the **Left**. Consequently, the first object will fill the entire width of the display and the second object will fill the left side of the display from the bottom of the first object to the bottom of the display.

Objects in a display have the dock property set to **Fill**, are laid out across a grid in the available space remaining after all docked objects are positioned. By default, the grid has one row and as many columns as there are objects in the display. You can modify the grid in the **Background Properties** dialog.

Once an object is docked, there are some limitations on how that object can be modified:

- Docked objects cannot be dragged or repositioned using **objX** and **objY** properties.
- Docked objects cannot be resized using the **objWidth** or **objHeight** properties. To resize you must drag on the resize handle.
- Docked objects can only be resized toward the center of the display (e.g. If an object is docked at the **Top**, only its height can be increased by dragging down toward the center of the display).
- Docked objects set to **Fill** cannot be resized at all.
- Docked objects cannot be moved using **Align**. Non-docked objects can be aligned against a docked object, but a docked object will not move to align against another object.
- Docked objects are ignored by **Distribute**.

Table Edits

In the Standard Table (class name: **obj_table02**), it is possible to add or remove rows and columns or edit the value of existing cells. To utilize table editing features, a Custom Message Handler must be implemented. Once the Custom Message Handler has been created, anyone can use RTView to easily update data throughout the enterprise. See "Table Edits - Custom Message Handler" for more information.

Enable Editing

To enable editing, you must set a variable and attach the table to data.

The **editDataEnabledFlag** allows you to control whether a table can be edited. In the **Object Properties** window when you select the **editDataEnabledFlag**, the **editDataLocalVarName** property will appear. In the **Property Value** field of **editDataLocalVarName** enter the name of the variable you would like to update with changes made to the editable table. This will allow you to identify the table in the Custom Message Handler as well as attach other objects in this display to your edited data.

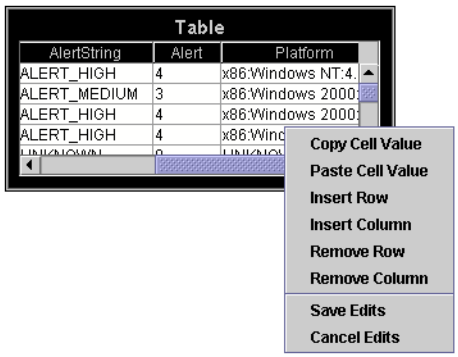
Note: The property **editDataLocalVarName** cannot be changed while the table is in Edit mode.

Attach data to the **valueTable** property to populate the table with initial data in order to enable editing. The table will not display data updates while in Edit mode, however once you save or cancel table edits current data for the **valueTable** attachment will be displayed.

Edit Table

To put the table in Edit mode, right-click on the table and select **Edit Table** from the popup menu. If the **Edit Table** option is not listed, the **editDataAllowFlag** has not been selected. If **Edit Table** is listed in the menu but is not active, the **valueTable** property is not attached to data.

To indicate your table is in Edit mode, the background rectangle of the table as well as the column headers will switch foreground and background colors.



To edit an existing cell value, double-click within a cell and enter a new value. When you are finished editing within a cell, you must press <Enter> or click on another cell before saving your edits. If the data entered is not the correct type (int, double, etc.) for that cell, the cell border will turn red and you will not be able to continue editing until the correct type of value is entered.

Right-click to access the following menu options:

Copy Cell Value	Select a cell and right-click to Copy Cell Value.
Paste Cell Value	Select a cell and right-click to Paste Cell Value.
Insert Row	Select a cell in the row that is immediately above or below where you want to insert a new row and right-click to Insert Row. A dialog will appear asking you to enter a row name and specify whether to insert the new row above or below the selected row. Click **OK** to exit the dialog and add the new row. **Note:** If the Property Value of **rowLabelMode** is **0**, row names will not appear in the table.
Insert Column	Select a cell in the column that is immediately to the right or left of where you want to insert a new column and right-click to Insert Column. A dialog will appear asking you to enter a column name, column type, and specify whether to insert the new column to the right or left of the selected column. Click **OK** to exit the dialog and add the new column.
Remove Row(s)	Select one or more cells and right-click to Remove Row(s) containing the selected cell(s).
Remove Column(s)	Select one or more cells and right-click to Remove Column(s) containing the selected cell(s). **Note:** The column that contains row names cannot be deleted. The table will only contain a column with row names if the Property Value of **rowLabelMode** is greater than 0.
Save/Cancel Edits	Right-click on the table to Save or Cancel Edits.

Save or Cancel Edits

If you exit a display while tables are in Edit mode without saving or canceling changes, a dialog will appear (for each table with unsaved changes) asking if you would like to save your table edits.

Once you save or cancel your changes, the table will display current values for the **valueTable** data attachment and return to its original color scheme to indicate you are out of Edit mode.

Table Edits - Custom Message Handler

Note: This section assumes you have a working knowledge of writing, compiling and deploying Java classes.

The Custom Message Handler is a Java class that allows you to extend the functionality of RTView by implementing code to update your system when the table is edited. Once the Custom Message Handler has been created, it can be deployed with your displays so that end users who are editing tables do not need to be familiar with the systems they are updating.

To implement a Custom Message Handler, you must create a Java class named **MyMessageHandler.java** that extends **GmsRtViewCustomMessageHandler**.

Define the following method in **MyMessageHandler.java**:

public boolean invokeMessage (String messageString, Object value)

The **invokeMessage** method will get called while the table is being edited. The **messageString** argument will contain one of the messages listed below. The **value** argument will be a 4 element Object array. The first two arguments in the Object array are constant, the rest vary based on the corresponding message.

The **invokeMessage** method must return true when it receives any of the following messages, otherwise it must return false.

Note: The **MessageHandler** will receive other messages, but the method should not return true or implement code when receiving anything other than the messages listed below.

The following lists messages and corresponding value arguments:

Message	**Description**	**Object Array Values**		
table_edit_start	The **invokeMessage** method will get called with this message when **Edit Table** is selected from the popup menu.	Argument	Definition	Type
		value[0]	Table object being edited.	GmsAppObject
		value[1]	Name of the variable assigned to **editDataLocalVarName**.	String
		value[2]	Initial table data prior to editing.	GmsTabularData
		value[3]	Null	Null
table_edit_save	The invokeMessage method will get called with this message when **Save Edits** is selected from the popup menu or when **Yes** is selected if prompted to save while exiting.	value[0]	Table object being edited.	GmsAppObject
		value[1]	Name of variable assigned to **editDataLocalVarName**.	String
		value[2]	Table data containing table edits.	GmsTabularData
		value[3]	Null	Null

table_edit_cancel	The **invokeMessage** method will get called with this message when **Cancel Edits** is selected from the popup menu or when No is selected if prompted to save while exiting.	value[0]	Table object being edited.	GmsAppObject
		value[1]	Name of variable assigned to **editDataLocalVarName**.	String
		value[2]	Table data containing the current value of **valueTable**. If data for the **valueTable** attachment was updated during editing, this may not be the same as data sent with the **table_edit_start** message. This message does not contain table edits.	GmsTabularData
		value[3]	Null	Null
table_edit_remove_row	This message is sent when **Remove Row** or **Remove Rows** is selected from the popup menu.	value[0]	Table object being edited.	GmsAppObject
		value[1]	Name of variable assigned to **editDataLocalVarName**.	String
		value[2]	**GmsTabularData** object with one or more data rows that were deleted.	GmsTabularData
		value[3]	Each entry is the index of a deleted row.	int[]
table_edit_insert_row	This message is sent when **Insert Row** is selected from the popup menu.	value[0]	Table object being edited.	GmsAppObject
		value[1]	Name of variable assigned to **editDataLocalVarName**.	String
		value[2]	**GmsTabularData** object with one row that contains the row data that was inserted.	GmsTabularData
		value[3]	Value is the index of the inserted row.	Integer
table_edit_update_row	This message is sent when a cell from the table is edited.	value[0]	Table object being edited.	GmsAppObject
		value[1]	Name of variable assigned to **editDataLocalVarName**.	String
		value[2]	**GmsTabularData** object with two rows. Row[0] contains row data before the cell was edited, Row[1] contains row data after the cell was edited.	GmsTabularData
		value[3]	Element[0] is the row number of the cell. Element[1] is the column number of the cell or the value **-1** for the row label column.	int[]

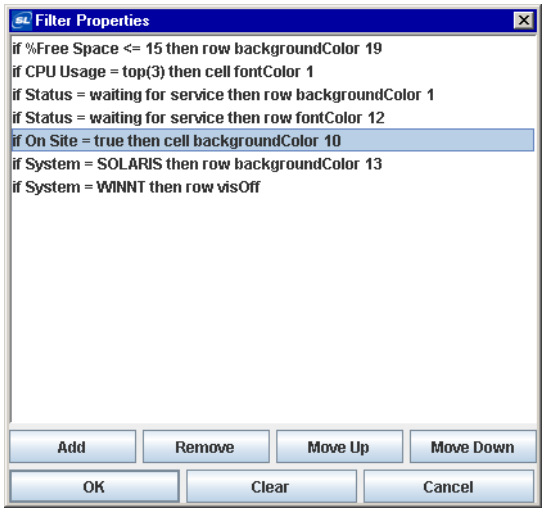
table_edit_remove_column	This message is sent when **Remove Column** or **Remove Columns** is selected from the popup menu.	value[0]	Table object being edited.	GmsAppObject
		value[1]	Name of variable assigned to **editDataLocalVarName**.	String
		value[2]	**GmsTabularData** object with columns that existed before the column was removed and no rows.	GmsTabularData
		value[3]	Each entry is the index of a deleted column.	int[]
table_edit_insert_column	This message is sent when **Insert Column** is selected from the popup menu.	value[0]	Table object being edited.	GmsAppObject
		value[1]	Name of variable assigned to **editDataLocalVarName**.	String
		value[2]	**GmsTabularData** object with columns that existed after the column was inserted and no rows.	GmsTabularData
		value[3]	Value is the index of the inserted column.	Integer

Table Filters

In the Standard Table (class name: **obj_table02**), it is possible to set font color, background color, row visibility and replace a cell value with an image based on the value of individual columns, rows and cells.

Note: It is recommended that you attach the table to data before attempting to define filters.

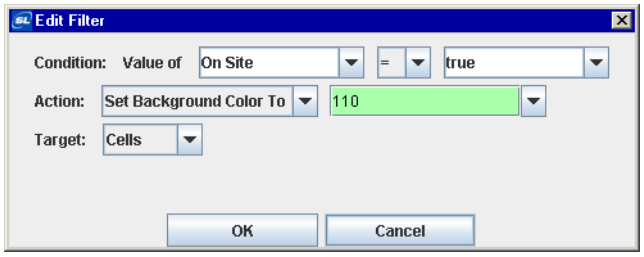
In the **Object Properties** window, double-click on **filterProperties** in the **Property Name** field to bring up the **Filter Properties** dialog. To create a filter, click on the **Add** button. Double-click on an existing filter to edit. Click the **Remove** button to delete a filter. Click **Clear** to remove all filters. Use the Move Up and Move Down buttons to control in what order filters will be applied to the table.



Filters are applied to the table in the order they are listed in the **Filter Properties** dialog. For example, in the table below, cells that are marked On Site are set to display a background color of green. However, table rows that are marked On Site and contain the SOLARIS system have a background color of gray throughout the entire row. The setting for the SOLARIS row background color overrides the setting for the On Site cell background color because the filters are being applied in the order they are listed above.

Create a Filter

In the **Object Properties** dialog, double-click in the **Property Name** field to open the **Filter Properties** dialog, then click Add. The Edit Filter dialog opens. Each filter is composed of three elements: **Condition**, **Action**, and **Target**:



Condition

Define the condition a cell must meet for the filter to be applied. Select a comparison field from the first drop down menu. The comparison field is typically the name of a column (e.g.: On Site) that contains the value (e.g.: **true**) to which you would like to apply a filter. The comparison field menu is populated with column names based on the table's data attachment, along with the options **Row Name** and **Column Header**. You may enter the name of a substitution or variable to use as the value of this field. Select **Row Name** to filter based on the value of a row name. Select **Column Header** to filter based on the value of a column header.

Note: If you have not attached the table to data, the only values listed in this menu will be **Row Name** and **Column Header**.

In the second drop down menu, select an operator (e.g.: =, >, <, etc.) for the comparison.

In the third drop down menu, select what value the comparison field must correspond to in order for the cell to meet the condition. The comparison value menu is populated with values from the table's data attachment, based on the selected comparison field, along with the options top(5) and bottom(5). Select top (5) to apply the filter to rows or cells that contain the five highest values in the selected comparison field. Select bottom(5) to apply the filter to rows or cells that contain the five lowest values in the selected comparison field. Once a selection is made from the comparison value menu, it is possible to edit the number of rows you would like to filter. You may enter the name of a substitution or variable to use as the value of this field.

Note: If you have not attached the table to data, the only values listed in this menu will be top(5) and bottom(5).

Action

Define the action that will be applied to cells, columns, or rows that meet the condition. Select one of four actions from the first drop down menu. Select **Set Background Color** to control the color of cells. Select **Set Font Color** to control the color of text. Select **Hide Rows** to control the visibility of rows. Select **Display Image** to replace a cell value with an image.

Note: The Hide Rows action only targets rows.

The second drop down menu is populated with options based on which action is chosen in the first drop down menu. Depending on the action you select; choose which color to apply to the background or font, enter the name of a variable or substitution to use as the value of this field, or select an image from the **Select Image** dialog. The **Select Image** dialog contains up to three directories:

- **Current Directory** - Contains images in the current directory and one level of subdirectories.
- **Custom Image Library** - If you have specified a custom image library, this directory contains those images (.gif, .jpg or .png). See Creating a **Custom Image Library** (below) for details.
- **Symbol Library** - Contains symbolic images (for example, symbols for various types of hardware, shapes, lights, arrows, etc.).

Navigate to the image you want to use and select it. A preview of the image appears in the pane to the right. Click **OK** or **Apply** to set the image on your object. If an image is not listed, enter the name of the file, including the relative path.

Target

Control how the filter is applied. There are three targets available. Select **Rows** to apply the filter to the entire row, if the cell from the comparison field in that row meets the condition. Select **Cells** to apply the filter to individual cells from the comparison field that meet the condition. Select **Columns** to apply the filter to the entire comparison field column, if any cell in that column meets the condition.

Note: The target does not apply if you chose **Hide Rows** as an action.

Creating a Custom Image Library

The custom image library enables you to make your own images available in the **Select Image** dialog. To add your own image library, perform the following steps.

1. Place your images **.jar** file and add it to the "RTV_USERPATH" environment variable. The images must be in a directory (not in the top level of the jar). They can be organized into subdirectories of one top level directory.

2. In the Display Builder, select **Tools/Builder Options** and, in the **Custom Image Library Path** field, set the path to the directory containing your images **.jar** file.

For example, if you have a jar with this directory structure:

```
com/mycompany/Images
com/mycompany/Images/Blue Images
com/mycompany/Images/Red Images
com/mycompany/Images/Green Images
```

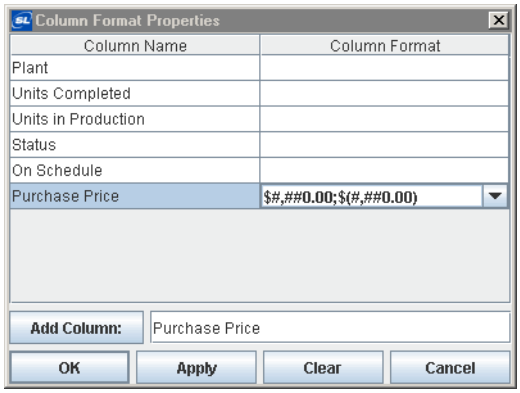
you would enter **com/mycompany/Images**. This adds a directory named **Images** to the tree in the **Select Image** dialog. The **Images** directory will have three subdirectories: **Blue Images**, **Red Images**, and **Green Images**. Only directories containing images are added to the **Select Image** dialog.

To access the images, you can edit any property that allows you to set an image on an object (for example, the **image**, **barImage**, and **filterProperties** properties), or edit the **File> Background Properties> Image Name** field.

Note: You can use animated .gif files in Standard Tables, but only in the Thin Client. In the builder/viewer, the animation will only update once every two seconds or after a mouse event.

Column Format Properties

In the **Object Properties** window, double-click on **columnFormat** in the **Property Name** field to display the **Column Format Properties** dialog. In the **Column Format Properties** dialog you can assign numerical and date formats to columns.



Field Name	Description
Column Name	This list is populated based on the table's data attachment. If you have not yet attached the table to data, this list will be empty.
Column Format	Enter or select a format from the drop down menu and press <Enter>. Specify numerical formats using the Java format specification, or with the following shorthand: $ for US dollar money values, $$ for US dollar money values with additional formatting, () for non-money values, formatted similar to money, or # for positive or negative whole values. Specify date formats using the Java date specification.
Add Column	Enter the name of the column and click the **Add Column** button to insert a column into the table.
Clear	Click the **Clear** button to remove all Column Formats listed.

Note: Text columns containing data where all the values represent numbers are treated as if they are numeric columns, so number formats can be applied.

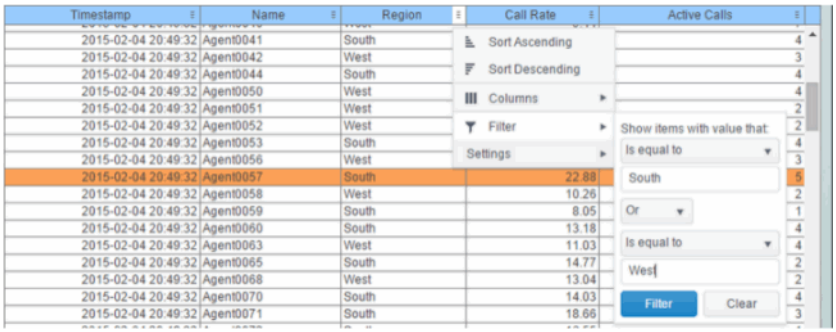
Web Grid

There are two different check boxes in the **Web Grid** Properties that affect the way the table behaves:

- "webGridFlag": provides enhanced filtering and layout capabilities in the thin client.
- "webTreeGridFlag": enables the first column of a table to act as a "tree," thus allowing the table rows to be expanded/collapsed in groups according to a defined hierarchy in the thin client.

webGridFlag

The web grid is an advanced HTML implementation of obj_table02 that provides enhanced filtering and layout capabilities. The web grid is available in the Thin Client only.



To display your data in a web grid, select the obj_table02 instance in the Builder and check its **webGridFlag** property in the property sheet. By default, the **webGridFlag** property is unchecked, which specifies use of the "classic" HTML grid in the Thin Client.

Alternatively, the web grid can be enabled on all obj_table02 instances by adding the following rule to an rtview stylesheet (**.rts**) file loaded by the Display Server:

```
obj_table02 {
webGridFlag : 1
 }
```

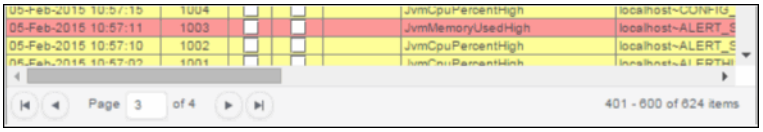
Requirements

If **webGridFlag** is enabled, the web grid appears in the Thin Client in any modern version of a supported browser. No plugin is required. For Internet Explorer users, version 9 or newer is required and version 11 is recommended for best performance. In IE8 or older, the web grid is not supported, so the classic grid is used regardless of the **webGridFlag** setting. In this release, the web grid is not supported on iPad but might be supported in the future.

Features

The web grid supports advanced, interactive table features in the Thin Client: sorting on multiple columns, filtering on multiple columns, column resizing, column reordering, and hiding columns. In addition, you can unsort a previously selected sort column and, in a grid with **rowHeaderEnabledFlag** = **true**, additional columns can be locked into the row header. You can save all of those column settings permanently so that they are restored when you return to the display later. Many of these features are accessed from the column menu, shown in the screen shot above, opened by clicking on the menu icon in each column's header.

Also, for improved performance and usability, if a data table contains more than 200 rows the web grid displays it in pages of 200 rows, using a page control that appears at the bottom of the grid:



Column Sorting

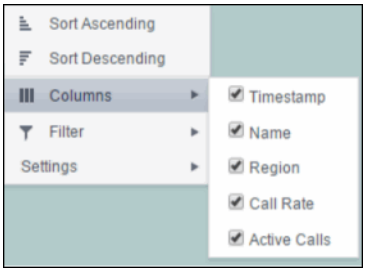
You can click on a column header to sort the table by that column. On the first click, the column is sorted in ascending order (smallest value at the top), on the second click the sort is in descending order, and on the third click, the column is returned to its original unsorted state. A sort on a string column is case-insensitive.

You can select multiple sort columns. In that case, the sorting is performed in the order that the column headers were clicked. Multiple column sorting is a very useful feature, but can also cause confusion if you intend to sort on a single column, but forget to "unsort" any previously selected sort columns first. You should check for the up/down sort icon in other column headers if a sort gives unexpected results.

Column sorting is reflected in an export to HTML and Excel.

Column Visibility

You can hide or show columns in the table by clicking on any column's menu icon, and choosing **Columns** from the menu. This opens a submenu with a check box for each column that toggles the visibility of the column. All columns in the data table appear in the Columns menu, even those that are initially hidden by the obj_table02 property **columnsToHide**.



If the grid has the **rowHeaderEnabledFlag** property checked then the leftmost column (the row header column) cannot be hidden.

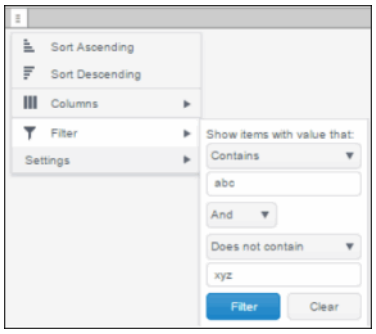
Column visibility changes are NOT reflected in an export to HTML and Excel.

Column Filtering

You can create a filter on any column. If filters are created on multiple columns, then only the rows that pass all of the filters are displayed. That is, if there are multiple filters they are logically "ANDed" together to produce the final result.

The background of a column's menu icon changes to white to indicate that a filter is defined on that column. This is intended to remind you which columns are filtered.

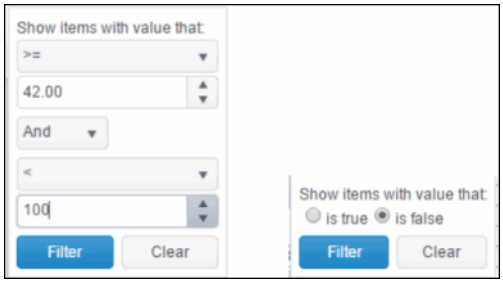
You can configure a filter on any column by clicking on the column's menu icon and choosing Filter from the menu. This opens the **Column Filter** dialog:



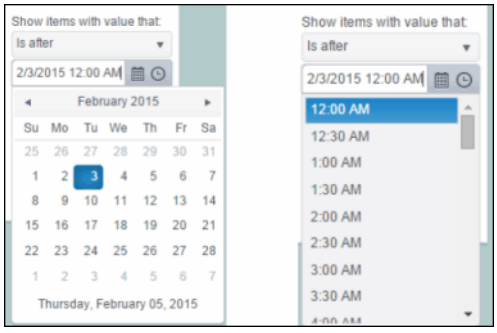
Options in the **Column Filter** dialog vary according to the data type of the selected column:

- **String columns**: You can enter a filter string such as "abc" and, from the dropdown list, select the operator (equal to, not equal to, starts with, contains, etc) to be used when comparing the filter string to each string in the column. All of the filter comparisons on strings are case-insensitive. You can optionally enter a second filter string (e.g. "xyz") and specify if an AND or OR combination should be used to combine the first and second filter results on the column.
- **Numeric columns**: You can enter numeric filter values and select arithmetic comparison operators, (=, !=, >, >=, <, <=). You can optionally enter a second filter value and comparison operator, and specify if an AND or OR combination should be used to combine the first and second filter results.
- **Boolean columns**: You simply select whether matching items should be true or false.

The numeric and boolean filter dialogs are shown below.



- **Date columns**: You can select a date and time and choose whether matching items should have a timestamp that is the same as, before, or after the filter time. The date is selected by clicking on the calendar icon and picking a date from a calendar dialog. The time is selected by clicking on the time icon and picking a time from a dropdown list:



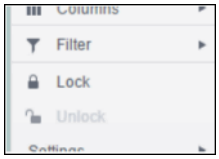
  Alternatively, a date and time can be typed into the edit box. See the "Unsupported obj_table02 Features" section for a note on time filtering when the client and server are located in different time zones.

Data updates to the grid are suspended while the filter menu is opened. The updates are applied when the menu is closed.

Column filtering is reflected in an export to HTML and Excel.

Column Locking

This feature is available only if the obj_table02 instance has the row header feature enabled (**rowHeaderEnabledFlag** is checked). If so, the leftmost column is "locked" in position, meaning that it does not scroll horizontally with the other columns in the table. If the row header is enabled, then two items labeled **Lock** and **Unlock** appear in the column menu. These can be used to add or remove additional columns from the non-scrolling row header area.



If the row header is enabled, at least one column must remain locked.

Column locking is NOT reflected in an export to HTML and Excel.

Column Reordering

You can reorder the grid columns by dragging and dropping a column's header into another position. If the grid has **rowHeaderEnabledFlag** checked, then dragging a column into or out of the row header area (the leftmost columns) is equivalent to locking or unlocking the column.

Column reordering is NOT reflected in an export to HTML and Excel.

Paging

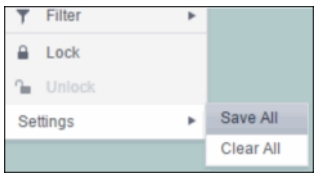
If the data table contains more than one page of rows, the page controls are displayed at the bottom of the grid. The default page size is 200 but can be set on each obj_table02 instance via the new property named **webGridRowsPerPage**. The default value of that property is zero, which indicates that the default size (200) should be used. If the height of the grid is less than about 64 pixels, there is insufficient space to display the page controls so only the rows on the first page is viewable.

Row Mouseover

A new property named **webGridHoverColor** is available on obj_table02. It is visible only if **webGridFlag** = **true**. The default value of **webGridHoverColor** is checked. If it is set to any other color index value, then that color is used to highlight the row that is under the mouse cursor. But if the obj_table02 **filterProperties** feature is used to color rows, that color takes precedence, so the **webGridHoverColor** may not be useful in those cases. Also, if the row header is enabled, the row header column and the other columns are highlighted separately, according to which section of the grid the mouse is over.

Saving Settings

You can permanently save all of the custom settings made to the grid, including filtering, sorting, column size (width), column order, column visibility, and column locking. This is done by opening any column menu, clicking **Settings**, and then clicking **Save All**:



The grid's settings are written as an item in the browser's local storage. The item's value is a string containing the grid's settings. The item uses a unique key comprised of the URL path name, the display name, and the obj_table02 instance's RTView object name. If the Thin Client's login feature is enabled, the key will also include the username and role, so different settings can be saved for each user and role for a grid on any given display, in the same browser and host.

If the user saves the grid settings and navigates away from the display or closes the browser, then the next time the user returns to the display in the same browser the settings are retrieved from the browser's local storage and applied to the grid. The browser's local storage items are persistent, so the grid settings are preserved if the browser is closed and reopened or if the host system is restarted.

If the obj_table02 has **autoResizeFlag** = **true** then the column widths are not restored from the saved settings, and the values computed by the auto-resize feature is used instead. This is by design.

You can delete the grid's item from local storage by clicking **Settings> Clear All** in any column menu. This permanently deletes the saved settings for the grid and returns the grid to the state defined in the display file.

Note that each browser has its own local storage on each host. The local storage items are not shared between browsers on the same host or on different hosts. So, if a user logs in as Joe with **role = admin**, in Internet Explorer on host H1, and saves grid settings for display X, then those grid settings are restored each time a user logs in as Joe, role admin, on host H1 and opens display X in Internet Explorer. But if all the same is true except that the browser is Chrome, then the settings saved in Internet Explorer are not applied. Or if the user is Joe and role is admin and the browser is IE and the display is X, but the host system is H2 not H1, then the grid settings saved on H1 are not applied.

Support for Large Tables

The web grid can support data tables with many rows and columns. However, for best performance the display server's **cellsperpage** property should be specified so that the server sends large tables to the client in pages, rather than sending all of the rows. In this server paging mode, large tables are also filtered and sorted in the Display Server, to improve performance and decrease data traffic. The **cellsperpage** option is not new or specific to the web grid, it has been available for several years. See the RTView documentation for a description of the **cellsperpage** property, and the related **cellsperexport** and **cellsperreport** properties. A typical value for **cellsperpage** is 20000.

Unsupported obj_table02 Features

The following are existing features of obj_table02 that are not supported by the web grid.

- The **rowHeaderEnabledFlag** property is supported, but **rowHeaderBgColor**, **rowHeaderTextColor**, **rowHeaderTextFont**, **rowHeaderTextSize** are ignored. Instead the row header column is rendered like all other columns.
- The **columnResizeEnabledFlag** is ignored if it is false, the web grid always allows column resizing.
- The **editDataEnabledFlag** is ignored, the table editing feature using custom commands is currently not supported.

Other limitations, and differences between the new and classic grids

- Time zones: The strings shown in a date column are formatted by the display server using its time zone. But if a filter is specified on a date column, the date and time for the filter are computed using the client system's time zone. This can be confusing if the display server and client are in different time zones.
- Selected rows: The grid's row selection is cleared if the sort is changed or if columns are resized or reordered.
- Scrollbars: In general the grid only displays scrollbars when they are needed. However, the web grid and the classic grid use different algorithms for deciding when to show or hide scrollbars, and do not use identical row heights and column widths. So the web grid may sometimes display scrollbars when the classic grid does not, for a grid instance with a given width and height.
- Keyboard traversal: In the classic grid, selecting a row and then using the up/down arrow keys changes the selection to the previous/next row. In the web grid, the arrow keys moves the keyboard focus to another row, as indicated by a highlight border around the focused table cell, but the user must press the space bar to select the row that contains the highlighted (focused) cell.
- Column widths: On a web grid with no locked columns (**rowHeaderEnabledFlag** = **false**), columns expand to fill any unused width in the table, even if **autoResizeFlag** = **false**. That is, if the total width of the columns is less than the grid width (i.e. the columns don't use all of the available width) then each column is expanded proportionally to fill the table. In contrast, the classic and Swing (Viewer) grids just leave unused space at the right edge of the grid. If the grid has locked columns (**rowHeaderEnabledFlag** = **true**), then the web grid behaves the same as the classic and Swing grids.
- Export: The export to HTML and Export to excel features are supported on the web grid, and behave much the same as on the classic grid. The exported table respects the grid's filter and sort settings but ignores any column reordering, sizing, or hiding changes made by the user.
- Data updates to the grid are suspended while the filter menu is opened. The updates are applied when the menu is closed.

Implementation Note for Custom Web Page Developers

The RTView Thin Client now uses two versions of the jQuery JavaScript library:

- jQuery 1.3.2, to support the jQuery UI components used for panels, layout, and (some) navigation controls.
- jQuery 1.9.1, to support the web grid.

Version 1.3.2 is bundled into the Thin Client js file named **rtvNav1.js**, while version 1.9.1 is loaded as a separate js file. To prevent conflicts between the two libraries, the jQuery **$** alias is no longer used by RTView and is intentionally removed. Instead the jQuery 1.3.2 library is referenced by the alias **rtv.jqNav** as needed, and the jQuery 1.9.1 library is referenced by the alias **rtv.jq** as needed.

This could affect existing custom JavaScript code that users have written. For example, the following is a snippet of JavaScript from a custom html page that uses jQuery UI tabs for navigation

```
<script src="rtvNav1.js"></script>
<script>
$(document).ready(function () {
var outerLayout = $('body').layout({
north__size:64, north__spacing_open: 0,
 });
```

```
$('#tabs').tabs();
...
```

The code above uses the jQuery **$** alias, which it assumes is defined in the **rtvNav1.js** file. But that is no longer true, so the custom code should be rewritten to use the **rtv.jqNav** alias for jQuery in place of the **$** alias, as follows:

```
<script src="rtvNav1.js"></script>
 <script>
 rtv.jqNav(document).ready(function () {
 var outerLayout = rtv.jqNav('body').layout({
 north__size:64, north__spacing_open: 0,
 });
 rtv.jqNav('#tabs').tabs();
 ...
```

webTreeGridFlag

The **webTreeGridFlag** property enables the first column of a table to behave as a tree, which allows for table rows to be expanded/collapsed in groups according to a hierarchy defined by the index columns in the data table. This mode can be configured in the builder, but the feature is only fully implemented in the thin client.

There are three properties that support this mode, which are visible only if **webGridFlag** is checked:

- **webTreeGridFlag**: select this check box to enable the "tree grid" mode.
- **webTreeLabelColumn**: the name of the column to be added as the tree (first) column of the table.
- **webTreeAggregateColumns**: a list of column names and calculations to be applied to data columns in parent rows.

Index Columns

To use the tree grid mode, the table object's **indexColumnNames** property must be specified. Each index column in the data table (attached to the **valueTable** property) defines a level in the tree -- the first index column specifies the values for the top level of the tree, the second index column specifies the values for the second level in the tree, and so on.

For example, here is a portion of a data table containing Major League Baseball (MLB) standings. The index columns are named **League**, **Division**, and **Team**:

League	Division	Team	W	L
American	East	Red Sox	93	69
American	East	Blue Jays	89	73
American	Central	Indians	94	67
American	Central	Tigers	86	75
American	West	Rangers	95	67
American	West	Mariners	86	76
National	East	Phillies	95	67
National	East	Mets	87	75
National	Central	Cubs	103	58

National	Central	Cardinals	86	76
National	West	Dodgers	91	71
National	West	Giants	87	75

Note: The full MLB table has 5 teams in each division.

The index column values in the above data table can be used to create a three level tree, as follows:

American
- East
  - Red Sox
  - Blue Jays
- Central
  - Indians
  - Tigers
- West
  - Rangers
  - Mariners

National
- East
  - Phillies
  - Mets
- Central
  - Cubs
  - Cardinals
- West
  - Dodgers
  - Giants

Data Table Format

When tree grid mode is enabled, the data table is automatically converted to the format required for the mode, as follows:

1. A row is added to the table for each parent level in the tree.

   For example, when the MLB table above is displayed in tree grid mode, it will contain eight additional "parent" rows that don't appear in the original table. These rows are added for the parent levels in the tree, one for the American League, one for the National League, and one for each of the three divisions in each league (American East, American Central, American West, National East, National Central, National West). The data columns (W, L) for each of the parent rows is assigned a value of zero because there is no data in the original table for the parent rows. The added rows will look like this:

American	*	*	0	0
American	East	*	0	0
American	Central	*	0	0
American	West	*	0	0
National	*	*	0	0
American	East	*	0	0
National	Central	*	0	0
National	West	*	0	0

2. A new column is added to the beginning of the table to display the tree levels.

   The column is named **Node Label** by default, but a different name can be specified in the **webTreeLabelColumn** property.

3) Columns named **Node ID** and **Parent ID** are added to the table.

   These define the parent-child relationships in the table, and are used by the thin client to construct the tree. These columns are automatically hidden, via the **columnsToHide** property.

Aggregation

As mentioned earlier, the parent rows added to the data table will contain a zero in each numeric data column. The **webTreeAggregateColumns** can be configured to show aggregated values (**sum**, **min**, **max**, or **average**) in those data columns. The property expects a list of column name and calculation pairs with semicolons between the pairs.

For example, to show the sum of the **W** (wins) and **L** (losses) columns of the child rows in each parent row of the MLB table above, you would define the following:

webTreeAggregateColumns: W,sum;L,sum

With that configuration, each **Division** row will show the sum of its teams **W** and **L** columns, and each **League** will show the sum of its divisions **W** and **L** columns.

To show the least number of wins by a team in the child rows in the **W** column and most losses by a team in the child rows in the **L** column in each parent row of the MLB table above, you would define the following:

webTreeAggregateColumns: W,min;L,max

Filter and Sort

In the thin client, all of the web grid filter and sort features are available by clicking on the column header. There are a few differences to note.

The column filter menu can be used to hide rows in the table. However, this does not affect the aggregate calculations (if any), since they are always performed on all rows in the table whether they are visible or not. Also, a parent row will be shown if any of its children pass the filter test, even if the parent row itself does not pass the filter. For example, if the filter **Team = Phillies** is applied to the MLB table, only one row passes that filter but the table will display both of that row's ancestor rows, for a total of 3 rows:

National

    East

        Phillies

The column sort feature in the thin client applies the sort within each level. That is, the sort will reorder rows within a tree level but will not move rows between levels of the tree, and parent rows will still appear above the rows of their children.

Limitations on webTreeGrid Mode

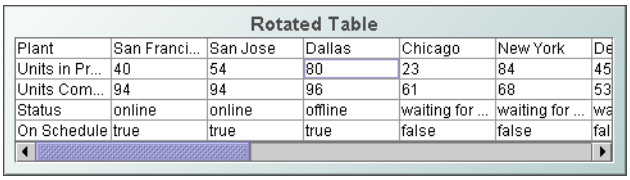
- The **webTreeGridMode** behavior only works in thin client, and only in desktop browsers that support the web grid. The table will not display a tree column in all other cases.
- The **webTreeGridMode** does not support server-side paging, filtering, and sorting. This means that **webTreeGridMode** mode does not perform well on large data tables because the entire data table must be downloaded to the browser. For best results, it should only be applied to data tables with 1000 rows or fewer. The intended use is to display the contents of an indexed cache table with fewer than 1000 rows.
- The export to html/excel/pdf feature does not recognize **webTreeGridMode** and instead exports the entire table.

Advanced Use

If the data table attached to **valueTable** already contains rows for the parent levels of the tree, with columns named exactly "Node ID" and "Parent ID" that uniquely identify each row and its parent row, then the table will not be converted as described above. In this case, the column to be used as the tree column should be identified by the **webTreeLabelColumn** property.

Rotated Table

Display your data in a rotated table (class name: **obj_table03**) by attaching it to the **valueTable** property. The rotated table is a standard table in which the rows and columns have been flipped. Row names are assigned the name of the column displayed in that row.



Background Properties

Specify how the background is displayed in a table.

Property Name	Description
bgBorderColor	Select the ... button and choose from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose from the palette to set the background color of the table.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **bgColor** property sets the first color in the gradient.

bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the table and the border.

Cell Properties

Property Name	Description
cellBgColor	Select the ... button and choose from the palette to set the background color of the cells.
cellBgStripeContrast	Set the contrast level for alternating row colors. **Note:** This property is only applicable if **cellBgStripedFlag** is selected.
cellBgStripedFlag	Select to display alternating row colors.
cellTextColor	Select the ... button and choose from the palette to set the color of the cell text.
cellTextFont	Select a font for the cell text.
cellTextSize	Specify (in pixels) the size of the cell text or enter **-1** to use the default size.

Column Properties

Property Name	Description
autoResizeFlag	If the **autoResizeFlag** check box is selected, table columns are automatically resized so that at least a portion of each column is visible within the current table width.
columnProperties	To resize a column in a rotated table, you must specify the size of each column. Click the ellipsis button  in the **columnProperties** property and enter a width for each column.

Data Properties

Property Name	Description
valueTable	Right-click in this property and select **Attach To Data > "Source"** (**XML**, for example) to select the table from which you want to pull in data. See "Attach to Data" for more information.

Grid Properties

Property Name	Description
gridHorizontalVisFlag	Select this property's check box to display horizontal lines in the table.
gridVerticalVisFlag	Select this property's check box to display vertical lines in the table.

Historian Properties

Property Name	Description
historyTableName	Specify name of table in your history database in which to store tabular data. See "Configuring the Historian" for information.
historyTableRowNameFlag	If selected, data from the row name field will be stored in the first column of the table specified in **historyTableName**.

Interaction Properties

Property Name	Description
command	Double-click on **command** to display the Define System Command window. See "Define System Command" for more information.

commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the  button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownColumnSubs	Double-click on **drillDownColumnSubs** or click the ellipsis button  to customize which substitutions will be passed into drill down displays. See "Drill Down Column Substitutions Dialog" and "Drill Down Displays" for more information.
drillDownSelectMode	Use **drillDownSelectMode** to control how a drill down display is activated. The standard table (class name: **obj_table02**) supports three modes for **drillDownSelectMode**: • Set to **Anywhere** (**0**) to activate a drill down display by double-clicking anywhere on the table. • Set to **Element Only** (**1**) to enable a drill down display only when you double-click on a cell in a table. Note that when set to **Element Only**, a drill down display can not be activated by double-clicking in the row label column, since this column does not correspond to an actual data column in the tabular data.
drillDownTarget	Double-click on the **drillDownTarget** property or click the ellipsis button  to specify a drill down target for the table. Based on your data attachment, preset substitutions are created automatically and passed into your drill down display.

Label Properties

Property Name	Description
label	Edit label text directly in the Property Value field or select the  button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. This property is only applies if **labelTextAlignY** is set to **TabTop**.
labelTextAlignX	Select x-axis position of label text from the drop down menu. You can select from **Left**, **Center**, or **Right**.

labelTextAlignY	Select y-axis position of label text from the drop down menu. **Outside Top** - Position label well above the outside border of the table. **Top** - Position label just above the outside border of the table. **Title Top** - Position label in the top line/border of the table. **Tab Top** - Position label tab just above the table. Height and width of the tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. **Inside Top** - Position label inside the top border of the table.
labelTextColor	Select the  button and choose a color from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop down menu.
labelTextHeight	Set the height of the label text in pixels.

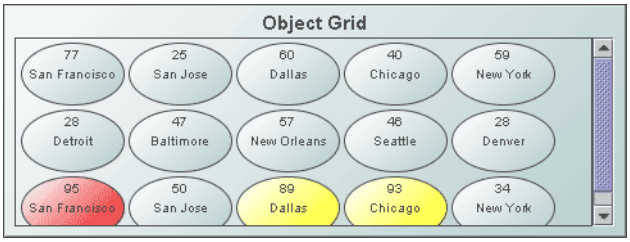
Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.

styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with rtv-.
visFlag	Control visibility of the object.

Object Grid

Display your data in an object grid (**obj_objectgrid**) by attaching it to the **valueTable** property. An object grid displays returned data as multiple objects, called icon objects, in a scrollable grid. Each icon object in the grid is analogous to a row of data in the standard table. Icon objects display the data using graphic attributes (e.g., color, dimension, orientation, etc.). The default icon object displayed in the object grid is **obj_circ2d_ilvx_ra4** (shown below).



Any object that appears in the Object Palette (with the exception of tables and graphs) can be used as an icon object. To specify which object to use in the object grid, in the **Object Properties** window, double-click on **iconProperties** in the **Icon** category. See "Icon Properties Dialog" for more information.

Background Properties

Specify how the background is displayed in the object grid.

Property Name	Description
bgBorderColor	Select the ... button and choose a color from the palette to set the color of the edge on the background rectangle. This property is only applicable if **bgBorderFlag** is selected.
bgBorderFlag	If selected, a border is drawn around the background rectangle.
bgColor	Select the ... button and choose a color from the palette to set the background color of the table.
bgEdgeWidth	Set the width of the 3D edge on the background rectangle. This property is only available if the **bgStyle** selected is **3D Rectangle**.
bgGradientColor2	Select the ... button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **bgColor** property sets the first color in the gradient.

bgGradientMode	Display a gradient in the background rectangle. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Center** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.
bgRaisedFlag	Reverses the direction of the gradient, as well as that of the 3D edge if the **bgStyle** selected is **3D Rectangle**.
bgRoundness	Set the arc length of the rounded corners. This property is only available if the **bgStyle** selected is **Round Rectangle**. **Note:** The value of **bgRoundness** cannot exceed half the value of the **objWidth** or the **objHeight**. If **bgRoundness** does exceed that value, then half (of the smaller of the two values) of **objWidth** or **objHeight** will be used instead. For example, if **objWidth** is **100** and **objHeight** is **50**, then the value of **bgRoundness** cannot exceed **25**. If it does, then half the value of **objHeight** (**25**) will be used instead.
bgShadowFlag	Select to display a drop shadow on the background rectangle.
bgStyle	Choose one of the following three options from the drop down menu: **Rectangle** - Select to display a background rectangle. **3D Rectangle** - Select to display a 3D edge on the background rectangle. If selected, use **bgEdgeWidth** to set the width of the 3D edge. **Round Rectangle** - Select to display a background rectangle with rounded edges. If selected, use **bgRoundness** to set the arc length of the rounded corners.
bgVisFlag	Select to display the background rectangle.
borderPixels	Set the amount of space (in pixels) between the table and the border.

Data Properties

Property Name	Description
valueTable	Right-click in this property and select **Attach To Data > "Source"** (**XML**, for example) to select the table from which you want to pull in data. See "Attach to Data" for more information.

Foreground Properties

Property Name	Description
fgColor	Set the color of the space between the objects. Select the  button and choose a color from the palette.
fgEdgeColor	Set the color of the box (line) surrounding the objects. Select the  button and choose a color from the palette.

fgGradientColor2	Select the button and choose a color for the second color in the gradient. Default is **white**. **Note:** The **fgColor** property sets the first color in the gradient.
fgGradientMode	Display a gradient in the space between the objects. Select from the following options: **None** - No gradient **Diagonal Edge** - Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object. **Diagonal Center** - Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object. **Horizontal Edge** - Gradient is drawn horizontally from the top to the bottom of the object. **Horizontal Top** - Gradient is drawn horizontally from the center to the top and bottom of the object. **Vertical Edge** - Gradient is drawn vertically from the left to the right of the object. **Vertical Center** - Gradient is drawn vertically from the center to the left and right of the object.

Historian Properties

The cache persistence feature is typically used instead of the following properties to configure the historian. See the **-persistCaches:true** command line option in "Command Line Options: Historian" for more information.

Property Name	Description
historyTableName	Specify name of table in your history database in which to store tabular data. See "Configuring the Historian" for information.
historyTableRowNameFlag	If selected, data from the row name field will be stored in the first column of the table specified in **historyTableName**.

Icon Properties

Property Name	Description
iconProperties	Double-click on **iconProperties** or select the button to display the "Icon Properties Dialog", which allows you to assign the icon object(s) you want to use in an Object Grid, as well as the attributes for each icon object. See "Icon Properties Dialog" for more information.

Interaction Properties

Property Name	Description
command	Double-click on **command** or click the ellipsis button  to display the Define System Command window. See "Define System Command" for more information.
commandCloseWindowOnSuccess	If selected, the window that initiates a system command will automatically close when the system command is executed successfully. This property only applies to system commands. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, this property is applied to each command individually. Therefore if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. **Note:** The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
commandConfirm	If selected, the command confirmation dialog is enabled. Use the **commandConfirmText** property to write your own text for the confirmation dialog, otherwise text from the command property will be used. For multiple commands, if you **Confirm** the execution then all individual commands will be executed in sequence with no further confirmation. If the you **Cancel** the execution, none of the commands in the sequence will executed.
commandConfirmText	Enter command confirmation text directly in the **Property Value** field or select the  button to open the **Edit commandConfirmText** dialog. If **commandConfirmText** is not specified, then text from the command property will be used.
drillDownColumnSubs	Double-click on **drillDownColumnSubs** or click the ellipsis button  to customize which substitutions will be passed into drill down displays. See "Drill Down Column Substitutions Dialog" and "Drill Down Displays" for more information.
drillDownSelectMode	Use **drillDownSelectMode** to control how a drill down display is activated. The standard table (class name: **obj_table02**) supports three modes for **drillDownSelectMode**: • Set to **Anywhere** (**0**) to activate a drill down display by double-clicking anywhere on the table. • Set to **Element Only** (**1**) to enable a drill down display only when you double-click on a cell in a table. Note that when set to **Element Only**, a drill down display can not be activated by double-clicking in the row label column, since this column does not correspond to an actual data column in the tabular data.
drillDownTarget	Double-click on the **drillDownTarget** property to specify a drill down target for the table. Based on your data attachment, preset substitutions are created automatically and passed into your drill down display.

menuItemGroup	Use the **menuItemGroup** and **rightClickActionFlag** properties to extend RTView context menu items. For details, see "Extending the Context Menu".
rightClickActionFlag	Use the **rightClickActionFlag** property so that a right-click by the user executes actions (a drill down or execute command) normally performed only with a left-click, before the right-click popup menu is shown. This is useful when you have a command string configured to set substitutions in the current window, and the drill down target is configured to drill down using those substitutions. In that case, the command is executed when you left-click and when you right-click so the substitutions are guaranteed to be set before you select **Drill Down** from the popup menu. This option should not be used if the left click is configured to drill down to another display. This property is ignored in the Thin Client on iOS Safari (iPad/iPhone).

Label Properties

Property Name	Description
label	Edit label text directly in the Property Value field or select the  button to open the **Edit Label** dialog.
labelMinTabWidth	Specify minimum width of the label tab. This property is only applies if **labelTextAlignY** is set to **TabTop**.
labelTextAlignX	Select x-axis position of label text from the drop down menu. You can select from **Left**, **Center**, or **Right**.
labelTextAlignY	Select y-axis position of label text from the drop down menu. **Outside Top** - Position label well above the outside border of the table. **Top** - Position label just above the outside border of the table. **Title Top** - Position label in the top line/border of the table. **Tab Top** - Position label tab just above the table. Height and width of the tab is dependent on the height and width of the text. Use the **labelMinTabWidth** property to specify a minimum tab width. **Inside Top** - Position label inside the top border of the table.
labelTextColor	Select the  button and choose a color from the palette to set the label text color.
labelTextFont	Select font style of label text from the drop down menu.
labelTextHeight	Set the height of the label text in pixels.

Layout

Property Name	Description
margins	Specifies the margin (in pixels) between the edge of the grid and the objects within the grid.
objSpacing	Specifies the spacing (in pixels) between the objects contained within the grid.

Object Properties

Property Name	Description
anchor	Select the  button to display the **Anchor Property** window, which allows you to specify where to anchor an object in the display. You can anchor to a location within a display, or to an anchor object within the display. The anchor property is only applied when the dimensions of the display are modified, either by editing "Background Properties" or resizing the window in Layout mode. See "Anchor Property Window" for more information. **Note:** If an object has the dock property set, the anchor property will be ignored.
dock	Specify the docking location of an object in the display. Select from the following options: **None** - Object is not docked. This is the default. **Top** - Dock object at top of display. **Left** - Dock object at left of display. **Bottom** - Dock object at bottom of display. **Right** - Dock object at right of display. **Fill** - Dock object in available space remaining in the display after all docked objects are positioned. See "Docking Objects" for more information.
isAnchorObject	Selecting this check box sets the object as an available anchor, which means that you can anchor other objects to this particular object. Objects defined as anchor objects display in the **Top**, **Left**, **Right**, and **Bottom** drop down lists in the "Anchor Property Window". See the **Anchor** property, above, for more information.
objHeight	Sets the height (in pixels) of the object.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objWidth	Sets the width (in pixels) of the object.
objX	Set the x position of the object.
objY	Set the y position of the object.
styleClass	Enter the style class name for this object as defined in your style sheet. If not specified, the object class name is used. **Note:** The value entered must not contain spaces and cannot start with **rtv-**.
transparencyPercent	Set transparency of the object. Enter a value between **0** and **100**. A value of **0**, the default, sets the object to be completely opaque. A value of **100** will render the object completely transparent.
visFlag	Control visibility of the object.

Sort Properties

Property Name	Description
sortAscendingFlag	When selected, data in the column specified in **sortColumnName** is listed in ascending order. When deselected, data in the column is listed in descending order.
sortColumnName	To sort the object grid by a column from the data attachment, enter the name of that column in this property. The column used to sort the object grid must be also be entered as a Property Value for **iconProperties** (e.g., PERCENTCOMPLETED). See "Icon Properties Dialog" (next) for more information.

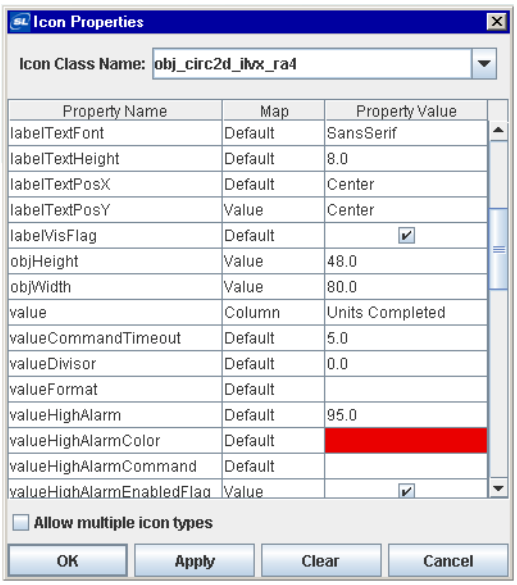
Icon Properties Dialog

Icon objects in the Object Grid display data using graphic attributes (e.g., color, dimension, orientation, etc.). The **Icon Properties** dialog, which can be accessed by double-clicking on the **iconProperties** option in the **Object Properties** window, allows you to assign the icon object(s) you want to use in an Object Grid, as well as the attributes for each icon object.

Before assigning attributes to icon objects, it is recommended that you attach the Object Grid to data and determine which object(s) you want to use. Objects can be seen in the Object Palette of the Display Builder. All objects in the Object Palette, with the exception of tables and graphs, can be used as an icon object.

Note: If you use a Composite object as the icon, then the **bgOpaqueFlag** property must be selected on the Object Grid.

Object class names appear in the status bar at the bottom of the window when an object is selected. For custom objects that are not in the standard palette, type in the class name. The object class name is what you need to specify the icon object.



Property Name	Description
Icon Class Name	Select an object from the drop down menu. The default icon object displayed in the Object Grids **obj_circ2d_ilvx_ra4**. For custom objects that are not in our standard palette, type in the class name.
Property Names	Property Names are populated based on the icon object you choose from the **Icon Class Name** drop down list. Properties appear in the **Property Names** column that are associated with the icon object selected from the **Icon Class Name** drop down list.
Map	Select one of three Map column options: **Default** - Select this option to use either the default value or the value specified in the style sheet (if a style sheet value is available) for the property. In this case, you do not need to set the Property Value. If the default value for this property changes in future releases of RTView, your icon object will reflect the new default value. **Value** - Select to specify a value. In this case, you need to set the value in **Property Value**. Specifying a value overrides any value defined in the style sheet. **Column** - Select to specify a column in the data source attached to the Object Grid's **valueTable** property. In this case, you need to select a column in **Property Value**. Specifying a column overrides any value defined in the style sheet.
Property Value	Select a value for the property from the drop down list. Options are populated based on the selections you make for Property Name and Map. For instance, when Map is set to **Column**, the **Property Value** drop down lists all columns from the table in **valueTable** as options to map to.
Allow multiple icon types	Check the box to enable. The **Add Icon** button appears. This allows you to use different graphic objects to represent different cells of data from each single row of data.
Add Icon	Click **Add Icon**, select an icon object from the **Add Icon** dialog drop down menu and click OK. The icon object is added to the list of available icon objects for the Object Grid.
Delete Icon	Removes the icon object from the list available to the Object Grid. If there are multiple icon objects in the list, properties of the selected icon is shown.

See the "Examples" section for an example of how to create an Object Grid with multiple icon objects.

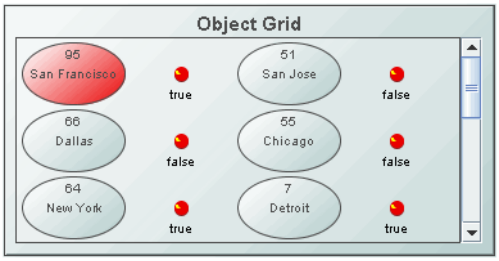
The following describes the **Icon Properties** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Examples

iconProperties

Objective: Create an object grid table using multiple icon objects



In this example, the object grid table will contain two icon objects. One icon object will display two columns of data from the data source: the **Units Completed** column and **Plant** name column. The other icon object will display data from the **On Schedule** column. The object grid table will look like this:

The data attachment for this exercise contains the following:

Plant	Units Completed	On Schedule
San Francisco	95	true
San Jose	51	false
Dallas	66	true
Chicago	55	false
New York	64	true
Detroit	7	true

To create the object grid table:

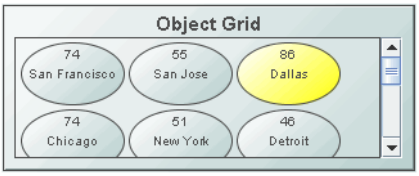
1. Attach an object grid table to the **valueTable** property.
2. Double-click on **iconProperties** (category: Icon). In the **Icon Properties** dialog:

   value - Select **Column** from the **Map** drop down menu. Select **Units Completed** from the **Property Value** drop down menu.

   label - Select **Column** from the **Map** drop down menu. Select **Plant** from the **Property Value** drop down menu.

   Click **OK**.

   We get the following results.



Notice that information from the **On Schedule** column is not displayed. We have not mapped it to an icon object. Also, notice that the numeric values have changed. This is because we are attaching to live data.

3. To add another icon object, double click on **iconProperties** (category: Icon).

4. In the **Icon Properties** dialog:

   Check the **Allow multiple icon types** box and click **Add Icon**.

   Select an icon object from the **Icon Class Name** drop down menu (in this example, **obj_indstatus_light**) and click **OK**. The icon object appears in the list of available icon objects for this object grid table.

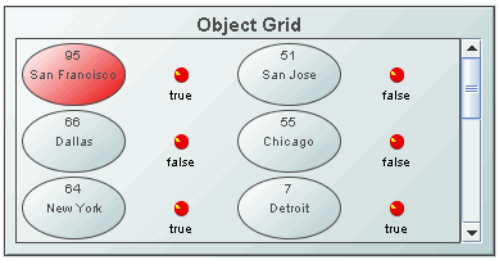
   With the icon object you just added selected in the list of available icon objects, edit the following properties:

   label - Select **Column** from the **Map** drop down menu. Select **On Schedule** from the **Property Value** drop down menu.

   status - Select **Column** from the **Map** drop down menu. Select **On Schedule** from the **Property Value** drop down menu.

   Click **OK**.

   We get the following results.



The display now shows three columns of data: the **Plant** and **Units Completed** columns are shown in the first icon object and the **On Schedule** column is shown in the second icon object.

Style Sheets

Style sheets allow you to set the values for object properties in your displays from one or more external (.rts) files. You can choose from a number of built-in style sheets or create your own or append your custom style sheets to built-in files in order to adjust the properties you want to change. See "Built-In Style Sheets" and "Creating Style Sheets" for more information.

You can also use style sheets to set the values of properties for your navigation controls, which makes it possible to standardize a default look and feel for all RTView applications or enable the deployment of several completely different look and feel styles that may be applied from the command line when starting RTView applications. See"Configuring Multiple Display Panels" for more information.

"Application Options Style Sheets"

Select **Tools> Options>General> Style Sheets** to manage style sheet (.rts) files applied to all displays in your applications.

Note: When multiple style sheet (.rts) files are applied, they are processed in the order specified. Therefore if the same property is specified in multiple style sheets, the value in the last style sheet applied will take precedence.

"Display-Specific Style Sheets"

In addition to application level style sheets, select **Tools> Style Sheet** to apply specific style values to individual displays.

Note: Display-specific style sheets are applied after application level styles sheets. Therefore if the same property is specified in both, the display-specific style sheet value will take precedence.

Disabling and Overriding Style Sheet Values

Style sheets are applied to the main Display Builder window (a.k.a Working Area) and in the Object Palette. To prevent display (.rtv) files from being saved with unwanted style values applied go to **Tools> Options> General> Style Sheet** and deselect the **Apply Style Sheets to Main Builder Window** check box.

In some cases, you might want to use a property value from your display (.rtv) file instead of the style sheet value. If a property value is set to apply a style sheet, then the property name is in italic. If you change that property (either by entering a value in the Object Properties dialog or by pasting properties onto the object) and save your display, the style sheet will no longer be applied to those properties. Property names with values that override the value set in the style sheet are in bold italic.

It is also possible to override the style sheet value without changing the property value by right-clicking on the property in the Object Properties dialog and selecting **Override Style Sheet**. To go back to using the style sheet value, right-click on the property and select **Use Style Sheet**.

To override all properties for an object defined in a style sheet, right-click on the object and select **Override Style Sheet**. To go back to using the style sheet values, right-click on the object and select **Use Style Sheet**.

Note: The options **Use Style Sheet** and **Override Style Sheet** are not applied immediately. You must save and reopen the display to accurately determine how style sheet properties are being applied.

Creating Style Sheets

Style sheets allow you to set the values for object properties in your displays from one or more external (.rts) files. You can choose from a number of built-in style sheets or create your own. See "Built-In Style Sheets" for more information.

Note: Style sheet file names must use a **.rts** extension.

Note: File names starting with **rtv_** are reserved for SL use only.

The syntax of an .rts style sheet is very similar to that of a .css file:

```
styleClassName {
        propertyName:propertyValue;
```

```
        propertyName2:propertyValue2;
}
styleClassName2 {
        propertyName:propertyValue;
}
styleClassName3,styleClassName4,styleClassName5 {
        propertyName:propertyValue;
}
```

If a property value contains any of the following characters, the entire value must be enclosed in quotations:

```
{} ; :
```

For example:

```
obj_bargraph {
    barProperties:"228,;46,;53,;101,;118,;161,;18,";
}
```

To add a comment to your style sheet, start the line with a #. For example:

```
# this is a comment
```

Style Class Names

By default, an object's style class name is it's object class name. The object class name is noted at the top of the Object Properties dialog.

Note: You must add an object to a display to view it's Object Properties.

Note: Specify m_basemodel as the style class name to set style values for the background model.

For example, look at style values set on an object from the General tab with the object class name obj_rect_il:

```
obj_rect_il  {
    bgColor: 1;
    bgGradientMode:0;
}
```

All obj_rect_il objects will have a red background and no gradient.

To apply a variety of styles to different instances of the same object class, edit the object's styleClass property in the Object Properties dialog. The value entered for styleClass must not contain spaces and must not start with **rtv-**.

For example, suppose you want a few instances of the obj_rect_il to have a different background (bgColor). For those objects, you can enter a styleClass value of **specialRect** and add it to your style sheet as follows:

```
specialRect  {
        bgColor: 7;
}
```

Objects with styleClassName set as specialRect will use bgColor 7, all other obj_rect_il objects will use bgColor 1.

Style Groups

In addition to style classes, there are three style groups which allow you to set properties on a group of objects:

- **rtv-all** -- Includes all objects (except the background model)
- **rtv-graphs** -- Includes all objects on the Graphs tab of the Display Builder Object Palette
- **rtv-controls** -- Includes all objects on the Controls tabs of the Display Builder Object Palette

If an object is in one or more style groups, styles are applied in order of the most specific reference. For example, suppose you have a bar graph with the **styleClassName** set to special_bar.

```
rtv-all {
            bgColor:7;
            labelTextColor:1;
            labelTextFont:2;
}
rtv-graphs {
            bgColor:2;
            labelTextColor:0;
}
obj_bargraph {
            bgColor:10;
            labelTextFont:7;
}
special_bar {
            labelTextColor:9;
}
```

All non-graph objects will use bgColor 7, labelTextColor 1 and labelTextFont 2. All graphs, except the bar graphs, will use bgColor 2, and labelTextColor 0. All bar graphs, except the special_bar graphs, will use labelTextFont 9, and all bar graphs, including the special_bar graphs, will use bgColor 10. The special_bar graphs will use labelTextColor 9.

If the same styleClassName or style group is specified more than once, or appears in more than one style sheet, the properties are merged. If the same property on the same object is specified more than once, the last reference takes precedence. For example, the following two style sheets end up applying the same style:

Style Sheet 1	Style Sheet 2

rtv-all {
bgColor:7;
labelTextColor:1;
labelTextFont:2;
}
rtv-graphs {
bgColor:2;
labelTextColor:0;
}
obj_bargraph {
bgColor:10;
labelTextFont:7;
}
rtv-graphs {
bgColor:12;
bgStyle:2;
}
obj_bargraph {
bgColor:20;
}
	rtv-all {
	bgColor:7;
	labelTextColor:1;
	labelTextFont:2;
	}
	rtv-graphs {
	bgColor:12;
	labelTextColor:0;
	bgStyle:2;
	}
	obj_bargraph {
	bgColor:20;
	labelTextFont:7;
	}

Object Properties

To view available properties for a particular object, add that object to a display in the Display Builder and look at the Object Properties list.

Some object properties are only available if others are selected. For example, **bgColor** is only available if the **bgVisFlag** check box is selected. When including properties like this in a style sheet, always include the controlling property (**bgVisFlag**) before the dependent property (**bgColor**).

The following Object Properties are not supported in style sheets:

- anchor
- command
- commandCloseWindowOnSuccess
- commandConfirm
- commandConfirmText
- dock
- drillDownTarget
- historyTableName
- historyTableRowNameFlag
- objName
- objX
- objY
- styleClassName
- varToSet

- all properties in the Composite category on the composite object (i.e. obj_composite)

Many property values are not formatted in style sheets as they are displayed in the Object Property list or require a dialog in the Display Builder to configure.

To determine the correct method of formatting of a property value within a style sheet, do one of the following:

- For color properties, add the object in the Display Builder and edit the selected color property. In the Color Dialog, hover the mouse over the color you want to use to see the tool tip. The first item in the tool tip is the color index number. This color index number should be used in the style sheet. For example if you hover over the first color (white), you will see it's index number is 0. Use a value of 0 to specify white.
- For properties where the Display Builder presents a list of options, such as bgGradientMode, add the object in the Display Builder and edit the selected property. The list will show the index number as well as the option, as follows:

  0 - None

  1 - Diagonal Edge

  2 - Diagonal Center

  3 - Horizontal Edge

  4 - Horizontal Center

  5 - Vertical Edge

  6 - Vertical Center

  The index number listed should be used in the style sheet. For example, use a value of 1 to specify a Diagonal Edge on the bgGradientMode property. NOTE: If the list does not show an index number, use the text value from the list.
- For properties where the Display Builder presents a dialog, add the object in the Display Builder and edit the selected property. Use the dialog to configure the property the way that you want it in your style sheet. Click OK to close the dialog, then select the text from the Property Value field in the Object Properties dialog and copy (Ctrl+C). For example if you select to edit barProperties on the Bar Graph, a dialog will open. If you choose white with the first fill pattern for the first bar and click OK, the Property Value field will contain the following text: 0,1;;;;;;. Paste this value into your style sheet.
- For properties where the Display Builder presents a check box, use 0 for false (deselected) and 1 for true (selected).
- For properties where the Display Builder allows you to type in a value, enter the same value in your style sheet.
- For properties that end in Height or Width that are specified in the Display Builder as pixels, but stored using internal coordinates, divide the pixel value by 8. The following properties are exceptions to this rule:

  a. If a Width property name includes one of the following, do NOT divide by 8:

  EdgeWidth e,g, bgEdgeWidth

  edgeWidth e.g. edgeWidth

  MinLabelWidth e.g. yAxisMinLabelWidth

  MeterMinWidth e.g. vuMeterMinWidth

  MinTabWidth e.g. labelMinTabWidth

  b. If a Height property name includes the following, do NOT divide by 8:

MinLabelHeight e.g. horizAxisMinLabelHeight

Application Options Style Sheets

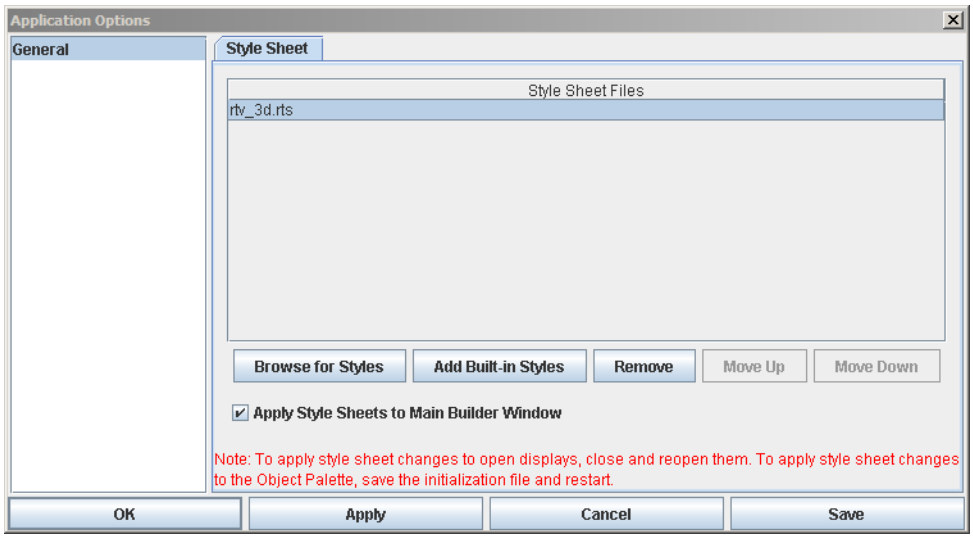
Select **Tools>Options>General>Style Sheet** to add, remove or reorder style sheets applied to your applications. It is also possible to apply specific style sheets to a single display. See "Display-Specific Style Sheets" for more information.

Note: In the Display Builder, style values are also applied to the Object Palette as well as objects added from the Display Builder toolbar.

Application level style sheets are read once, at startup, and applied when display (.rtv) files are opened. If you edit a style sheet, then you need to restart to see those changes. If you add, remove or re-order style sheets, changes will not be applied to open displays. To see these changes, close and reopen your displays. To see changes in the Object Palette you must always restart.

Note: In the following cases, style changes are applied immediately:

- Objects added from the Display Builder toolbar
- Objects added to Object Grids
- Objects added to Composites



Field Name	Description
Browse for Styles	Locate a specific style sheet (.rts) file.
Add Built-in Style	Choose from available built-in style sheets. See "Built-In Style Sheets" for more information.
Remove	Select a style sheet (.rts) file from the Style Sheet Files list and click to remove.

Move Up / Move Down	Re-order the Style Sheet Files list. **Note:** When multiple style sheet (.rts) files are applied, they are processed in the order specified. Therefore if the same property is specified in multiple style sheets, the value in the last style sheet applied will take precedence.
Apply Style Sheets to Main Builder Window	If selected, style sheets are applied to the main Display Builder window (a.k.a Working Area) and in the Object Palette. To prevent display (.rtv) files from being saved with unwanted style values applied you can choose to deselect this option.

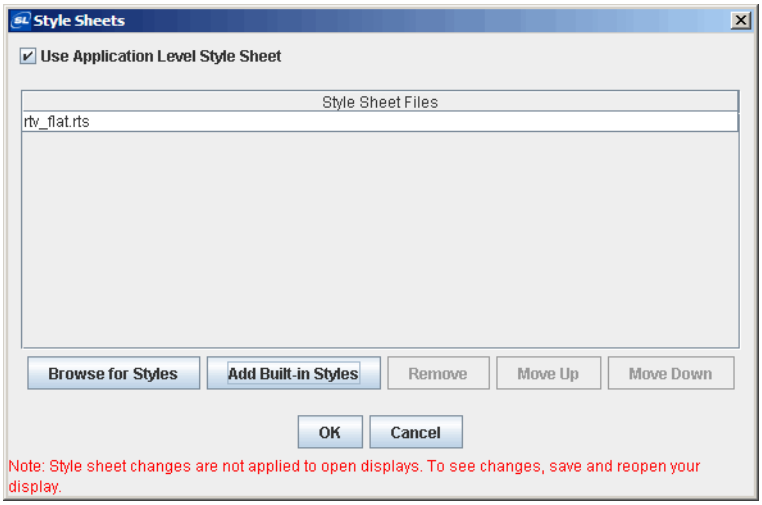
Display-Specific Style Sheets

Select **Tools>Style** to apply specific style sheets to a single display. To select style sheets to apply to all displays in your applications, select **Tools>Options>General>Style Sheet**. See "Application Options Style Sheets" for more information.

Display-specific style sheets are read once, the first time they are referenced in a display, and applied as subsequent display (.rtv) files are opened. If you edit a style sheet, then you need to restart to see those changes. If you add, remove or re-order style sheets, changes will not be applied to open displays. To see these changes, close and reopen your displays.

Note: In the following cases, style changes are applied immediately:

- Objects added from the Display Builder toolbar
- Objects added to Object Grids
- Objects added to Composites



Field Name	Description
Browse for Styles	Locate a specific style sheet (.rts) file.
Add Built-in Style	Choose from available built-in style sheets. See "Built-In Style Sheets" for more information.
Remove	Select a style sheet (.rts) file from the Style Sheet Files list and click to remove.

Move Up/Move Down	Re-order the Style Sheet Files list. **Note:** When multiple style sheet (.rts) files are applied, they are processed in the order specified. Therefore, if the same property is specified in multiple style sheets, the value in the last style sheet applied will take precedence.
Use Application Level Style Sheet	If selected, application level style sheets will be applied to this display. If not, application level style sheets will not be applied to this display. See "Application Options Style Sheets" for more information. **Note:** Display-specific style sheets are applied after application level styles sheets. Therefore if the same property is specified in both, the display-specific style sheet value will take precedence.

Built-In Style Sheets

Instead of modifying built-in style sheets, it is recommended that you append custom style sheets to them and adjust the properties you want to change. For example, **rtv_darkstyles.rts** specifies a color scheme and sets the background style (**bgStyle**) to (**flat**) Rectangle. The **rtv_3d.rts** style sheet sets the **bgStyle** to **3D Rectangle**, but specifies no color scheme. Since **rtv_3d.rts** does not specify colors, it can be appended to **rtv_darkstyles.rts** to apply the **3D Rectangle** background style to all objects.

Note: When multiple style sheet (.rts) files are applied, they are processed in the order specified. Therefore if the same property is specified in multiple style sheets, the value in the last style sheet applied will take precedence.

The following style sheets are included with RTView:

rtv_3d.rts	Use in conjunction with any other style sheet to set background style (bgstyle) of all objects to **3D Rectangle**.
rtv_darkstyles.rts	Uses a dark color scheme with the background style (**bgStyle**) set to (**flat**) Rectangle. Use in conjunction with **rtv_3d.rts** or **rtv_rounded.rts** to modify the **bgStyle** of all objects.
rtv_default.rts	Uses default RTView color scheme. Objects appear just as they do in the Display Builder's Object Palette when no style sheet is applied. Use in conjunction with **rtv_flat.rts** or **rtv_rounded.rts** to modify the **bgStyle** of all objects.
rtv_flat.rts	Use in conjunction with any other style sheet to set background style (**bgStyle**) of all objects to (**flat**) **Rectangle**.
rtv_rounded.rts	Use in conjunction with any other style sheet to set the background style (**bgStyle**) of all objects to **Rounded Rectangle**.

Note: Copies of these style sheet (.rts) files can be found in demos/features.

Drill Down Displays

Drill Down Targets

The ability to assign drill down targets to objects allows you to build a customizable hierarchy of displays. Any display (.rtv) file can be targeted as a drill down. Once a drill down target has been set, double-click on the object in the Display Builder or single-click in the Display Viewer to activate the drill down.

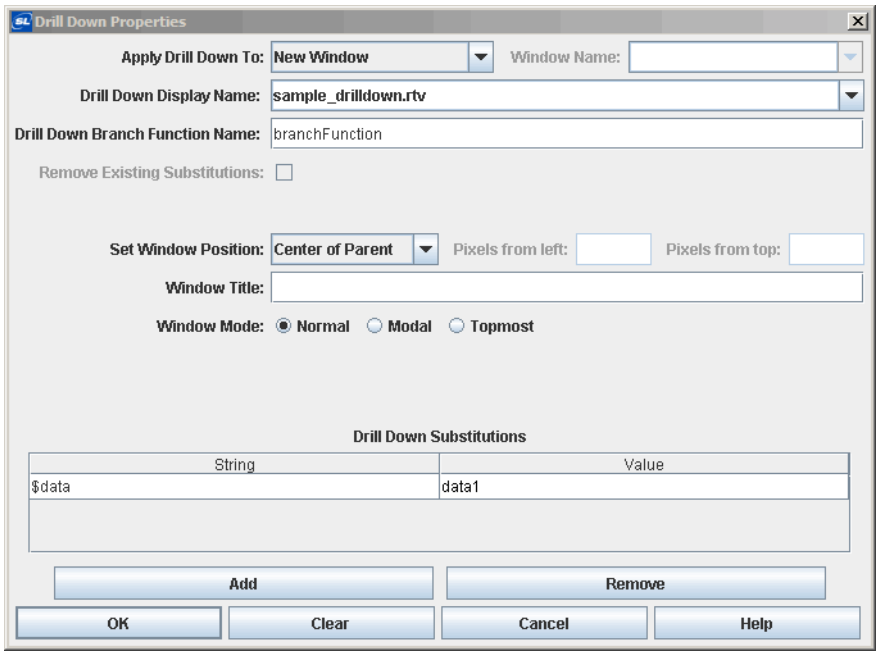
Note: To activate drill downs in the Display Viewer with a double-click, select **Tools>Options** in the Display Builder. If your "Login" doesn't allow you to view a particular drill down display, the drill down display will not open when you click on the object.

Drill down displays can be activated in the same window that contains the source object or open in a separate window. The file **drilldown.rtv** (located in the tutorials directory) is provided as an example of the different ways that drill down targets can be utilized in a display.

If a command (see "Define/Execute Command" for details) has been assigned to an object, then you must right-click and select Drill Down from the popup menu to open the targeted display.

Drill Down Properties

In the **Object Properties** window, double-click on **drillDownTarget** in the **Property Name** field to bring up the **Drill Down Properties** dialog.



Field Name	Description
Apply Drill Down To	**New Window*** - Open the targeted display file in a new display window. A new window will be created each time this drill down is activated. **Current Window** - Open the targeted display file in same window as the source object. When viewing in tabbed panels, selecting Current Window in the Apply Drill Down To field (in the Drill Down Properties dialog) opens your specified display in another tab if the display is already open. If the specified display is not already open, it will open in the selected tab. See "Multiple Display Panels" for more information. **Named Window*** - Open the targeted display file in a separate window defined by a specific name. The same window will be reused each time this drill down is activated or if you activate another drill down with the same window name. If you choose this option, you must also enter a Window Name.
Window Name	Enter a name for the window. The same window will be reused for all drill down targets that reference this window name. Entering main as a Window Name will open the targeted display in your top level window. When viewing multiple display panels, main will open the drill down in *panelcenter*. See "Multiple Display Panels" for more information. **Note:** This field is only valid if the drill down is applied to a Named Window*.
Drill Down Display Name	Select the name of the targeted display (.rtv) file. The drop down menu contains the names of files located in the current directory, as well as files located in first level of subdirectories. If a display is not listed, enter the name (including relative path) of the file. If the file path is a URL and it contains spaces, the spaces must be replaced with %20. Select **Current Display** to target the display that is currently in the target window. This is most useful when Current Display is used in conjunction with Current Window or Named Window. Only substitutions specified in the Drill Down Properties dialog will be applied when the drill down is activated and this allows you to use the source object to control data displayed by all objects in the window. To attach the Drill Down Display Name to data, right-click and choose **Attach to Data** or double-click in the field.
Drill Down Branch Function Name	Enter the name of a function (in your current display) that returns the text string you want appended to the end of the Drill Down Display Name. This enables to you drill down to different displays based on the result of the function. Functions make it possible to perform calculations on your data before it displayed in RTView. For more information, see the Add/Edit Functions section. See "Functions" for more information. **Note:** If the **Drill Down Display Name** is set to **Current Display** this option will not be enabled.
Remove Existing Substitutions	Select the **Remove Existing Substitutions** check box to remove existing drill down substitutions on the drill down window. **Note:** This option is only enabled when you drill down to the Current Window or a Named Window.

Window Position

Set the position of a new drill down window.

Note: This option only applies when the drill down opens in a new window.

There are five Window Position options:

Default Positioned by OS window manager.

Center of Screen -- Center on the screen the parent window occupies.

Center of Parent -- Center relative to the parent window.

Relative to Screen -- Position on the screen the parent window occupies, offset from the top left corner by the number of pixels specified in **Pixels from left** and **Pixels from top**.

Relative to Parent -- Position relative to the parent window, offset from the top left corner by the number of pixels specified in **Pixels from left** and **Pixels from top**.

Window Title

Specify text in the title bar. If this field is left blank, then the title bar will display **application name+drill down display file (.rtv) name**.

Note: This option only applies when the drill down opens in a new window.

To attach the Window Title to data, right-click and choose **Attach to Data** or double-click in the field.

Window Mode

Specify modality and stacking order of drill down windows.

Note: This option only applies when the drill down opens in a new window.

There are three Window Mode options:

Normal -- Allow user interaction in all RTView windows. Stacking order is determined by the Drill Down Windows **Always on Top** setting in the **General** tab of the "Application Options Dialog".

Modal -- Allow user interaction only in this drill down window while it is open. Stacking order is on top of all other RTView windows.

Topmost -- Allow user interaction in any RTView window. Stacking order is on top of all other RTView windows. Additionally, all windows targeted from a **Topmost** window will automatically assume the topmost position.

Note: Some platforms do not support this functionality. If more than one window is set to be in the **Topmost** position, stacking order is platform dependent.

Drill Down Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. In this way, a single display can be reused to show data from a number of different sources. The ability to specify substitutions for drill down displays allows you to design a multi-level hierarchy of parameterized displays. For more information, see the "Substitutions" section.

Note: Some drill down substitutions are automatically added for displays targeted from table objects.

Add -- Click **Add** to create a new row in the Drill Down Substitutions table. Enter a **String** (e.g. $data) and a **Value** (e.g. $data1).

To attach a **String** or **Value** to data, right-click on the desired cell in the Drill Down Substitutions table and select Attach to Data.

Note: Substitution strings cannot contain the following:

: | . tab space , ; = < > ' " & / \ { } [ ] ( )

Remove -- Select a substitution from the table and click **Remove** to delete.

Note: Drill down displays opened in a New Window, or a Named Window (with the exception of main) do not feature menus or a toolbar. Object Properties can be accessed within these drill down displays, but editing is not allowed. You can copy and paste an object, or properties of an object, from these drill down displays into the top level display, however no object or property values can be pasted into these drill down displays. It is possible to double-click on objects within drill down windows to access further displays.

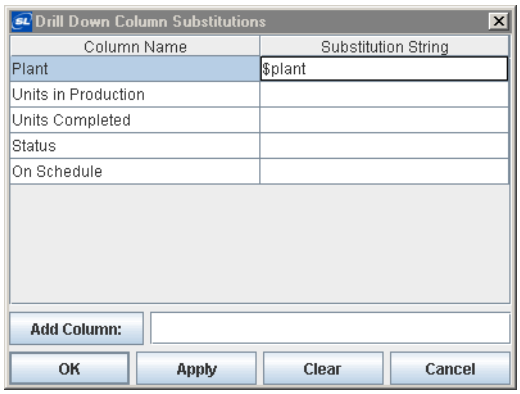
The following describes the **Drill Down Properties** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Clear	Clears all fields. Removes drill down target (once OK is selected).
Cancel	Closes the dialog with last values applied.
Help	Open the Help dialog.

Drill Down Column Substitutions Dialog

In many objects that display tabular data, it is possible to specify which column values will be passed as substitutions into drill down displays. In the **Object Properties** window, double-click on **drillDownColumnSubs** in the **Property Name** field to display the **Drill Down Column Substitutions** dialog.

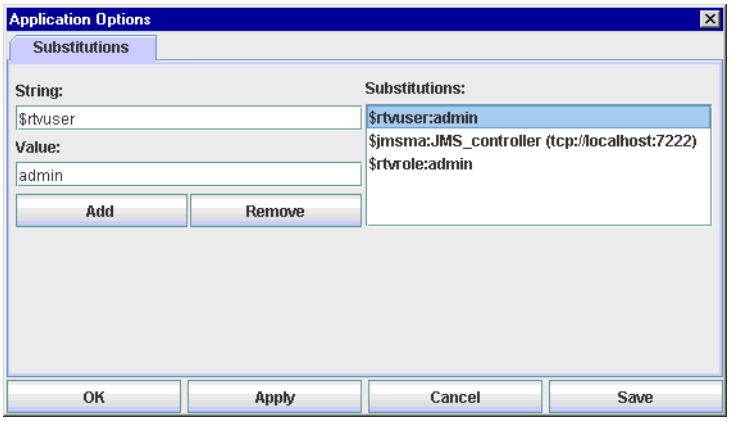
Once you have selected which column values to pass in as substitutions, double-click on any element in your object to open a drill down window that displays corresponding values.



Field Name	Description
Column Name	This list is populated based on the table's data attachment. If you have not yet attached the table to data, this list will be empty.
Substitution String	Enter the substitution string for the column it applies to and press <Enter>. **Note:** Substitution strings cannot contain the following: : | . tab space , ; = < > ' " & / \ { } [ ] ( )
Add Column	Enter the name of the column and click the **Add Column** button to insert a column into the table.
Clear	Click the **Clear** button to remove all substitution strings listed.

Substitutions

Substitutions allow you to build open-ended displays in which data attachments and commands depend on values defined at the time the display is run. In this way, a single display can be reused to show data and execute commands from a number of different sources. When data attachments are created in the Display Builder, generic values are used instead of the actual value of any field in the Attach To Data and Define Command dialogs. Later when the display is running, these generic values are defined. Substitution values can either be defined for all displays or for a single drill down display.



To specify a value for substitution, select **Tools>Options**. Click the **Substitutions** tab to add, remove or edit substitution values. These values are inherited by drill down displays. Substitutions specified in the "Application Options Dialog" can be saved to **OPTIONS.ini**. For more information on how to use the **Application Options** dialog to specify substitutions, see "Application Options".

Note: The substitution string **$value** is reserved for internal use.

Note: Substitution strings cannot contain the following:

: | . tab space , ; = < > ' " & / \ { } [ ] ( )

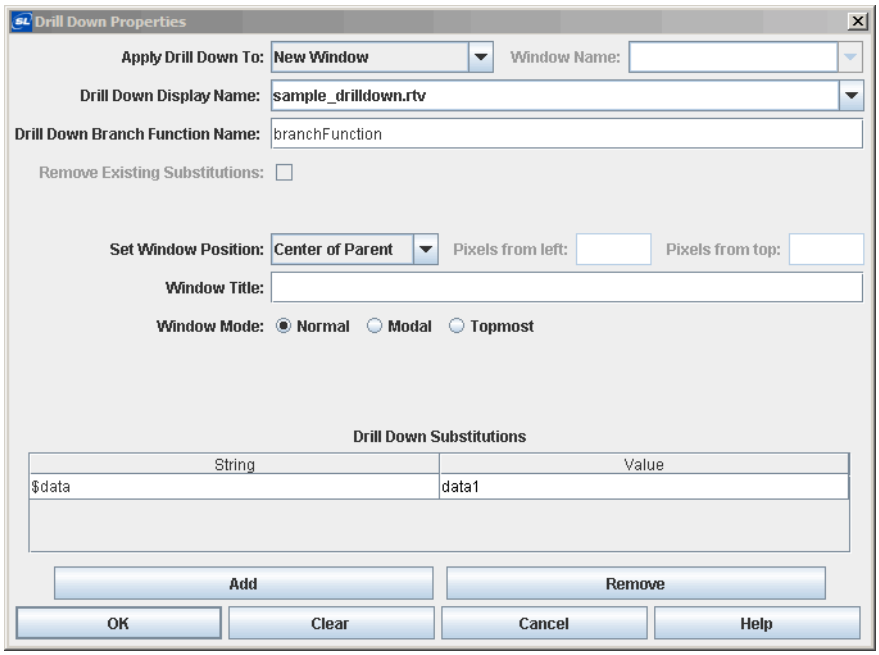
Linking Substitutions with Variables

Substitutions can also be set by adding or editing variables. To link a variable with a substitution by the same name, select **Use As Substitution** when creating a variable. If no substitution by that name exists, one will be created and given the value of the variable. If a substitution by that name already exists, the variable will assume the value of the substitution instead of the value entered in the **Initial Value** field. Whenever the value of the variable is updated, the associated substitution will also be updated and when the substitution is updated the associated variable will update. The **Use As Substitution** feature is not available when editing existing variables. See "Add/Edit Variables" for more information.

Note: If you create a variable in the **Attach to Variable Data** dialog and it begins with a $, the variable will be linked to a substitution with the same name.

Drill Down Substitutions

The ability to specify substitutions for drill down displays allows you to design a multi-level hierarchy of parameterized displays. To specify a substitution value for a drill down display, right-click on the source object in the main display and select Object Properties. In the Object Properties window, double-click on **drillDownTarget** in the **Property Name** field to open the "Drill Down Properties" dialog.



Some substitutions are automatically set on drill down displays that are activated from an object that displays tabular data (i.e., table or graph). The data attachment for the source object is used to create substitutions that will be passed into the drill down display. The row which corresponds to the selected element in the source object (i.e., cell, object, bar, or wedge) will be used to construct the substitution when the drill down is activated. Which substitutions are set on the drill down display depends on the data attachment of the source object. To customize which substitutions will be passed into drill down displays, double-click on **drillDownColumnSubs** in the **Object Properties** window to open the "Drill Down Column Substitutions Dialog".

For information on Substitutions for your data source, refer to the "RTView Data Sources" section of this documentation.

Data Source	Substitution Value	Definition
The following substitutions will be set no matter what data source is attached:	$col1	Data from first cell in selected row or object. **Note:** Substitutions set for **drillDownColumnSubs** will be used instead of $col1.
	$celldata	Data from selected cell. In an Object Grid this value will return the same data $col1.
	$colName	Name of the selected column.
If the source object is attached to Function data, the following substitutions will be set:	$XrowName (where X is the # of the rowname)	Data from a selected row or object. A substitution is defined for each part of the rowname. For example, if the rowname was plant1:load:station7:machine8 the substitutions would be: $1rowName:plant1 $2rowName:load $3rowName:station7 $4rowName:machine8
	$filterfield	Filter field name from the selected row or object.
	$filtervalue	Filter field value from the selected row or object.

For more information on objects that display tabular data, see the "Table Objects" and "Graph Objects" sections. To learn more about setting drill down targets, open the display file **drilldown.rtv** (located in **demos/tutorials**) for an example of the different ways drill down targets can be utilized in a display.

Attach to Data

From the **Object Properties** window you can access the **Attach to Data** dialog, which is used to connect an object property to any active data source, as well as alerts, functions and variables. See "Attach to Alert Data", "Attach to Function Data", and "Attach to Variable Data" for more information. Depending on your version of RTView, there are several data sources available. For detailed information on the **Attach to Data** dialog for your data source, refer to the "RTView Data Sources" section of this documentation.

To bring up the **Attach to Data** dialog, right-click on a Property Name from the Object Properties window and select **Attach to Data** for your data source. Once a property has been attached to data, it receives continuous updates.

When an object property is attached to data, the Property Name and Value in the Object Properties window will be displayed in green. This indicates that editing this value from the Object Properties window is no longer possible. To remove the data attachment, and resume editing capability in the Object Properties window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

If an object property is attached to data, right-click on the Property Name and select **Display Data** to open a window that contains a table showing data available from that attachment. If the object property is attached to tabular data, then the number of columns and rows in the attached table will also be displayed.

Note: Only one **Display Data** window can be opened for each individual data attachment. Select **Show Column Types** to display the data type of each column. By default, old data is replaced by new data on each update. To insert new rows to display incoming data, select **Insert New Rows**. Select the **Scroll Columns** check box to set the width of all columns to fit the visible area.

Substitutions

The "Substitutions" feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. In this way, a single display can be reused to show data from a number of different sources. When data attachments are created in the Display Builder, generic values (i.e.: $sub1, $sub2, etc.) are used instead of the actual value of any field in the Attach To Data dialog. Later when the display is running, these generic values are defined. Substitution values can either be defined for all displays or for a single drill down display.

Row Filtering

In Attach to Data dialogs that feature a Filter check box, it is possible to indicate multiple columns for the filter and multiple values to compare against for each column. Multiple Filter Column names should be entered as a semicolon (;) delimited list. Multiple Filter Values should be entered as a nested list, where values for a given column are separated by commas within a semicolon (;) delimited list. For example:

Filter Column: col1;col2;col3

Filter Values: val1,val2;val3,val4;val5,val6

The resulting table would contain only those rows where the values of **val1** OR **val2** are in **col1** AND the values of **val3** OR **val4** are in **col2** AND the values of **val5** OR **val6** are in **col3**.

The following describes Attach to Data dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.

Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Define/Execute Command

There are several types of commands you can assign to objects with RTView, including custom commands. Commands can be setup to execute when you click on an object or automatically execute the command when the value of an object reaches a specified threshold.

Depending on which data sources are licensed in your RTView application, there are several commands that can be assigned to an object. For information on the Define Command dialog for your data source, refer to the "RTView Data Sources" section of this documentation.

Note: Some data sources do not support commands.

How your command is executed depends on the type of command, which are described in this section.

Commands

Commands work with the Display Builder, Display Viewer, and the Display Server. When you deploy RTView as a Thin Client Browser or Served Data deployment, depending on the command, it is executed either on the server or the client. See command descriptions for information on where commands are executed.

Define Command

For "Control Objects", assign a command by right-clicking on **actionCommand** in the Object Properties window. For all other objects, assign a **command** by right-clicking on command in the Object Properties window. Select **Define Command** and choose **SYSTEM**, **ALERT** (see "Define Alert Command"), **MULTIPLE** (see "Define Multiple Commands"), or your data source. For information on the **Define Command** dialog for your data source, refer to the "RTView Data Sources" section of this documentation.

Note: Some data sources do not support commands.

Execute Command

Once a command has been assigned to a control object, it is executed when the control object is activated. For more information on how each control object is activated, see the "Control Objects" section. For all other objects, double-click on the object in the Display Builder, single-click in the Display Viewer, or right-click and select **Execute Command** from the popup menu to execute the command. In the Display Server, commands execute on a single-click or by right-clicking and selecting **Execute Command** from the popup menu.

Note: Select **Tools>Options** to execute commands on non-control objects in the Display Viewer with a double-click. See "General Tab" for more information.

Multiple commands are executed as a series of individual commands that are launched in sequence.

Note: There is no guarantee that any individual command will have completed before the execution of the next command due to their potentially asynchronous nature.

It is possible to execute a command from an individual element in an object that displays tabular data (i.e., a cell in a table or a bar in a bar graph). Based on your data attachment, substitutions are created that will be applied to the command before it is executed. See the "Drill Down Substitutions" section for information on how to customize which substitutions will be passed into the command.

Command Confirmation

By default, the command confirmation dialog is disabled. To control this option for each individual object, use the **commandConfirm** check box in the **Object Properties** window. If confirmation is required for a multiple command, a single confirmation dialog is presented. If you confirm the execution, all individual commands will be executed in sequence with no further confirmation. If the you cancel the execution, none of the commands in the sequence will executed.

Use the **commandConfirmText** property to write your own text for the confirmation dialog. Otherwise, text from the command property will be used.

It is also possible to override the confirmation status of individual objects with an application wide policy. Select **Tools>Options** to choose from three confirmation values:

Policy	Description
Do not confirm	Indicates that no commands require confirmation (regardless of each object's confirmation status).
Confirm all	Indicates that all commands require confirmation (regardless of each object's confirmation status).
Use object confirm flag	Indicates that the confirmation status of each object will determine whether confirmation is required. This is the default policy.

Close Window Command

Use the **commandCloseWindowOnSuccess** check box to automatically close the window that initiates a system command when the system command is executed successfully. This applies to system commands only. With data source commands, the window is closed whether or not the command is executed successfully. For multiple commands, the **commandCloseWindowOnSuccess** is applied to each command individually. Therefore, if the first command in the multiple command sequence succeeds, the window will close before the rest of the commands are executed. See "Define Multiple Commands" for more information.

Note: The **commandCloseWindowOnSuccess** property is not supported in the Display Server.

Threshold Commands

Threshold commands work with the Display Builder, Display Viewer, and the Display Server. There are four objects that support threshold commands: **obj_circ2d_ilvx_ra4**, **obj_rect_ilvx_ra4**, **obj_circ2d_ilvx_da3**, and **obj_rect_ilvx_da3** (found in the **General** tab labeled either **Range Dynamic** or **Discrete Dynamic**). A threshold command can be defined using **valueCommandResetTrigger** and **valueCommandTimeout** on those objects. Both the range dynamic and the discrete dynamic objects have the thresholding functionality turned off by default. In the descriptions below, the object's value is the current value of the **value** property.

Range Dynamic

Range Dynamic objects allow you to setup your threshold so that if the object's value is within a range of numbers, the threshold functionality executes. To enable the high alarm threshold, check the **valueHighAlarmEnabledFlag**. This will enable several related properties. When the object's value is greater than or equal to the **valueHighAlarm** property, the background of the object will change to the **valueHighAlarmColor**, the bitmap on the image will change to the **valueHighAlarmImage** and the **valueHighAlarmCommand** will be executed. To enable the high warning threshold, check the **valueHighWarningEnabledFlag**. When the object's value is greater than or equal to the **valueHighWarning**, but less than the **valueHighAlarm**, the background of the object will change to the **valueHighWarningColor**, the bitmap on the image will change to the **valueHighWarningImage** and the **valueHighWarningCommand** will be executed. The low alarm and low warning threshold properties work the same way, but activate when the object's value is less than or equal to the **valueLowAlarm** or **valueLowWarning** properties.

Discrete Dynamic

Discrete Dynamic objects are similar to Range Dynamic objects, except that the threshold functionality is executed when the object's value equals the threshold value instead of when it is within the range of threshold values. To enable the high alert threshold, check the **valueHighAlertEnabledFlag**. When the value property equals the **valueHighAlert**, the background of the object will change to the **valueHighAlertColor**, the bitmap on the image will change to the **valueHighAlertImage** and the **valueHighAlertCommand** will be executed. To enable the medium alert threshold, check the **valueMediumAlertEnabledFlag**. When the value property equals the **valueMediumAlert**, the background of the object will change to the **valueMediumAlertColor**, the bitmap on the image will change to the **valueMediumAlertImage** and the **valueMediumAlertCommand** will be executed. The low alert threshold property works the same way, but activates when the object's value is less than or equal to the **valueLowAlert** property.

For both the Discrete Dynamic and Range Dynamic objects, threshold commands will only execute if the number of seconds specified in the **valueCommandTimeout** has elapsed since the previous threshold command was executed and the value has changed to another threshold. For example, if the high warning threshold range is between 40 and 50, when object's value enters this range, the **valueHighWarningCommand** will be executed. The **valueHighWarningCommand** will not be executed again until the object's value leaves the high warning threshold range and then re-enters it. If this occurs in less time than is specified in the **valueCommandTimeout** property, it will not be executed until that time elapses. If the **valueCommandTimeout** is set to 0, the internal default, 5 seconds, is used. However, if the value of **valueCommandResetTrigger** property is changed, then the object's threshold command will be executed again even if the value property has not changed. Changing the value of **valueCommandResetTrigger** when the object's value is not at or above a limit has no effect.

If the **webLabelFlag** property is checked on an object with a threshold command, the threshold command will be executed in a thin client deployment when appropriate. If the threshold command is a supported client-side command it will be executed in the browser. The client-side commands are: **Play Audio File**, **Drilldown / Set substitution**, **Execute Custom Command** (if the custom command has a javascript implementation), and **Open Browser**. All other commands are executed by the display server or (for a data source command) in the data server.

Define Threshold Command

To assign a threshold command, right-click on a threshold command property in the "Object Properties" window. Select **Define Command** and choose **SYSTEM**, **ALERT** (see "Define Alert Command"), **MULTIPLE** (see "Define Multiple Commands"), or your data source.

Execute Threshold Command

Once a threshold command has been assigned to an object, it is executed when the current value of the object (based on the attached data source) reaches the specified threshold. Multiple commands are executed as a series of individual commands that are launched in sequence.

Note: There is no guarantee that any individual command will have completed before the execution of the next command due to their potentially asynchronous nature.

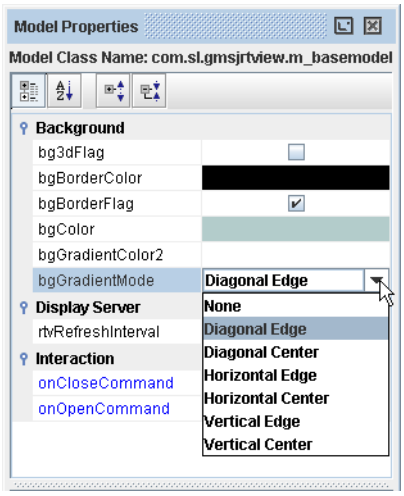
Open/Close Display Commands

You can define a command to be activated when a display (**.rtv**) file is opened or closed.

Define Open/Close Display Command

Select **Edit>Model Properties** and right-click on either **onCloseCommand** or **onOpenCommand**. Select **Define Command** and choose **SYSTEM**, **ALERT** (see "Define Alert Command"), **MULTIPLE** (see "Define Multiple Commands"), or your data source.

Note: The **SYSTEM** drill down command is not supported for **onCloseCommand** and **onOpenCommand**.



Execute Open/Close Display Command

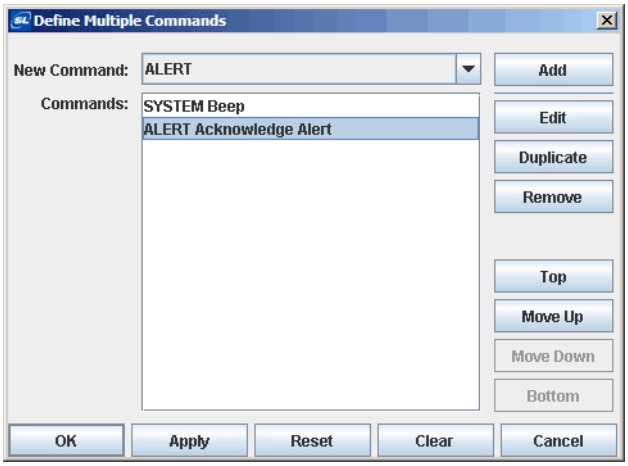
Open Display and **Close Display** commands are executed when the display (.rtv) file (where you set the Model Properties) is opened/loaded or closed/replaced.

Multiple Commands

You can define multiple commands to execute when a command is activated.

Define Multiple Commands

To define multiple commands, right-click on a **command property** in the Object Properties window. Select **Define Command** and choose **MULTIPLE**. To create multiple commands, click on the **Add** button. Select an existing command and click **Edit** to modify, **Duplicate** to copy, or **Remove** to delete.



Execute Multiple Commands

Multiple commands are executed as a series of individual commands that are launched in sequence. There is no guarantee that any individual command will have completed before the execution of the next command due to their potentially asynchronous nature. Therefore, you cannot count on a result of the first command to apply to the rest of the commands. Command failure dialogs will appear for any command in the sequence that fails.

Command Failure

Commands executed by a data source or a system command can return a dialog window or a console message indicating whether the command was successful. Where and how the dialog window appears depends on which type of command is executed and the type of RTView deployment:

Application or Rich Client Browser with Direct Data Connection:

System command - Dialog window appears on the client

Data source command - Console message appears on the client. For the data sources that support error dialogs for command failure, a dialog will also appear on the client. See the Define Command section for your data source to see if it supports error dialogs.

Application or Rich Client Browser with Served Data:

System command - Dialog window appears on the client

Data source command - Message appears in Data Server console

Thin Client Browser with Served Data:

System command - Message appears in Display Server console

Data source command - Message appears in Data Server console

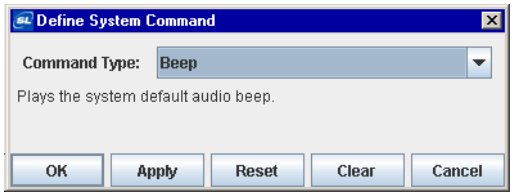
Thin Client Browser with Direct Data:

For both command types - Message appears in Display Server console

Define System Command

You can access the **Define System Command** dialog from the **Object Properties** window. This dialog is used to assign system commands allowing you to issue commands. Commands can be executed from within an RTView display in the client (the Builder, Viewer or Display Server). Additionally, several commands can be executed on a Data Server (the Execute Custom Command, Execute URL, Run DOS Command or Unix Shell, Send Email and Send SNMP Trap commands). For details, see "Execute Commands on a Data Server". By default, commands execute on the client.

To bring up the **Define System Command** dialog, right-click on the appropriate command property in the **Object Properties** window and select **Define Command>SYSTEM**. The **Define System Command** dialog provides a drop down menu with available commands. When a command can execute on a Data Server, the Data Server drop down menu appears at the bottom of the dialog. Argument fields vary based on the selected command type. See the "Define/Execute Command" section for information on how to execute a command.



Command Types

The following commands are executed in the client or, where indicated, can also be redirected to a Data Server.

Command Type	Description
Beep	Plays the system default audio beep. This command does not work in the Display Server. **Note:** If you execute this command from one of the Served Data deployments, this command will execute on the client.
Close Window	Closes the window specified in the **Window Name** argument. If no Window Name is given it closes the window containing the object that generated the command. This command cannot be used to close the main window. See "Drill Down Displays" for more information. **Note:** If you execute this command from a Thin Client with Direct Data Connection or any Served Data deployment, this command will execute on the client.

Drill Down or Set Substitution	Drill down to a new display or set substitutions on the current display. Click on **Edit Drill Down Target** to open the "Drill Down Properties" dialog and define your drill down or substitution. Substitutions are not applied to this command before it is executed. In the Drill Down Properties dialog, to attach one of the following fields to data, right-click and choose **Attach to Data** or double-click in the field: • Drill Down Display Name • Window Title • Drill Down Substitution Value **Note:** If you execute this command from a Thin Client with Direct Data Connection or any Served Data deployment, this command will execute on the client.
Execute Custom Command	Executes the custom command in the Command Name field with the specified Command Value. You may define your own custom commands in the Custom Command Handler. The specified Command Name and Command Value will be passed into the **invokeCommand** method in the Custom Command Handler. If you execute this command in a Thin Client deployment, it will execute on the server if implemented in Java and on the client if implemented in JavaScript. See "Define Custom Command" for more information. **Note:** If you execute this command from one of the Served Data deployments, this command will execute on the server. Otherwise, this command executes on the client. This command can also execute on a Data Server. See "Execute Commands on a Data Server" for more information.
Execute URL	Executes the specified URL String. This can be used to execute JSP, ASP or CGI scripts. This command will not open the specified URL in a browser. **Note:** If you execute this command from a Thin Client with Direct Data Connection or any Served Data deployment, this command will execute on the client. This command can also execute on a Data Server. See "Execute Commands on a Data Server" for more information.
Open Browser	Opens the specified URL in a browser window or a new tab. Select **Current**, **Parent**, **Top**, **New**, or **Named** in the Browser Window property to set in which browser window the page will display. **New** is the default setting. If you specify **Browser Window** as **Named** and **Window Name** as **_tab**, then the URL will open in a new browser tab (in the thin client). **Note:** When trying to open the URL in a new tab, your browser's settings must be configured to automatically open a new tab for popup windows. If your browser's settings are configured to open popups in a new window, then the Open Browser command will open the URL in a (new) browser regardless of the settings. **Note:** If you execute this command from a Thin Client with Direct Data Connection or any Served Data deployment, this command will execute on the client.

Play Audio File	Plays the specified audio file. The File Name should include a relative path from your RTView display to the audio file. The support for audio file formats varies by platform, as follows. • The Play Audio File command is not supported in Internet Explorer version 8 and older. • In a desktop browser (Internet Explorer 9 or newer, Firefox, Chrome), .wav and .mp3 files are supported. However, Internet Explorer uses the Windows Media Player to play .wav files, which may require user confirmation or may fail due to security settings. • In mobile browsers (for example, Safari on iOS) only .mp3 files are supported, and the user must click on the **Enable Audio** button that appears in a display the first time an audio command is executed. • The **Beep** command is still not supported in the thin client. **Note:** Only .wav files are supported by the Play Audio File command in the builder/viewer. **Note:** If you execute this command from one of the Served Data deployments, this command will execute on the client.
Run DOS Command or UNIX Shell	Executes the specified DOS command or UNIX Shell. This will be executed in the directory where you started RTView. **Note:** This command does not work in the Display Server. This command can also execute on a Data Server. See "Execute Commands on a Data Server" for more information.
Send Email	Sends the specified email message. **Note:** Multiple addresses or attachments should be separated by spaces or commas. To use the Send Email system command you must modify the Java security settings on each client to include the following permissions: **permission java.net.SocketPermission "SMTPHostName", "resolve";** **permission java.net.SocketPermission "SMTPHostIP", "accept, connect, listen, resolve";** **permission java.util.PropertyPermission "user.name", "read"; };** Where **SMTPHostName** is the name of the SMTP server, **SMTPHostIP** is the IP address of the SMTP server. Contact your system administrator for SMTP Host address and SMTP Port number. SMTP Port field is optional. If left blank, SMTP Port 25 is used. User and Password fields are optional, unless authenticated mail sever access is required. This command executes on the server. This command can also execute on a Data Server. See "Execute Commands on a Data Server" for more information.
Send SNMP Trap	Enables a display to send an SNMP Trap message, such as would be sent from an SNMP Agent to a SNMP Management Station. Requires outgoing access to UDP port 162 (or a different port as specified in your command). **Note:** If you execute this command from a thin-client deployment it will execute on the server, otherwise it will execute on the client. This command can also execute on a Data Server. See "Execute Commands on a Data Server" for more information.

Use the following fields to define your SNMP Trap Message:

Trap Type --SNMP version of the trap.

Destination Address -- System name or IP address of the receiving system. Default value is 127.0.0.1 (localhost). To attach the Destination Address to data, right-click and choose **Attach to Data** or double-click in the field.

Destination Port -- UDP port on the receiving system. Default value is 162. Value must be an integer greater than 0. To attach a Destination Port to data, right-click and choose Attach to Data or double-click in the field.

Community Name -- SNMPv2 Community Name string. Default value is public. To attach the Community Name to data, right-click and choose Attach to Data or double-click in the field.

Enterprise OID -- SNMP Object ID to be used as the sender's Enterprise. Default value is 1.3.6.1.6.3.1.1.5.

Generic Trap -- List of generic traps.

Specific Trap --List of specific traps available will depend on the property for which you are defining a command.

The Generic Trap selected must be **enterpriseSpecific** to enable this menu.

Currently the only Specific Trap option is Alert, which is available if you are defining a command for the **alertCommand** property. The Alert trap is sent with the name **rtviewAlert** and includes the following MIB definition:

```
rtview OBJECT IDENTIFIER ::= { sl 1 }
 rtviewNotifications OBJECT IDENTIFIER ::= { rtview 1 }
 rtviewAlert NOTIFICATION-TYPE
 OBJECTS {
 rtviewAlertName,
 rtviewAlertIndex,
 rtviewAlertTime,
 rtviewAlertLastUpdateTime,
 rtviewAlertSeverity,
 rtviewAlertText,
 rtviewAlertID,
 rtviewAlertLabel,
 rtviewAlertCommandText,
 rtviewAlertCurrentValue,
 rtviewAlertComparisonValue
 }
 ::= { rtviewNotifications 2 }
```

Note: Traps sent by RTView use Enterprise OID private.enterprises.sl (.1.3.6.1.4.1.34605).

Execute Commands on a Data Server

The **Data Server** drop down menu appears at the bottom of the dialog when a command can be executed on a Data Server. Drop down menu options are:

Option	Description
none	Sets the command to execute on the client. This is the default.
default	Sets the command to execute on the default Data Server. If there is no connection defined for that Data Server, the command is not executed.

Enter a named Data Server	Enter a named Data Server in the Data Server field on which to execute the command. If there is no connection defined for that Data Server, the command is not executed.
Substitution	Enter a substitution in the Data Server field, such as **$server**.

For example:

$server:	An empty string sets the command to execute on the client.
$server:__none	**__none** sets the command to execute on the client. This value starts with two underscore characters.
$server:__default	**__default** sets the command to execute on the default Data Server. If there is no connection defined for that Data Server, the command is not executed. This value starts with two underscore characters.
$server:SalesDS	A named Data Server sets the command to execute on the named Data Server. If there is no connection defined for that Data Server, the command is not executed.

Substitutions

Substitutions allow you to build open-ended displays in which commands depend on values defined at the time the display is run. Generic names are used instead of arguments for specific commands. Later when the display is running, these generic values are defined by the actual arguments. In this way, a single display can be reused to issue a number of different commands. Substitutions are not applied to Drill Down or Set Substitution commands. For more information on creating displays using substitution values, see "Substitutions".

Special Values

$value	When an **actionCommand** is executed $value is replaced with the value from the control. This value may be used in any field in the Define System Command dialog. **Note:** This value may only be used for Action Commands. See "Commands" for more information.

The following describes **Define System Command** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

File Options

Working with Files

Creating a New Display (.rtv) File

To create a new display, click the New button  on the toolbar (or **Ctrl-N**). This will open a new file with assigned default properties, including background and model width and height.

Opening a Display (.rtv) File

To open a display, click the Open button  on the toolbar (or **Ctrl-0**). From the Open dialog, either double-click on a file name or enter the file name of a specific display (the .rtv extension is not required) and click Open. The last four displays you've opened will be listed in the File menu.

Note: If your "Login" doesn't allow you to view a particular display, the display will not open when you select or enter the file name in the **Open** dialog.

Saving a Display (.rtv) File

To save a display, click the Save button  on the toolbar (or **Ctrl-S**). If the display file is new or if it does not have write permission, the Save dialog will appear. Type in a name for the display file and click Save. Display files are saved with the **.rtv** extension, which is added automatically if you do not enter it.

Note: The following characters can be used in display file names:

- ASCII alpha-numeric (a-z, 0-9)
- . (dot)
- - (dash)
- _ (underbar)
- (space)

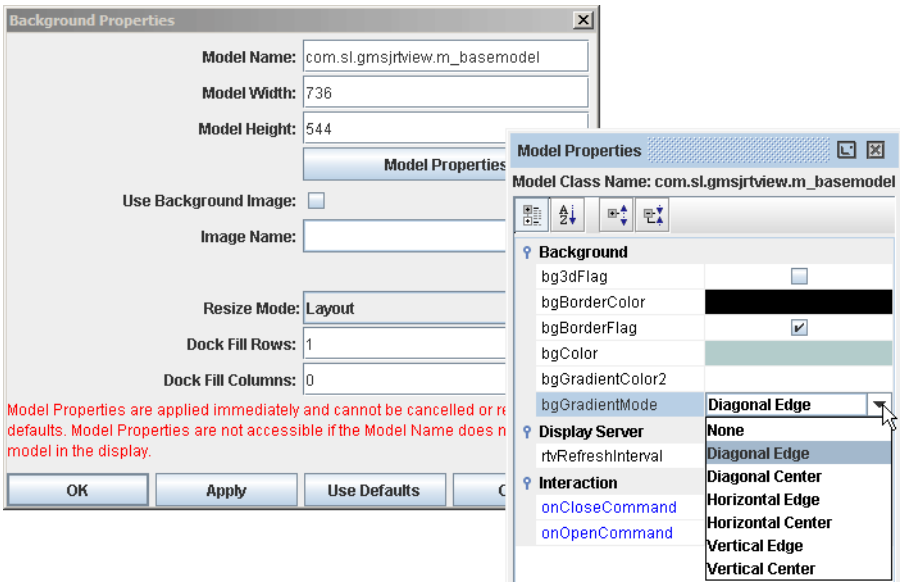
To save a display file under a different name, select **File>Save As** and enter a file name. File names may include spaces, however they are not recommended.

Printing a Display (.rtv) File

To print a display, select **File>Print** (or **Ctrl-P**)

Background Properties

Select **File>Background Properties** to modify the following settings.



Field Name	Description
Model Name	Current background model. Default model is **com.sl.gmsjrtview.m_basemodel**. **Note:** SL-GMS J/Developer and J/Net customers may import subclasses of GmsModelCode and access renamed variables via the **Model Properties** dialog.
Model Width / Model Height	Model width and height in pixels. If you set the model height to a size larger than the display area, a dialog will appear to confirm whether to add space to the top or bottom of the display area. If you decrease the size of the model height, a dialog will appear asking you whether to take space from the top or the bottom of the display area. In addition, if you decrease the model size and the objects no longer fit within the display area, you will be asked to confirm this in a separate dialog. **Note:** Model Width/Height options are disabled if you are not using the default model **com.sl.gmsjrtview.m_basemodel**.
Model Properties	Select to open the **Model Properties** dialog, which lists attributes of the background model and enables you to edit property values. Model properties are applied immediately and cannot be canceled or reverted to defaults. If the Model Name does not match the model in the display, the **Model Properties** dialog is not accessible. To access **Model Properties** directly, select **Edit>Model Properties**. See the "Object Properties" section for details on editing property values. If you are using the default model **com.sl.gmsjrtview.m_basemodel**, the Model Properties are:

Background

bg3dFlag -- Control visibility of bevel on background.

bgBorderColor -- Select the  button and choose a color from the palette to set the color of the background outline. This property is only applicable if **bgBorderFlag** is selected.

bgBorderFlag -- Control visibility of outline on background.

bgColor -- Select the  button and choose a background color from the palette.

bgGradientColor2 -- Select the  button and choose a second color for background gradient. Default is white. **Note:** The **bgColor** property sets the first color in the gradient.

bgGradientMode -- Control style of gradient fill on background. Select from the following options:

- **Diagonal Edge** -- Gradient is drawn at a 45 degree angle from the top left to the bottom right corner of the object.
- **Diagonal Center** -- Gradient is drawn at a 45 degree angle from the center to the top left and the bottom right corners of the object.
- **Horizontal Edge** -- Gradient is drawn horizontally from the top to the bottom of the object.
- **Horizontal Center** -- Gradient is drawn horizontally from the center to the top and bottom of the object.
- **Vertical Edge** -- Gradient is drawn vertically from the left to the right of the object.
- **Vertical Center** -- Gradient is drawn vertically from the center to the left and right of the object.

Display Server

rtvRefreshInterval -- Specify the interval (in seconds) to refresh a display in the client browser. Minimum interval is 5 seconds. If left blank, the interval defined in the rtvdisplay.properties file will be used. Set to 0 if you do not want displays automatically refreshed. It is also possible to enter a substitution or attach **rtvRefreshInterval** to data.

Note: The refresh rate specified in the URL or DIV tag used to open a display will override the **rtvRefreshInterval** property.

See "Display Servlet Configuration","Creating Display Servlet HTML Files", and "Creating Display Servlet JSP Files" for more information.

Interaction

onCloseCommand -- Select **Define Command** and choose **SYSTEM**, **ALERT**, **MULTIPLE**, or your data source. The specified command will be activated when this display (.rtv) file is opened/loaded.

Note: When this property is defined by a **SYSTEM** drill down command, this functionality is not supported in the Display Server.

onOpenCommand -- Select **Define Command** and choose **SYSTEM**, **ALERT**, **MULTIPLE**, or your data source. The specified command will be activated when this display (.rtv) file is closed/replaced.

Note: When this property is defined by a **SYSTEM** drill down command, this functionality is not supported in the Display Server.

resizeHeightMin / resizeWidthMin -- Specify (in pixels) the minimum height and width of a display. Default value for each property is 0, which means the display has no minimum size.

In the Display Viewer Application, a panel cannot be resized smaller than the minimum size of the display in that panel.

In the Thin Client, a panel can be resized smaller than the minimum size of the display in that panel, but (since the default **Resize Mode** is **Crop**) scrollbars will be added.

To set the minimum size for multiple displays, **resizeWidthMin** and **resizeHeightMin** could be specified in a style sheet, as follows:

```
m_basemodel {resizeWidthMin:600; resizeHeightMin:400}
```

See "Style Sheets" for more information.

Note: If **resizeWidthMin** or **resizeHeightMin** are set to values larger than the specified Model Width and Model Height, then no minimum size will be enforced on the display.

Use Background Image

Select this check box to specify a model file or a bitmap image (.gif, .jpg, or .png) as the display background.

Note: If you select to use a background image, all background model properties will be disabled.

Image Name

Use Background Image must be selected in order to place an image (.gif, .jpg, or .png) in the background of your display.

Enter the name (including relative path) of the file or select the  button to open the **Select Image** dialog containing up to three directories:

Current Directory - Contains images in the current directory and one level of subdirectories.

Custom Image Library - If you have specified a custom image library, this directory contains those images. See the *Creating a Custom Image Library* section (below) for details.

Symbol Library - Contains symbolic images (for example, symbols for various types of hardware, shapes, lights, arrows, etc.).

Navigate to the image you want to use and select it. A preview of the image appears in the pane to the right. Click **OK** or **Apply** to set the image on your object. If an image is not listed, enter the name of the file (including relative path).

Creating a Custom Image Library

The custom image library enables you to make your own images available in the **Select Image** dialog. To add your own image library, perform the following steps.

1. Place your images in a .jar file and add it to the "RTV_USERPATH" environment variable.

Note: The images must be in a directory and not at the top level of the .jar file. They can be organized into subdirectories of one top level directory.

2. In the Display Builder, select **Tools>Builder Options** and, in the **Custom Image Library Path** field, set the path to the directory containing your .jar file.

For example, suppose you have a .jar file with the following directory structure:

com/mycompany/Images

com/mycompany/Images/Blue Images

com/mycompany/Images/Red Images

com/mycompany/Images/Green Images

In the **Custom Image Library Path** field you would enter **com/mycompany/Images** to add a directory named **Images** to the tree in the Select Image dialog. The **Images** directory will have three subdirectories: **Blue Images**, **Red Images**, and **Green Images**. **Note:** Only directories containing images will be added to the Select Image dialog.

Note: You can also access your custom image library by editing any property that allows you to set an image on an object (e.g. image, barImage, and filterProperties properties).

Resize Mode

Control object layout when a window is resized in the current display (.rtv) file. It is also possible to globally specify a Window Resize Mode, select **Tools>Options>General** or set -**resizemode** on the command line.

In the Display Builder, the selected Resize Mode is only applied to drill down windows. The main window of the Display Builder is always in Crop mode.

Select from the following options:

Default - Use application level default mode. Defaults are: **Scale** for the Display Builder and Display Viewer Application and **Crop** for the Thin Client.

Crop - When the window is resized, the display stays the same size. If the window is bigger than the display, empty space will show around the display. If the window is smaller than the display, scrollbars will be added. The window is not forced to maintain its aspect ratio. This is the default for the Thin Client.

Scale - When the window is resized, the display and all of the objects in it are scaled to fit the available space. The window is forced to maintain its aspect ratio. This is the default for the Display Builder and Display Viewer Application.

Layout - When the window is resized, the display is resized to fit the available space. The objects in the display are positioned according to their **anchor** and **dock** properties. The window is not forced to maintain its aspect ratio.

Objects that are not docked or anchored will move relative to their offset from the top left corner of the display. For example, if the object is centered on the display, the object will move 50% of the resize amount. If the object is centered at 3/4 of the display, it will move 75% of the resize amount.

To prevent objects from overlapping, set **resizeWidthMin** and **resizeHeightMin**. If a panel containing a display is resized, the display will not reduce smaller than the specified minimum size.

Note: All three resize modes support zooming the display (right-click -> zoom). In both Scale and Layout modes if the window is resized while the display is zoomed, then the resize will further zoom the display.

Dock Fill Rows/ Doc Fill Columns

Objects in a display that have the **dock** property set to Fill are laid out across a grid in the available space remaining in the display after all docked objects have been positioned. By default, the grid has one row and as many columns as objects.

If **Dock Fill Rows** is set to 0 and the **Doc Fill Columns** value is specified, then the number of rows will be calculated based on the number of objects. If a value is specified for both **Dock Fill Rows** and **Doc Fill Columns**, the row value will be used and the number of columns will be calculated based on the number of objects. If both are set to 0, the default will be used.

Objects are positioned left to right, top to bottom according to the order in which objects were added to the display.

The following describes the **Background Properties** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Use Defaults	Resets all fields to default values (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied.

View Options

Several options are available to control how you view a display.

Zoom In - To initiate, click the button in the toolbar. To activate, click in the display window. To terminate, press the **<Esc>** key. **Note:** To zoom out, hold down the <Shift> key while clicking in the display window.

Zoom Out - To initiate, click the button in the toolbar. To activate, click in the display window. To terminate, press the <Esc> key.

Zoom Rect - To initiate a cross bar that allows you to select and zoom in on a specific area of the display, select **View>Zoom Rect**. To activate, click and drag across the display window.

Pan - To initiate, select **View>Pan**. To activate, click in the display window. To terminate, press the <Esc> key. **Note:** It is not possible to pan at 100% visibility.

100% - To resume a full view of the display, select **View>100%**.

Pause Display - To disable screen updates, select Tools>Pause Display. Select again to resume updates. **Note:** While the display is suspended, incoming data is still processed.

Multiple Display Panels

With Multiple Display Panels you can deploy several displays, arranged in separate panels within a single window. This option is not available in the Display Builder, but is supported in all RTView deployments. See **demos/multipanels** for examples of using Multiple Display Panels.

Configuring Multiple Display Panels

By default, the Display Viewer Application, and Display Server read the **PANELS.ini** file to populate panels with the specified displays. If the panel configuration file is not found in the current directory, the Display Viewer and Display Server applications will search under **lib** in your installation directory.

Display Viewer Application

You may specify a different panel configuration file on the command line:

command line: **-panelconfig:filename**

Note: To run the Display Viewer Application without a panel configuration file, when PANELS.ini is present in the current directory or the **lib** directory, use the above command line argument with a value for **<filename>** that does not correspond to a file in either directory, for example: **run_viewer -panelconfig:X**

Display Server

You may specify a different panel configuration in the URL:

http://host:8080/rtvdisplay/panels.jsp?file=filename.

To define initial substitution values in the Display Server, you may include a **subs** parameter in URLs that specify **panels.jsp**:

http://SomeHost/rtvdisplay/panels.jsp?subs=$table:production_table

The **panels.jsp** file must use the **rtvLayout** tag (described below). Substitutions will be applied to the initial display(s) opened in the panel layout. To apply to displays that are subsequently opened, substitutions should be mapped to global variables or you can specify the **clearsubs** option either on the command line or in the DISPLAYSERVER.ini file.

PANELS.ini

Note: PANELS.ini formats and associated tags supported in previous releases are still supported. However, the new tags listed below cannot exist in the same PANELS.ini file with any older tags.

Tag	Description
rtvLayout	The **rtvLayout** tag can appear only once in PANELS.ini and expects each child tag to specify a region (i.e. north, south, east, west, or center) in order to determine the location of each child panel. Typically, an rtvLayout will contain one of the following combinations: • A main **rtvDisplayPanel** with region="center", plus either an **rtvAccordionPanel** or **rtvTreePanel** with region="west" or "east", and possibly other secondary **rtvDisplayPanels** in other regions. • A main rtvTabbedDisplayPanel with region="center", and possibly other secondary rtvDisplayPanels in other regions. **Note:** The tags listed below can only be used within the rtvLayout tag. PANELS.ini formats and associated tags supported in previous releases are still supported. However, tags listed below cannot exist in the same PANELS.ini file with any older tags. Alternately, it is possible to use style sheets to specify colors, fonts, and other properties of the navigation panels and controls. See "Style Sheets" for more information.

Attributes	
Name	**Description**
dividers	If true, then a movable divider (via dragging) is drawn between the child panels.
title	Set the title of the main window.

Child Tags	Description
rtvDisplayPanel	Create a panel containing the specified display.

Attributes	Description

display	The name of the display (.rtv) file to load into the panel.
	You can use an application level substitution or login substitution for the display value as long as the substitution name starts with a "**$**." For example:
	`<?xml version="1.0" ?>`
	`<panels xmlns="www.sl.com" version="1.0">`
	`<rtvLayout title="Example">`
	`<rtvDisplayPanel region="north" name="title"`
	`display="$my_title"/>`
	`<rtvDisplayPanel region="center" name="main"`
	`display="$my_main"/>`
	`</rtvLayout>`
	`</panels>`

minimized	For the Thin Client. If true, specifies that a panel is initially hidden. The dividers attribute must also be true for the **rtvLayout** tag. In the following example the "north" panel is initially minimized:
	`<?xml version="1.0" ?>`
	`<panels xmlns="www.sl.com" version="1.0">`
	`<rtvLayout title="Panel Example" dividers="true">`
	`<rtvDisplayPanel region="north" minimized="true"`
	`display="title.rtv"/>`
	`<rtvDisplayPanel region="center" name="main"`
	`display="test1.rtv"/>`
	`</rtvLayout>`
	`</panels>`
	A minimized panel is made visible by clicking in the dark area in the center of its divider bar. The minimized attribute is ignored in the Viewer.

name	Corresponds to Window Name entered in the "Drill Down Properties" dialog. When using tabbed panels, if the name is not specified, a name is constructed internally using the display name and substitutions to make it easy to drill down between tabs. In this case, when you drill down from a tab using the Current Window option and the specified display with the specified substitutions is already loaded in another tab, the Display Viewer will switch to that tab. To configure Synchronize Drill-down and Navigation Controls, this name must be nav. See "Drill Down Displays" and "Synchronizing Drill-down and Navigation Controls" for more information.
nodivider	For the thin client only. If movable dividers are enabled in the layout (see the **dividers** attribute, above), then the divider for a specific panel can be hidden by setting **nodivider="true"**.
region	Position of panel in **rtvLayout**: west, east, center, north or south.
subs	Specify initial substitutions for this panel. Substitutions are optional and must use the following syntax: **$subname:subvalue $subname2:subvalue2** If a substitution value contains a single quote, it must be escaped using a / : **$filter:Plant=/'Dallas/'** If a substitution value contains a space, it must be enclosed in single quotes. Do not escape these single quotes: **$subname:subvalue $subname2:'sub value 2'** A substitution string cannot contain the following: : | . tab space , ; = < > ' " & / \ { } [ ] ( ) **Note:** Substitutions set in Application Options will apply to all displays. See "Substitutions Tab" for more information.

rtvAccordionPanel

Create a panel containing an accordion control for display navigation. The accordion control assumes there is a **rtvDisplayPanel** in the center region and sends its navigation commands to that panel. You can configure this control to show/hide displays based on role definitions. For details, see "Role Definitions".

rtvAccordionPanel contents cannot be more then two levels deep (not including the root node). If deeper nesting is required, use **rtvTreePanel** instead.

Attributes

Name	Description
navdata	Name of the navigation tree definition file. This XML file should describe content of the accordion control.
width	Width (in pixels) of the panel. Default is 125.
region	Position of panel in rtvLayout: west, east, center, north and south. Default is center.

rtvTreePanel

Create a static tree navigation panel. The tree control assumes there is a **DisplayPanel** in the center region and sends its navigation commands to that panel.

There are two methods for creating a tree-driven multi-panel application: the static tree navigation panel and the tree control. Use the static tree navigation panel method if you know the specific sources that are to populate the tree and they remain constant for the life of the application. For example, if you know all the displays that compose your application and the static representation of a tree will only be used for navigating those displays, the static tree navigation panel is suitable (and is easier to configure).

Use the tree control method if the number of nodes or leaves, labels or icons of the tree change during the lifetime of the application. Data can be provided that will change the nodes and leaves of the tree and also change the labels, and icon representations on the tree with dynamic data. For details, see "Tree Control".

You can configure this control to show/hide displays based on role definitions. For details, see "Role Definitions".

Attributes

Name	Description
navdata	Name of the "Navigation Tree Definition File". This XML file should describe content of the tree.
width	Width (in pixels) of the panel. Default is 125.
region	Position of panel in rtvLayout: west, east, center, north and south. Default is center.

rtvTabbedDisplayPanel

Create a panel with tabs for navigation. You can configure this control to show/hide displays based on role definitions. For details, see "Role Definitions".

Attributes

Name	Description
tabs	Name of the "Tab Definition File". This XML file should describe content of the tabs.

display	Name of display (**.rtv**) file to load into the panel.
subs	Specify initial "Substitutions" for this panel. Substitutions are optional and must use the following syntax: **$subname:subvalue $subname2:subvalue2** If a substitution value contains a single quote, it must be escaped using a / : **$filter:Plant=/'Dallas/'** If a substitution value contains a space, it must be enclosed in single quotes. Do not escape these single quotes: **$subname:subvalue $subname2:'sub value 2'** A substitution string cannot contain the following: : | . tab space , ; = < > ' " & / \ { } [ ] ( ) **Note:** Substitutions set in Application Options will apply to all displays. See "Substitutions Tab" for more information.
region	Position of panel in **rtvLayout**: west, east, center, north and south. Default is center.
placement	Position of tabs: top or bottom. Default is top.

Example 1

This example of PANELS.ini creates a title panel at the top, an accordion panel on the left, and a main display panel in the center, with movable dividers that can be dragged between the west and center panels:

```
<?xml version="1.0" ?>
<panels xmlns="www.sl.com" version="1.0">
<rtvLayout title="Accordion Example" dividers="true">
    <rtvDisplayPanel region="north" name="title_panel" nodivider=”true” display="title.rtv"/>
    <rtvAccordionPanel region="west" width="200" navdata="navtree.xml"/>
    <rtvDisplayPanel region="center" name="main_panel" display="chart_main.rtv"/>
</rtvLayout>
</panels>
```

Example 2

This example of PANELS.ini uses the new tags to create a tabbed display panel at the top, and a title panel at the bottom:

```
<?xml version="1.0" ?>
<panels xmlns="www.sl.com" version="1.0">
<rtvLayout title="Tab Example">
    <rtvTabbedDisplayPanel region="center" tabs="navtabs.xml" display="stock_chart"/>
    <rtvDisplayPanel region="south" name="title_panel" display="title.rtv"/>
</rtvLayout>
</panels>
```

Synchronizing Drill-down and Navigation Controls

A panel configuration file (**PANELS.ini**) can be created where one **rtvDisplayPanel** contains a navigation control (the tree control (**obj1_c1tree**) or the accordion control (**obj_c1accordion**)) that drills down in a second **rtvDisplayPanel**. The navigation control remains synchronized with the second **rtvDisplayPanel** when a user clicks on an object in the second **rtvDisplayPanel**--rather than a node in the navigation control--to drill-down to another display.

To configure this behavior for the panels configuration file:

1. The specified name for the **rtvDisplayPanel** containing the display with the tree or accordion control is **nav**.
2. The specified region for the other **rtvDisplayPanel**, which is the target of the tree/accordion control's drilldown, is center.

If the above rules are followed, for example, and the user drills down to a display named **X** in the center panel by clicking on an object in the center panel display, the node in the navigation control that corresponds to **X** (if any) is automatically selected.

To view an example of this panel configuration, see **PANELS_treecon.ini** or **PANELS_accordcon.ini**, located in the **RTV_HOME/demos/features** directory. To start the example, type the following commands in an RTView command prompt:

```
cd %RTV_HOME%\demos\features
run_viewer -panelconfig:PANELS_accordcon.ini
```

In the accordion control, click the **Navigate Displays** button. In the main display right-click the object labeled **Current Window**, then select **Drill Down** from the popup menu. The **object_variety2** display opens in the main window, which corresponds to the button labeled **Table Objects** in the accordion, and which should now be visible and highlighted.

Limitation

Note that only the display name is compared when determining which node should be selected, substitutions are not compared. If several nodes in the tree use the same display as their value, but with different substitutions (via **drilldownColumnSubs**), this feature always selects the first node in the control whose display name matches, regardless of the substitutions.

Style Sheets

Within the rtvLayout tag, style sheets can be used to specify colors, fonts and other properties of the navigation panels and controls. For detailed information about using style sheets, go to "Style Sheets".

The following style class names and associated properties are supported in a panels configuration file, such as in the following example:

```
rtnav {bgColor: 7; fgColor: 12; }
rtnav-state-default {bgColor: 256; font: 1; }
rtnav-state-selected {bgColor: #4c5d65; font: 7;}
rtnav-state-hover {bgColor: #004157; }
rtnav-accordion {buttonWidth: 200;}
```

Note: If a property is specified for more than one class name, the value set for the class name with highest precedence applied. Class names (with the exception of rtnav-accordion) are listed below in order of precedence, from lowest to highest:

Style Class Name	Description	Properties
rtnav	Properties specified for this class are applied to all the navigation panels and controls (i.e. tabs, trees, and accordion).	**bgColor / fgColor** -- Background and foreground (text) colors. Specify color index number from the RTView color palette or RGB value (in #rrggbb format where rr, gg, bb are the red, green, and blue intensities in hexadecimal values between 00 and ff). **font** -- Font style for navigation controls. Specify index number, as follows: **1** sans-serif **2** monospaced bold **3** monospaced **4** serif **5** monospaced italic **6** serif bold **7** sans-serif bold **8** sans-serif italic **9** serif bold italic **10** serif italic **11** sans-serif bold italic **12** monospaced bold italic **fontSize** -- Font size (in pixels) for navigation controls. **Note:** Unless the **font** property is specified, **fontSize** is ignored. **bgGradientFlag** -- Select gradient fill or solid fill for tabs or accordion control. Default value is **true**. Set to **false** for solid fill.
rtnav-state-default	Properties for this class are applied to all navigation controls that are in the default (i.e. unselected) state.	**bgColor / fgColor** -- Background and foreground (text) colors. Specify color index number from the RTView color palette or RGB value (in #rrggbb format where rr, gg, bb are the red, green, and blue intensities in hexadecimal values between 00 and ff). **font** -- Font style for navigation controls. Specify index number, as follows: **1** sans-serif **2** monospaced bold **3** monospaced **4** serif **5** monospaced italic **6** serif bold **7** sans-serif bold **8** sans-serif italic **9** serif bold italic **10** serif italic **11** sans-serif bold italic **12** monospaced bold italic

		fontSize -- Font size (in pixels) for navigation controls. **Note:** Unless the **font** property is specified, **fontSize** is ignored. **bgGradientFlag** -- Select gradient fill or solid fill for tabs or accordion control. Default value is **true**. Set to **false** for solid fill.
rtnav-state-hover	Properties for this class are applied to the navigation control that the mouse cursor is currently positioned over (affects tab and accordion controls only, and only in the Thin Client)	**bgColor / fgColor** -- Background and foreground (text) colors. Specify color index number from the RTView color palette or RGB value (in #rrggbb format where rr, gg, bb are the red, green, and blue intensities in hexadecimal values between 00 and ff). **font** -- Font style for navigation controls. Specify index number, as follows: **1** sans-serif **2** monospaced bold **3** monospaced **4** serif **5** monospaced italic **6** serif bold **7** sans-serif bold **8** sans-serif italic **9** serif bold italic **10** serif italic **11** sans-serif bold italic **12** monospaced bold italic **fontSize** -- Font size (in pixels) for navigation controls. **Note:** Unless the **font** property is specified, **fontSize** is ignored. **bgGradientFlag** -- Select gradient fill or solid fill for tabs or accordion control. Default value is **true**. Set to **false** for solid fill.
rtnav-state-selected	Properties for this class are applied to all navigation controls that are in the selected state (e.g. the selected tab in a tabbed panel).	**bgColor / fgColor** -- Background and foreground (text) colors. Specify color index number from the RTView color palette or RGB value (in #rrggbb format where rr, gg, bb are the red, green, and blue intensities in hexadecimal values between 00 and ff). **font** -- Font style for navigation controls. Specify index number, as follows: **1** sans-serif **2** monospaced bold **3** monospaced **4** serif **5** monospaced italic **6** serif bold **7** sans-serif bold **8** sans-serif italic **9** serif bold italic **10** serif italic **11** sans-serif bold italic **12** monospaced bold italic

		fontSize -- Font size (in pixels) for navigation controls. **Note:** Unless the **font** property is specified, **fontSize** is ignored. **bgGradientFlag** -- Select gradient fill or solid fill for tabs or accordion control. Default value is **true**. Set to **false** for solid fill.
rtnav-accordion	Properties for this class are applied to the accordion navigation controls.	**buttonWidth** -- Width (in pixels) of all buttons in the accordion. Default is 160. **spacing** -- Vertical space (in pixels) between buttons. Default is 5. **indent** -- Left indent (in pixels) of leaf buttons in the accordion. Default is 10.

Navigation Tree Definition File

The content of **rtvAccordionPanel** or **rtvTreePanel** is determined by an XML file specified as the value for the **navdata** attribute.

Note: rtvAccordionPanel only supports two levels of nodes, not counting the root node. Therefore, in the following example, the node labeled Displays would not be visible.

The navdata file must start with the following:

```
<?xml version="1.0" ?>
```

The navdata file must end with the following:

```
</navtree>
```

For example:

```
<?xml version="1.0" ?>
 <navtree version="1.0">
 <node label="Displays">
     <node label="Charts" display="chart_overview.rtv">
         <node label="Chart One" display="chart1.rtv" subs="$sub1:123"/>
         <node label="Chart Two" display="chart2.rtv"/>
     </node>
     <node label="Reports" display="report_overview.rtv">
         <node label="Report 1" display="rep1.rtv"/>
         <node label="Report 2" display="rep2.rtv"/>
     </node>
 </node>
 </navtree>
```

The following tags are supported:

Tag	**Description**	**Attrributes**	
		Name	**Description**
node	Add a node to the navigation tree.	display	Name of the display (.rtv) file.

label	Label for this node in the navigation tree.
subs	Substitutions to apply to the display. Substitutions are optional and must use the following syntax: **$subname:subvalue $subname2:subvalue2** If a substitution value contains a single quote, it must be escaped using a / : **$filter:Plant=/'Dallas/'** If a substitution value contains a space, it must be enclosed in single quotes. Do not escape these single quotes: **$subname:subvalue $subname2:'sub value 2'** A substitution string cannot contain the following: : | . tab space , ; = < > ' " & / \ { } [ ] ( )

Tab Definition File

The content of tabs in **rtvTabbedDisplayPanel** are determined by an XML file specified as the value for the tabs attribute.

The tabs file must start with the following:

```
<?xml version="1.0" ?>
```

The tabs file must end with the following:

```
</navtree>
```

For example:

```
<?xml version="1.0" ?>
 <navtree>
     <node label="Bar Chart" display="disp1.rtv"/>
     <node label="History Graph" display="disp2.rtv" subs="$v1:xyz"/>
 </navtree>
```

The following tags are supported:

Tag	Description	Attributes	
		Name	**Description**
node	Add a node to the navigation tree.	display	Name of the display (.rtv) file.
		label	Label for this node in the navigation tree.
		subs	Substitutions to apply to the display. Substitutions are optional and must use the following syntax: **$subname:subvalue $subname2:subvalue2** If a substitution value contains a single quote, it must be escaped using a / : **$filter:Plant=/'Dallas/'** If a substitution value contains a space, it must be enclosed in single quotes. Do not escape these single quotes: **$subname:subvalue $subname2:'sub value 2'** A substitution string cannot contain the following: : | . tab space , ; = < > ' " & / \ { } [ ] ( )

Viewing Multiple Display Panels

Display Viewer Application

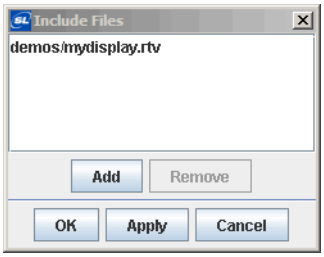
At startup, the Display Viewer Application reads the panel configuration from the directory where it is started to populate panels with the specified displays. If the panel configuration file is not found in the current directory, the Display Viewer will search under **lib** in your installation directory. When viewing multiple display panels in the Display Viewer, you cannot open displays using the command line or the **File>Open** dialog.

Display Server

At startup, the Display Server reads the panel configuration from the directory where it is started to populate panels with the specified displays. If the panel configuration file is not found in the current directory, the Display Server will search under **lib** in your installation directory. See the "Display Server" section for more details on viewing multiple display panels.

Include Display Files

If several displays use a common set of functions, variables or objects, it is possible to include the contents of another display (**.rtv**) file within those displays. Common elements, such as a navigation bar, can be defined and maintained in a separate display (**.rtv**) file and then included in each display.



To include a display file, select **Tools>Include Files** in the Display Builder. It is possible to include multiple display files within a single display. Displays can be nested, that is an included display file can include other files, however a display cannot include the same display (.rtv) file more than once.

Functions, variables and objects from an included display file cannot be modified in the current display, with the exception of the initial value of a variable. It is possible to copy objects from an included display file and paste them into your current display. Functions and variables associated with objects in an included display file can be used as data attachments by objects in the current display. However when a display file is removed from the Include Files list, all functions and variables associated with that file will be removed from the current display. Background properties (image, model properties, etc.) defined within an included display file are ignored and therefore not active in the current display.

If an included display file is in the same directory or subdirectory as the current display, a relative path to the included display file is saved in the current display (**.rtv**) file. If not, an absolute path to the included display file is saved.

Functions

Functions from an included display file are listed in the **Functions** dialog of the current display, but cannot be removed or modified. To edit an included function, you must directly edit the included display file. In the **Edit Function** dialog, the **Include File** field shows the name of the included function's display file.

An included function's result can be used in data attachments in the current display, however if the included display file is removed all associated functions will be removed from the current display.

Variables

Variables from an included display file are listed in the Variables dialog of the current display, but cannot be removed. However, the initial value of a variable from an included display file can be changed in the Variables dialog. For example, several displays may include a display file named **Header.rtv** in which a label object is attached to a variable named **headerTitle**. In the Variables dialog of the current display, the initial value of **headerTitle** (e.g.: Your Title Goes Here) can be changed (e.g.: My Display Title) and the new value will be saved in the current display (.rtv) file.

Objects

Properties of objects from an included display file are listed in the **Object Properties** window. The name of the included object's display file is noted in the **objName** property field. Included objects cannot be moved, deleted or modified in any way. To edit an included object, you must edit the included display file. It is also possible to copy objects from included displays and paste them into your current display.

If objects from an included display file have the same value for the **objName** property as objects in the current display, or other included displays, this may cause links to attach to the wrong object. To avoid this overlap, assign a unique value to the **objName** property of objects in files you intend to include in other displays.

In the Display Viewer, objects that are loaded from an included display file are treated exactly as objects that are loaded from the current display.

CHAPTER 8 Functions

This section contains the following:

Add/Edit Functions

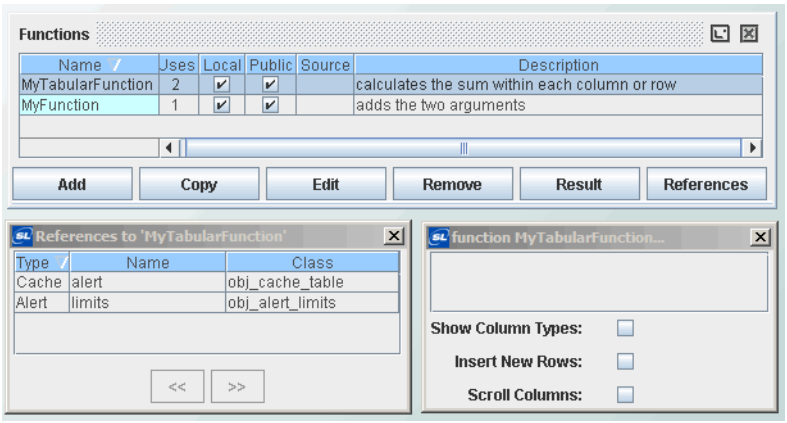
It is possible to perform calculations on your data before it displayed in RTView. For example, you can create a function that will take the average value of a table column or add the values of multiple data attachments. To display the results, attach your functions to objects using the "Attach to Function Data" dialog.

Note: Functions are only available for use within the display where they are created. The file functions.rtv (located in the tutorials directory) is provided as an example of the different ways that functions can be utilized in a display.

This section contains:

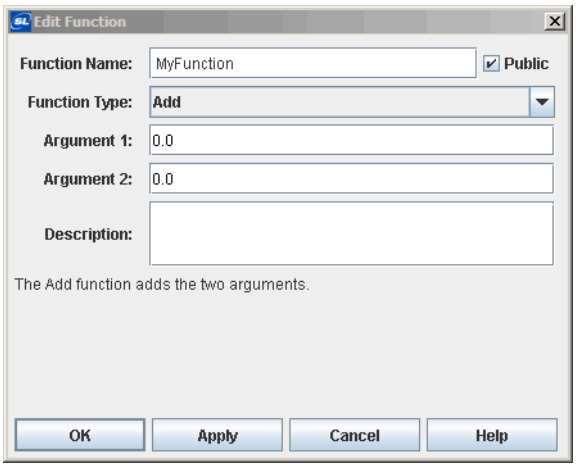
Adding Functions

In the Display Builder, select **Tools>Functions** to open the **Functions** window.



Field Name	Description
Functions	the functions listed can be sorted by clicking on any column name. **Name** -- Function Name as entered in the Edit Function dialog. **Uses** -- Number of times function is used in current display. **Local** -- Selected if function is defined in current display. **Public** -- Selected if function is available in display (.rtv) files, other than display where it was defined. **Source** -- Name of file in which global function or included display file is defined. **Description** -- Description as entered in the **Edit Function** dialog.
Add	To add a new function, click **Add** to open the **Edit Function** dialog. See "Editing Functions" for more information.
Copy	Select a function from the list and click **Copy** to create a duplicate of that function. To copy a function from your current display to another display (**.rtv**) file, select a function from the list and click the **Copy** button  in the toolbar (or Ctrl-C). Then open the other display (**.rtv**) file and click the Paste  button (or Ctrl-V). You must enter a unique name for each copy you make. If a function by that name already exists in that display (**.rtv**) file, you will need to rename the function. **Note:** A function pasted into another display (**.rtv**) file will have the same data attachments as the original function.
Edit	Select a function from the list and click **Edit** to open the **Edit Function** dialog. See "Editing Functions" for more information. **Note:** Functions associated with global functions or included display files can only be modified from the display in which they were originally defined.
Remove	Select a function from the list and click **Remove** to delete.
Result	Select a function from the list and click **Result** to view the result of executing the selected function.
References	Select a function from the list and click **References** to view a list of items that reference the selected function. Make a selection from the References list to bring up dialogs and/or object properties associated with that reference item. If the selected reference item is an object, then that object will be highlighted in the Working Area. If the selected item is another function and that function has references, then the References dialog will add an additional page of items referenced by that function. Use the **<< >>** buttons to navigate between the references of interrelated functions. If the **References** button is disabled, then the selected function is not referenced by any other items.

Editing Functions



Edit Functions Window

Field Name	Description
Function Name	Enter a unique name for your function. Each function must have a unique name (within that display). Function Names cannot contain spaces and the name function is not allowed. **Note:** If a global function shares the same name as a local function, only the local function will be displayed and the global function will be ignored.
Public	If the Function Name specified will be defined as a global function or in an included display file, then selecting the **Public** check box will make this function available in all displays. When the Public check box is deselected, this function will only be available for data attachments in the display (.rtv) file where it was defined. It is recommended that you deselect the **Public** check box if the function you defined may generate intermediary results that are only useful in the context of the current display and would be confusing or unusable during the creation of other displays. See "Global Functions and Variables" and "Include Display Files" for more information.
Function Type	The Function Type drop down menu lists all available calculations. Function types are divided into two groups: Scalar and Tabular. RTView comes with an array of built-in, pre-defined functions or you can create your own Custom Functions. See "Function Types" and "Custom Functions" for more information.

Arguments	Argument fields populate based on the selected Function Type. You may either enter a value into the Argument field or right-click and select Attach to Data to choose a data source. An argument that has been attached to data is displayed in green, this indicates that editing is no longer possible in the Argument field. Double-click in the Argument field to edit the data attachment. Once an argument is attached to data, right-click to choose from the following options: **Attach to Data** -- Change the data source. **Detach from Data** -- Remove the data attachment. **Display Data** -- Open a window that contains a table showing data available from that attachment. **Note:** Only one Display Data window can be opened for each data attachment. **Show Column Types** -- Display the data type of each column. **Insert New Rows** -- Insert new rows to display incoming data, By default, old data is replaced by new data on each update. **Edit Function** -- If the argument references another function (i.e. is attached to Function data), you can edit the properties of the attached function. The current Edit Function dialog will be replaced with that of the attached function. Click the Back button to return to the function you were previously editing. **Note:** Save changes in your current Edit Function dialog before attempting to open another.
Description	Enter a description of any length, which will be visible in the Functions dialog as well as the "Attach to Function Data" dialog. This field is optional.

Function Types

Functions are separated into two groups according to whether they act on Scalar or Tabular data. This section contains the following:

- "Scalar Functions" on page 406
- "Tabular Functions" on page 421

Scalar Functions

This section contains the following:

- "Available Scalar Functions" on page 406
- "Expression Syntax Definition" on page 413
- "Execute Java Method Function Type Requirements and Examples" on page 417

Available Scalar Functions

Add	Adds the two arguments. **Return Type** Numeric **Arguments** Argument1 (type = Numeric) Argument2 (type = Numeric)

Average	Calculates average of the two arguments. **Return Type** Numeric **Arguments** Argument1 (type = Numeric) Argument2 (type = Numeric)
Concatenate	Combines the two Values into a single text string. **Return Type** Text **Arguments** Value1 (type = Numeric or Text) Value2 (type = Numeric or Text)
Date Add	Adds the specified Number of Date Part intervals, which may be negative, to the specified Date and returns a string representing the resulting date/time. The Date must be either a formatted date/time string or a Java standard date/time argument in milliseconds from Jan 1, 1970. For Date Part, specify **s**, **m**, **h**, **d**, **w**, **M**, **q**, or **y** for seconds, minutes, hours, days, weeks, months, quarters or years. The Date Format specified is the same as the Java SimpleDateFormat class. For example, the format **MMMM dd, yyyy hh:mm:ss** a would result in the form A**ugust 30, 2003 05:32:12 PM**. If no Date Format is given, the string is returned in the form **08/30/03 05:32 PM**. Use **q**, **qqq**, or **qqqq** for short, medium or long versions of quarter notation. For example, **qqq-yyyy** will result in a string of the form **Qtr 1-2005**. This function returns a text string. **Return Type** Text **Arguments** Date (type = Text (Java standard Date/Time argument in milliseconds)) Number (type = Text (e.g., MMMM dd, yyyy hh:mm:ss a)) Date Part (type = Text (e.g., s, m, h, d, w, M, q, y)) Date Format (type = Text (e.g., MMMM dd, yyyy hh:mm:ss a))
Date Ceiling	Determines which Date Part interval contains the Date, and returns a string representing the start value of the next Date Part interval. The Date must be either a formatted date/time string or a Java standard date/time argument in milliseconds from Jan 1, 1970. For Date Part, specify **s**, **m**, **h**, **d**, **w**, **M**, **q**, or **y** for seconds, minutes, hours, days, weeks, months, quarters, or years. The Date Format specified is the same as the Java SimpleDateFormat class. For example, the format **MMMM dd, yyyy hh:mm:ss** a would result in the form **August 30, 2003 05:00:00 PM**. If no Date Format is given, the string is returned in the form **08/30/03 05:00 PM**. Use **q**, **qqq** or **qqqq** for short, medium or long versions of quarter notation. For example, qqq-yyyy will result in a string of the form **Qtr 1-2005**. This function returns a text string. **Return Type** Text **Arguments** Date (type = Text (Java standard Date/Time argument in milliseconds)) Date Part (type = Text (e.g., s, m, h, d, w, M, q, y)) Date Format (type = Text (e.g., MMMM dd, yyyy hh:mm:ss a))

Date Compare	Compares Date 1 and Date 2 and returns -1 if Date 1 is less than Date 2, 0 if the values are equal and 1 if Date 1 is greater than Date 2. Date 1 and Date 2 must be either formatted date/time strings or Java standard date/time arguments in milliseconds from Jan 1, 1970. For Date Part, specify **s**, **m**, **h**, **d**, **w**, **M**, **q**, or **y** for seconds, minutes, hours, days, weeks, months, quarters or years. Date Part controls the resolution of the comparison. For example, comparing **08/30/03 05:32 PM** to **08/30/03 04:47 PM** with Date Part set to m will return 1, while setting Date Part to **d** will cause this function to return **0**. This function returns a number. **Return Type** Numeric **Arguments** Date 1 (type = Text (Java standard Date/Time argument in milliseconds)) Date 2 (type = Text (Java standard Date/Time argument in milliseconds)) Date Part (type = Text (e.g., s, m, h, d, w, M, q, y))
Date Difference	Calculates the integer number of Date Part intervals by which Date 1 is less than Date 2. Date 1 and Date 2 must be either formatted date/time strings or Java standard date/time arguments in milliseconds from Jan 1, 1970. For Date Part, specify **s**, **m**, **h**, **d**, **w**, **M**, **q**, or **y** for seconds, minutes, hours, days, weeks, months, quarters or years. For example, comparing **05/12/05 05:32 PM** to **05/15/05 04:47 PM** with Date Part set to **d** will return **3**, This function returns a number. **Return Type** Numeric **Arguments** Date 1 (type = Text (Java standard Date/Time argument in milliseconds)) Date 2 (type = Text (Java standard Date/Time argument in milliseconds)) Date Part (type = Text (e.g., s, m, h, d, w, M, q, y))
Date Floor	Determines which Date Part interval contains the Date, and returns a string representing the starting date/time value of that interval. The Date must either be a formatted date/time string or a Java standard date/time argument in milliseconds from Jan 1, 1970, For Date Part, specify **s**, **m**, **h**, **d**, **w**, **M**, **q**, or **y** for seconds, minutes, hours, days, weeks, months, quarters or years. The Date Format specified is the same as the Java SimpleDateFormat class. For example, the format **MMMM dd, yyyy hh:mm:ss** a would result in the form A**ugust 30, 2003 05:00:00 PM**. If no Date Format is given, the string is returned in the form **08/30/03 05:00 PM**. Use **q**, **qqq**, or **qqqq** for short, medium or long versions of quarter notation. For example, **qqq-yyyy** will result in a string of the form **Qtr 1-2005**. This function returns a text string. **Return Type** Text **Arguments** Date (type = Text (Java standard Date/Time argument in milliseconds)) Date Part (type = Text (e.g., s, m, h, d, w, M, q, y)) Date Format (type = Text (e.g., MMMM dd, yyyy hh:mm:ss a))

Date Format	Returns a formatted string representing the specified Date. The Date must be a Java standard date/time argument in milliseconds from Jan 1, 1970. The Date Format specified is the same as the Java SimpleDateFormat class. For example, the format **MMMM dd, yyyy hh:mm:ss** a would result in the form **August 30, 2003 05:32:12 PM**. If no Date Format is given, the string is returned in the form **08/30/03 05:32 PM**. This function returns a text string. **Return Type** Text **Arguments** Date (type = Text (Java standard Date/Time argument in milliseconds)) Date Format (type = Text (e.g., MMMM dd, yyyy hh:mm:ss a))
Date Now	Returns a string representing the current date and time. The Date Format specified is the same as the Java SimpleDateFormat class. For example, the format **MMMM dd, yyyy hh:mm:ss** a would result in the form **August 30, 2003 05:32:12 PM**. If no Date Format is given, the string is returned in the form **08/30/03 05:32 PM**. This function returns a text string. **Return Type** Text **Arguments** Date Format (type = Text (e.g., MMMM dd, yyyy hh:mm:ss a))
Delta	Calculates the rate of change of the Value over the Time Interval specified in seconds. If no Time Interval is given, the absolute delta is returned. **Return Type** Numeric **Arguments** Value (type = Numeric) Interval (type = Numeric)
Divide	Divides the first argument by the second. **Return Type** Numeric **Arguments** Argument1 (type = Numeric) Argument2 (type = Numeric)
Duration	Takes a Duration argument in milliseconds and returns a text string representing that Duration. If no Duration Format is specified, the string is returned in the form "15:32" (hours:minutes). The Duration Format may contain any of the characters **d**, **s**, or **.** indicating that days, seconds, or milliseconds are to be included in addition to hours and minutes in the returned string. **Return Type** Text **Arguments** Duration (type = Text) Duration Format (type = Text (e.g., ds))

Evaluate Expression As Double	Evaluates the given expression and returns the result as a double. NOTE: Boolean true or false values will be returned as 1 and 0 respectively. **Expression** - Write an expression using variables including standard arithmetic and logical operators as well as a variety of mathematical and string functions. Variables specified in the expression must include a % prefix and cannot begin with a number (e.g. %var1 + %var2). String constants must be enclosed in double quotations (e.g. "%var1 + %var2"). When an Expression is specified and the field activated (by pressing Enter or navigating to another field) the dialog will update with a text field for each variable that requires a value. Values whose form is numeric are substituted into the expression as numbers, otherwise they are substituted into the expression as strings. If a value whose column is numeric needs to be treated as a string, perhaps to serve as an argument to a string function, enclose the variable in the expression in double quotations (e.g. "length("%var1") + %var2".) **Return Type** Numeric **Arguments** Expression (type = Expression) See "Expression Syntax Definition" for more information.
Evaluate Expression As String	Evaluates the given expression and returns the result as a text string. **Expression** - Write an expression using variables including standard arithmetic and logical operators as well as a variety of mathematical and string functions. Variables specified in the expression must include a % prefix and cannot begin with a number (e.g. %var1 + %var2). String constants must be enclosed in double quotations (e.g. "%var1 + %var2"). When an Expression is specified and the field activated (by pressing Enter or navigating to another field) the dialog will update with a text field for each variable that requires a value. Values whose form is numeric are substituted into the expression as numbers, otherwise they are substituted into the expression as strings. If a value whose column is numeric needs to be treated as a string, perhaps to serve as an argument to a string function, enclose the variable in the expression in double quotations (e.g. "length("%var1") + %var2".) **Return Type** Text **Arguments** Expression (type = Expression) See "Expression Syntax Definition" for more information.

Execute Java Method	Allows you to write code for a Java method that will be executed when the function is updated. When you select this option, the following string displays containing the **eval** method in the **Code** field:
	`public Object eval() {`
	`return "";`
	`}`
	This method can take zero or more arguments of type String, Integer, Double, or GmsTabularData. The method's return type must be declared as an Object, but can return a String, Integer, Double, or GmsTabularData object.
	The Java code you define is saved with the display (.rtv) file that contains the function and, on the first update, the function compiles the given Java code. See "Execute Java Method Function Type Requirements and Examples" for more information.

Format Number	Applies the specified Format to the Number To Format and returns a formatted string. The Format can be specified as a Java format specification, or with the following shorthand: $ for US dollar money values, $$ for US dollar money values with additional formatting, or () for non-money values, formatted similar to money. For example, if Number To Format is 50, and Format is $ this function would return $50.00. Both positive and negative formats can be supplied, for example: #,###;(#,###). **Return Type** Numeric **Arguments** Number to Format (type = Numeric) Format (type = Text (e.g., #,###;(#,###)))
Get Substitution	Returns the current value of the given Substitution String. **Return Type** Text **Arguments** Substitution String (type = Text)
isWindowsOS	Returns a value of 1 if RTView is running on Windows, otherwise it returns 0. **Return Type** Numeric **Arguments** Not applicable
Max	Returns larger of the two arguments. **Return Type** Numeric **Arguments** Argument1 (type = Numeric) Argument2 (type = Numeric)
Min	Returns smaller of the two arguments. **Return Type** Numeric **Arguments** Argument1 (type = Numeric) Argument2 (type = Numeric)
Modulo	Divides the Value by the Divisor and returns the remainder. **Return Type** Numeric **Arguments** Value (type = Numeric) Divisor (type = Numeric)
Multiply	Multiplies the two arguments. **Return Type** Numeric **Arguments** Argument1 (type = Numeric) Argument2 (type = Numeric)

Percent	Computes the percentage of the Value, given the range defined by Min Value and Max Value. This function returns a number between 0 and 100. **Return Type** Numeric **Arguments** Value (type = Numeric) Min Value (type = Numeric) Max Value (type = Numeric)
Replace Value	Replaces the Value with associated text from Replacement Values. Replacement Values should contain pairs of values and replacement values separated by a :. If the names of values or replacement values listed in Replacement Values contain a space or a colon, they must be enclosed in single quotes. For example, if the Value is **Windows NT** and pairs listed in Replacement Values are **'Windows NT':winnt Windows2000:win2k**, the text string returned will be winnt. If Return Value if No Match is set to **1**, then the returned text will be the Value if no match. If Return Value if No Match is set to **0**, then an empty text string will be returned if no match. If there is no match found in the Replacement Values, the Default Value will be returned. Unless Return Value If No Match is set to **1**, then the returned text will be the Value. **Return Type** Text **Arguments** Value (type = Text) Replacement Values (type = Text (e.g., WindowsNT:winnt)) Return Value if No Match (type = Numeric) Default Value (type = Text)
Set Substitution	Sets the Substitution String to the given Value. This function returns the value that has been set. **Return Type** Text **Arguments** Substitution String (type = Text) Value (type = Numeric or Text)

Set Substitutions By Lookup	Sets multiple substitutions based on the values in the Lookup Table. The Key is compared against the values in the first column of the Lookup Table to determine which row to use to set substitution values. For each additional column in the Lookup Table where the column name starts with $, a substitution will be set on the display, where the substitution name is the name of the column and the substitution value is the value from that column in the row where the first column matched the Key. Add Dollar To Column Names - If Add Dollar To Column Names is set to **1**, each column after the Key column is used to set a substitution regardless of whether or not it starts with **$**. For columns that do not start with **$**, a **$** is added to the beginning of the column name when it is used as a substitution name. If Add Dollar To Column Names is set to **0**, only columns starting with **$** are used as described above. **Return Type** Text **Arguments** Key (type = Numeric or Text) Lookup Table (type = Table) Add Dollar To Column Names (type = Numeric)
Subtract	Subtracts the second argument from the first. **Return Type** Numeric **Arguments** Argument1 (type = Numeric) Argument2 (type = Numeric)
Validate Substitution	Validates a substitution against a table of valid values. The Substitution String is compared to values found in the first column of the Valid Value Table. If the substitution value is not found, the Substitution String will be reset to the first value found in the table. If Clear If Invalid is set to 1, the function will return an empty string if the substitution value is not found in the table. To enter the Substitution String as a semicolon delimited list of values, set Allow Multiple Values to 1. Each value will be tested for validity and the final result will be assembled from all (if any) valid values found in the table. **Return Type** String **Arguments** Substitution String (type = String) Valid Value Table (type = Table (Only the first column is used.)) Numeric (type = Clear If Invalid) Numeric (type = Allow Multiple Values)

Expression Syntax Definition

The expression syntax for the functions Evaluate Expression As Double, Evaluate Expression As String, and Evaluate Expression By Row follows standard Java syntax and includes these operators and functions:

Operators	Precedence
unary	+ - !
multiplicative	* / %

Operators	Precedence
additive	+ -
relational	< > <= >=
equality	== !=
logical	&& ||

The following operators are not supported:

- bitwise NOT, AND, OR, XOR
- arithmetic shift

All operators may be applied to double variables. In addition the relational and equality operators may be applied to string variables, and the addition operator may be used to concatenate strings.

The following arithmetic functions are supported:

- abs
- acos
- asin
- atan
- atan2
- ceil
- cos
- exp
- floor
- IEEEremainder
- log
- max
- min
- pow
- random
- rint
- round
- sin
- sqrt
- tan
- toDegrees
- toRadians

The following *string functions* are supported:

- charAt
- compareTo
- compareToIgnoreCase
- concat
- condExpr
- endsWith
- equals
- equalsIgnoreCase
- indexOf
- lastIndexOf
- length
- replace
- startsWith
- substring
- toLowerCase
- toUpperCase
- trim

Definitions of Math Functions (per Java.Math)

Math Function	Definition
double abs(double a)	Returns the absolute value of a double value.
double acos(double a)	Returns the arc cosine of an angle, in the range of 0.0 through pi.
double asin(double a)	Returns the arc sine of an angle, in the range of -pi/2 through pi/2.
double atan(double a)	Returns the arc tangent of an angle, in the range of -pi/2 through pi/2.
double atan2(double y, double x)	Converts rectangular coordinates (x, y) to polar (r, theta).
double ceil(double a)	Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer.
double cos(double a)	Returns the trigonometric cosine of an angle.
double exp(double a)	Returns Euler's number e raised to the power of a double value.
double floor(double a)	Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.
double IEEEremainder(double f1, double f2)	Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.

Math Function	Definition
double log(double a)	Returns the natural logarithm (base e) of a double value.
double max(double a, double b)	Returns the greater of two double values.
double min(double a, double b)	Returns the smaller of two double values.
double pow(double a, double b)	Returns the value of the first argument raised to the power of the second argument.
double random()	Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.
double rint(double a)	Returns the double value that is closest in value to the argument and is equal to a mathematical integer.
long round(double a)	Returns the closest long to the argument.
double sin(double a)	Returns the trigonometric sine of an angle.
double sqrt(double a)	Returns the correctly rounded positive square root of a double value.
double tan(double a)	Returns the trigonometric tangent of an angle.
double toDegrees(double angrad)	Converts an angle measured in radians to an approximately equivalent angle measured in degrees.
double toRadians(double angdeg)	Converts an angle measured in degrees to an approximately equivalent angle measured in radians.

Definitions of String Functions (per Java.String)

String Function	Definition
char charAt(int index)	Returns the char value at the specified index.
int compareTo(String anotherString)	Compares two strings lexicographically.
int compareToIgnoreCase(String str)	Compares two strings lexicographically, ignoring case differences.
String condExpr(String expression, String value1, String value2)	Evaluated as true or false, returns value1 if true or value2 if false.
boolean endsWith(String suffix)	Tests if this string ends with the specified suffix.
boolean equals(Object anObject)	Compares this string to the specified object.
boolean equalsIgnoreCase(String anotherString)	Compares this String to another String, ignoring case considerations.
int indexOf(String str, int fromIndex)	Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.
int lastIndexOf(int ch, int fromIndex)	Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.
int length()	Returns the length of this string.
String replace(char oldChar, char newChar)	Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.
boolean startsWith(String prefix, int toffset)	Tests if this string starts with the specified prefix beginning a specified index.

String Function	Definition
String substring(int beginIndex, int endIndex)	Returns a new string that is a substring of this string.
String toLowerCase()	Converts all of the characters in this String to lower case using the rules of the default locale.
String toUpperCase()	Converts all of the characters in this String to upper case using the rules of the default locale.
String trim()	Returns a copy of the string, with leading and trailing white.

Execute Java Method Function Type Requirements and Examples

The Execute Java Method function allows you to write code for a Java method that will be executed when the function is updated and is intended for implementing a 'one-of-a-kind' java method that is useful on a single RTView display. This section describes the basic requirements and provides some examples.

Requirements

- The Java Development Kit (JDK) must be installed on systems that will be used to configure or run the **Execute Java Method** function. The Java Runtime Kit (JRE) is not sufficient since it does not include the Java compiler tools.
- The person configuring the **Execute Java Method** function must be familiar with Java as well as with the **com.sl.gmsjrt.GmsTabularData** class if the function will deal with tabular data. See "Custom Functions" for more information.
- The heap usage of the RTView application increases by about 6 to 10 MB to support the compiler.
- The **Execute Java Method** function should only be used when appropriate to avoid maintenance and performance problems.
- Since the **Execute Java Method** function is intended for a single RTView display then, if a method is needed on multiple RTView displays, it should be implemented in an RTView custom function handler class instead.

Code Argument

The function takes one predefined string argument named Code, plus any number of user-defined arguments as described later. The Code string must be a Java method named **eval** with this signature: *public Object eval(args)*. For each new instance of the function, the **Edit Function** dialog will set the Code string to a "skeleton" implementation of the **eval** method as shown below:

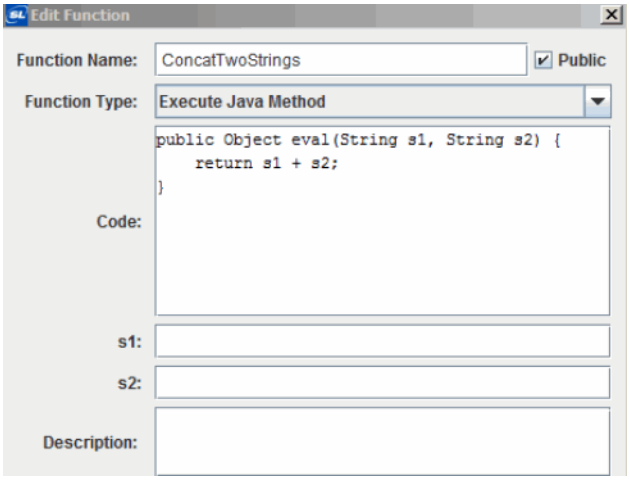
```
public Object eval() {
 return "";
}
```

The **eval** method can take zero or more arguments of type String, Integer, Double, or GmsTabularData. The method's return type must be declared as **Object**, but it can return a String, Integer, Double, or GmsTabularData object.

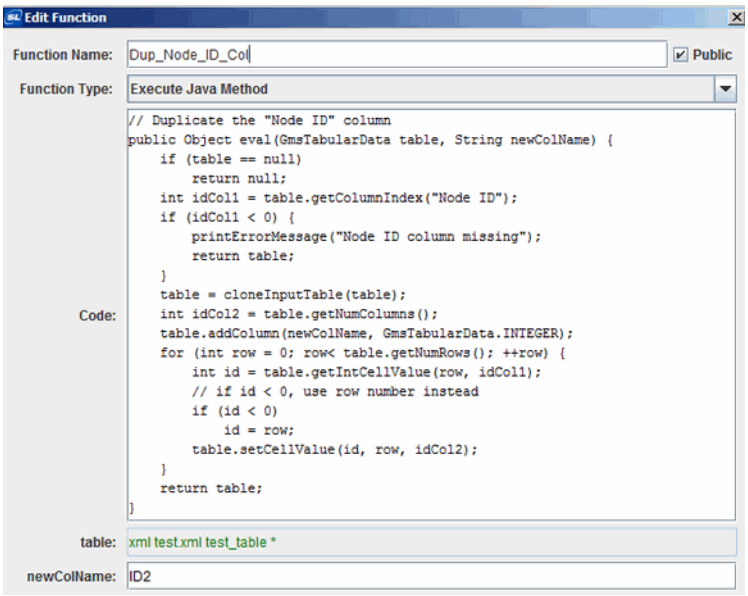
The arguments declared for the **eval()** method become arguments to the rtview function instance, just like the %x, %y tokens in an **Evaluate Expression** function. For example, here is a simple **eval** method that concatenates two String args named **s1** and **s2**:

```
public Object eval(String s1, String s2) {
       return s1 + ":" + s2;
}
```

In the **Edit Function** dialog, **s1** and **s2** will appear as argument fields so their values can be assigned to constants or attached to data:



Here is a more complex example that takes a **GmsTabularData** argument. The input table is assumed to have an integer column named **Value**, and the function returns a copy of the input table with all rows removed where the **Value** column item is < 25 or > 75. The method makes use of utility methods named **cloneInputTable** and **printErrorMessage**, which are defined in the base class named **com.sl.gmsjrtview.GmsJavaCodeExecutor**:



For documentation on the **GmsTabularData** and **GmsJavaCodeExecutor** classes, see "Customization" for more information.

Compilation

The Java code entered as the value of the Code argument for a function named **F101** is used to generate the source code for a subclass of **com.sl.gmsjrtview.GmsJavaCodeExecutor**, as follows:

```
package rtvfuncs;
```

```
import com.sl.gmsjrt.GmsTabularData;
import com.sl.gmsjrtview.*;
import java.util.*;
public class F101_Cnn extends com.sl.gmsjrtview.GmsJavaCodeExecutor {
// begin user-defined code
public Object eval() {
 return "";
}
// end user-defined code
}
```

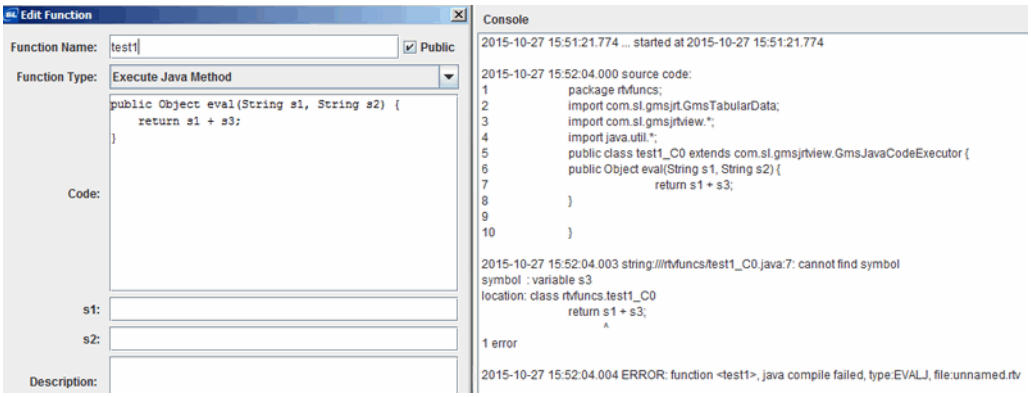
The suffix **_Cnn** on the generated class name is not significant and will vary. The source code for the class will be generated and compiled the first time the function is updated after the .rtv file is loaded. In the Builder, source code for the class will also be generated and compiled each time the user changes the code for the **eval** method and clicks **OK** or **Apply**. The compiler will use the same classpath as the RTView application.

No external .java or .class files are produced for an **Execute Java Method** function. All source generation, compilation, and class loader operations are performed using memory buffers. The user-defined Java code for the **eval** method is saved with the function in the .rtv file.

As shown above, the generated source code imports **com.sl.gmsjrtview.***, **com.sl.gmsjrt.GmsTabularData**, and **java.util.***. If additional imports are needed they can be defined on lines above the **eval()** method in the Code argument, or globally with a property named **sl.rtview.function.compiler_import**. For example, these lines could be added to an rtview **.properties** file to import **java.sql.*** and **java.math.*** for all function instances:

```
sl.rtview.function.compiler_import=java.sql.*;
sl.rtview.function.compiler_import=java.math.*;
```

Compiler errors are logged to the console. The complete generated source code, as described above, is shown before the compiler error so that the user can make sense of the line numbers in the error message. For example, the **eval()** method shown below uses an undefined variable "s3", so the console shows the corresponding source code and the Java compiler error:



Examples

The following **eval()** implementation returns a table with 2 columns, "Name" and "Value", where the Name column contains "this is row N" and the Value column contains a random integer between 0 and 100. The number of rows in the table is determined by the **numRows** argument. The "trigger" argument could be attached to the result of a "Date Now" function to trigger the function periodically; otherwise, it will only update once.

```
public Object eval(int numRows, String trigger) {
```

```
 GmsTabularData t = new GmsTabularData();
 t.addColumn("Name", GmsTabularData.STRING);
 t.addColumn("Value", GmsTabularData.INTEGER);
 for (int row = 0; row < numRows; ++ row) {
 String name = "this is row " + row;
 int val = (int) (Math.random() * 100);
 t.addRow("");
 t.setCellValue(name, row, 0);
 t.setCellValue(val, row, 1);
 }
 return t;
}
```

This **eval** method takes a table as an argument. The input table is assumed to have an integer column named **Value** (for example, the table returned from the previous example), and the function returns a copy of the input table with all rows removed where the **Value** column item is < 25 or > 75:

```
public Object eval(GmsTabularData t) {
 if (t == null)
 return t;
 int valCol = t.getColumnIndex("Value");
 if (valCol < 0)
 return t;
 ArrayList<Integer> rowsToRemove = new ArrayList<Integer>();
 for (int row = 0; row < t.getNumRows(); ++row) {
 int val = t.getIntCellValue(row, valCol);
 if (val < 25 || val > 75) {
 rowsToRemove.add(row);
 }
 }
 if (rowsToRemove.size() > 0) {
 // don't modify input table, clone it
 t = cloneInputTable(t);
 t.removeRows(rowsToRemove);
 }
 return t;
}
```

This **eval** method implements a simple counter function. It declares an integer member variable named "count" and increments it on each update and returns the new count. The "trigger" argument could be attached to the result of a "Date Now" function, to trigger the function periodically. Otherwise, it will only update once.

```
private int count = 0;
public Object eval(String trigger) {
 return ++count;
}
```

This **eval** method implements a counter function, similar to the previous example, but it demonstrates the use of the **skipUpdate()** method to prevent updating the function result in certain cases:

```
private int count = 0;
public Object eval(String trigger) {
 ++count;
 if (count % 5 == 0) {
 // don't update result for multiples of 5
 System.out.println("skip counter update: " + count);
 skipUpdate();
 }
```

```
 return count;
}
```

This **eval** method takes a table as an argument. The input table is assumed to have an string column named **Status** (for example, **production_table** in **update.xml** produced by the RTView XML data simulator). The function returns a copy of the input table with the string in the **Status** column truncated to the length specified by the second argument, **numCharsForStatusCol**:

```
public Object eval(GmsTabularData prodTable, int numCharsForStatusCol) {
 if (prodTable == null || prodTable.getNumRows() < 1)
 return prodTable;
 prodTable = cloneInputTable(prodTable);
 int statusCol = prodTable.getColumnIndex("Status");
 if (statusCol < 0) {
 printErrorMessage("no Status column");
 return prodTable;
 }
 for (int row = 0; row < prodTable.getNumRows(); ++row) {
 String sts = prodTable.getCellValue(row, statusCol);
 if (sts != null && sts.length() > numCharsForStatusCol) {
 sts = sts.substring(0, numCharsForStatusCol);
 prodTable.setCellValue(sts, row, statusCol);
 }
 }
 return prodTable;
}
```

Tabular Functions

Add All Rows or Columns	Calculates the sum within each row or column of the specified Table. All numerical columns will be included in the calculation. **Result Column** - Controls whether to return a column or a row of result values. To get a column of result values, one value for each row, set Return Column to 1. To get a row of result values, one value for each column, set Return Column to 0. **Result Label** - Used to specify a label for the result row or column. To have the Result Label placed in a particular column (when Return Column is 0), specify a column name in Result Label Column. **Return Type** Table **Arguments** Table (type = Table) Return Column (type = Numeric) Result Label (type = Text for Column or Row Label. Default label is Total) Result Label Column (type = Name of Table Column)

Add Columns	Calculates the sum of the specified columns within the specified Table. **Note:** If a specified column is not found, the function attempts to convert the string argument to a numeric value. If this succeeds, the value is used as a constant in each row-wise operation. **Return Type** Table **Arguments** Table (type = Table) First Column Name or Numeric Value (type = Text or Numeric) Second Column Name or Numeric Value (type = Text or Numeric) Result Column Name (type = Name of Table Column)
Average All Rows or Columns	Calculates the average within each row or column of the specified Table. All numerical columns will be included in the calculation. **Result Column** - Controls whether to return a column or a row of result values. To get a column of result values, one value for each row, set Return Column to 1. To get a row of result values, one value for each column, set Return Column to 0. **Result Label** - Used to specify a label for the result row or column. To have the Result Label placed in a particular column (when Return Column is 0), specify a column name in Result Label Column. **Return Type** Table **Arguments** Table (type = Table) Return Column (type = Numeric) Result Label (type = Text for Column or Row Label. Default label is Average) Result Label Column (type = Name of Table Column)
Average Columns	Calculates the average of the specified columns within the specified Table. **Note:** If a specified column is not found, the function attempts to convert the string argument to a numeric value. If this succeeds, the value is used as a constant in each row-wise operation. **Return Type** Table **Arguments** Table (type = Table) First Column Name or Numeric Value (type = Text or Numeric) Second Column Name or Numeric Value (type = Text or Numeric) Result Column Name (type = Name of Table Column)

Baseline Over Time	Calculates a baseline average of the values in the specified Table over the number of Date Part Intervals and offsets the timestamp to a Reference Time. The Table must contain a time column and a number column. This function returns a table containing a baseline calculated for the Number of Intervals specified or for all of the data in the Table if the Number of Intervals is 0. **Date Part** - The date unit to use. Enter **s**, **m**, **h**, **d**, **w**, **M**, **q**, or **y**, for seconds, minutes, hours, days, weeks, months, quarters or years. If left blank, the Date Part will default to seconds. **Reference Time** - After the baseline average has been calculated over the range of data specified, all values in the resulting time column are offset to start at the given Reference Time. This provides an easy way for the baseline to be plotted in a trend graph against a current set of values. **Return Type** Table **Arguments** Table (type = Table) Date Part (type = Text (e.g., s, m, h, d, w, M, q, y)) Date Parts per Interval (type = Numeric) Number of Intervals (type = Numeric) Reference Time (type = Text)
Buffer Table Rows	Appends all rows of the input table to a buffer table that contains rows from previous updates. This function is useful for buffering a table argument to another function in cases where the updates to the table may arrive rapidly (e.g. from an event-driven data source) to ensure that the other function receives all rows. **Note:** If the result of this function is used as the input to another function, all rows will be removed from the buffer table after the other function is updated regardless of the Number of Rows specified. **Table** - Name of input table. **Number Of Rows** - Specify the number of rows in the returned buffer table. If necessary, older rows will be removed to maintain this value. **Return Type** Table **Arguments** Table (type = Table) Number of Rows (type = Numeric)

Combine	Combines Table 1 and Table 2 into a single table. When Combine Rows is 0, the combined table will contain the columns from Table 1 followed by the columns from Table 2. When Combine Rows is 1, the combined table will contain the rows from Table 1 followed by the rows from Table 2. In addition, when Combine Rows is 1, set Ignore Column Names to 0 to reorder the columns of Table 2 so its column names match Table 1 before merging rows. Set Ignore Column Names to 1 to merge rows without trying to match columns by name. **Return Type** Table **Arguments** Table 1 (type = Table) Table 2 (type = Table) Combine Rows (type = Numeric) Ignore Column Names (type = Numeric)
Combine Multi-Server Tables	Combine tables from a multi-server data attachment* into a single table, using the **DataServerName** column as the index column. **Table** - A column named **DataServerName** will be added as the first column of the returned table and contain the name of each data server from which a table is received. The combined table is returned by the function and stored internally for use on the next update. For example, when a table is received from a data server named ds101 it is combined with the previous result table, replacing any existing rows from ds101. **Note:** It is assumed that the value of this argument will be supplied by a multi-server data attachment. A multi-server data attachment is one in which a single data attachment is directed to multiple data servers. For details, refer the Attach to Data section for your data source(s). **Return Type** Table **Arguments** Table (type = Table)
Concatenate Columns	Creates a string concatenation of the values in the given table columns separated by the given Separator(s) and returns the results in a new table column. Specify a semi-colon (;) delimited list of column names for Columns to Concatenate. The Separator can be a single character such as period (.) or forward slash (/), but it can also be a string such as and. **Return Type** Table **Arguments** Table (type = Table) Columns to Concatenate (type = Text) Separator (type = Text)

Convert Columns	Converts the specified columns to the specified type and replaces the input columns with the results. Specify a single column name or a semi-colon (;) delimited list of column names for Columns to Convert. When converting from numeric data (other than Long) to the date-time (Time) type, columns are first converted to Long data and then to Time data. If a String column entry cannot be parsed as a date, then the resulting entry will be blank. **Note:** Convert to Type options (Boolean, Integer, Long, Float, Double, String or Time) may be abbreviated to the first letter. **Return Type** Table **Arguments** Table (type = Table) Columns to Convert (type = Text) Convert to Type (type =Text (Boolean, Integer, Long, Float, Double, String, or Time.))
Copy	Copies the specified Table. **Return Type** Table **Arguments** Table (type = Table)
Count	Counts rows in the specified Table Column. **Return Type** Numeric **Arguments** Table Column (type = Table Column)
Count By Bands	Divides the range given by the Min Value and Max Value into bands and counts the number of rows in the Table argument that contain a value that lies within each band. This function accepts as arguments a Min Value, Max Value and Number of Bands and returns a table containing one column that holds the midpoint values of each band (one row for each band), and N additional columns, one for each column that was contained in the specified Table. These columns contain the total counts of values that fall within the calculated bands. If the Return Cumulative Percents argument is set to true (1) then the returned columns will contain the cumulative percentage of the total count in each cell, rather than the individual counts. **Return Type** Table **Arguments** Table (type = Table) Number of Bands (type = Numeric) Include Min/Max (type = Numeric) Min Value (type = Numeric) Max Value (type = Numeric) Return Cumulative Percent (type = Numeric)

Count Unique Values	Returns a table listing unique values and their counts from the specified Table Column. To include rows in the returned table for values that are not always present in the Table Column, specify a table column in Value List that contains all possible values. If the Value List is not specified, all unique values from Table Column will be included in the returned table. If Restrict to Value List is set to 0, all unique values from the Table Column will be included in the returned table. If Restrict to Value List is set to 1 and Value List is specified, only rows from the Value List will be included. **Return Type** Table **Arguments** Table Column (type = Table Column) Value List (type = Table Column) Restrict to Value List (type = Numeric)
Count Unique Values By Time	Returns a table listing unique values and their counts from a specified Table, sorted by the number of Date Part Intervals. The Table must contain a time and a value column. The returned table contains an interval column, a column for each unique value, and counts for Number of Intervals specified or for all data in the Table if the Number of Intervals is 0. For Date Part, specify s, m, h, d, w, M, q or y for seconds, minutes, hours, days, weeks, months, quarters or years. The optional Date Format argument controls the format in the returned table using the Java SimpleDateFormat class. Use q, qqq or qqqq for short, medium or long versions of quarter notation. To include columns in the returned table for values not always present in the value column, specify a table column in the Value List argument containing all possible values. If Restrict to Value List is 0, or if the Value List is not specified, all unique values from the value column are included in the returned table, otherwise only values from the Value List are included. **Return Type** Table **Arguments** Table (type = Table) Date Parts Per Interval (type = Numeric) Number Of Intervals (type = Numeric) Date Part (type = Text (e.g., s, m, h, d, w, M, q, y)) Date Format (type = Text (e.g., MMMM dd, yyyy hh:mm:ss a)) Value List (type = Table Column) Restrict to Value List (type = Numeric)

Create Selector List	Returns a two column table containing a list of values to be presented in a dropdown list. If the Selector Table has two columns, selector names will be taken from the first column and their values from the second column. If the Selector Table has only one column, its contents will be used for both the names and values in the returned table. If you enter an All Selector name, then the first row of the first column in the returned table will contain the specified All Selector name and the second column will contain a value of *. If Sort Values is set to 1, your selector names column will be sorted alphabetically in the returned table; unless the text consists entirely of numbers in which case it will be sorted numerically. To sort values in descending order set Sort Descending to 1, otherwise (if left blank or set to 0) values will be sorted in ascending order. This function returns a table where the first column contains selector names and the second column contains their values. **Return Type** Table **Arguments** Selector Table (type = Table) All Selector Name (type = Text) Sort Values (type = Numeric) Sort Descending (type = Numeric)
Delta Rows	Computes the delta between incoming rows of tabular data. The Delta Rows function returns a table including, for the specified columns, new values for the difference between this update and the previous. **Delta Column Names** - The name of one or more columns for which deltas will be calculated. **Note:** This field cannot be left blank. **Index Column Names** - The name of one or more columns that uniquely identify a row in the table. If left blank, the default is to calculate deltas for all rows as if they had the same value. The values contained in each index column are concatenated to form a unique index used to organize the resulting summary data. **Replace Data With Deltas** - If set to 1, delta values will replace original values in the same column of the specified Table. If set to 0, new columns will be added. The new columns will use the delta columns names with Delta prefixed. **Display Negative Values** - If set to 1, delta values less than zero will be displayed with a negative sign. If set to 0, they will be displayed as zero. **Return Type** Table **Arguments** Table (type = Table) Delta Column Names (type = Name of Table Column) Index Column Names (type = Name of Table Column) Replace Data with Deltas (type = Numeric) Display Negative Values (type = Numeric)

Delta And Rate Rows	The Delta And Rate Rows function returns a table for the specified columns, including: new values for the difference between this update and the previous, along with the rate of change per second. The new values may be appended to the input table in named columns by prefixing Delta and Rate to the column name or they may replace the values in the input columns. **Table** - The table of interest. **Delta Column Names** - The names of one or more columns for which deltas will be calculated. At least one name must be given. The Time Column Name must be included after the other columns. **Index Column Names** - The names of one or more columns that uniquely identify a row in the table. If left blank, the default is to calculate deltas for all rows as if they had the same value. The values contained in each index column are concatenated to form a unique index used to organize the resulting summary data. **Time Column Name** - The name of a timestamp column that willbe used to calculate the rate of change. If left blank, the rate will not be computed and the Rate column will not be appended. **Replace Data With Deltas** - If set to 1, the delta values replace the original values in the same column in the returned table; otherwise they are in new columns appended to the table. **Display Negative values** - If set to 1, the delta values less than zero will be displayed with a negative sign and the value; otherwise they will be displayed as zero. **Return Type** Table **Arguments** Table (type = Table) Delta Column Names (type = Name of Table Column) Index Column Names (type = Name of Table Column) Replace Data with Deltas (type = Numeric) Display Negative Values (type = Numeric)
Distinct Values	Returns a table that lists all unique values from the specified Column Name. If Sort Values is set to 1, your selector names column will be sorted alphabetically in the returned table; unless the text consists entirely of numbers in which case it will be sorted numerically. To sort values in descending order set Sort Descending to 1, otherwise (if left blank or set to 0) values will be sorted in ascending order. If Use Column Names is set to **1**, then the original column name will be used in the resulting table. If set to **0**, the resulting column will be given the name **Values**. **Note:** Generic column names like this are useful when the data attachment for the Table argument uses a substitution that will cause the column names to change when the substitution changes. **Return Type** Table **Arguments** Table (type = Table) Column Name (type = Name of Table Column) Sort Values (type = Numeric) Sort Descending (type = Numeric) Use Column Names (type = Numeric)

Divide Columns	Divides the value of each row in the second column into the corresponding rows in first column of the specified Table. **Note:** If a specified column is not found, the function attempts to convert the string argument to a numeric value. If this succeeds, the value is used as a constant in each row-wise operation. **Return Type** Table **Arguments** Table (type = Table) First Column Name or Numeric Value (type = Text or Numeric) Second Column Name or Numeric Value (type = Text or Numeric) Result Column Name (type = Name of Table Column)
Ensure Columns	Returns a copy of the specified table with given column name(s) guaranteed to be of the given column type(s). **Table** - The table of interest. **Column Name(s)** - A single column name or a semi-colon (;) delimited list of column names. **Column Type(s)** - A single column type or a semi-colon (;) delimited list of column types. **Note:** Column Type(s) (i.e. Boolean, Integer, Long, Float, Double, String or Time) may be abbreviated to the first letter. Value(s) - A single value or a semi-colon (;) delimited list of values. Semi-colons are supported as literals in the Value(s) argument if: • there is only one value in the Column Name(s) argument, and • the Value(s) argument is enclosed in single quotes. If those conditions are not met, the semi-colon is assumed to be a delimiter for multiple values. If the Table input parameter to this function does not have a column defined in the Column Name(s) parameter, it creates the column of type Column Type(s) with the designated initial Value(s). For columns of type String, Integer and Long, if either no value is provided, or it is not possible to convert the provided value to the column type, the default setting is placed in the column: blank for String and 0 for Integer and Long. For columns of type Double, if no value is provided the column displays 0. And if a value is provided that cannot be converted to Double, the column displays **NaN**. The Value(s) parameter can be used to supply a list of values for initializing new columns added to the table (existing columns do not change). The values are applied to the new columns in the order they are created and converted to the corresponding column type if possible. Otherwise, the result is blank for string, 0 for integer/long, and **NaN** for double. **Return Type** Table **Arguments** Table (type = Table) Column Name(s) (type = Name of Table Column(s)) Column Type(s) (type = Text (Boolean, Integer, Long, Float, Double, String or Time.)) Value(s) (type = Text or Numeric)

Evaluate Expression By Row	Evaluates the expression for all rows of the specified Table, using values from the columns specified, and returns the results in a new table column. **Expression** - Write an expression using variables including standard arithmetic and logical operators as well as a variety of mathematical and string functions. Variables specified in the expression must include a % prefix and cannot begin with a number (e.g. %col1 + %col2). String constants must be enclosed in double quotations (e.g. "%col1 + %col2"). If a value whose column is numeric needs to be treated as a string, perhaps to serve as an argument to a string function, enclose the variable in the expression in double quotations (e.g. "length("%col1") + %col2".) When an Expression is specified and the field activated (by pressing Enter or navigating to another field) the dialog will update with a text field for each variable. Enter the names of columns in the specified Table from which values will be substituted for the corresponding variables in the expression. The types of the values are taken from the types of the columns. **Note:** Numeric and Boolean values are converted to Double. Date columns are not supported. **Result Column Name** - Specifies the name of the column that will be added to the table. **Result Column Type** - Specifies the type of that column that will be added to the table: Double or String (may be abbreviated to the first letter). **Return Type** Table **Arguments** Table (type = Table) Expression (type = Expression) Result Column Name (type = Name of Table Column) Result Column Type (type = Text (Double or String)) See "Expression Syntax Definition" for more information.

Filter And Extract Matches	Returns a table containing all rows from the specified Table in which the value of a column matches a pattern. For each matching row, each token from the specified column that matches a group in the pattern is extracted to a new column. **Filter Column** - Name specified the column to which the pattern is to be applied. If left blank, the pattern is applied to the row name. If Pattern Is Regular Expression is set to 0 (the default), then Pattern should contain a string where * is a wildcard. Otherwise, Pattern should contain a regular expression as described at: http://java.sun.com/j2se/1.5/docs/api/java/util/regex/Pattern.html Number of New Columns specifies the number of new columns that should be added to the result table to contain the matching groups extracted from the filter column. New Column Names specifies the names for each of the new columns, separated by semicolon. For example, if Number of New Columns is 2 and New Column Names is FirstName;LastName then the result table will contain 2 additional columns named FirstName and LastName, after all the columns from the input table. Furthermore, if Filter Column Name is CustomerName and Pattern is * * and PatternIsRegExpr is zero, then if a row in the input table contains Joe Smith in the CustomerName column, the corresponding row in the result table will contain Joe in the FirstName column and Smith in the LastName column. **Return Type** Table **Arguments** Table (type = Table) Filter Column (type = Name of Table Column) Pattern (type = Text) Pattern is a Regular Expression (type = Numeric) Number of New Columns (type = Numeric) New Column Names (type = Text)
Filter By Pattern	Returns a table containing all rows from the specified Table in which the value of a column matches a pattern. **Filter Column Name** - A string that specifies the name of the column in the input table to which the pattern is applied. If this argument is blank, then the pattern is applied to the row name. **Pattern** - A string that specifies a pattern. For each row in the Table, if the value in the specified column matches the pattern then the row is included in the result table. **Pattern Is Regular Expression** - If 0 (the default), then Pattern should contain a string where * is a wildcard that represents any character, including none. If nonzero then Pattern must contain a regular expression as described at: http://java.sun.com/j2se/1.5/docs/api/java/util/regex/Pattern.html **Return Type** Table **Arguments** Table (type = Table) Filter Column (type = Name of Table Column) Pattern (type = Text) Pattern is a Regular Expression (type = Numeric)

Filter By Row	Returns a copy of the specified Table containing only those rows in which the value of the Filter Column exactly matches the Filter Value. It is possible to indicate multiple columns for the filter and multiple values to compare against for each column. **Note:** If the number of specified column names does not correspond to the number of values listed, then extra names and/or values are ignored. Multiple Filter Column Names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col 3). If your column name contains a space or a semicolon, then the entire name must be enclosed in single quotes. Enter * for Filter Value to display all rows in the table. To use * as a literal comparative value, it must be enclosed in single quotes. Multiple Filter Values should be entered as a nested list, where values for a given column are separated by commas within a semicolon (;) delimited list (i.e. val1,val2;val3,val4;val5,val6). If your filter value contains a space or a semicolon, then the entire value must be enclosed in single quotes. **Note:** Spaces around separators are not allowed. **Return Type** Table **Arguments** Table (type = Table) Filter Column Name (type = Name of Table Column) Filter Value (type = Text or Numeric)
Filter By Time Range	Returns a copy of the specified Table, containing only those rows in which the value in the Date/Time Column falls within the given Time Range. **Date/Time Column Name** - Name of a column containing a timestamp. If left blank, the first column is assumed to contain a timestamp. **Time Range Start** - (Optional) A date/time included in the range. **Time Range End** - (Optional) A date/time included in the range. Note: the actual end time used is one second less than this value, so it is non-inclusive. **Return Type** Table **Arguments** Table (type = Table) Date/Time Column Name (type = Name of Table Column) Time Range Start (type = Text) Time Range End (type = Text)
First Table Rows	Returns the first N rows of the given Table or the first N rows for each unique combination of values in the given Index Column(s), where N is the specified Number of Rows to return. If you choose to specify one or more Index Column Names, the values contained in each index column are concatenated to form a unique index and N rows are returned for each index. **Return Type** Table **Arguments** Table (type = Table) Index Column Names (type = Name of Table Column(s)) Number of Rows (type = Numeric)

Format Table Columns	Applies the specified Column Format(s) to the Table To Format and returns a formatted table. Only the columns included in Column Format(s) will be formatted. The Column Format must include column name(s) and the column format(s) separated by :'s. The column name must be enclosed in single quotes if it contains a space. The column format can be specified as a Java format specification, or with the following shorthand: $ for money values, $$ for money values with additional formatting, or () for non-money values, formatted similar to money. For example, if Column Format(s) contains 'Units Completed':$, the Units Completed column in the returned table would be formatted for money. Both positive and negative formats can be supplied, for example: #,###;(#,###). **Return Type** Table **Arguments** Table to Format (type = Table) Column Format(s) (type = Text ('Units Completed':$)) Format Mode -- If 0 (default), then all numeric columns are converted to strings in the result using the local default format for any numeric columns not specified in Column Format(s). If Format Mode is nonzero, then only the numeric columns specified in Column Format(s) are formatted, all other numeric columns in the table remain numeric.
Get Data Server Connection Status	Returns a table with status information about connections from the Display Builder and Display Viewer Application to the default Data Server and any Named Data Servers. The returned table contains one row for each connection, with the following columns: **Name** -- __default for the default Data Server or the name of a Named Data Server. **Connected** -- True if the server connection is operational, otherwise False. **Status** **OK** -- Connection is operational. **no connection** -- No connection to the Data Server. **no service** -- Valid http connection to the rtvdata servlet, but the servlet has no connection to its Data Server. **inactive** --A Data Server connection designated to activate As Needed is inactive because it is not needed. **ConnectionString** -- URL for an http connection to the rtvdata servlet or hostname:port for a direct socket connection to a Data Server. **ReceiveCount** -- Number of data transmissions (pushes) received from the server. **ReceiveTime** -- Time of the most recent data transmission from the server. **Config** -- Configuration string that identifies the RTView version of the Data Server. **Return Type** Table **Arguments** Not applicable

Group By Time	Returns a table containing a summary of all the data in the given Table, subtotaled or aggregated over time as indicated by the Group Type. The summary data in the returned table are grouped into time intervals specified as a number of Date Parts over a Time Range. **Group Type** - The type of aggregation to perform. If left blank the default is **sum**. Permitted values are **sum**, **avg**, **min**, and **max**. When only one value is entered, that value will be applied to all columns in the given Table. To specify multiple values, enter a semi-colon (;) delimited list. For example: **avg;sum;avg. Note:** The order of multiple values should correspond the order of columns in the given Table. To specify column name(s) in the returned Table, add the suffix :**ColumnName**. For example: **avg:Memory;sum:Threads**. **Date/Time Column Name** - Name of a column containing a timestamp. If left blank, the first column is assumed to contain a timestamp. **Date Part** - The date unit to use. Enter **s**, **m**, **h**, **d**, **w**, **M**, **q**, or **y**, for seconds, minutes, hours, days, weeks, months, quarters or years. If left blank, the Date Part will default to seconds. **Date Parts Per Interval** - The size of each interval in Date Parts. **Number Of Intervals** - The number of intervals to return in the summary table, one row for each interval. If set to 0, the number of intervals is determined from the range of data in the Table, or the given Time Range. **Time Range Start** - (Optional) A date/time included in the range. **Time Range End** - (Optional) A date/time included in the range. Note: the actual end time used is one second less than this value, so it is non-inclusive. **Restrict To Time Range** - If set to 1, the resulting summary table includes only those time intervals within the specified Range. **Use Column Names** - Permits the retention of the original column names. When Use Column Names is set to 1 the original column names are retained. If Use Column Names is set to 0, generic column names will be used. The first column will be named Value and the second and subsequent columns will be named Value1, Value2, etc. Generic column names are useful if the attached data uses a substitution that will cause the column names to change when the substitution changes. **Return Type** Table **Arguments** Table (type = Table) Group Type (type = Text (e.g., sum, count, avg, min, max) Date/Time Column Name (type = Table Column) Date Part (type = Text (e.g., s, m, h, d, w, M, q, y)) Date Parts per Interval (type = Numeric) Number of Intervals (type = Numeric) Time Range Start (type = Text) Time Range End (type = Text) Restrict to Time Range (type = Numeric) Use Column Names (type = Numeric)

Group By Time and Unique Values	Returns a table containing a summary of all the data in the given Table, subtotaled or aggregated over time and index columns as indicated by the Group Type. The summary data in the returned table are grouped into time intervals specified as a number of Date Parts over a Time Range, with a further breakdown by Unique Values in the Index Columns. **Group Type** - The type of aggregation to perform. If left blank the default is sum. Permitted values are **sum**, **avg**, **min** and **max**, and **count**. When only one value is entered, that value will be applied to all columns in the given Table. When the Group Type **count** is specified, a column named Count (that displays the number of rows from the input table that were grouped) will automatically be added as the last column in the returned Table. **Note:** When multiple Group Type values are specified, count must be entered last (e.g. **avg;sum;count**). To specify multiple values, enter a semi-colon (;) delimited list. For example: **avg;sum;avg**. **Note:** The order of multiple values should correspond the order of columns in the given Table. To specify column name(s) in the returned Table, add the suffix :**ColumnName**. For example: **avg:Memory;sum:Threads**. **Date/Time Column Name** - Name of a column containing a timestamp. If left blank, the first column is assumed to contain a timestamp. **Date Part** - The date unit to use. Enter s, m, h, d, w, M, q or y, for seconds, minutes, hours, days, weeks, months, quarters or years. If left blank, the Date Part will default to seconds. **Date Parts Per Interval** - The size of each interval in Date Parts. **Number Of Intervals** - The number of intervals to return in the summary table, one row for each interval. If set to 0, the number of intervals is determined from the range of data in the Table, or the given Time Range. **Time Range Start** - (Optional) A date/time included in the range. **Time Range End** - (Optional) A date/time included in the range. Note: the actual end time used is one second less than this value, so it is non-inclusive. **Restrict To Time Range** - If set to 1, the resulting summary table includes only those time intervals within the specified Range. **Index Column Names** - The names of one or more columns that uniquely identify a row in the table. If left blank, the default is to aggregate only by time interval. The values contained in each index column are concatenated to form a unique index used to organize the resulting summary data. **Value List** - A table column containing a set of values which will be included in the set of values for the first Index Column. This is useful if you want the summary table to include values that may or may not be in the Table data. **Restrict To Value List** - If set to 1, the returned table contains only rows that include the values of the specified Value List.

	Return Type Table **Arguments** Table (type = Table) Group Type (type =Text (e.g., sum, count, avg, min, max)) Date/Time Column Name (type =Table Column) Date Part (type =Text (e.g., s, m, h, d, w, M, q, y)) Date Parts per Interval (type =Numeric) Number of Intervals (type =Numeric) Time Range Start (type =Text) Time Range End (type =Text) Restrict to Time Range (type =Numeric) Index Column Names (type =Name of Table Column(s)) Value List (type =Name of Table Column) Restrict to Value List (type =Numeric)

Group By Unique Values	Returns a table containing a summary of all the data in the given Table, subtotaled or aggregated as indicated by the Group Type. The summary data in the returned table are broken down by all combinations of unique values found in the specified Index Columns. **Group Type** - The type of aggregation to perform. If left blank the default is sum. Permitted values are **sum**, **avg**, **min** and **max**, and **count**. When only one value is entered, that value will be applied to all columns in the given Table. When the Group Type count is specified, a column named Count (that displays the number of rows from the input table that were grouped) will automatically be added as the last column in the returned Table. **Note:** When multiple Group Type values are specified, count must be entered last (e.g. **avg;sum;count**). To specify multiple values, enter a semi-colon (;) delimited list. For example: **avg;sum;avg**. **Note:** The order of multiple values should correspond the order of columns in the given Table. To specify column name(s) in the returned Table, add the suffix :**ColumnName**. For example: **avg:Memory;sum:Threads**. **Index Column Names** - The names of one or more columns that uniquely identify a row in the table. If left blank, the default is to use the first column in the Table as the index column. The values contained in each index column are concatenated to form a unique index used to organize the resulting summary data. **Value List** - A table column containing a set of values which will be included in the set of values for the first Index Column. This is useful if you want the summary table to include values that may or may not be in the Table data. **Restrict To Value List** - If set to 1, the returned table contains only rows that include the values of the specified Value List. **Use Column Names** - Permits the retention of the original column names. When Use Column Names is set to 1 the original column names are retained. If Use Column Names is set to 0, generic column names will be used. The first column will be named Value and the second and subsequent columns will be named Value1, Value2, etc. Generic column names are useful if the attached data uses a substitution that will cause the column names to change when the substitution changes. **Restrict To Data Combinations** - If set to 1, the returned table is restricted to only those combinations of values found in the specified Index Columns that occur in the data. If set to 0, the returned table contains all possible combinations of unique values found in the specified Index Columns. **Return Type** Table **Arguments** Table (type = Table) Group Type (type = Text (e.g., sum, count, avg, min, max)) Index Column Names (type = Name of Table Column(s)) Value List (type = Name of Table Column) Restrict to Value List (type = Numeric) Use Column Names (type = Numeric) Restrict To Data Combinations (type = Numeric)

Join	Join Performs an inner join of the Left Table and the Right Table on the columns specified in the Left Column Name and the Right Column Name fields. Multiple column names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col 3). The Left Column Name list must contain the same number of column names as the Right Column Name List and corresponding columns in each list must be of the same type. If a Column Name field is left blank, the row name up to the first : (if the row name contains a :), will be used instead of a column value. The joined table will contain all columns from the Left Table followed by all columns from the Right Table, and include all rows where the value in the Left Column exactly matches the value in the Right Column. Enter a Column Include Mode to specify whether to include in the returned table the columns specified in the Left Column Name and Right Column Name fields. Available options (may be abbreviated to the first letter) are: • **N**one - Do not include columns specified in either Left Column Name or Right Column Name • **L**eft - Include only columns specified in Left Column Name • **R**ight - Include only columns specified in Right Column Name • **A**ll - Include all columns specified in both Left Column Name and Right Column Name If Column Include Mode is left blank, then all columns specified in Left Column Name and Right Column Name will be included. **Return Type** Table **Arguments** Left Table (type = Table) Right Table (type = Table) Left Column Name (type = Name(s) of Table Column, or leave blank to use row name)) Right Column Name (type = Name(s) of Table Column, or leave blank to use row name) Column Include Mode (type = Text (**N**one, **L**eft, **R**ight or **A**ll), or leave blank to include all)

Join Outer	Performs an outer join of the Left Table and the Right Table on the columns specified in the Left Column Name and the Right Column Name fields. Multiple column names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col 3). The Left Column Name list must contain the same number of column names as the Right Column Name List and corresponding columns in each list must be of the same type. If a Column Name field is left blank, the row name up to the first : (if the row name contains a :), will be used instead of a column value. The joined table will contain all columns from the Left Table followed by all columns from the Right Table, and include all rows where the value in the Left Column exactly matches the value in the Right Column. Specify an Outer Join Type (may be abbreviated to the first letter) to select additional rows to include in the joined table: **L**eft (all rows from Left Table), **R**ight (all rows from Right Table), or **F**ull (all rows from both tables). If left blank, a Full outer join will be performed. **Note:** If **Full** is selected, then the **Column Include Mode>Merge** option is recommended to ensure that the returned table contains only one set of the join columns and that all of those columns contain valid values for all rows in the result. Enter a Column Include Mode to specify whether to include in the returned table the columns specified in the Left Column Name and Right Column Name fields. Available options (may be abbreviated to the first letter) are: • **N**one - Do not include columns specified in either Left Column Name or Right Column Name • **L**eft - Include only columns specified in Left Column Name • **R**ight - Include only columns specified in Right Column Name • **A**ll - Include all columns specified in both Left Column Name and Right Column Name • **M**erge - Include columns, names and values, specified in Left Column Name along with values from columns specified in Right Table Column Name. **Note:** To use the Merge option you should specify **Outer Join Type>Full**. If Column Include Mode is left blank, then all columns specified in Left Column Name and Right Column Name will be included. **Note:** In any row where there is no match for the joined column value, the cells from the other table will contain null values. Null values will be represented as follows: blank for strings, 0 for integers and longs, NaN for floats and doubles, and NULL_DATE for dates. **Return Type** Table **Arguments** Left Table (type = Table) Right Table (type = Table) Left Column Name (type = Name of Table Column, or leave blank to use row name) Right Column Name (type = Name of Table Column, or leave blank to use row name Outer Join Type (type = Text (Left, Right or Full)) Column Include Mode (type = Text (None, Left, Right, All or Merge), or leave blank to include all)

Last Table Rows	Last Table Rows Returns the last N rows of the given Table or the last N rows for each unique combination of values in the given Index Column(s), where N is the specified Number of Rows to return. If you choose to specify one or more Index Column Names, the values contained in each index column are concatenated to form a unique index and N rows are returned for each index. **Return Type** Table **Arguments** Table (type = Table) Index Column Names (type = Name of Table Column(s)) Number of Rows (type = Numeric)
Mark Time Gaps	Examines a table of time-stamped data in which data is expected at regular intervals, and if any time gaps in the data are found, marks them by inserting rows containing NaNs into the returned table. The **NaN** values appear as breaks in a trace line if the data table is plotted on a trend graph. If the time interval between any two rows in a table is greater than the expected interval, two new rows are inserted between those rows in which the value of each column to be marked is set to NaN or other specified value. (**NaN** indicates "not a number".) The timestamp of the first new row is set to a value of **1 msec** more than the timestamp of the last row before the time gap, and the timestamp of the second new row is set to a value of **1 msec** less than the timestamp of the next row after the time gap. **Note:** This feature assumes that the table is sorted by timestamp in ascending order. On the second and subsequent updates of this function, the timestamp of the first row in the table is compared to the timestamp of the last row from the previous update. **Table** - The table to be checked for time gaps. The table must have a timestamp column and must be sorted by timestamp in ascending order. **Name of Timestamp Column** - Name of the table timestamp column. **Expected Interval** - Maximum time interval that should occur between consecutive rows in the table. If this interval is exceeded, it is considered a gap. Specify the interval in seconds, or specify a value followed by **m**, **h**, **d**, for minutes, hours, or days. **Names of Columns to Mark** - Names of the columns to be marked with the specified value when rows are added to mark a gap. Multiple column names should be entered as a semicolon (;) delimited list (i.e. **col1;col2;col3**). If no column names are specified, all columns with floating point values are marked. **Mark Columns With** - Numerical value to be assigned when marking columns in the rows added to mark a gap. The default value is **NaN**. **Return Type** Table **Arguments** Table (type = Table) Name of Timestamp Column (type = Text) Expected Interval (type = Numeric, specify s, m, h or d) Names of Columns to Mark (type = Text) Mark Columns With (type = Numeric)

Max All Rows or Columns	Finds the maximum value within each column or row of the specified Table. All numerical columns will be included in the calculation. **Result Column** - Controls whether to return a column or a row of result values. To get a column of result values, one value for each row, set Return Column to **1**. To get a row of result values, one value for each column, set Return Column to **0**. **Result Label** - Used to specify a label for the result row or column. To have the Result Label placed in a particular column (when Return Column is 0), specify a column name in Result Label Column. **Return Type** Table **Arguments** Table (type = Table) Return Column (type = Numeric) Result Label (type = Text for Column or Row Label. Default is Maximum.) Result Label (type = Column Name of Table Column)
Max Columns	Finds the maximum value of specific columns within the specified Table. To use a column containing a date, both the **First Column Name** or **Numeric Value** and the **Second Column Name** or **Numeric Value** arguments must specify the name of a column of type date. This function does not support entering a date string for the argument. **Note:** If a specified column is not found, the function attempts to convert the string argument to a numeric value. If this succeeds, the value is used as a constant in each row-wise operation. **Return Type** Table **Arguments** Table (type = Table) First Column Name or Numeric Value (type = Text or Numeric) Second Column Name or Numeric Value (type = Text or Numeric) Result Column Name (type = Name of Table Column)
Min All Rows or Columns	Finds the minimum value within each column or row of the specified Table. All numerical columns will be included in the calculation. **Result Column** - Controls whether to return a column or a row of result values. To get a column of result values, one value for each row, set Return Column to **1**. To get a row of result values, one value for each column, set Return Column to **0**. **Result Label** - Used to specify a label for the result row or column. To have the Result Label placed in a particular column (when Return Column is 0), specify a column name in Result Label Column. **Return Type** Table **Arguments** Table (type = Table Column) Return Column (type = Numeric) Result Label (type = Text for Column or Row Label. Default is Minimum.) Result Label Column (type = Name of Table Column)

Min Columns	Finds the minimum value of specific columns within the specified Table. To use a column containing a date, both the **First Column Name** or **Numeric Value** and the **Second Column Name** or **Numeric Value** arguments must specify the name of a column of type date. This function does not support entering a date string for the argument. **Note:** If a specified column is not found, the function attempts to convert the string argument to a numeric value. If this succeeds, the value is used as a constant in each row-wise operation. **Return Type** Table **Arguments** Table (type = Table) First Column Name or Numeric Value (type = Text or Numeric) Second Column Name or Numeric Value (type = Text or Numeric) Result Column Name (type = Name of Table Column)
Modulo Columns	Divides the specified columns and returns the remainder in a new table column. **Note:** If a specified column is not found, the function attempts to convert the string argument to a numeric value. If this succeeds, the value is used as a constant in each row-wise operation. **Return Type** Table **Arguments** Table (type = Table) First Column Name or Numeric Value (type = Text or Numeric) Second Column Name or Numeric Value (type = Text or Numeric) Result Column Name (type = Name of Table Column)
Multiply Columns	Multiplies the specified columns. **Note:** If a specified column is not found, the function attempts to convert the string argument to a numeric value. If this succeeds, the value is used as a constant in each row-wise operation. **Return Type** Table **Arguments** Table (type = Table) First Column Name or Numeric Value (type =Text or Numeric) Second Column Name or Numeric Value (type =Text or Numeric) Result Column Name (type =Name of Table Column)

Percent Columns	Divides the value of each row in the second column into corresponding rows in the first column of the specified Table and converts to a percentage. **Note:** If a specified column is not found, the function attempts to convert the string argument to a numeric value. If this succeeds, the value is used as a constant in each row-wise operation. **Return Type** Table **Arguments** Table (type = Table) First Column Name or Numeric Value (type = Text or Numeric) Second Column Name or Numeric Value (type = Text or Numeric) Result Column Name (type =Name of Table Column)
Pivot On Unique Values	Returns a table in which row data from the given Table has been rotated into columns. The Pivot Name Column contains values that become new column names in the returned table. The Key Column is used to group rows containing unique names in the Pivot Name Column into a single row. Enter a single column name or a semi-colon (;) delimited list of column names. When multiple column names are specified, results are grouped by unique occurrences of the combined values of all columns. The Pivot Value Column contains the data of interest. All consecutive rows that contain the same value in the Key Column will have the data in the Pivot Value Column subtotaled into the same row of the resulting table, in the appropriate column. To include columns in the returned table for names that are not present in the Pivot Name Column, specify a table column in the Name List argument which contains all possible names. If Restrict to Name List is set to **0** or if the Name List is not specified, all unique values from the Pivot Name Column are included in the returned table, otherwise only values from the Name List will be included. **Return Type** Table **Arguments** Table (type = Table) Key Column (type = Name(s) of Table Column) Pivot Name Column (type = Name of Table Column) Pivot Value Column (type = Name of Table Column) Name List (type = Text) Restrict To Name List (type = Numeric)
Reference	Makes a reference to the specified Table without copying the contents. This function returns a table. **Return Type** Table **Arguments** Table (type = Table)

Rename Columns	Returns a copy of the specified table with Column Name(s) replaced by the specified New Name(s). Column Name(s) - Enter a single column name or a semi-colon (;) delimited list of column names. New Name(s) - Enter a single column name or or a semi-colon (;) delimited list of column names to replace the specified Column Name(s). **Note:** The number of Columns Name(s) entered cannot exceed the number of New Name(s) entered. **Return Type** Table **Arguments** Table (type = Table) Column Name(s) (type = Name of Table Column(s)) New Name(s) (type = Renamed Table Column(s))
Replace Value in Rows	Replaces text in the specified Column Name with text from Replacement Values. **Column Name** - Input column containing value to be replaced with associated text from specified Replacement Values. **Result Column Name** - Name of column to be created. Replacement Values - String that contains pairs of values and replacement values separated by (:) colons. NOTE: Any values or replacement values that contain a space or a colon must be enclosed in single quotes. **Return Value If No Match** - Determines what value is stored in the Result Column for a row with no match. If greater than 0, then the original value from the input column will be used, otherwise the Default Value is used. **Default Value** - String that is stored in the result column if there is no match and Return Value If Not Match is less than or equal to 0. **Return Type** Table **Arguments** Table (type = Table) Column Name (type = Name of Table Column) Result Column Name (type = Name of Result Column) Replacement Values (type = String) Return Value If No Match (type = Numeric) Default Value (type = String)
RTView Info	Returns a table containing information about the selected RTView **Item**. Select from the following options: • RTView Version Info • Product Version Info • Data Sources **Return Type** Table **Arguments** Item (type = Not Applicable)

Select Column	Returns a table containing only the column specified in Select Column Name from the given Table. This function returns table that contains all rows from the original table. **Return Type** Table **Arguments** Table (type = Table) Select Column Name (type = Name of Table Column)
Send Table	This function can be used to send a table to a remote instance of the Data Server via socket or http. The RTVAgent data adapter is used to receive the table in the Data Server. **Connection** - Connection for the remote RTVAgent. This should either be **hostname:port** (e.g. localhost:5665) or a URL for the rtvagent servlet (e.g. http://SomeServer/rtvagent). **Agent Name** - Name of agent (sender) to the RTVAgent. NOTE: This name should be unique among all agents in the same Agent Class (e.g. District 5 Agent). **Agent Class** - Name that uniquely identifies the class (type) to which this agent belongs (e.g. MyCompany.DistrictAgent). **Table Name** - Name that identifies the **Table to Send** when it is received by the RTVAgent (e.g. Sales Table). **Table to Send** - Table to be sent to the remote RTVAgent. **Note:** This is the only argument that should change on each update. **Insert Column For Agent Name** - If **1**, then a column named **AgentName**, that will contain the value of the Agent Name argument, is inserted into the specified **Table to Send**. This column is useful for identifying which rows were received from which agents, especially when multiple agents in the same class send tables with the same **Table Name**. **Insert Column For Agent Time -** If **1**, then a column named **AgentTime** (that contains the current time) is inserted into the specified **Table to Send**. **Return Type** Table **Arguments** Connection (type = String) Agent Name (type = String) Agent Class (type = String) Table Name (type = String) Table to Send (type = Table) Insert Column For Agent Name (type = Numeric) Insert Column For Agent Time (type = Numeric)

Sort Table	Returns a table sorted by the specified **Sort Column Name**. Specify a single column name or a semi-colon (;) delimited list of column names. If multiple column names are entered, the returned table will be sorted on the first column specified, then the second column, etc. If an invalid column name is entered the original table will be returned. If **Sort Descending** is set to **0**, the Table is sorted in descending order. If **Sort Descending** is set to **1**, the Table is sorted in ascending order. **Note:** If your column contains text it will be sorted alphabetically, unless the text consists entirely of numbers, in which case it will be sorted numerically. If **Use Fast String Compare** is set to **1**, the **java.lang.String.compareTo** method is used to compare strings, which is fast but non-localized. If set to **0**, the **java.text.Collator.compare** method is used to provide a localized string sort, but the sort will be somewhat slower. Default is **0**. **Note: Use Fast String Compare** may produce different results than a standard sort, since the sort is based on the value of the Unicode characters set. **Return Type** Table **Arguments** Table (type = Table) Sort Column Name (type = Name of Table Column) Sort Descending (type = Numeric) Use Fast String Compare (type = Numeric)
Split String	This function returns a table with the given String split using the regular expression in the specified Separator. The Separator should contain a regular expression as described at: *http://java.sun.com/j2se/1.5/docs/api/java/util/regex/Pattern.html* Use Results Column Name to specify the name of the column in the returned table that contains the split results, one per row. **Return Type** Table **Arguments** String (type = String) Separator (type = Regular Expression) Results Column Name (type = Name of Table Column)
Store Table in Cache	This function provides a useful way of adding data from multiple sources into a single cache. Multiple functions can be created with various sources of data that all supply data to a single cache. A value of 0 is returned if the named cache is not defined, otherwise 1 is returned. **Table** - Name of table to be stored in a cache. **Cache Name** - Name of the cache in which the table should be stored. **Note:** This cache must already be defined. **Return Type** Numeric **Arguments** Table (type = Table) Cache Name (type = Name of Cache)

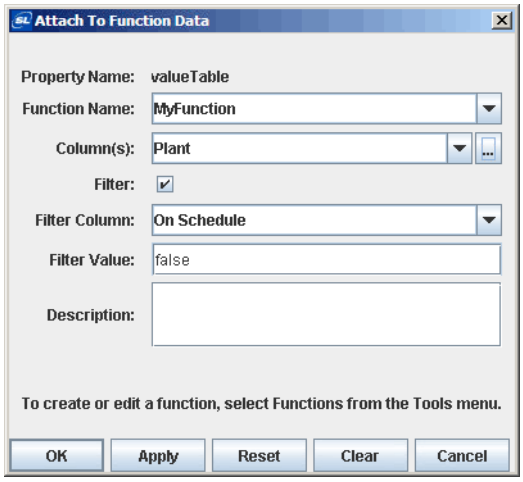
Subtotal By Time	Subtotals the values in the specified Table by the number of Date Part Intervals. The Table must contain a time column and a number column. This function returns a table containing subtotals for the Number of Intervals specified or subtotals for all of the data in the Table if the Number of Intervals is 0. For Date Part, specify **s**, **m**, **h**, **d**, **w**, **M**, **q**, or **y** for seconds, minutes, hours, days, weeks, months, quarters or years. If left blank, the Date Part will default to seconds. The optional Date Format argument controls the format of the time in the returned table, using the Java SimpleDateFormat class. For example, the format **EEE, hh:mm a** will result in a string of the form **Wed, 05:32 PM**. Use **q**, **qqq**, or **qqqq** for short, medium or long versions of quarter notation. For example, **qqq-yyyy** will result in a string of the form **Qtr 1-2005**.If Date Format is left blank the Return Type of the first column in the table will be Date, otherwise it will be String. **Use Column Names** - Permits the retention of the original column names. When Use Column Names is set to 1 the original column names are retained. If Use Column Names is set to 0, generic column names will be used. The first column will be named Interval and the second and subsequent columns will be named Subtotal 1, Subtotal 2, etc. Generic column names are useful if the attached data uses a substitution that will cause the column names to change when the substitution changes. **Return Type** Table **Arguments** Table (type = Table (must contain time column and numeric column)) Interval (type = Numeric) Number of Intervals (type = Numeric) Date Part (type = Text (e.g., s, m, h, d, w, M, q, y)) Date Format (type = Text (e.g., MMMM dd, yyyy hh:mm:ss a)) Use Column Names (type = Numeric)
Subtotal By Unique Values	Returns a table listing all of the unique values found in the first column of the Table Columns argument, along with the sum of the values in the corresponding fields of the remaining Table Columns. To include rows in the returned table for values that are not always present in the Table Columns, specify a table column in the Value List argument which contains all possible values. If Restrict to Value List is set to 0 or if the Value List is not specified, all unique values from the Table Columns will be included in the returned table. Otherwise, only values from the Value List will be included. This function returns a table. **Use Column Names** - Permits the retention of the original column names. When Use Column Names is set to 1 the original column names are retained. If Use Column Names is set to 0, generic column names will be used. The first column will be named Value and the second and subsequent columns will be named Total 1, Total 2, etc. Generic column names are useful if the attached data uses a substitution that will cause the column names to change when the substitution changes. **Return Type** Table **Arguments** Table Columns (type = Table Columns (enter two table columns)) Value List (type = Table Column) Restrict to Value List (type = Numeric) Use Column Names (type = Numeric)

Subtract Columns	Subtracts the value of each row in the second column from the corresponding rows in the first column of the specified Table. **Note:** If a specified column is not found, the function attempts to convert the string argument to a numeric value. If this succeeds, the value is used as a constant in each row-wise operation. **Return Type** Table **Arguments** Table (type = Table) First Column Name or Numeric Value (type = Text or Numeric) Second Column Name or Numeric Value (type = Text or Numeric) Result Column Name (type = Name of Table Column)
Table Contains Values	Returns a copy of the specified Table with a new boolean column containing a value of true for each row where the value in the Comparison Column is in the Comparison Table. **Return Type** Table **Arguments** Table (type = Table containing column in which to search for values from the Comparison Table) Comparison Column Name (type = Name of column in Table in which to search for the values in the Comparison Table.) Result Column Name (type = Name of boolean result column to add to Table. If no name is specified, the column will be named Result.) Comparison Table (type = One column table containing values to look for in the Comparison Column of the Table.)

Attach to Function Data

From the **Object Properties** window you can access the **Attach to Function Data** dialog, which is used to connect an object property to a function. A function enables you to perform calculations on data before it is attached to object in RTView displays. See "Add/Edit Functions" for details on creating functions. The file **functions.rtv** (located in the **tutorials** directory) is provided as an example of the different ways that functions can be utilized in a display.

When an object property is attached to data, the Property Name and Value in the Object Properties window will be displayed in green. This indicates that editing this value from the Object Properties window is no longer possible. To remove the data attachment, and resume editing capabilities in the Object Properties window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.



To bring up the **Attach to Function Data** dialog, right-click on the Property Name from the **Object Properties** window and select **Attach to Data>Function**.

Field Name	Description
Function Name	List of all available local, global, and include file functions. See "Global Functions and Variables" and "Include Display Files" for more information.
Column(s)	If your function contains tabular data you can select which column(s) to display.
Filter	Check box to indicate whether or not to filter this function. Filters can only be used for functions that return tabular data.
Filter Column	Name of the table column to use as a filter.
Filter Value	Value that the filter column must equal. Enter * to display all rows in the table. Enter "*" to use * as a literal comparative value. To list multiple values, separate with a semicolon. For example: **value1**;**value2**;**value3**. If your value contains a semicolon, enclose it in single quotes.
Description	Displays the description of this function. This description can be edited from the "Functions" window.

The **Function Name** drop down menu lists all available functions. If you have not defined any functions this list will be empty. Select **Tools>**"Functions" to create a function. It is possible to type in a function name that has not yet been defined, but it will not be valid until you create a function by that name.

Note: Functions are only available for use within the display where they are created. The **Filter Column** and **Filter Value** fields are only applicable if the attached function returns tabular data.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information entered into the dialog is validated against the list of functions you created for this display. The following describes the significance of the Attach to Function Data validation colors:

Blue	Unknown	Entry does not match any known function or column.
White	Valid state	Entry is valid.
Red	Invalid state	Function is valid, but Filter Column is not recognized.
Gray	Not Required	Field does not require a value. This applies to the Filter Column or Filter Value fields for functions that do not return tabular data.

Substitutions

Substitutions allow you to build open-ended displays in which data attachments depend on values defined at the time the display is run. Generic names, such as $function1 and $function2, are used instead of values for specific functions. Later when the display is running, these generic values are defined by the actual names of specific functions. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see the "Substitutions" section.

Select Columns

From the **Attach to Function Data** dialog you can specify which table columns to display and in what order they will appear. In order to populate the listing of available columns, you must first select a function. See "Add/Edit Functions" for details on creating functions.

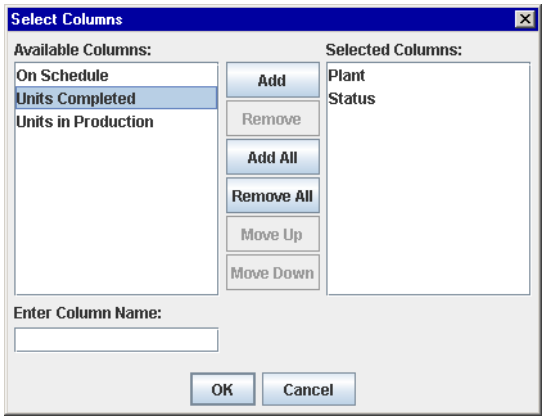
To bring up the **Select Columns** dialog, click on the ellipse ... button in the Column(s) field (or right-click in the Column(s) field and choose **Select Columns**). The dialog should contain a list of Available Columns.

To add a column, select an item from the **Available Columns** list and click on the **Add** button. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of columns in a table by arranging the items in the **Selected Columns** list with the Move Up and Move Down buttons.

"Validation Colors" indicate whether selected columns are valid. However, if even one column selected is invalid the Column(s) field in the **Attach to Function Data** dialog will register as an invalid entry.

Note: Invalid columns will not update.

If no data is available for a table row within a selected column, then the table cell will display one the following values: N/A, false, 0, or 0.0.



The following describes the **Attach to Function Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from function (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied.

Global Functions and Variables

It is possible to specify global functions and variables that will be available to objects in any display. Global functions and variables are defined in Global Definition files that are read by the Display Builder, Display Viewer, Display Server, Data Server and Historian. One instance of each global variable and function exists for each client in all deployments. Depending on how you will be using your function or variable, you may want to define them locally or in an include display file instead.

Use a global function or variable when:

- the same function result is needed in most of your displays
- you want the same variable value in all of your displays

Global functions are updated even if currently open displays do not use those results, therefore avoid defining functions globally unless the results are commonly required information. If you have a function that will only be used in a few displays, you may get better performance using a local or include file function.

Use a local function or variable when:

- the function result is only needed in a single display
- the variable is only needed in a single display

Use a function or variable from an include file when:

- the function result is needed in only a few displays
- the variable value needs to be different in each display

See "Include Display Files" for more information.

To view global function or variable data, you must create a Global Definition file and then configure Application Options to read that file.

Creating a Global Definition File

A Global definition file is an RTView display (**.rtv**) file that contains your function and variable definitions. To create a Global Definition file, create a new display (**.rtv**) file and add your function and variable definitions. Multiple function and variable definitions can be added to a single Global Definition file. You may also create multiple Global Definition files. See "Functions" and "Add/Edit Variables" for more information.

When you have finished adding all of your function and variable definitions and configuring their properties, save this file as your Global Definition (**.rtv**) file. You are now ready to add this file to the Globals tab of the Application Options dialog.

Creating a Reusable Global Definition File

You can create a reusable Global Definition file by specifying substitution value(s) in your function definitions and in the **Application Options** dialog. See "Substitutions" for more information.

To give an example, let us say that your production data is broken down by Plant, Units Completed, and Units in Production. Instead of manually creating a function to sort each column in your table, you could create a single global function named Sort (in a Global Definition file named SortFunc.rtv) and reuse it as a template for the others.

In the Application Options dialog, select the **Globals** tab and separately add the following files and corresponding substitutions:

SortFunc.rtv	$fname:completed $sortColumn:'Units Completed'
SortFunc.rtv	$fname:inProd $sortColumn:'Units in Production'
SortFunc.rtv	$fname:plant $sortColumn:Plant

Once **SortFunc.rtv** is added and applied in the **Globals** tab, the **Sort.$fname** function will show up in the Function dialog three times: Sort.completed, Sort.inProd and Sort.plant.

You can also include variables in your Reusable Global Definition file. If the **SortFunc.rtv** file in the example above had a Public local variable named $fname, the $fname variable would show up in your Variable dialogs three times: $fname.completed, $fname.inProd, and $fname.plant.

Note: In this example we created a function named **getSortCol** that is intended to be used only as an input to the **Sort.$fname** function and should not be defined as a global function. When the function **getSortCol** was created the Public option was not selected, which prevents the function from appearing in the Attach to Function data dialog in any files except the global function definition file.

Adding a Global Definition File to Application Options

In the Display Builder, select **Tools>Options>General>**"Globals Tab" to specify any number of Global Definition files and, optionally, corresponding substitutions.

Viewing Global Data

See "Attach to Function Data" for information on viewing the results of your global functions and "Attach to Variable Data" for information on viewing global variables.

CHAPTER 9 Variables

This section contains the following:

Add/Edit Variables

Variables allow you to use "Control Objects" to update other objects in your RTView display. To setup a control object to update the value of a variable, attach its **varToSet** property to a variable using the "Attach to Variable Data" dialog. To display the results, attach another object to the variable.

Note: Variables are only available for use within the display where they are created.

Select **Tools>Variables** to open the Variables dialog. To create a variable, enter the **Variable Name** and **Initial Value** and click on the **Add** button. If the variable ends in Color (i.e.: bgColor), click anywhere in the Initial Value field and choose a color from the color chooser palette to set the Initial Value. To delete, select a variable from the list and click **Remove**.

Linking Substitutions with Variables

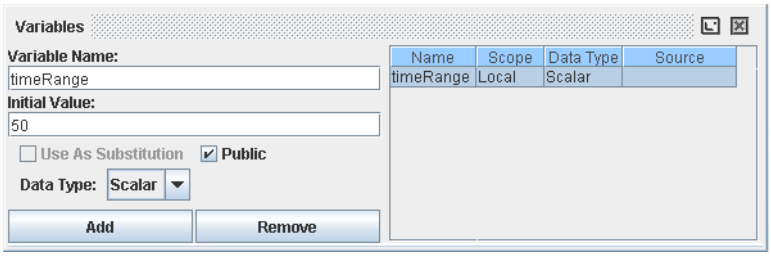
To link a variable with a substitution by the same name, select **Use As Substitution** when creating a variable. If no substitution by that name exists, one will be created and given the value entered in **Initial Value**. If a substitution by that name already exists, the variable will assume the value of the substitution instead of the value entered in the **Initial Value** field. Whenever the value of the variable is updated, the associated substitution will also be updated and when the substitution is updated the associated variable will update. The **Use As Substitution** feature is not available for variables that use tabular input data or when editing existing variables.

Note: If the **Variable Name** begins with a $, the **Use As Substitution** check box will automatically be selected.

If the variable specified will be defined as a global variable or in an included display file, then selecting the **Public** check box will make this variable available in all displays. When the **Public** check box is deselected, this variable will only be available for data attachments in the display (.rtv) file where it was defined. See "Global Functions and Variables" and "Include Display Files" for more information.

Variables associated with global or included display files can only be modified from the display in which they were originally defined.

Note: If a global variable shares the same name as a local variable, only the local variable will be displayed and the global variable will be ignored.

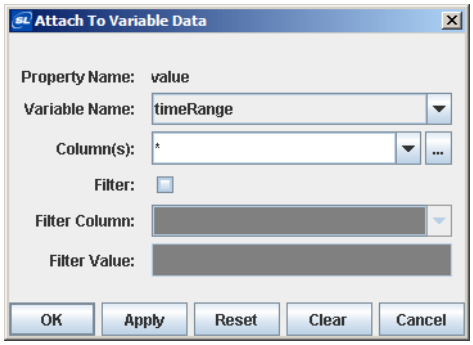


Name	**Name of the variable**
Scope	Scope of the variable: **Local** - Variable is defined in the current display. **Included** - Variable is defined in an included display file. See "Include Display Files" for more information. **Global** - Variable is defined in a Global Definition file. See "Global Functions and Variables" for more information.
Data Type	Data type of the variable: **Scalar** - Variable uses scalar input data. This is the default data type. **Tabular** - Variable uses table input data. **Note:** The **Use As Substitution** feature is not available when this data type is selected.
Source	Name of the file in which a global variable or included display file is defined.

Attach to Variable Data

From the **Object Properties** window you can access the **Attach to Variable Data** dialog, which is used to connect an object property to a variable. Variables can be updated by "Control Objects" allowing you to control the values in other objects in your RTView display. See "Add/Edit Variables" for details on creating variables.

When an object property is attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing this value from the **Object Properties** window is no longer possible. To remove the data attachment, and resume editing capabilities in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.



To bring up the **Attach to Variable Data** dialog, right-click on the Property Name from the Object Properties window and select **Attach to Data>VARIABLE**.

Variable Name	List of all available variables.
Column(s)	If the data type of the attached variable is tabular, you can select which column(s) to display.
Filter	Check box to indicate whether or not to filter the variable. Filters can only be used for tabular data.
Filter Column	Name of the column to use as a filter.
Filter Value	Value that the filter column must equal. Single or multiple values may be listed. Enter * to display all rows in the table. Enter "*" to use * as a literal comparative value. To list multiple values, separate with a semicolon. For example: **value1;value2;value3**. If your value contains a semicolon, enclose it in single quotes.

The **Variable Name** drop down menu lists all available local, global and include file variables. If you have not defined any variables this list will be empty. Select **Tools>Variables** to create a variable. See "Global Functions and Variables", "Include Display Files", and "Add/Edit Variables" for more information.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid.

The following describes the significance of the Attach to Variable Data validation colors:

White	Valid state	Entry is valid.
Red	Invalid state	Variable is invalid; or Variable is valid, but Column(s) or Filter Column selected are not.
Gray	Not Required	Field does not require a value. This applies to the Column(s), Filter Column or Filter Value field for variables that are not tabular.

The following describes the **Attach to Variable Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.

Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from the variable (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied.

Global Functions and Variables

It is possible to specify global functions and variables that will be available to objects in any display. Global functions and variables are defined in Global Definition files that are read by the Display Builder, Display Viewer, Display Server, Data Server and Historian. One instance of each global variable and function exists for each client in all deployments. Depending on how you will be using your function or variable, you may want to define them locally or in an include display file instead.

Use a global function or variable when

- the same function result is needed in most of your displays
- you want the same variable value in all of your displays

Global functions are updated even if currently open displays do not use those results, therefore avoid defining functions globally unless the results are commonly required information. If you have a function that will only be used in a few displays, you may get better performance using a local or include file function.

Use a local function or variable when

- the function result is only needed in a single display
- the variable is only needed in a single display

Use a function or variable from an include file when

- the function result is needed in only a few displays
- the variable value needs to be different in each display

See "Include Display Files" for more information.

To view global function or variable data, you must create a Global Definition file and then configure Application Options to read that file.

Creating a Global Definition File

A Global definition file is an RTView display (.rtv) file that contains your function and variable definitions. To create a Global Definition file, create a new display (.rtv) file and add your function and variable definitions. Multiple function and variable definitions can be added to a single Global Definition file. You may also create multiple Global Definition files. See "Functions" and "Add/Edit Variables" for more information.

When you have finished adding all of your function and variable definitions and configuring their properties, save this file as your Global Definition (**.rtv**) file. You are now ready to add this file to the **Globals** tab of the **Application Options** dialog.

Creating a Reusable Global Definition File

You can create a reusable Global Definition file by specifying substitution value(s) in your function definitions and in the **Application Options** dialog. See "Substitutions" for more information.

To give an example, let us say that your production data is broken down by Plant, Units Completed and Units in Production. Instead of manually creating a function to sort each column in your table, you could create a single global function named **Sort** (in a Global Definition file named **SortFunc.rtv**) and reuse it as a template for the others.

In the Application Options dialog, select the Globals tab and separately add the following files and corresponding substitutions:

SortFunc.rtv	$fname:completed $sortColumn:'Units Completed'
SortFunc.rtv	$fname:inProd $sortColumn:'Units in Production'
SortFunc.rtv	$fname:plant $sortColumn:Plant

Once **SortFunc.rtv** is added and applied in the Globals tab, the **Sort.$fname** function will show up in the Function dialog three times: Sort.completed, Sort.inProd and Sort.plant.

You can also include variables in your Reusable Global Definition file. If the **SortFunc.rtv** file in the example above had a Public local variable named $fname, the $fname variable would show up in your Variable dialogs three times: $fname.completed, $fname.inProd, and $fname.plant.

Note: In this example we created a function named **getSortCol** that is intended to be used only as an input to the **Sort.$fname** function and should not be defined as a global function. When the function **getSortCol** was created the **Public** option was not selected, which prevents the function from appearing in the Attach to Function data dialog in any files except the global function definition file.

Adding a Global Definition File to Application Options

In the Display Builder, select **Tools>Options>General>**"Globals Tab" to specify any number of Global Definition files and, optionally, corresponding substitutions.

Viewing Global Data

See "Attach to Function Data" for information on viewing the results of your global functions and "Attach to Variable Data" for information on viewing global variables.

CHAPTER 10 RTView Data Sources

RTView provides a variety of data sources that can be used to populate dashboards, generate reports and drive alerts. For ease of use and improved performance, RTView has been optimized to reduce the impact of data acquisition, federate information to interested clients and provide options specific to each data source.

RTView comes standard with XML, other data sources can be licensed individually. To find out which data sources are available in your installation, select **Help>About RTView** in either the Display Builder or the Display Viewer Application. For information on licensing additional data sources, including custom data sources, contact SL Sales at **800.548.6881** or **415.927.8400**.

Data sources supported by RTView can be event-driven in real-time (XML, JMS, JMX™ [notifications], StreamBase®, TIBCO Rendezvous® and TIBCO Hawk®), or persistent and require queries (SQL, OSIsoft® PI, IBM WebSphere® MQ and TIBCO® Enterprise Message Service™). RTView features a unique **Attach to Data** dialog for each data source to facilitate the acquisition of available data and the rapid configuration of dashboards, reports and alerts.

Some RTView data sources, including OSIsoft® PI and IBM WebSphere® MQ, are exclusive to monitoring, while others provide write-back facilities – known as commands. RTView commands allow applications to be built that can, for example, write to a database, send a JMS message, or call a JMX™ MBean operation.

The **Data Sources** section of the documentation provides the following sections with detailed information specific to each data source.

Note: This is an overview of all the information provided and some of sections described below do not apply to all data sources.

System Requirements and Setup

This section will help you determine system requirements and setup necessary for your particular data source. You must also review the main "System Requirements" section of this documentation.

Attach to Data

For each data source there is a different way to select the appropriate data to input to dashboard objects, alerts, analytic functions and reports. In the Display Builder, a unique dialog is provided for each data source that allows you to discover available data, as well as subscribe to or query a particular data value.

Define Command

Some data sources allow write-back facilities, known as commands. With commands it is possible to configure RTView to write to a data source based on user interaction or activation of alerts.

Substitutions

Many data sources have standard built-in substitutions that allow for the parameterization of described data values, which can be realized at run-time. This allows you to build open-ended dashboards in which data attachments and commands depend on values defined by user interaction. In this way, a single dashboard template can be reused to show data and execute commands from a number of different sources.

Application Options

In this section you will learn how to describe the connection, as well as other default attributes, of your particular data source. Options available via command line arguments are described in the section titled **Command Line Options**.

Deployment

This section contains details about the deployment process specific to your data source. Please begin in the main "Deployment" section of this documentation for information on available options, the pros and cons of each, and how to choose and implement the optimal deployment.

Examples

Sections titled **Quick Start**, **Demos**, **Simulators**, or **Sample Application** will assist you in understanding how to best utilize RTView with your particular data source.

The Data Sources section contains the following:

IBM WebSphere MQ Data Source

The IBMMQ data adapter allows access to IBM WebSphere® MQ servers and the ability to query metrics from key MQ entities such as Queue Managers, Queues and Channels. Place raw metrics into RTView functions to calculate SLAs and then present data in dashboards, reports and automated alerts. These metrics can be used to monitor the health and internal state of the MQ servers.

This section includes:

System Requirements and Setup - IBMMQ

System Requirements

In addition to basic "System Requirements", the IBMMQ data source requires IBM WebSphere® MQ and is only available if you include your IBM WebSphere MQ provider's IBM WebSphere MQ jars in "RTV_USERPATH". See the **README_sysreq.txt** file in your installation's home directory for the current version(s) supported.

Setup

In addition to general environment variables (see "Setup"), you must include the IBM WebSphere MQ jars in "RTV_USERPATH":

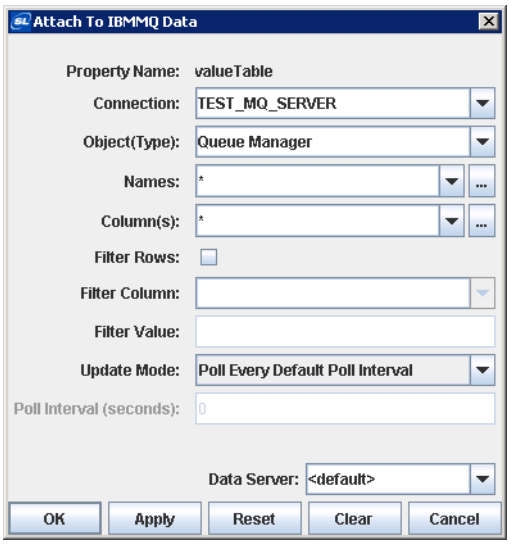
Name	Description	Example
RTV_USERPATH	Location of these IBM WebSphere MQ jars: com.ibm.mq.jar, connector.jar, com.ibm.mq.pcf-6.0.jar. **Note:** If RTV_USERPATH already exists, append IBM WebSphere MQ jars to it.	C:\Program Files\IBM\WebSphere MQ\Java\lib\com.ibm.mq.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\connector.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\com.ibm.mq.pcf-6.0.jar **Note:** com.ibm.mq.pcf-6.0.jar can be obtained from the MS0B: IBM WebSphere MQ Java classes for PCF support pack: http://www-1.ibm.com/support/docview.wss?rs=203&uid=swg24000668&loc=en_US&cs=utf-8&lang=en

Attach to IBMMQ Data

Note: The IBMMQ data source may not be licensed in your RTView installation.

From the **Object Properties** window you can access the **Attach to IBMMQ Data** dialog, which allows you to query IBM MQ Objects for selected metrics on one or more MQ servers. Once a property has been attached to a metric, it receives continuous updates.

To display the **Attach to IBMMQ Data** dialog, right-click on the Property Name from the **Object Properties** window and select **Attach to Data>IBMMQ**. The **Attach to IBMMQ Data** dialog provides several drop down menus. If the drop down menu does not contain the item you require, type your selection into the text field.



Field Name	Description
Connection	Select name of a connection from the drop down menu. Connections can be defined on the "IBMMQ Connections Tab" of the Applications Options dialog. Enter * to select all defined connections. Data is queried across defined connections and is updated as it is returned from individual connections. Since the data is updated asynchronously, it is recommended that attached data are applied to caches in order to aggregate.

Object (Type)	Select type of object to be queried from the drop down menu. Choose from the following: **Connections** -- Defined connections used by the data source. **Queue Manager** -- Queue Manager object used by this connection. **Channel Names** -- List of named channels used by the Queue Manager. **Channel Status** -- Channel Status information for named channels. **Channels** -- Channel attributes for named channels. **Queue Names** -- List of named Queues used by the Queue Manager. **Queues** -- Inquired Queue Attributes for named Queues. **Note:** If the named Queue is a Model Queue, then (as per the API) a dynamic queue is created on the server and an error message is displayed that identifies the cause, disconnects, and marks the connection invalid to prevent re-occurrences. If this happens, investigate the Queues data attachment and modify it so that model queues are not referenced by the Queues query or use the **Queues (PCF)** query instead. **Queues (PCF)** -- Queue Attributes for named Queues queried using PCF (Program Control Format) messages.
Names	Enter an object name associated with the specified **Object (Type)** or click on the ... button to open the "Select Object Name(s)" dialog. Enter * to select all object names associated with the specified **Object (Type)** or a value of * can be used to select object names that share a similar prefix (e.g. **SYSTEM.DEF***). Multiple object names can be entered manually as a semicolon (;) delimited list (i.e. name1;name2;name3). If your object name contains a space or a semicolon, then the entire name must be enclosed in single quotes.
Columns	Select a column from the drop down menu. Click on the ... button to open the "Select Columns" dialog.
Filter Rows	Check box to indicate whether or not to filter rows. See "Row Filtering" for more information.
Filter Column	Name of the column to use as a filter. Multiple column names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col 3). If your column name contains a space or a semicolon, then the entire name must be enclosed in single quotes.
Filter Value	Value that the Filter Column must equal. Multiple filter values should be entered as a nested list, where values for a given column are separated by commas within a semicolon (;) delimited list (i.e. val1,val2;val3,val4;val5,val6). If your filter value contains a space or a semicolon, then the entire value must be enclosed in single quotes. When * is entered as a filter field value, data for all values in the specified filter column will be used to update the object property. When "*" is entered, only the literal comparative value will be used. These are only allowed for objects which display tabular data.
Update Mode	Specify which mode to use to update the Connection: **Poll Every Default Poll Interval** -- Update connection each Default Poll Interval. See the "IBMMQ Options Tab" in the **Application Options** dialog for information on setting the **Default Poll Interval**. This is the default **Update Mode**. **Every Poll Interval** -- Update connection each Poll Interval. If this option is selected, you must specify a **Poll Interval**. **Poll On Demand** -- Update connection each time a display that uses this data attachment is opened and each time a substitution string that appears in the data attachment has changed. **Poll Once (Static Data)** -- Poll for data only once. Select if the data returned by this attribute or operation is static.

Poll Interval

Specify the interval (in seconds) to update connection. This option is only available if the **Update Mode** selected is **Every Poll Interval**.

Note: Because the Poll Interval is superseded by the General Update Period, the amount of time elapsed between updates may be longer than the value entered. For example, if the General Update Period is 2 seconds and the Poll Interval is 5 seconds, the connection will be updated every 6 seconds. See "General Tab" for more information.

Data Server

Select to read data through your configured Data Server and not directly from the The IBMMQ data source.

Default - Select the default Data Server you configured in the **Application Options>**"Data Server Tab".

None - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source.

Named Data Servers - Select a Named Data Server that you configured in **Application Options>Data Server**. See "Data Server Tab" for more information.

Multi-Server Attachment - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more **Named Data Servers** (e.g. **ds101;ds102**). Each name specified must correspond with a Named Data Server that you configured in Application Options>Data Server. It is also possible to specify **__default** and **__none** (e.g. **__default;ds101;ds102**).

Note: The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of * can be entered to specify all data servers, including **__default** and **__none**.

When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received.

A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways:

1. The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows.
2. The multi-server attachment can be applied as the **Table** argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

When an object property has been attached to data, the Property Name and Value in the Object Properties window will be displayed in green. This indicates that editing values from the Object Properties window is no longer possible. To remove the data attachment and resume editing capabilities in the Object Properties window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid.

The following describes the significance of the Attach to IBMMQ Data validation colors:

Blue	Unknown	Cannot validate entry.*

White	Valid state	Entry is valid.
Red	Invalid state	Entry is not valid.

*If a Connection is validated as Unknown, RTView will attempt to validate it when you click OK or Apply.

Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. For example, a generic connection value such as **$conn** is used instead of a defined connection or **$names** instead of a list of names. Later when the display is running, **$conn** is defined by a specific named connection and **$names** by data from a specific set of named objects, such as queues relating to a particular application. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

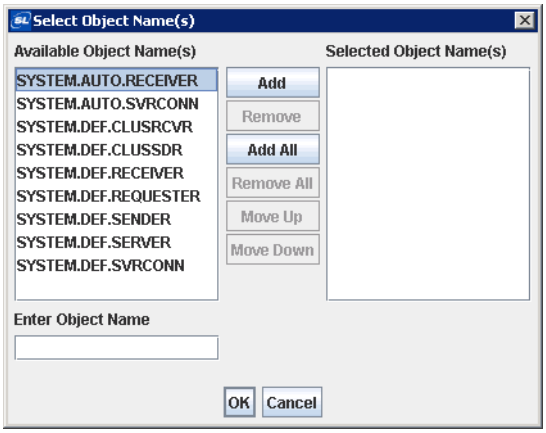
Select Object Name(s)

To bring up the **Select Object Name(s)** dialog, click on the ... button in the **Names** field (or right-click in the **Names** field and choose **Select Object Names**). The dialog should contain a list of available object names.

Note: In order to populate the list of available names, you must first select a valid **Object (Type)**.

To add a name, make a selection from the Available Object Name(s) list and click on the **Add** button. If the item you require is not listed, you can type your selection into the **Enter Object Name** field. Click the **Remove** button to delete an item previously added to the Selected Object Name(s) list. You can control the order of items in this list with the **Move Up** and **Move Down** buttons.

Validation colors indicate whether selected object names are valid. However if even one name selected is invalid, the **Names** field in the **Attach to IBMMQ Data** dialog will register as invalid.



Select Columns

To bring up the **Select Columns** dialog, click on the ... button in the **Columns** field (or right-click in the **Columns** field and click **Select Columns**). The dialog should contain a list of Available Columns.

Note: In order to populate the listing of available columns, you must first select a valid Object (Type) and Name(s).

To add a column, make a selection from the **Available Columns** list and click on the **Add** button. If the item you require is not listed, you can type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the Selected Columns list. You can control the order of items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

Validation colors indicate whether selected columns are valid. However, if even one column selected is invalid, the **Columns** field in the **Attach to IBMMQ Data** dialog will register as invalid.

The following describes **Attach to IBMMQ Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Application Options - IBMMQ

To access the **Application Options** dialog, select **Tools>Options** in the Display Builder.

Options specified in IBMMQ options tabs can be saved in an initialization file (**IBMMQOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server and Historian to set initial values. If no directory has been specified for your initialization files and **IBMADMOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

Note: Options specified using command line arguments will override values set in initialization files. See "Command Line Options - IBMMQ" for more information.

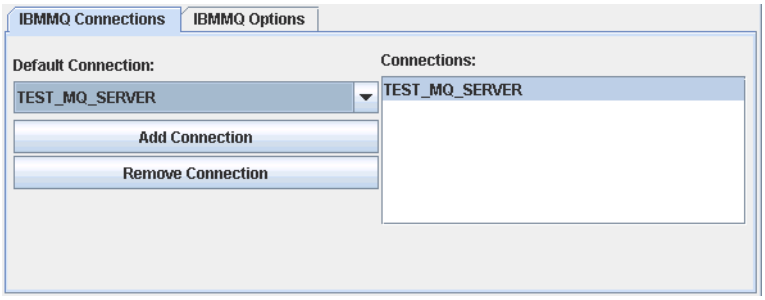
There are two Application Options tabs for IBMMQ: "IBMMQ Connections Tab" and "IBMMQ Options Tab".

IBMMQ Connections Tab

This tab allows you to add or remove connections and set your default connection.

When you add an IBMMQ connection to the list it will be highlighted in yellow indicating that RTView has not connected to it. To attempt to connect, click **OK**, **Apply**, or **Save**. If the background remains yellow, then RTView was unable make a connection. Check that your connection was setup correctly and that the server is running.

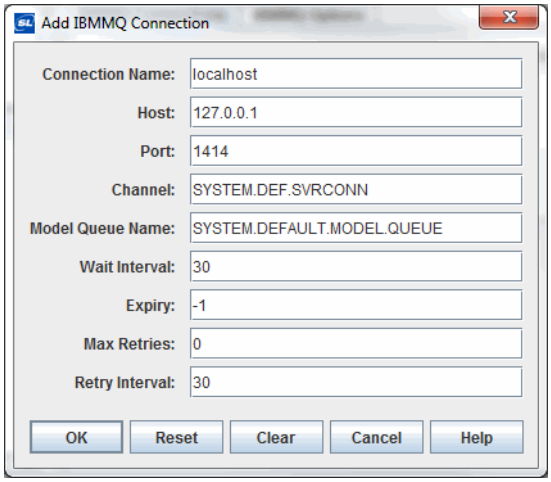
Note: Regardless of which tab you are currently working from in the **Application Options** dialog, each time you click **OK**, **Apply**, or **Save**, RTView will attempt to connect to all unconnected connections.



Field Name	Description
Default Connection	Name of connection used as the default for data attachments. Select from drop down menu to change default setting.

Add Connection

Click to open the **Add IBMMQ Connection** dialog. To edit, select a connection from the list and double-click. Connections that are updating objects in a current display cannot be renamed.



Connection Name -- Name to use when referencing this IBMMQ connection. This name will appear in the **Connections** drop down menu of the "Attach to IBMMQ Data" dialog.

Host -- Name or IP address of the host computer.

Port -- Port number of the connection.

Channel -- Client Channel to use for this connection (use **SYSTEM.DEF.SVRCONN**)

Model Queue Name -- Named model queue of the connection.

Wait Interval -- Wait interval (in seconds) between attempts to create a connection.

Expiry -- The time (in seconds) in which the attempt to connect to the IBMMQ data source will expire. The default value is -1, which means the connection will not expire.

Max Retries -- Maximum number of subsequent connection retry attempts. A retry will only be attempted when the initial connection failed or when a broken connection error has occurred. When a successful connection has been made, the connection attempt counter is reset to zero.

If Max Retries is set to **0**, only an initial connect attempt is made for this connection.

If Max Retries is set to **-1**, unlimited connection attempts will be made until a connection is established. This may be useful when MQ servers are cycled for maintenance, or the servers are expected to be down for a period and a connection is wanted when the servers come back up.

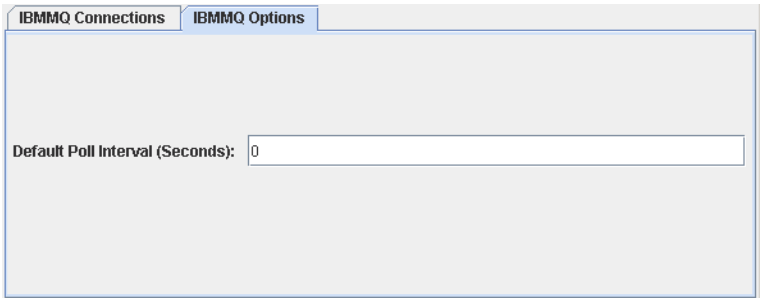
Retry Interval -- Minimum interval (in seconds) between connection retry attempts.

Remove Connection

Select a connection from the list and click **Remove Connection** to delete. Connections that are updating objects in a current display cannot be removed.

IBMMQ Options Tab

This tab allows you to control how often defined connections will be queried.



Field Name	Description
Default Poll Interval	Enter the time in seconds to control how often the defined connections will be queried. Default is 0, which queries according to the General Update Period specified in Application Options on the "General Tab". Because the Default Poll Interval is superseded by the General Update Period, the amount of time elapsed between updates may be longer than the value entered. For example, if the General Update Period is 2 seconds and the Default Poll Interval is 5 seconds, the connection will be updated every 6 seconds.

RTView Deployment - IBMMQ

This page contains details about the deployment process that are specific to your data source. Please go to the **Deployment** section of this documentation for instructions on how to implement your RTView deployment option. Return to this page whenever you are instructed to refer to deployment information that is specific to your data source.

System Requirements and Setup

The IBMMQ data source has additional "System Requirements and Setup".

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options" and "RTV_JAVAOPTS" for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
IBMMQOPTIONS.ini	Contains data source options for IBMMQ.

Note: Options specified using command line parameters override values set in these initialization files.

Command Line Options - IBMMQ

In addition to General Options, the following command line arguments are enabled with the IBMMQ data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: **"-sub:$data:my Data"**).

Name	Description
-ibmmqtrace:N	Enable data source specific tracing. N is an Integer and higher numbers yield more trace output. Default is no tracing enabled (i.e. -ibmmqtrace:0). Example: **-ibmmqtrace:6**
-ibmmqoption:MQTrace=N	Enable low level internal IBMMQ communication library tracing. N is an Integer and higher numbers yield more trace output. Default is no tracing enabled (i.e. **-ibmmqoption:MQTrace=0**). Example: **-ibmmqoption:MQTrace=1**

IBMMQ Custom Handler

Note: This section assumes you have a working knowledge of writing, compiling, and deploying Java classes.

IBM MQ Connections created with RTView can have a handler associated with them, which allows the programmatic setting of additional properties associated with that connection. The handler associated with a connection is specified as a fully qualified class name, in the IBM connection dialog thus:

- Handler Class: Fully Qualified Class Name of GmsRtViewIbmMqDsConnectionHandler subclass, which is equivalent to the **handler=<fully qualified class name>** field in the ibmmq connection string.
- The named class is loaded at runtime using the user specified class path (RTV_USERPATH environment variable), and thus the user may specify a unique handler per connection, or share one handler between multiple connections.
- The handler must be a subclass of the **(com.sl.gmsjibmmqds)GmsRtViewIbmMqDsConnectionHandler** defined in the **gmsjibmmqds.jar** file in the **RTV_HOME/lib** directory.
- Your handler should override the public Hashtable **getConnectionProperties(String connectionName)** method and return a hashtable populated with additional properties for the named connection.
- Example uses might be to set user name and password on a per connection basis, or establish a secure connection to using user configured SSL.

Example - Set username / password in MQ Data Source

This example will create a java class named **MyMqHandler** that extends the **GmsRtViewIbmMqDsConnectionHandler** class.

In **MyMqHandler.java**, define the following method:

```
public Hashtable getConnectionProperties(String connectionName)
```

This method is called to retrieve the additional (userID, password) properties and will be used when RTView creates an MQ connection, where the handler field has been set to **MyMqHandler**.

Add **RTV_HOME/lib/gmsjibmmqds.jar** to your classpath when you compile **MyMqHandler**. The compiled **MyMqHandler** class file must be included in the RTView classpath by adding it to the definition for the RTV_USERPATH environment variable.

The following source code is an example of **MyMqHandler**:

```
// import com.sl.gmsjibmmqds to reference the
com.sl.gmsjibmmqds.GmsRtViewIbmMqDsConnectionHandler class
 import com.sl.gmsjibmmqds.*;
// import java.util to reference the java.util.Hashtable
 import java.util.*;
/**
  * This example class MyMqHandler extends the GmsRtViewIbmMqDsConnectionHandler class
  * to set userID and password properties
  */
 public class MyMqHandler extends GmsRtViewIbmMqDsConnectionHandler {
 /**
  * Subclasses should override this method (getConnectionProperties) to return
  * a Hashtable of connection properties suitable to pass into the
  * MQQueueManager constructor
  * as described in the IBM WebSphere MQ classes for Java API documentation.
  * <p>
  *
  * @param connectionName rtview ibmmqds connection name
  * @return populated Hashtable of connection properties
  */
 @Override
 public Hashtable getConnectionProperties (String connectionName)
 {
 Hashtable connectionProperties = new Hashtable();
System.out.println("MyMqHandler: Setting connection properties for connection: " +
connectionName);
// Set userID and password variables
String userID = "MyUserID";
String password = "MyPassword";
// Set userID and password variables, differently, based on connection name
if (connectionName.equals("SomeConnection")) {
userID = "MyOtherUserId";
password = "MyOtherPassword";
 }
// Set userID and password properties in Hastable to be returned
System.out.println("MyMqHandler:[" + connectionName + "] Setting userID: " + userID);
connectionProperties.put("userID" /* CMQC.USER_ID_PROPERTY */ , userID);
System.out.println("MyMqHandler:[" + connectionName + "] Setting password: " + password);
connectionProperties.put("password" /* CMQC.PASSWORD_PROPERTY */ , password);
// return populated connection properties
 return connectionProperties;
 }
```

```
}
```

Note: You should replace "MyUserID", "MyPassword" / "MyOtherUserId", "MyOtherPassword" as appropriate, or use some approved method to fetch these values, to avoid having them exposed in plain text.

This example uses the literal values ("userID", "password") for properties for simplicity to avoid dependencies on the equivalent property constant values, in the relevant IBM MQ Java client libraries. The constant values could be used by adding the appropriate import statement and adding the relevant jar file to the classpath at build and run-time.

Using SSL for MQ connections

To use SSL with RTView, you need to create a Java class that extends the **GmsRtViewIbmMqDsConnectionHandler** class to provide the relevant properties to establish an SSL connection.

For example, let's say we create a handler called **MySSLMqHandler**.

In **MySSLMqHandler.java**, define the following method:

```
public Hashtable getConnectionProperties (String connectionName)
```

This method is called to set and return the additional SSL properties required for a named MQ connection. In this example, we assume a shared truststore / keystore to be used for multiple connections.

Add **RTV_HOME/lib/gmsjibmmqds.jar** to your classpath when you compile **MyMqHandler**. The compiled **MyMqHandler** class file must be included in the RTView classpath by adding it to the definition for the RTV_USERPATH environment variable.

The following is an example of **MySSLMqHandler**:

```
// import com.sl.gmsjibmmqds to reference the
com.sl.gmsjibmmqds.GmsRtViewIbmMqDsConnectionHandler class
import com.sl.gmsjibmmqds.*;
// import java.util to reference the java.util.Hashtable
import java.util.*;
/**
  * This example class MySSLMqHandler extends the GmsRtViewIbmMqDsConnectionHandler class
   * to set SSL connection properties
   */
 public class MySSLMqHandler extends GmsRtViewIbmMqDsConnectionHandler {
 /**
   * Subclasses should override this method (getConnectionProperties) to return
   * a Hashtable of connection properties suitable to pass into the
   * MQQueueManager constructor
   * as described in the IBM WebSphere MQ classes for Java API documentation.
   * <p>
   *
   * @param connectionName rtview ibmmqds connection name
   * @return populated Hashtable of connection properties
   */
@Override
public Hashtable getConnectionProperties (String connectionName)
{
Hashtable connectionProperties = new Hashtable();
System.out.println("MySSLMqHandler: Setting connection properties for connection: " +
```

```
connectionName);
// Set userID and password variables
String userID = "MyUserID";
String password = "MyPassword";

 // Set userID and password variables, differently, based on connection name
 if (connectionName.equals("SomeConnection")) {
 userID = "MyOtherUserId";
 password = "MyOtherPassword";
 }
// Set userID and password properties in Hastable to be returned
 System.out.println("MySSLMqHandler:[" + connectionName + "] Setting userID: " + userID);
 connectionProperties.put("userID" /* CMQC.USER_ID_PROPERTY */ , userID);
System.out.println("MySSLMqHandler:[" + connectionName + "] Setting password: " +
password);
connectionProperties.put("password" /* CMQC.PASSWORD_PROPERTY */ , password);
// set SLL variables ( trust store, key store, cipher suite)
String trustStore = "MyTrustStore.jks";
String trustStorePassword = "MyTrustStorePassword";
String keystore = "MyKeyStore.jks";
String keystorePassword = "MyKeyStorePassword";
String transport = "MQSeries Client";
String cipherSuite = "TLS_RSA_WITH_AES_256_CBC_SHA";
// Set connection SSL properties
System.setProperty("javax.net.ssl.trustStore", trustStore);
System.setProperty("javax.net.ssl.trustStorePassword", trustStorePassword);
System.setProperty("javax.net.ssl.keyStore", keystore);
System.setProperty("javax.net.ssl.keyStorePassword", keystorePassword);
System.out.println("MySSLMqHandler:[" + connectionName + "] Setting trustStore: " +
trustStore);
connectionProperties.put("javax.net.ssl.trustStore", trustStore);
System.out.println("MySSLMqHandler:[" + connectionName + "] Setting trustStorePassword:
" + trustStorePassword);
connectionProperties.put("javax.net.ssl.trustStorePassword", trustStorePassword);
System.out.println("MySSLMqHandler:[" + connectionName + "] Setting keyStore: " +
keystore);
connectionProperties.put("javax.net.ssl.keyStore", keystore);
System.out.println("MySSLMqHandler:[" + connectionName + "] Setting keyStorePassword: "
+ keystorePassword);
connectionProperties.put("javax.net.ssl.keyStorePassword", keystorePassword);
System.out.println("MySSLMqHandler:[" + connectionName + "] Setting transport: " +
transport);
 connectionProperties.put("transport" /* CMQC.TRANSPORT_PROPERTY */ , transport /
*CMQC.TRANSPORT_MQSERIES_CLIENT) */) ;
 System.out.println("MySSLMqHandler:[" + connectionName + "] Setting SSL Cipher Suite:
" + cipherSuite);
 connectionProperties.put("SSL Cipher Suite" /* CMQC.SSL_CIPHER_SUITE_PROPERTY */ ,
cipherSuite);
// return populated connection properties
return connectionProperties;
}
}
```

Note: Users should replace the "My..." variable values as appropriate, or use some approved method to fetch these values, to avoid having them exposed in plain text. The Cipher suite given is an example. Supported cipher suites vary by MQ server version, JDK/JRE version, and JVM configuration.

This example uses the literal values ("userID", "password", etc) for properties for simplicity to avoid dependencies on the equivalent property constant values, in the relevant IBM MQ Java client libraries. The constant values could be used by adding the appropriate import statement and adding the relevant jar file to the classpath at build and run-time.

If using the Oracle JRE/ JVM you may need to use the following JVM flags to achieve the same run time behaviour as the IBM JRE/JVM:

```
-Dcom.ibm.mq.cfg.preferTLS=false
-Dcom.ibm.mq.cfg.useIBMCipherMappings=false
```

JMS Data Source

The Java Message Service (JMS) defines the standard for reliable enterprise messaging. Enterprise messaging, often referred to as Messaging Oriented Middleware (MOM), is universally recognized as an essential tool for building enterprise applications. By combining Java™ technology with enterprise messaging, the JMS API provides a powerful tool for solving enterprise computing problems.

With RTView, the JMS data source is designed to take advantage of this standard and therefore can be used with any JMS implementation. SL tests for compatibility against these commercial JMS offerings: IBM WebSphere® MQ, TIBCO Enterprise Message Service™, Progress SonicMQ® and Sun SeeBeyond™.

With RTView you can subscribe to JMS topics, browse JMS queues, or send JMS messages and then build sophisticated monitoring and management applications that allow you to:

- present real-time business content included in JMS messages;
- archive business metrics in the RTView Historian for trend analysis;
- provide a look at application health and troubleshooting by analyzing message content; and/or
- instigate interactive measures by sending JMS messages to applications.

This section includes:

JMS System Requirements and Setup

System Requirements

In addition to basic "System Requirements", the JMS data source is only available if you include your JMS provider's JMS jars in "RTV_USERPATH". See the **README_sysreq.txt** file in your installation's home directory for details.

Setup

In addition to general environment variables (see "Setup"), you must include the JMS jars for your provider in "RTV_USERPATH":

Name	Description	Example
RTV_USERPATH	Location of your JMS provider's JMS jars. **Note:** If **RTV_USERPATH** already exists, append JMS jars to it.	**TIBCO EMS:** **C:\TIBCO\ems\clients\java\tibjms.jar;C:\TIBCO\ems\clients\java\jms.jar** If you are using SSL, also include: **tibcrypt.jar** **slf4j-api-1.4.2.jar** **slf4j-simple-1.4.2.jar** which are located in the **EMS_HOME\lib** directory. **SonicMQ:** **C:\Sonic\MQ6.1\lib\sonic_Client.jar** **IBM WebSphere MQ Administration:** Include the following client jars: **jms.jar**, **com.ibm.mq.jar**, **com.ibm.mqjms.jar**, **connector.jar** and **dhbcore.jar**. These jars can be found in the **Java\lib** directory where you installed the IBM WebSphere MQ server. **BEA Weblogic:** **c:\BEA\server\weblogic92\lib\weblogic.jar** **Other:** See JMS provider documentation.

Attach to JMS Data

Note: The JMS data source may not be licensed in your RTView installation.

From the **Object Properties** window you can access the **Attach to JMS Data** dialog, which is used to register an RTView object as a listener for a JMS message topic or queue browser. In the **Attach to JMS Data** dialog, enter the name of a message topic or queue and specify which message field should be used to update an object property. RTView supports attaching to the following message types:

- MapMessage
- TextMessage
- BytesMessage - This message type requires a "JMS Handler".

- ObjectMessage - This message type requires a "JMS Handler".

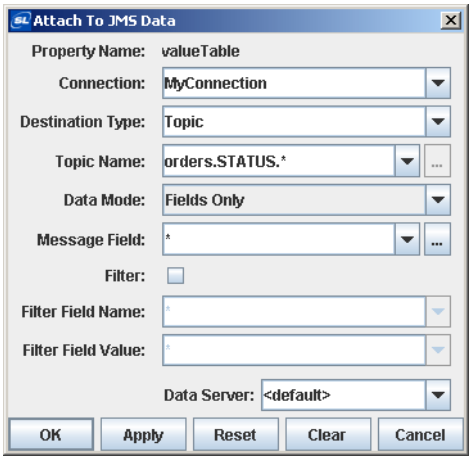
Once an object property has been attached to a message, it receives continuous updates. For topics, it updates whenever a new message is received. For queue browsing, the current contents of the queue are polled every update period, but not more often than once every six seconds. For example, if your update period is two seconds, the queue will only be polled once every third update period. It is also possible to set up a filter based on a particular field in the message.

Note: IBM WebSphere MQ does not allow you to browse queues, subscribe to topics or send messages on the same connection. Only one of these can be performed on each connection. Therefore, you have to define a separate connection for each one.

For example, a message with the topic orders.STATUS.London has two possible message fields: PERCENTCOMPLETED and PLANT. The first field, PERCENTCOMPLETED, tracks the status of an order. The second field, PLANT, indicates which London plant (L1 or L2) the message is describing.

Setting the Message Field to PERCENTCOMPLETED will set the attached object to indicate the real-time status of an order. Without a filter, the attached object would receive simultaneous updates from both London plants. To receive updates for a specific plant, you would set up a filter for the PLANT field and enter a Filter Field Value of either L1 or L2 to indicate which London plant will update the attached object.

In the **JMSALIAS.ini** file, you may create an alias for (top level or nested) messages, message fields and XML data embedded in messages (see "Create JMS Message Alias" for more information). Once **JMSALIAS.ini** is saved and RTView is restarted, you will be able to access these aliases from the Topic Name drop down menu:



To bring up the **Attach to JMS Data** dialog, right-click on the Property Name from the **Object Properties** window and select **Attach to Data>JMS**. The **Attach to JMS Data** dialog provides several drop down menus that allow you to specify information regarding a message. If the drop down menu does not contain the item you require, type your selection into the text field.

Attach to JMS Data Dialog

Field Name	Description
Connection	Connection name. You may define a connection on the "JMS Connections Tab" of the **Application Options** dialog.
Destination Type	Select **Topic** or **Queue Browser** from the drop down menu.
Topic Name	Only available if Topic is selected in Destination Type. For Topic Name, enter a specific topic name or use * as a wild card character. For example: **orders.STATUS.***. **Note:** IBM WebSphere MQ support for separators and wildcards differ from other vendors. See your IBM WebSphere MQ documentation for information on wildcard support. If the selected connection is a JNDI Connection, click on the ... button to bring up a tree browser of available topics. See "JMS JNDI Connections Tab" for more information.
Queue Name	Only available if Queue Browser is selected in Destination Type. For Queue Name, enter the name of the queue you wish to browse. For example: **myqueue.***. **Note:** IBM WebSphere MQ support for separators and wildcards differ from other vendors. See your IBM WebSphere MQ documentation for information on wildcard support.
Data Mode	Select **Fields Only**, **Fields and Properties**, or **Properties Only** from the drop down menu.
Message Field	Message field chosen to update the attached object. When the Data Mode is set to Fields and Properties or Properties Only, the following standard JMS Message Header properties are available: JMSCorrelationID, JMSMessageID, JMSType, JMSTimestamp, JMSExpiration, JMSDeliveryMode, JMSPriority, JMSRedelivered and JMSReplyTo.
Filter	Check box to indicate whether or not to filter the message.
Filter Field Name	Name of the message field to use as a filter.

Filter Field Value	Value that the filter field must equal.
Data Server	Select to read data through your configured Data Server and not directly from the JMS data source. **Default** - Select the default Data Server you configured in "Data Server Tab". **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". **Multi-Server Attachment** - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more Named Data Servers (e.g. **ds101;ds102**). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default;ds101;ds102**). **Note:** The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of * can be entered to specify all data servers, including **__default** and **__none**. When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received. A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways: **1.** The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows. **2.** The multi-server attachment can be applied as the **Table** argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

Drop down menus populate based on message topics added from the **Application Options** dialog or those typed directly into the **Attach to JMS Data** dialog. Message topics will not be added to drop down menus until at least one JMS message with that topic has been received by RTView.

When an object property has been attached to data, the **Property Name** and **Value** in the **Object Properties** window will be displayed in green. This indicates that editing values from the **Object Properties** window is no longer possible. To remove the data attachment and resume editing capabilities in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information is validated against the message topic added from Application Options dialog, or those typed directly into the Attach to JMS Data dialog. Message topics will not be validated until at least one JMS message with that topic has been received by RTView.

Note: Some topics using wild card characters are not validated at this time.

The following describes the significance of the validation colors:

Blue	Unknown	Entry is not recognized. When a Connection or Topic is unknown, the Message Field, Filter Field Name, and Filter Field Value are also unknown.
White	Valid state	Entry is valid.
Red	Invalid state	Entry is not valid.

Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. A generic Message Topic such as **$topic** is used instead of a specific message. Later when the display is running, this generic value is defined by the actual name of a specific message, such as **orders.STATUS.London**. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

Special Values

The following special values can be entered for message and filter fields:

Message Field	*	When * is entered as a message field, values from all message fields will be used to update the object property. This is only allowed for objects which display tabular data. **Note:** IBM WebSphere MQ support for separators and wildcards differ from other vendors. See your IBM WebSphere MQ documentation for information on wildcard support.
Filter Field Value	*	When * is entered as a filter field value, data for all values in the specified filter field will be used to update the object property. When "*" is entered, only the literal comparative value will be used. These are only allowed for objects which display tabular data.

Select Message Fields (Tables Only)

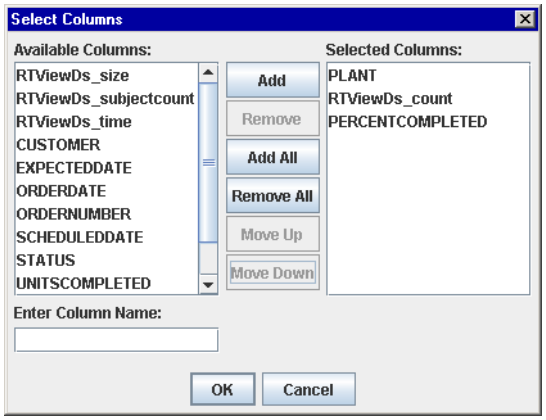
From the Attach to Data dialog you can specify which message fields to display as columns in a table and in what order they will appear. In order to populate the listing of available message fields, you must first select a valid topic.

To bring up the Select Columns dialog, click on the ... button in the Message Field field (or right-click in the Message Field field and click **Select Columns**). The dialog should contain a list of Available Columns that you can add to your table.

To add a field, select an item from the **Available Columns** list and click on the **Add** button. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of fields in a table by arranging the items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

Validation colors indicate whether selected message fields are valid. However, if even one field selected is invalid the **Message Field** field in the **Attach to JMS Data** dialog will register as an invalid entry.

If no data is available for a table row within a selected column, the table cell will display one of the following values: N/A, false, 0, or 0.0.



The following describes **Attach to JMS Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Create JMS Message Alias

Note: The JMS data source may not be licensed in your RTView installation.

In the **JMSALIAS.ini** file, you can create aliases for (top level or nested) messages, message fields and XML data embedded in JMS messages. Once **JMSALIAS.ini** is saved and RTView is restarted, you will be able to access these aliases from the **Message Topic** drop down menu in the "Attach to JMS Data" dialog.

It is possible to specify a directory for your initialization files. If no directory has been specified for initialization files and **JMSALIAS.ini** is not found in the directory where you started the application, then RTView will search under lib in your installation directory. See "RTV_JAVAOPTS" for more information.

To define an alias in JMSALIAS.ini use the following syntax:

aliasName connection topic partialUpdatesOK fieldNameList

aliasName	Create a name to use when selecting this alias from the "Attach to JMS Data" dialog. **Note:** Alias names cannot contain spaces.
connection	The connection to use when listening for this topic. If the specified connection is not defined, the alias will not be added. If the connection name contains a space, it must be enclosed in single quotes.
topic	JMS message topic. It is possible to use * as a wild card character (e.g.: **orders.STATUS.*** or *.*.*). **Note:** If a message topic contains a space or a colon, then the topic must be enclosed in single quotes.
partialUpdatesOK	This keyword is optional. If present, new row data will be merged into existing row data when a new message comes in with the same topic. For example, if a message comes in which contains fields **FieldA**, **FieldB**, and **FieldC**, then another message with the same topic comes in which contains fields **FieldC**, **FieldD**, and **FieldE**, the new row will contain fields **FieldA** and **FieldB** from the previous message and **FieldC**, **FieldD**, and **FieldE** from the current message.
fieldNameList	Message field names listed in hierarchical order separated by a :: (e.g., **fieldName1::fieldName2::fieldName3 ...** etc.) ending in the name of the field that contains the data to display. If no **fieldNameList** is specified, top level message fields will be displayed. If a message field in this list contains XML data, then add the **$xml:** prefix to the field name that contains the XML data. (e.g., **fieldName1::$xml:fieldName2::fieldName3** ... etc.) The name of the message field containing XML data may be followed by a list of XML tags in hierarchical order that ends in an XML tag with the specific data you would like to display. If the last **fieldName** listed contains more than one level of XML tags, then all subsequent tags will be converted to column names using the _ symbol. **Note:** If a field name contains a space or a colon, then the entire **fieldNameList** must be enclosed in single quotes.

Nested MapMessage Field Aliases

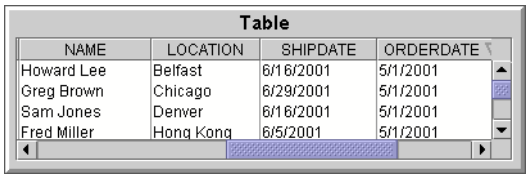
In the alias defined below named allOrders, a message with the topic **orders.STATUS.*** contains the **customer_info** MapMessage field, which is nested within the **sales** MapMessage field.

Note: Not all JMS providers support nested MapMessages.

---

allOrders myConnection orders.STATUS.* sales::customer_info

A table attached to **allOrders** will contain a column for each field in the **customer_info** message field.



It is also possible to create an alias for a specific field in the **customer_info** message field.

allNames myConnection orders.STATUS.* sales::customer_info::NAME

A table attached to **allNames** would contain one column for the **NAME** field.

Embedded XML Data Aliases

To display XML data embedded within a JMS message, add the **$xml:** prefix to the field name that contains the XML data. In the alias defined below named **OrderInfo**, the **OrderData** message field (containing XML data) is nested within the **Production** MapMessage field:

OrderInfo myConnection orders.STATUS.* Production::$xml:OrderData

It is also possible to display XML from a TextMessage:

OrderInfo myConnection orders.STATUS.* $xml:<TEXT>

There are several ways that the following XML values and attributes (contained in the message field named **OrderData**) can be defined in an alias.

```
<Orders>
        <Order Date="March 1, 2004" Time="12:00:00">
                <OID>12345</OID>
                <Customer>
                        <Name>John Smith</Name>
                        <CID>6789</CID>
                </Customer>
        </Order>
       <Order Date="March 1, 2004" Time="12:00:00">
                <OID>67891</OID>
                <Customer>
                        <Name>Alice Chen</Name>
                        <CID>1001</CID>
                </Customer>
        </Order>
</Orders>
```

Displaying XML Values

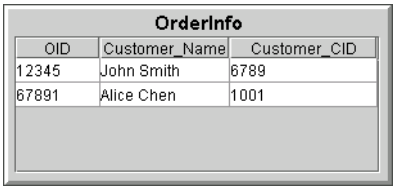
To display all XML values contained within **Order** tags of the **OrderData** MapMessage field, you would use the following alias.

Note: Hierarchical XML tags are converted to column names using the _ symbol (e.g.: **Customer_Name**, **Customer_CID**, etc.).

OrderInfo myConnection orders.STATUS.* $xml:OrderData::Orders::Order

To display the same data from a TextMessage:

$xml:<TEXT>::Orders::Order

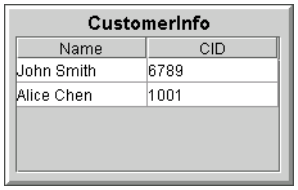


You can display specific information contained within **Customer** tags of the **OrderData** message field using the following alias:

**CustomerInfo myConnection orders.STATUS.*
$xml:OrderData::Orders::Order::Customer**

To display the same data from a TextMessage:

$xml:<TEXT>::Orders::Order::Customer



Displaying XML Values and Attributes

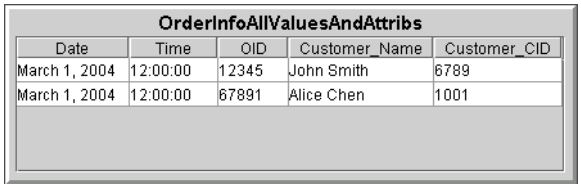
To display all XML values and attributes contained within **Order** tags of the **OrderData** message field, you would use the following alias.

Note: Hierarchical XML tags are converted to column names using the _ symbol (e.g.: **Customer_Name**, **Customer_CID**, etc.).

**OrderInfoAllValuesAndAttribs myConnection orders.STATUS.*
$xml:OrderData::Orders::Order:$attrib=**

To display the same data from a TextMessage:

$xml:<TEXT>::Orders:Order:$attrib=



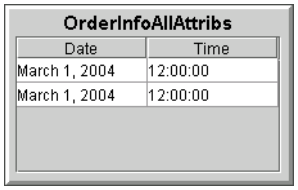
Displaying XML Attributes Only

You can display all of the XML attributes contained within **Order** tags of the **OrderData** message field using the following alias:

**OrderInfoAllAttribs myConnection orders.STATUS.*
$xml:OrderData::Orders::Order:$attrib=***

To display the same data from a TextMessage:

$xml:<TEXT>::Orders::Order:$attrib=*



It is also possible to display only a particular attribute (**Date**) contained within the **Order** tags.

OrderInfoDate myConnection orders.STATUS.*
$xml:OrderData::Orders::Order:$attrib=Date

To display the same data from a TextMessage:

$xml:<TEXT>::Orders::Order:$attrib=Date

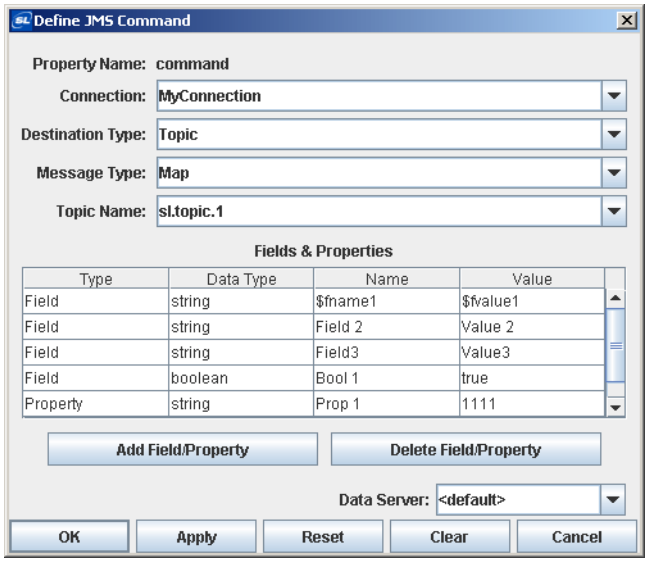
Define JMS Command Window

Note: The JMS data source may not be licensed in your RTView installation.

You can access the **Define JMS Command** window from the **Object Properties** window. The **Define JMS Command** window is used to assign a JMS message to an object's command property, giving you the ability to send messages from within an RTView display. If you execute a JMS command from a Thin Client with Direct Data Connection or any Served Data deployment, the command will execute on the server.

Note: IBM WebSphere MQ does not let you send a command on a connection that you receive messages on. Therefore, you need to define separate connections for executing commands and attaching to data.

To open the **Define JMS Command** window, right-click on the appropriate command property in the **Object Properties** window and select **Define Command>JMS**. The information supplied assigns a message to the command property. See the "Define/Execute Command" section for information on how to execute a command.



Field Name	Description
Connection	Enter a connection name. You may define a connection on the "JMS Connections Tab" of the **Application Options** dialog.
Destination Type	Topic is currently the only supported destination type. To attach the Destination Type to data, right-click and choose **Attach to Data** or double-click in the field.
Message Type	Select **Map** to send a Map Message or **Text** to send a Text Message. To attach the Message Type to data, right-click and choose **Attach to Data** or double-click in the field.
Topic Name	Enter the name of the topic to send. To attach the Topic Name to data, right-click and choose **Attach to Data** or double-click in the field.
Fields & Properties Type	Select **Field** to define a message field or **Property** to define a message property. **Data Type** - Select the data type for this field or property. **Name** - Specify the name for this field or property. **Value** - Specify the value for this field or property. To attach the Value to data, right-click and choose **Attach to Data** or double-click in the field.
Add Field / Property	Add a field or property.

Delete Field / Property	Delete the selected field or property.
Data Server	Select to read data through your configured Data Server and not directly from the JMS data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab". **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a Named Data Server that you configured in **Application Options>**"Data Server Tab". **Multi-Server Command** - When multiple data servers are specified, the command will be executed on each data server in the list. To configure multiple data servers, enter a semicolon (;) delimited list containing two or more **Named Data Servers** (e.g. **ds101;ds102**). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and __none (e.g. **__default;ds101;ds102**). **Note:** The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of * can be entered to specify all data servers, including **__default** and **__none**.

Drop down menus populate based on message topics added from the **Application Options** window or those typed directly into the **Attach to Data** window. Message topics will not be added to drop down menus until at least one JMS message with that topic has been received by RTView.

Substitutions

The Substitutions feature allows you to build open-ended displays in which commands depend on values defined at the time the display is run. A generic topic such as **$topic** is used instead of a specific topic. Later when the display is running, this generic value is defined by the actual name of a specific topic, such as **orders.STATUS.London**. In this way, a single display can be reused to send a number of different messages. Substitutions can be used in any field in this dialog. For more information on creating displays using substitution values, see "Substitutions".

Special Values

$value	When an **actionCommand** is executed, **$value** is replaced with the value from the control. This value may be used in any field in the **Define JMS Command** dialog. **Note:** This value may only be used for Action "Commands".

The following describes **Define JMS Command** window commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.

Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

JMS Data Source Substitutions

In addition to standard built-in substitutions, this data source also sets the following drill down substitutions:

Substitution Value	Definition
$conn	The connection name from the selected row or object.
$topic	Message topic from the selected row or object.
$filterfield	Filter field name from the selected row or object.
$filtervalue	Filter field value from the selected row or object.

See "Drill Down Substitutions" for more information.

Application Options -- JMS

To access the **Application Options** dialog, in the Display Builder select **Tools>Options**.

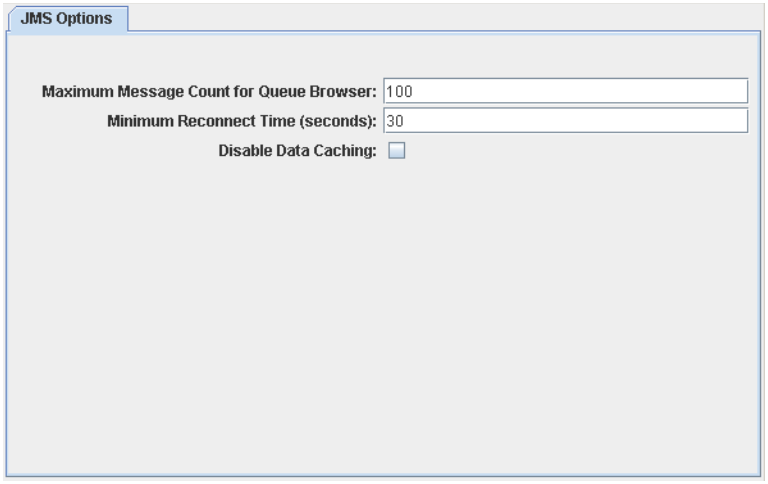
Options specified in JMS tabs can be saved in an initialization file (**JMSOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server and Historian to set initial values. If no directory has been specified for your initialization files and **JMSOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information

---

Note: Options specified using command line arguments will override values set in initialization files. See "Weather Data" for more information.

There are three Application Options tabs for JMS: "JMS Options Tab", "JMS Messages Tab", and the "JMS JNDI Connections Tab".

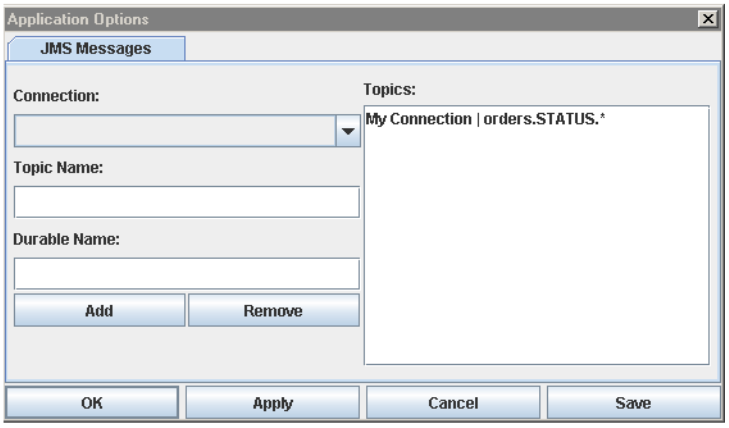
JMS Options Tab



Field Name	Description
Maximum Message Count for Queue Browser	Enter the maximum number of messages that will be queried from a queue browser for any queue. Default is 100.
Minimum Reconnect Time (seconds)	Enter the minimum number of seconds that will elapse before attempting to reconnect to the server. Default is 30.
Disable Data Caching	Select to disable caching in the JMS data source so that listeners are only updated with new rows instead of a combination of all rows for a topic. **Note:** This option should be utilized when using JMS as the input to a cache file so that duplicate rows are not sent to the cache.

JMS Messages Tab

Message topics listed on the **JMS Messages** tab are used to populate drop down menus in the "Attach to JMS Data" dialog. RTView starts listening for new topics added to the Topics list after you click **OK**, **Apply**, or **Save**.



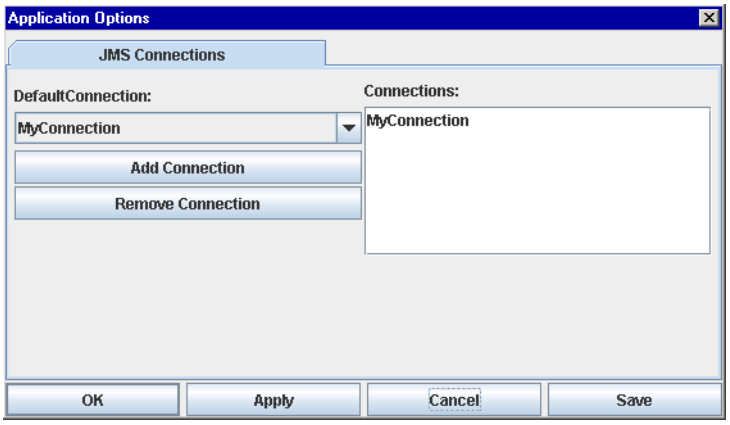
Field Name	Description
Connection	Enter the name of the connection to use when listening for this topic.
Topic Name	Enter the name of a JMS Message Topic.
Durable Name	Enter the name of the durable. If specified, this will be used to create a durable subscription to this topic.
Remove	Select from Topics and click **Remove** to delete.

JMS Connections Tab

This tab allows you to add or remove connections and set your default connection.

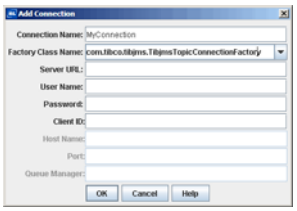
When you add a JMS connection to the list it will be highlighted in yellow indicating that RTView has not connected to it. To attempt to connect to a JMS connection, click **OK**, **Apply**, or **Save**. If the background remains yellow, then RTView was unable make a connection. Check that your JMS connection was setup correctly and that the JMS message server is running.

Note: Regardless of which tab you are currently working from in the **Application Options** dialog, RTView will attempt to connect to all unconnected connections each time you click **OK**, **Apply**, or **Save**.



Note: Additional setup is required to connect to your JMS Server using SSL. This is only supported for JMS connections to TIBCO EMS. See "JMS SSL Parameters" for more information.

Field Name	Description
Default Connection	Name of connection used as the default for data attachments. Select from drop down menu to change default setting.
Add Connection	Click to open the **Add Connection** dialog. To edit, select a connection from the list and double-click. Connections that are updating objects in a current display or that are listening for topics defined on the "JMS Messages Tab" cannot be renamed.



Connection Name -- Name to use when referencing this JMS connection in your data attachments.

Factory Class Name -- The fully qualified name of the topic connection factory class to use when creating this connection. The path to this class must be included in the **RTV_USERPATH** environment variable. Refer to your JMS provider documentation if you do not know the name of this class.

To use IBM Websphere MQ as a JMS Connection, select either a Topic connection which can be used to subscribe to topics, or a Queue connection which can be used to browse queues.

For a Topic connection select:

com.ibm.mq.jms.MQTopicConnectionFactory

For a Queue connection select:

com.ibm.mq.jms.MQQueueConnectionFactory

Then specify the Host Name, Port and Queue Manager used by IBM Websphere MQ.

Server URL -- Complete URL for your JMS message server. This parameter is optional.

User Name -- User name to use when creating this connection. This parameter is optional.

Password -- Password to use when creating this connection. This parameter is optional.

If you need to provide an encrypted password (rather than expose server password names in a clear text file, use the **encode_string** command line option with the following syntax:

encode_string type mypassword

where **type** is the key for the data source and **mypassword** is your plain text password.

Note: The **type** argument is only required when you encrypt a string for a data source.

For example, enter the following in an initialized command window:

encode_string jms mypassword

and you will receive an encrypted password:

encrypted value:
0134301355013460133101349013530134501348013 34

Copy the encrypted value, paste it into the password field, and click **Save** to save this value to the initialization (***.ini**) file. Or, if necessary, manually edit the (***.ini**) file to include the encrypted value.

Note: If you need to manually edit a configuration (***.ini**) file, contact SL Technical Support at support@sl.com for information about supported syntax.

Client ID --Client identifier to set on this connection. This parameter is optional.

Host Name -- For IBM WebSphere MQ only. Name of the host to use for this connection.

Port -- For IBM WebSphere MQ only. Name of the port to use for this connection.

Queue Manager -- For IBM WebSphere MQ only. Name of the queue manager to use for this connection.

Remove Connection — Select a connection from the list and click **Remove Connection** to delete. Connections that are updating objects in a current display or that are listening for topics defined on the "JMS Messages Tab" cannot be removed.

Note: IBM WebSphere MQ does not let you send a command on a connection that you receive messages on. Therefore, you need to define separate connections for executing commands and attaching to data.

JMS SSL Parameters

This section assumes you have a working knowledge of writing, compiling and deploying Java classes.

Note: This is only supported for JMS connections to TIBCO EMS.

To use SSL with JMS, you will need to create a Java class named **MyJmsDsSSLHandler** that extends the **com.sl.gmsjjmsds.GmsRtViewJmsDsSSLHandler** class.

In **MyJmsDsSSLHandler**, define the following method:

```
public Map getSSLParams (String serverUrl)
```

This method will get called to retrieve the list of SSL parameters to pass in when RTView creates each JMS Connection. See the TIBCO EMS documentation for information on creating a map of SSL parameters suitable to pass into the JMS Connection.

Add **gmsjjmsds.jar**, located in the lib directory (found in your installation directory), to your classpath when you compile **MyJmsDsSSLHandler**. The compiled **MyJmsDsSSLHandler** class must be included in the RTView classpath by adding it to the definition for the **RTV_USERPATH** environment variable.

The following is an example of **MyJmsDsSSLHandler**:

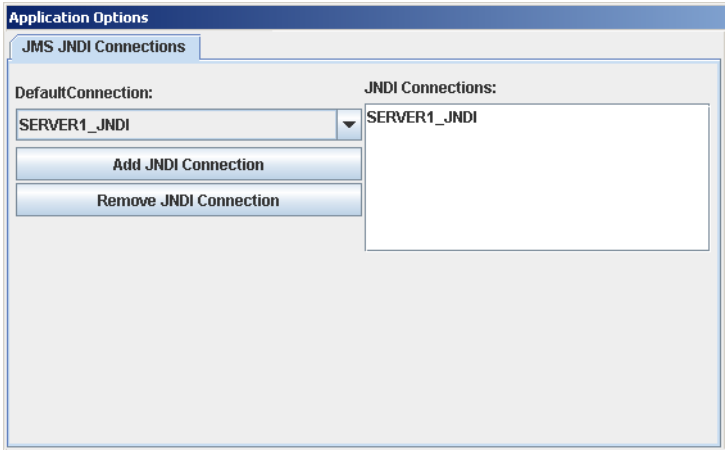
```
import java.util.*;
import com.tibco.tibjms.TibjmsSSL;
import com.sl.gmsjjmsds.GmsRtViewJmsDsSSLHandler;

public class MyJmsDsSSLHandler extends GmsRtViewJmsDsSSLHandler
{

public Map getSSLParams (String serverUrl)
{
        System.out.println("==> MyJmsDsSSLHandler.getSSLParams");
        return null;
}

}
```

JMS JNDI Connections Tab

This tab allows you to add or remove JNDI connections and set your default connection. JNDI connections are only supported for BEA Weblogic 9.2+. Other JNDI JMS vendors may work, but are not guaranteed.

When you add a JNDI connection to the list it will be highlighted in yellow indicating that RTView has not connected to it. To attempt to connect to a JNDI connection, click **OK**, **Apply**, or **Save**. If the background remains yellow, then RTView was unable make a connection. Check that your connection was setup correctly and that the JMS message server is running.

Note: Regardless of which tab you are currently working from in the Application Options dialog, RTView will attempt to connect to all unconnected connections each time you click **OK**, **Apply**, or **Save**.



Field Name	Description
Default Connection	Name of connection used as the default for data attachments. Select from drop down menu to change default setting.
Add JNDI Connection	Click to open the Add JNDI Connection dialog. To edit, select a connection from the list and double-click. Connections that are updating objects in a current display or that are listening for topics defined on the "JMS Messages Tab" cannot be renamed.

Connection Name -- Name to use when referencing this JNDI connection in your data attachments.

Initial Context Factory -- The fully qualified name of the Initial Context Factory class to use when creating this JNDI connection.

Note: The path to this class must be included in the "RTV_USERPATH" environment variable.

Provider URL -- Complete URL for your JNDI server.

Connection Factory -- JNDI name of the JMS connection factory to use when creating this connection

Security Principal -- Security principal to use when creating this connection. This parameter is optional.

Security Credentials -- Security credentials to use when creating this connection. This parameter is optional.

	Additional Properties -- This parameter is optional. **Property Name** -- Specify property name to add to the list. **Property Value** -- Specify property value to add to the list.
Remove Connection	Select a connection from the list and click Remove Connection to delete. Connections that are updating objects in a current display or that are listening for topics defined on the "JMS Messages Tab" cannot be removed.

JMS Handler

The JMS data source supports a custom JMS handler to parse BytesMessage and ObjectMessage content. To get the content from BytesMessage or ObjectMessage, create a subclass of **com.sl.gmsjjmsds.GmsJRtViewCustomJmsHandler** and add it to RTV_USERPATH. By default, RTView looks for a class named **MyJmsHandler**. To specify an alternate name, use the **-jms_customhandlername:className** command line option. Include **gmsjrtview.jar** and **gmsjjmsds.jar** (in the **lib** directory of your RTView installation) and the **jms.jar** from your JMS vendor in the classpath when building this subclass.

The **GmsJRtViewCustomCommandHandler** has three methods you may override in your subclass:

- **public void initialize()** - This method is called once, after the class is instanced, in case the subclass needs to perform an initialization.
- **public String getBytesMessageStringContent (BytesMessage bytesMsg)** - This method is called every time a BytesMessage is received. Subclasses should override this method to return a String representation of the byte[] from the message body.
- **public GmsTabularData getObjectMessageTableContent (ObjectMessage objectMsg)** - This method is called every time an ObjectMessage is received. Subclasses should override this method to return a GmsTabularData representation of the Object in the message body.

For example:

```
import com.sl.gmsjjmsds.*;
import javax.jms.*;
import javax.jms.*;
import com.sl.gmsjrt.*;
public class MyJmsHandler extends GmsRtViewCustomJmsHandler
{
public void initialize ()
{
}
public String getBytesMessageStringContent (BytesMessage bytesMsg)
{
try {
byte[] bytes = new byte[(int)bytesMsg.getBodyLength()];
bytesMsg.readBytes(bytes);
return new String(bytes);
} catch (Exception e) {
System.out.println("ERROR: Can't parse BytesMessage &lt; " +
bytesMsg + "&gt;. Caught exception: " + e);
}
return null;
}
public GmsTabularData getObjectMessageTableContent (ObjectMessage objMsg)
```

```
{
try {
Object obj = objMsg.getObject();
if (obj == null)
return null;
if (obj instanceof Vector)
return processVector((Vector)obj);
else
return processString(String.valueOf(obj));
} catch (Exception e) {
System.out.println("ERROR: Can't parse Object &lt; " +
objMsg + "&gt;. Caught exception: " + e);
}
return null;
}
private GmsTabularData processVector (Vector&lt;String[]&gt; v)
{
GmsTabularData data = new GmsTabularData();
data.addColumn("Plant", GmsTabularData.STRING);
data.addColumn("Status", GmsTabularData.STRING);
data.addColumn("Units Completed", GmsTabularData.STRING);
String[] plants = v.elementAt(0);
String[] statuses = v.elementAt(1);
String[] units = v.elementAt(2);
if (plants == null || statuses == null || units == null)
return data;
for (int i = 0; i &lt; plants.length; i++) {
data.addRow("");
data.setCellValue(plants[i], i, 0);
data.setCellValue(statuses[i], i, 1);
data.setCellValue(units[i], i, 2);
}
return data;
}
private GmsTabularData processString (String s)
{
if (s == null)
s = "";
GmsTabularData data = new GmsTabularData();
data.addColumn("Value", GmsTabularData.STRING);
data.addRow("");
data.setCellValue(s, 0, 0);
return data;
}
}
```

- XML data embedded in both BytesMessage content and ObjectMessage content can be used to define a JMS alias. To show all XML content of a bytes message add the following to your **JMSALIAS.ini** file:

  OrderInfo myConnection orders.STATUS.* $xml:<TEXT>

- To show the data from a specific XML tag in the bytes message content, add something similar to the following:

  $xml:<TEXT>::Orders::Order

  Where **Orders::Order** specifies the path to the XML element to display.

- Since the ObjectMessage callback returns a GmsTabularData, use the column name of the field containing the XML data:

OrderInfo myConnection orders.STATUS.* Production::$xml:OrderData

Where **OrderData** is the name of the column containing the XML data. Note that only the value from the cell in the first row is used.

- To show the data from a specific XML tag of an **ObjectMessage** table, add something similar to the following:

  CustomerInfo myConnection orders.STATUS.* $xml:OrderData::Orders::Order::Customer

  Where **Orders::Order::Customer** defines the path to the XML element to display.

RTView Deployment - JMS

This section contains details about the deployment process that are specific to your data source. Please go to the Deployment section of this documentation for instructions on how to implement your RTView deployment option. Return to this page whenever you are instructed to refer to deployment information that is specific to your data source.

System Requirements and Setup

The JMS data source has additional "JMS System Requirements and Setup".

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options", "Application Options -- JMS", and "RTV_JAVAOPTS" for more information.

Include the following initialization files when you deploy RTView with this data source:

File Name	Description
JMSOPTIONS.ini	Contains data source options for JMS.
JMSALIAS.ini	Contains JMS alias definitions. See "Create JMS Message Alias" for more information. **Note:** This file is optional.

Note: Options specified using command line override values set in initialization files.

Setup Client

No additional client setup is required for this data source.

JMS Demos

Note: Demos that utilize the JMS data source will not work with IBM WebSphere MQ.

Except where noted, all demos can be run in three ways, as an application, or via rich or thin client in a browser.

Before You Begin

Start the Demo Server

Thin Client Demo only.

There is a thin client demo already installed on the "RTView Demo Server".

In an initialized command window (see "Initializing a Command Prompt or Terminal Window"):

Type **run_startup_demoserver**

Data Source Demo

The Data Source Demo is designed to illustrate each data source.

1. Start the Simulators

   Start the simulators for each data source you will be using. To run the "JMS Data Simulator":

   In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory and type:

   run_jmssimdata

2. **Run Demos - Application or Thin Client Browser**

   Application Demo

   1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
   2. To view the demo, type:

      run_viewer

   3. To edit the demo, type:

      run_builder

   Thin Client Browser Demo

   "Start the Demo Server" if it is not running.

   1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
   2. Start the Display Server by typing:

      run_displayserver

   3. Open a browser and navigate to **http://localhost:8068/dstutorial**.

Quick Start Tutorial: JMS

This Quick Start Tutorial provides you with the fundamentals on how to use RTView with a JMS data source. Once completed, you can swiftly apply this knowledge to building your own real-time dashboard displays for visual access to your JMS data.

Learn to:

- Animate graphic objects with JMS data
- Create a drill down display with JMS data

Note: The JMS data source may not be licensed in your RTView installation.

Get Started

This tutorial requires the following:

- Register for a license key. If you have not, you must do so before continuing. See "Registration" for more information.
- "Quick Start Tutorial" - This tutorial requires that you have a working knowledge of RTView. We recommend that you complete the Quick Start Tutorial before continuing.
- Setup your "RTV_USERPATH" to include your JMS provider's JMS jars.
- Your JMS message server must be running.

Note: This tutorial will not work with IBM WebSphere MQ, which does not support the way wildcard characters are used in topic names in this example.

Start the JMS Simulator

In this exercise you start the "JMS Data Simulator" which is the data source used in this tutorial. The data simulator sends JMS messages that are used to animate objects in your display

1. In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"), type:

   On Windows: **start run_jmssimdata**

   On UNIX: **run_jmssimdata &**

   The JMS simulator runs as a background process and is ready when dots appear across the screen.

Note: Note: You must follow this initialization process for each new terminal window you open. See the "Setup" section for more details about setting up your environment.

Start the Display Builder

If you are already logged onto the Display Builder, skip this section and go to the **Create a Display** section, below.

1. In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"):

   type **run_builder**

2. Login to the Display Builder. By default, the Display Builder does not require a login. Login can be enabled at setup to support "Role-based Security". The default user name and password are:

   User Name: **admin**

   Password: **admin**

   Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role.

   You are now ready to create a display using the data source.

Create a Display

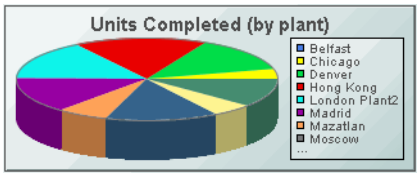
At this point you have:

- Registered for a license key. See "Registration" for more information.
- Logged on to the Display Builder
- Setup your "RTV_USERPATH" to include your JMS provider's JMS jars
- Started your JMS message server
- Completed the "Quick Start Tutorial"

In this tutorial you use the JMS simulator as your data source to create an animated pie graph that displays, as seen below.

As you saw in the Quick Start Tutorial, the data structure of tables and graphs (tabular data) enables RTView to automatically create several data source specific, built-in Substitutions for you. You will see these built-in Substitutions used in the target display when you create the drill down. For more information on Substitutions, see Substitutions.

In this exercise, you create a drill down using the previously created display, jms_dd_qs.rtv, as the target display. First you will set the pie graph to display units completed per plant. Then you will create a drill down that will open a bar graph that shows more detailed data for each plant.



Add a JMS Connection to List of Available Connections

Adding the JMS connection to the list of available connections makes it available for animating graphic objects in your display.

1. In the Display Builder, select **Tools>Options** to open the **Application Options** dialog. See "Application Options -- JMS" for more information.

2. Select the **JMS Connections** tab and click on **Add Connection** to open the Add Connection dialog.

3. In the Add Connection dialog:

Connection Name - **Enter My Connection**

Factory Class Name - If you are a TIBCO EMS user, select **com.tibco.tibjms.TibjmsTopicConnectionFactory** from the drop down menu. If not, enter the name of your JMS provider's connection factory class. The path to this class must be included in the "RTV_USERPATH" environment variable. Refer to your JMS provider documentation if you do not know the name of this class.

Server URL - The complete URL for your JMS message server if it is not running on localhost.

User Name - The user name to use when creating this connection. This parameter is optional.

Password - The password to use when creating this connection. This parameter is optional.

If you need to provide an encrypted password (rather than expose server password names in a clear text file, use the **encode_string** command line option with the following syntax:

encode_string type mypassword

where **type** is the key for the data source and **mypassword** is your plain text password.

Note: The **type** argument is only required when you encrypt a string for a data source.

For example, enter the following in an initialized command window (see "Initializing a Command Prompt or Terminal Window"):

encode_string jms mypassword

and you will receive an encrypted password:

encrypted value: 013430135501346013310134901353013450134801334

Copy the encrypted value, paste it into the password field and click Save to save this value to the initialization (***.ini**) file. Or, if necessary, manually edit the (***.ini**) file to include the encrypted value.

Note: If you need to manually edit a configuration (***.ini**) file, contact SL Technical Support at support@sl.com for information about supported syntax.

Client ID - The client identifier to set on this connection. This parameter is optional.

4. Click **Apply** and **OK**.

5. Select the **JMS Messages** tab:

   Connection - Select **My Connection**.

   Topic Name - Enter **orders.STATUS.***

6. Click **Add**.

   All of the messages contained in **My Connection** with topic names that begin with **orders.STATUS.** are generated by the JMS data simulator.

7. Click **OK** to apply and close the Application Options dialog.

   The connection and message topics are now available for animating graphic objects in your display.

Display Data in a Pie Graph

In this exercise you add a pie graph and then display the message topic in the pie graph by attaching it to the data source.

1. Click the button and click again in the Working Area to place the pie graph.
2. In the **Object Properties** dialog:

   label (category: Label) - Change the name of the label to **Units Completed (by Plant)**.

   legendWidthPercent (category: Legend) - Increase to **50** to widen the legend.

   valueTable (category: Data) - Right-click on the ... button and select **Attach to Data>JMS**.
3. In the **Attach to JMS Data** dialog:

   Connection - Select **My Connection**.

   Destination Type - Topic should already be selected.

   Topic Name - Enter **orders.STATUS.***

   Message Field - Select the ... button to open the **Select Columns** dialog.
4. In the **Select Columns** dialog:

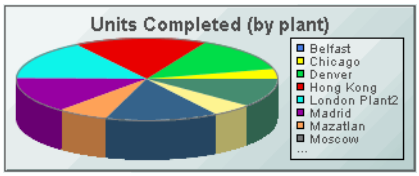
   Select **PLANT** in the Available Columns list and click **Add**.

   Select **UNITSCOMPLETED** in the Available Columns list and click **Add**.
5. Click **OK** to close the **Select Columns** dialog.
6. In the **Attach to JMS Data** dialog:

   Filter - Click to select the check box.

   Filter Field Name - Select **PLANT**.

   Filter Field Value - * should already be selected.
7. Click **OK** to apply these values and close the Attach to JMS Data dialog.



The pie graph is now animated by the JMS data. Since the values in the Units Completed column are numeric, this data is graphed in the pie. Since the values in the Plant column contain text, they are shown in the legend.

You are now ready to create the drill down.

Create a Drill Down Target in the Pie Graph

In this exercise, you create a drill down in the pie graph using the previously created display, jms_dd_qs.rtv, as the target.

1. In the Object Properties window:

   drillDownTarget (category: Interaction) - Double-click in the Property Name field to bring up the "Drill Down Properties" dialog.

2. In the "Drill Down Properties" dialog:

   Apply Drill Down To - Select **Named Window** from the drop down menu. This option lets you re-use the window when you drill down multiple times.

   Window Name - Enter **jms**. This name should be unique unless the display is to open in an existing window.

   Drill Down Display Name - Select **dstutorial\jms_dd_qs.rtv** from the drop down menu.

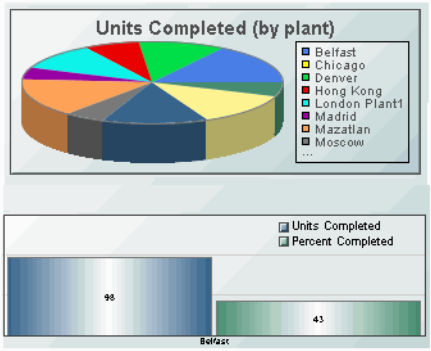
3. Click **OK** to attach the drill down target and close the Drill Down Properties dialog.

View the Drill Down Display

In this exercise, you drill down to the target display.

1. Double-click on any wedge in the pie graph to drill down to detailed data. The target display opens.

   Note: You must select the top of a wedge for the drill down to open.



2. Double-click on another wedge in the pie and the same display is used to show different data based on the wedge you select.

3. Close the drill down window.

4. In the Display Builder select **File>Save**.

JMS Data Simulator

A JMS data simulator allows customers to work with RTView without setting up their own messages. The simulator creates, updates, and sends out messages with the following topics:

MapMessages:

- orders.STATUS.Belfast
- orders.STATUS.Chicago
- orders.STATUS.Denver
- orders.STATUS.Hong Kong
- orders.STATUS.London
- orders.STATUS.Madrid
- orders.STATUS.Mazatlan
- orders.STATUS.Moscow
- orders.STATUS.Palo Alto
- orders.STATUS.Seattle
- orders.STATUS.Tokyo
- WEATHER.REGIONAL.WEST
- WEATHER.REGIONAL.EAST

TextMessages:

- orderStatus.simple
- WEATHER.XML.WEST
- WEATHER.XML.EAST

Note: IBM WebSphere MQ does not support spaces in topic names. When the simulator is run with the IBM WebSphere MQ connection factory, spaces in topic names will be replaced with underbar (_) symbols.

Running the Simulator

From an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"), type:

run_jmssimdata

Your JMS server must be running before you start the JMS data simulator.

Note: You must run the JMS data simulator from a command prompt or terminal window to use command line parameters.

Command line parameters for the simulator include:

Name	Description
-u (milliseconds)	Set update rate in milliseconds. Default is **2000**.
-jmsclient:	The client identifier for the message server connection. For IBM WebSphere MQ, this must be set to null.

-jmsfactory:	The fully qualified name of the topic connection factory class to use when creating this connection. The path to this class must be included in the "RTV_USERPATH" environment variable. Refer to your JMS provider documentation if you do not know the name of this class. The default is **com.tibco.tibjms.TibjmsTopicConnectionFactory**.
-jmshostname:	For IBM WebSphere MQ only. The name of the host to use for this connection. Example: **-jmshostname:myServer**
-jmspassword:	The password for the message server connection.
-jmsport:	For IBM WebSphere MQ only. The name of the port to use for this connection.
-jmsqueuemanager:	For IBM WebSphere MQ only. The name of the queue manager to use for this connection. Example: **-jmsqueuemanager:myQueueManager**
-jmsserver:	The URL for your message server.
-jmsuser:	The user name for the message server connection.

The data simulator **MapMessages** for topic **orders.STATUS.*** contain the following fields and properties.

Note: IBM WebSphere MQ does not support spaces in field or property names. When the simulator is run with the IBM WebSphere MQ connection factory, spaces in field and property names will be replaced with underbar (_) symbols.

Field Name	Field Type	Field Description
CUSTOMER	String	Customer name
EXPECTEDDATE	String	Expected date of completion
NAME	String	Plant name
ORDERDATE	String	Date of the order
ORDERNUMBER	String	Order number
PERCENTCOMPLETED	double	Percent of the order that is completed
PLANT	String	Plant name
SCHEDULEDDATE	String	Date the order is scheduled for completion
STATUS	String	Order status: WORKING, BROKEN, COMPLETED
UNITSCOMPLETED	double	Number of units completed for this order
X	double	X location coordinate
Y	double	Y location coordinate

Property Name	Property Type	Property Description
ClientID	String	The client identifier for the message server connection
Continent	String	Name of continent
OrderType	int	Order type code
Plant Property	String	Plant name

The data simulator **MapMessages** for topic **WEATHER.REGIONAL.*** contain the following fields and properties:

Field Name	Field Type	Field Description
Name	String	WEST or EAST
NumberReports	int	Number of weather reports
RegionalData	XML String	XML string containing "Weather Data"
timeStamp	String	Current time

Property Name	Property Type	Property Description
ClientID	String	JmsSimData or value passed in on command line for -jmsclient
Number of Stations	int	Number of stations reporting
Region Property	String	WEST or EAST

The data simulator **TextMessages** for topic **orderStatus.simple** contains the following properties. The body of the message toggles between WORKING, BROKEN, and COMPLETED.

Property Name	Property Type	Property Description
ClientID	String	The client identifier for the message server connection
OrderType	int	Order type code
Value	String	Same value as body

The data simulator **TextMessages** for topic **WEATHER.XML.*** contains the following properties. The body of the message contains an XML string containing "Weather Data".

Property Name	Property Type	Property Description
Number of Stations	int	Number of stations reporting
Time	String	Current time

Weather Data

```
<?xml version="1.0" ?>
<xmldata xmlns="www.sl.com">
   <weatherUpdate>
      <timeStamp>2/23/2005 04:10:45</timeStamp>
      <state MeasurementMethod="every 6 hours" Observations="Upper Air">
        <name>CA</name>
        <numberRegionsReporting>2</numberRegionsReporting>
        <regions>
           <region>
              <name>coast northern</name>
              <reportingIDNum>831</reportingIDNum>
              <currentWeatherConditions>
                 <tempHigh>16</tempHigh>
                 <tempLow>-16</tempLow>
                 <wind>
                    <direction>North</direction>
                    <speed>30</speed>
                 </wind>
                 <rainfall>0.0</rainfall>
```

```
            <snowPack>6.7</snowPack>
          </currentWeatherConditions>
          <toDateTotals>
              <rainfall>0.0</rainfall>
              <snowPack>6.7</snowPack>
              <tempHigh>16</tempHigh>
              <tempLow>-16</tempLow>
           </toDateTotals>
        </region>
        <region>
           <name>inland northern</name>
           <reportingIDNum>95</reportingIDNum>
           <currentWeatherConditions>
              <tempHigh>17</tempHigh>
              <tempLow>2</tempLow>
              <wind>
                <direction>East</direction>
                <speed>24</speed>
             </wind>
             <rainfall>4.0</rainfall>
             <snowPack>1.5</snowPack>
          </currentWeatherConditions>
          <toDateTotals>
             <rainfall>4.0</rainfall>
             <snowPack>1.5</snowPack>
             <tempHigh>17</tempHigh>
             <tempLow>2</tempLow>
          </toDateTotals>
       </region>
    </regions>
    <currentStateTotals>
       <currentMaxRainfall>4.0</currentMaxRainfall>
       <currentMaxSnowfall>6.7</currentMaxSnowfall>
       <currentHighTemp>17</currentHighTemp>
       <currentLowTemp>-16</currentLowTemp>
    </currentStateTotals>
    <todateStateTotals>
       <toDateTotalRainfall>28.0</toDateTotalRainfall>
       <toDateTotalSnowfall>6.7</toDateTotalSnowfall>
       <todateHighTemp>17</todateHighTemp>
       <todateLowTemp>-16</todateLowTemp>
    </todateStateTotals>
 </state>
 <state MeasurementMethod="daily" Observations="Satellite">
    <name>OR</name>
    <numberRegionsReporting>2</numberRegionsReporting>
    <regions>
      <region>
         <name>central eastern</name>
         <reportingIDNum>848</reportingIDNum>
         <currentWeatherConditions>
            <tempHigh>64</tempHigh>
            <tempLow>5</tempLow>
            <wind>
               <direction>South East</direction>
               <speed>42</speed>
            </wind>
            <rainfall>0.0</rainfall>
            <snowPack>4.0</snowPack>
```

```
            </currentWeatherConditions>
            <toDateTotals>
               <rainfall>0.0</rainfall>
               <snowPack>4.0</snowPack>
               <tempHigh>64</tempHigh>
               <tempLow>5</tempLow>
            </toDateTotals>
         </region>
         <region>
            <name>southern</name>
            <reportingIDNum>54</reportingIDNum>
            <currentWeatherConditions>
               <tempHigh>49</tempHigh>
               <tempLow>39</tempLow>
               <wind>
                  <direction>South</direction>
                  <speed>10</speed>
               </wind>
               <rainfall>0.0</rainfall>
               <snowPack>0.0</snowPack>
            </currentWeatherConditions>
            <toDateTotals>
               <rainfall>0.0</rainfall>
               <snowPack>0.0</snowPack>
               <tempHigh>49</tempHigh>
               <tempLow>39</tempLow>
            </toDateTotals>
         </region>
      </regions>
      <currentStateTotals>
         <currentMaxRainfall>0.0</currentMaxRainfall>
         <currentMaxSnowfall>4.0</currentMaxSnowfall>
         <currentHighTemp>64</currentHighTemp>
         <currentLowTemp>5</currentLowTemp>
      </currentStateTotals>
      <todateStateTotals>
         <toDateTotalRainfall>15.0</toDateTotalRainfall>
         <toDateTotalSnowfall>4.0</toDateTotalSnowfall>
         <todateHighTemp>64</todateHighTemp>
         <todateLowTemp>5</todateLowTemp>
      </todateStateTotals>
   </state>
   <state MeasurementMethod="twice a day" Observations="Buoy Reports">
      <name>WA</name>
      <numberRegionsReporting>2</numberRegionsReporting>
      <regions>
         <region>
            <name>north west</name>
            <reportingIDNum>225</reportingIDNum>
            <currentWeatherConditions>
               <tempHigh>44</tempHigh>
               <tempLow>-18</tempLow>
               <wind>
                  <direction>North</direction>
                  <speed>3</speed>
               </wind>
               <rainfall>2.0</rainfall>
               <snowPack>4.5</snowPack>
            </currentWeatherConditions>
```

```
            <toDateTotals>
               <rainfall>2.0</rainfall>
               <snowPack>4.5</snowPack>
               <tempHigh>44</tempHigh>
               <tempLow>-18</tempLow>
            </toDateTotals>
         </region>
         <region>
            <name>central</name>
            <reportingIDNum>135</reportingIDNum>
            <currentWeatherConditions>
               <tempHigh>59</tempHigh>
               <tempLow>41</tempLow>
               <wind>
                  <direction>North East</direction>
                  <speed>7</speed>
               </wind>
               <rainfall>2.0</rainfall>
               <snowPack>0.0</snowPack>
            </currentWeatherConditions>
            <toDateTotals>
               <rainfall>2.0</rainfall>
               <snowPack>0.0</snowPack>
               <tempHigh>59</tempHigh>
               <tempLow>41</tempLow>
            </toDateTotals>
         </region>
      </regions>
      <currentStateTotals>
          <currentMaxRainfall>2.0</currentMaxRainfall>
          <currentMaxSnowfall>4.5</currentMaxSnowfall>
          <currentHighTemp>59</currentHighTemp>
          <currentLowTemp>-18</currentLowTemp>
       </currentStateTotals>
       <todateStateTotals>
          <toDateTotalRainfall>32.0</toDateTotalRainfall>
          <toDateTotalSnowfall>4.5</toDateTotalSnowfall>
          <todateHighTemp>59</todateHighTemp>
          <todateLowTemp>-18</todateLowTemp>
       </todateStateTotals>
    </state>
  </weatherUpdate>
</xmldata>
```

JMS Data Source Command Line Options

In addition to General Options, the following command line arguments are enabled with the JMS data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window. See "Command Line Options: Display Builder and Display Viewer" for more information

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: **"-sub:$data:my Data"**).

Name	Description
-jmsconn	Add a connection definition using the following arguments: conName factoryName serverURL userName password clientID It must contain at least the connection name and factory name. The remaining fields are optional. Use a - to specify no value for a field. **Note:** This command line argument must be enclosed in quotes. If any of the arguments contain a space, it must be enclosed in single quotes. Example: **"-jmsconn:'My Con' com.tibco.tibjms.TibjmsTopicConnectionFactory tcp:\\myserver:7222 user1 pass1 client1"**
-jmstopic	Add a message topic for a connection that RTView will listen for and use to populate dialog drop down menus. This topic will only be added if the specified connection is already defined. **Note:** This command line argument must be enclosed in quotes. If the connection name or the topic contains a space, it must be enclosed in single quotes. Example: **"-jmstopic:'My Con' orders.STATUS.*"**
-jms_customhandlername:className	Specify an alternate class name. Include gmsjrtview.jar and gmsjjmsds.jar (in the lib directory of your RTView installation) and the jms.jar from your JMS vendor in the classpath when building this subclass. By default, RTView looks for a class named MyJmsHandler.
-jms_maxqueuemsgcount:(number of messages)	Enter the maximum number of messages that will be queried from a queue browser for any queue. Default is 100. Example: **-jms_maxqueuemsgcount:100**
-jms_minreconnecttime:(seconds)	Enter the minimum number of seconds that will elapse before attempting to reconnect to the server. Default is 30. Example: **-jms_minreconnecttime:30**

JMX Data Source

Java Management Extensions (JMX™) technology is rapidly becoming a common, standardized way to provide basic monitoring of Java™/J2EE™ applications and application servers. By equipping applications with components known as MBeans, IT Departments can expose application information (i.e.: availability, downtime, upgrades, performance, sizing, security, integration) and other key metrics to enable the monitoring and control of applications. In addition, many middleware products, such as application servers and business process management tools, are being offered with performance metrics automatically exposed via JMX™.

While many companies have enabled JMX™ for their mission-critical J2EE™ applications, they find they are still unable to effectively manage and monitor them. While some solutions do exist, they lack key functionality such as alerts, real-time analytics, or the ability to deliver information in context by comparing real-time data to historical data. Without these capabilities, companies are unable to provide adequate application management, nor true real-time operational visibility.

With RTView, the JMX™ data source allows you to:

- browse available MBean attributes and assign their values as input to a variety of graphic objects in web-based dashboards;
- archive the MBean data in the RTView Historian for trend analysis;
- aggregate MBean attributes to, for example, provide subtotals over dimensions or time series;
- define thresholds and actions based on MBean attributes or aggregated data; and
- create interactive dashboards or automated alert behavior that can manage specific applications with the ability to execute an MBean operation.

This section includes:

System Requirements and Setup - JMX

System Requirements

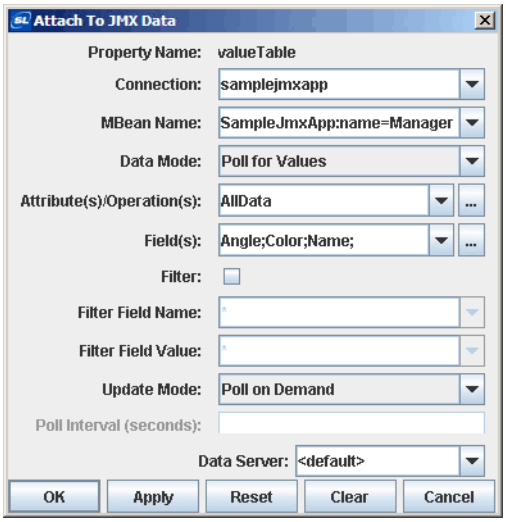
The JMX data source has no additional "System Requirements".

Setup

In addition to general environment variables (see "Setup"), you must include the **tools.jar** file (from your JDK installation) in "RTV_USERPATH" to utilize the **Automatically Discover Local MBean Servers** application option and **-jmxautodiscover** command line option. See "JMX Administration Tab" and "JMS Data Source Command Line Options" for more information.

Attach to JMX Data

From the Object Properties window you can access the Attach To JMX Data dialog. This dialog is used to attach an object property to an MBean attribute or operation, giving you the ability to display the results within an RTView display. Once an object property has been attached, it receives continuous updates. By default, each MBean method is executed once each update period, but you can set the default poll interval on the **JMX Administration** tab in the **Application Options** dialog.



To bring up the **Attach to JMX Data** dialog, right-click on the Property Name from the **Object Properties** window and select **Attach to Data>JMX**. The **Attach to JMX Data** dialog provides several drop down menus that allow you to specify information regarding the MBean method to execute. If the drop down menu does not contain the item you require, type your selection into the text field.

Field Name	Description
Connection	Connection name. Select **Tools>Options** and click on **JMX** to define a JMX Connection or JMX Connection Group. See "JMX Connections Tab" and "JMX Connection Groups Tab" for more information.
MBean Name	Name of the MBean containing the method to execute.
Data Mode	**Poll for Values** – Poll for attribute values or execute operations. If this option is selected, you can control how often the data is polled in the **Update Mode** field. **Listen for Notifications** – Register as a listener for attribute notification events. When this option is selected, initial data is polled for the attribute and then data is only updated when notification events are received.
Attribute(s)/Operation(s)	Name of the attribute or operation to execute. If the attribute or operation you select requires arguments, then the dialog will populate with a text field for each argument. The values entered in these argument fields are not validated. Values listed populate based on the selected **MBean** and **Data Mode**. If Data Mode selected is **Poll for Values**, all available attributes and operations that return data will be listed. Otherwise if Data Mode selected is **Listen for Notifications**, only attributes returned by the MBean **getNotificationInfo()** method will be listed.

Field(s)	Name of the items to display from the selected attribute or operation. This field is not required if a single simple attribute is selected from **Attribute(s)/Operation(s)**.
Filter	Select this check box to filter the MBean method result.
Filter Field Name	MBean method field to use as filter.
Filter Field Value	Value that the filter field must equal. Enter * to display all rows in the table. Enter "*" to use * as a literal comparative value. To list multiple values, separate with a semicolon. For example: **value1;value2;value3**. If your value contains a semicolon, enclose it in single quotes.
Update Mode	If **Data Mode** selected is **Poll for Values**, the following options are available: **Poll Once** - Poll for data only once. Select if the data returned by this attribute or operation is static. **Poll Every Default Poll Interval** - Poll for data each Default Poll Interval. See "Application Options - JMX" for information on setting the **Default Poll Interval**. This is the default **Update Mode**. **Poll Every Poll Interval** - Poll for data each Poll Interval. If this option is selected, you must specify a **Poll Interval**. **Poll On Demand** - Poll for data each time a display that uses the data attachment is opened and each time a substitution string that appears in the data attachment has changed.
Poll Interval	Enter the time (in seconds) to poll for data. This option is only available if the Update Mode selected is Poll Every Poll Interval. **Note:** Because the Poll Interval is superseded by the General Update Period, the amount of time elapsed between MBean polls may be longer than the value entered. For example, if the General Update Period is 2 seconds and the Poll Interval is 5 seconds, MBean attributes and operations will be polled every six seconds. See "General Tab" for more information.
Data Server	Select to read data through your configured Data Server and not directly from the JMX data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab". **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a Named Data Server that you configured in **Application Options>**"Data Server Tab". **Multi-Server Attachment** - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more **Named Data Servers** (e.g. **ds101;ds102**). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default;ds101;ds102**). **Note:** The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of * can be entered to specify all data servers, including **__default** and **__none**. When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received.

A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways:

1. The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows.
2. The multi-server attachment can be applied as the Table argument of the RTView function named Combine Multi-Server Tables. See "Tabular Functions" for more information.

Drop down menus populate based on connections added in the **Application Options** window.

When an object property has been attached to data, the **Property Name** and **Value** in the **Object Properties** window will be displayed in green. This indicates that editing values from the **Object Properties** window is no longer possible. To remove the data attachment and resume editing capabilities in the **Object Properties** window, right-click on the **Property Name** and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the **Property Name** and **Value** are no longer green.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information is validated against the connections added in **Application Options** window.

Note: If the Connection is *, the MBean field is validated against the first active connection. If the MBean name uses * for a key property, the Attribute(s)/Operation(s) field is validated against the first MBean found matching the name.

Note: The Filter Field Value field and argument fields related to selected Attribute(s)/Operation(s) are not validated.

The following describes the significance of the validation colors:

Blue	Unknown Entry	Entry is not recognized. When a Connection is unknown, the MBean, Attribute(s)/Operation(s), Field(s), Filter Field Name, and Filter Field Value are also unknown.
Yellow	Offline	Connection is offline.
White	Valid state	Entry is valid.
Red	Invalid state	Entry is not valid.
Grey	Not Required	Field does not require a value. This applies to Field(s) if the selected attribute or operation returns a single simple field.

Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. A generic connection such as **$conn** is used instead of a specific connection. Later when the display is running, this generic value is defined by the actual name of a specific connection, such as **samplejmxapp**. In this way, a single display can be reused to show data from a number of different sources. Substitutions can be used in any field in this dialog. For more information on creating displays using substitution values, see "Substitutions".

RTViewDs

The JMX data adapter provides an MBean called **RTViewDs:type=Connection**, which provides a table containing JMX connection information. In the **Attach to JMX Data** dialog, select **RTViewDs:type=Connection** from the **MBean Name** drop-down menu.

RTViewDs Table Name	Column Available	Description
Connection	Connection	Name of the JMX Connection for the data attachment.
	Host	JMX MBean Server host name. This field is available if it was used to create the connection (the **Enter URL** option in the **Add JMX Connection** dialog was NOT enabled when the JMX connection was created). See "JMX Connections Tab" for more information.
	Port	Port number for the JMX MBean Server. This field is available if it was used to create the connection (the **Enter URL** option in the **Add JMX Connection** dialog was NOT enabled when the JMX connection was created). See "JMX Connections Tab" for more information.
	URL	URL of the JMX MBean Server. This field is available if it was used to create the connection (the **Enter URL** option in the **Add JMX Connection** dialog was enabled when the JMX connection was created).See "JMX Connections Tab" for more information.
	Connected	True if the connection is connected.
	Display Name	Display name of the connection. Used when Auto Discover is true and is the name of the discovered connection.

Special Values

The following special values can be entered.

Connection	*	All connections are displayed. A Connection attribute is added to the Attribute(s)/Operation(s) list. This attribute displays the name of the connection.

MBean Name	*	One * can be used in the domain of the MBean name in order to display MBeans with similar names. When * is used in the domain, a Domain column containing the domain name is added to the Attribute(s)/Operation(s) list. For example, a connection has the following domains: SampleJmxAgent1 SampleJmxAgent2 MyAgent You could specify * to get all three or **SampleJmxAgent*** to get the first two. However, you could not specify ***Agent*** because only one * is allowed. A * can be used for any key property in the MBean name. When * is used, an attribute named for that key property is added to the Attribute(s)/ Operation(s) list. For example, a connection has the following MBeans: Catalina:type=Manager,path=/,host=localhost Catalina:type=Manager,path=/demoapps,host=localhost Catalina:type=Manager,path=/productionapps,host=otherhost
		You can specify either of the following for the MBean Name in your data attachment: Catalina:type=Manager,*,host=localhost or Catalina:type=Manager,* An attribute will be added to the Attribute(s)/Operation(s) list for each key property following the *. In both examples above, a path and host attribute will be added to the Attribute(s)/Operation(s) list. Since the first example lists host=localhost, the data attachment will return a two row table, the first row showing the information for path=/ and the second row showing the information for path=/demoapps. The host for the MBean with the path=/ productionapps is not localhost, so it will not be included in the returned table. The second example lists * for both path and host, so the data attachment will return a three row table, the first row showing the information for path=/ , host=localhost, the second row showing the information for path=/ demoapps, host=localhost and the third row showing the information for path=/productionapps, host=otherhost.
Attribute(s) / Operation(s)	*	All attributes and operations for the selected MBean method are displayed. **Note:** <TABULAR> will be displayed in cells for MBean methods that return tabular data. The way composite attributes will be displayed depends on the **Expand Composite Attributes** setting in Application Options. See "JMX Administration Tab" for more information.

Select Multiple Attributes / Operations or Attribute Fields (Tables Only)

From the **Attach to JMX Data** dialog you can select multiple attributes or operations to display or, for a single attribute or operation, you can specify which items to display as columns in a table and in what order they will appear. In order to populate the **Attribute(s) / Operation(s)** list, you must first select a valid MBean Name. In order to populate the **Field(s)** list, you must first select a valid entry in the **Attribute(s) / Operation(s)** field.

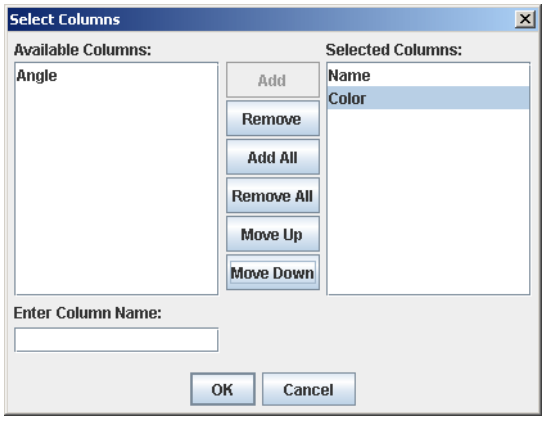
Note: When specifying multiple attributes or operations, you can only select MBean methods that return simple single values or MBean methods that return composite data. The way composite attributes are displayed in a table with other attributes depends on the **Expand Composite Attributes** setting in **Application Options**. See "JMX Administration Tab" for more information.

Click on the ellipsis button or right-click in the **Attribute(s) / Operation(s)** or **Field(s)** fields and choose **Select Columns** to display the **Select Columns** dialog. The dialog should contain a list of Available Columns that you can add to your table.

To add a field, select an item from the **Available Columns** list and click on the **Add** button. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of fields in a table by arranging the items in the **Selected Columns** list with the Move Up and Move Down buttons.

Validation colors indicate whether selected message fields are valid. However, if even one field selected is invalid the field in the **Attach to JMX Data** dialog registers as an invalid entry.

If no data is available for a table row within a selected column, the table cell displays one of the following values: N/A, false, 0, or 0.0.



The following describes **Attach to JMX Data** dialog commands:

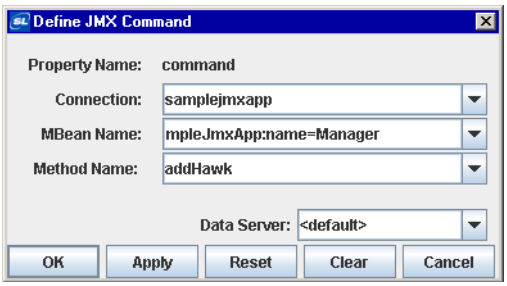
Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Define JMX Command

Note: The JMX data source may not be licensed in your RTView installation.

You can access the **Define JMX Command** dialog from the **Object Properties** window. This dialog is used to assign an MBean operation to an object's command property, giving you the ability to execute an MBean operation from within an RTView display. If you execute a JMX command from a Thin Client with Direct Data Connection or any Served Data deployment, the command will execute on the server. If your command fails on a Direct Data deployment, an error dialog will popup with information on why the command failed. In other deployments, the error is only output to the console.

To open the **Define JMX Command** dialog, right-click on the appropriate command property in the Object Properties window and select **Define Command>JMX**. See the "Define/Execute Command" section for information on how to execute a command.



Field Name	Description
Connection	Connection name. You may define a connection on the "JMX Connections Tab" of the **Application Options** dialog. The "JMX Connection Groups Tab" allows you to define Connection Groups and add them to the drop down menu in the Define Command dialog.
MBean Name	Name of the MBean containing the operation you want to execute.
Method Name	Name of the operation you want to execute.
Method Argument(s)	Method argument fields populate based on which MBean Name and Method Name are selected. To attach a method argument to data, right-click and choose **Attach to Data** or double-click in the field.
Data Server	Select to read data through your configured Data Server and not directly from the JMX data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab" **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". **Multi-Server Command** - When multiple data servers are specified, the command will be executed on each data server in the list. To configure multiple data servers, enter a semicolon (;) delimited list containing two or more **Named Data Servers** (e.g. **ds101**;**ds102**). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**). **Note:** The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

Drop down menus populate based on connections added in the **Application Options** dialog. Once a method is selected, argument fields are added to the dialog based on the selected method. See "Application Options - JMX" for more information.

Substitutions

The Substitutions feature allows you to build open-ended displays in which commands depend on values defined at the time the display is run. A generic connection such as $conn is used instead of a specific connection. Later when the display is running, this generic value is defined by the actual name of a specific connection, such as **samplejmxapp**. In this way, a single display can be reused to send a number of different messages. Substitutions can be used in any field in this dialog. For more information on creating displays using substitution values, see "Substitutions".

Special Values

$value	When an **actionCommand** is executed $value is replaced with the value from the control. This value may be used in any field in the **Define JMX Command** dialog. **Note:** This value may only be used for Action Commands. See "Define/Execute Command" for more information.

The following describes **Define JMX Command** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from assigned message subject (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Substitutions - JMX

In addition to standard built-in substitutions, this data source also sets the following drill down substitutions:

Substitution Value	Definition
$conn	Connection name from the selected row or object.
$mbean	MBean name from the selected row or object.
$rowindex	Row index from the selected row or object.
$filterfield	Filter field name from the selected row or object.
$filtervalue	Filter field value from the selected row or object.

See "Drill Down Substitutions" for more information.

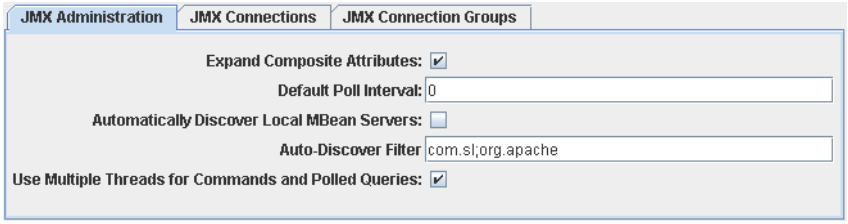
Application Options - JMX

To access the **Application Options** dialog, in the Display Builder select **Tools>Options**. There are three Application Options tabs for JMX: "JMX Administration Tab", "JMX Connections Tab", and "JMX Connection Groups Tab".

Options specified in JMX tabs can be saved in an initialization file (**JMXOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server and Historian to set initial values. If no directory has been specified for your initialization files and **JMXOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under lib in your installation directory. See "RTV_JAVAOPTS" for more information.

Note: Options specified using command line arguments will override values set in initialization files. See "JMX Data Source Command Line Options" for more information.

JMX Administration Tab



Field Name	Description
Expand Composite Attributes	Select this option so that you display one column for each item when a composite attribute is displayed in a table with other attributes. The column name will be *compositename.itemname*.
Default Poll Interval	Enter the time in milliseconds to control how often MBean attributes are polled or operations in data attachments are executed if no Poll Interval is specified in the data attachment. Default is 0, which polls MBean attributes and operations according to the **General Update Period** specified in **Application Options** on the "General Tab". Because the Default Poll Interval is superseded by the General Update Period, the amount of time elapsed between MBean polls may be longer than the value entered. For example, if the General Update Period is 2000 milliseconds and the Default Poll Interval is 5000 milliseconds, MBean attributes and operations will be polled every six seconds.
Automatically Discover Local MBean Servers	If selected, RTView will get the list of local MBean Servers once each poll interval. **Note:** Poll interval is determined by either the **Default Poll Interval** or according to the **General Update Period** specified in **Application Options** on the "General Tab". All new servers that are discovered are added as connections using the process ID for the connection name. Connections to servers that have exited will be removed. Auto-discovered connections are not saved to **JMXOPTIONS.ini**. All auto-discovered connections are added to a JMX Connection Group named RTViewDs-Auto. This group cannot be removed or edited and is not saved to **JMXOPTIONS.ini**. **Note:** You must include the tools.jar file (from your JDK installation) in "RTV_USERPATH".

Auto-Discover Filter

Filter auto-discovered connections. The filter can be a single value or a ; (semicolon) delimited list of values.

For example, **com.sl;org.apache** would limit auto-discovered connections to those that start with **com.sl** or **org.apache**.

Use Multiple Threads for Commands and Polled Queries

Select to enable multiple threads for commands and polled queries. This option enhances performance in the case where some JMX servers are slow, thus preventing a single server (or servers) from delaying updates from all JMX servers.

Note: RTView will create one thread per connection, so this is not a practical option for cases where hundreds of connections are defined.

When multiple threads are enabled, the thread that created the JMX connection will be used for all commands and polled queries on that connection. The polled queries will initiate based on the Poll Interval for the data attachment and each connection will update their listeners asynchronously when data becomes available.

If multiple threads are disabled, the polled queries are executed synchronously and the listener is not updated until the data from all connections in the data attachment has returned.

For example, a data attachment for a table has * in the **Connection** field and there are five defined JMX connections:

- If multiple threads are disabled, then RTView (every Update Period or Poll Interval) will synchronously query all five JMX connections and then update the listener once with a table containing the data for all five connections.

When multiple threads are enabled, then RTView (every Update Period or Poll Interval) will asynchronously initiate queries on all five JMX connections. As each query returns data, the listener will be updated with data for that connection. Therefore, the table will be updated five times per Update Period or Poll Interval with each update only containing the data for one connection.

Due to the difference in how listeners are updated, users need to confirm that the listeners for their JMX data attachments to multiple connections can handle partial updates. For example:

- If you want to see all connections in a single table, you must cache the data. The **indexColumnNames** property on the cache must be configured correctly.
- For existing caches attached to JMX data that want to use this feature, users should confirm that the **indexColumnNames** property is properly configured. Since the previous behavior wrote all of the JMX data on every update, a cache could have previously worked correctly even if the **indexColumnNames** property was not configured correctly. When multiple threads are enabled, this will no longer be the case.

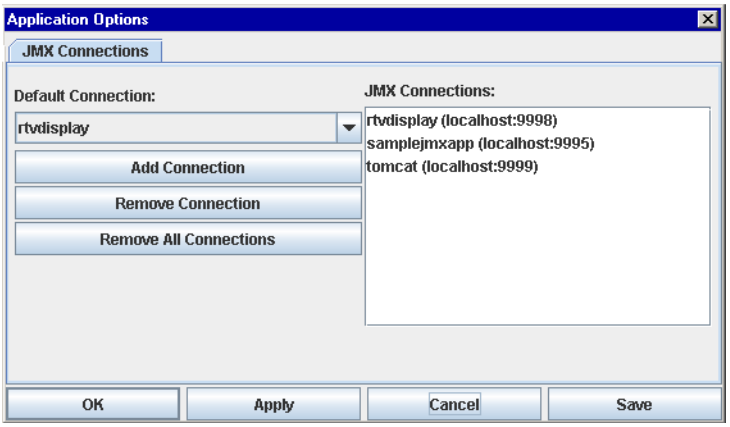
For existing functions attached to JMX, users must confirm that the function will operate as expected when receiving update containing only one connection at a time. For example, a **Join** function attached to JMX might assume that all connections are present in every update. If your function requires all of the connections in each update in order to operate correctly, first put the JMX data into a cache and then use the cache as input to the function.

JMX Connections Tab

This tab allows you to add or remove connections and set your default connection.

When you add a JMX connection to the list it will be highlighted in yellow indicating that RTView has not connected to it. To attempt to connect to a JMX connection, click **OK**, **Apply**, or **Save**. If the background remains yellow, then RTView was unable make a connection. Check that your JMX connection was setup correctly and that the JMX instrumented application is running.

Note: Regardless of which tab you are currently working from in the **Application Options** dialog, each time you click **OK**, **Apply**, or **Save** RTView will attempt to connect to all unconnected connections.



Field Name	Description
Default Connection	Name of connection used as the default for data attachments. Select from drop down menu to change default setting.
Add Connection	Click to open the **Add JMX Connection** dialog. To edit an existing connection, double-click on a selection from the JMX Connections list.

Connection Name -- Name to use when referencing this JMX connection in your data attachments.

Host or IP -- Host name or IP address of the MBean server.

Port -- Port that your JMX instrumented application is using to expose MBean methods.

Enter URL -- Select to enter the URL (below) that should be used to create this connection. Otherwise, the Host/IP and Port specified will be used to construct a URL for this connection.

URL -- URL that should be used to create this connection. Select **Enter URL** to enable this field. Enter **local** to make a local connection to an in-process MBean server.

Use Client Credentials -- If selected, the User Name and Password from the RTView login will be used instead of the User Name and Password entered in the Add JMX Connection dialog. Connections to this database will only be made when you are running with login enabled. See "Login" for more information.

If you will be using the Data Server or the Display Server with a JMX connection that has this option enabled, you must enable **Use Client Credentials for Database Login** in these applications. In both of these deployments, a connection will be made for each client login, as the users log in. Once all of the clients for a login have closed, the connection for that login is removed. See "Running the Data Server" and "Thin Client Browser with Direct Data - Manual Setup" for more information.

User Name -- User name to use when creating this connection. This parameter is optional.

Password -- Password to use when creating this connection. This parameter is optional.

If you need to provide an encrypted password (rather than expose server password names in a clear text file), use the **encode_string** command line option with the following syntax:

encode_string type mypassword

where **type** is the key for the data source and **mypassword** is your plain text password. **Note:** The **type** argument is only required when you encrypt a string for a data source.

For example, enter the following in an initialized command window (see "Initializing a Command Prompt or Terminal Window"):

encode_string jmx mypassword

and you will receive an encrypted password:

encrypted value:
013430135501346013310134901353013450134801334

Copy the encrypted value, paste it into the password field and click Save to save this value to the initialization (***.ini**) file. Or, if necessary, manually edit the (*.ini) file to include the encrypted value.

Note: If you need to manually edit a configuration (*.ini) file, contact SL Technical Support at support@sl.com for information about supported syntax.

Additional Properties -- Add any properties necessary to initiate connections with specific MBean servers.

Property Name -- Enter a property name. To specify an encrypted password, use **SL.** as a prefix in the Property Name. For example: **SL.yourpropertyname**

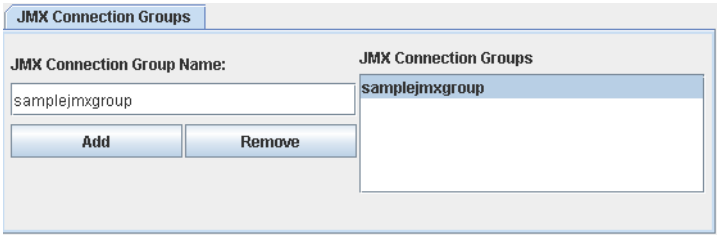
Property Value -- Enter a property value. To specify an encrypted password, execute the encode_string command line option (above) and paste the encrypted value here.

Add Property -- Once you have entered a Property Name and Property Value and click **Add Property** to insert.

Remove Property -- Select a Property Name from the list and click **Remove Property** to delete.

Remove Connection	Select a connection from the list and click Remove Connection to delete.
Remove All Connections	Removes all connections.

JMX Connection Groups Tab



Field Name	Description
JMX Connection Group Name	Enter a JMX Connection Group Name and click **Add** to insert. Once you have added a group, double-click on the name of that group to define its connections in the **Select Connections** dialog. **Note:** It is possible to enter a Connection Name that contains one or more wildcard (i.e. *****) characters. For example, **rtvdisplay*** or ***jmx***.
JMX Connection Groups	Select a group from the list and click **Remove** to delete.

Connecting to Multiple Secure Servers

You can create a JMX connection to connect to multiple secure servers for the following types of servers:

- "Connecting to Multiple Secure WebSphere Servers" on page 525
- "Connecting to Multiple Secure Tomcat Servers" on page 527
- "Connecting to Multiple Secure WebLogic Servers" on page 529

Connecting to Multiple Secure WebSphere Servers

If you need to connect to multiple secure WebSphere servers, you can create a JMX connection to a WebSphere server that, during setup, allows you to add the following WebSphere properties:

com.ibm.CORBA.ConfigURL (required)

com.ibm.SOAP.ConfigURL (optional)

com.ibm.SSL.ConfigURL (required)

These properties point to WebSphere properties files that can be used, among other things, to connect to other WebSphere servers.

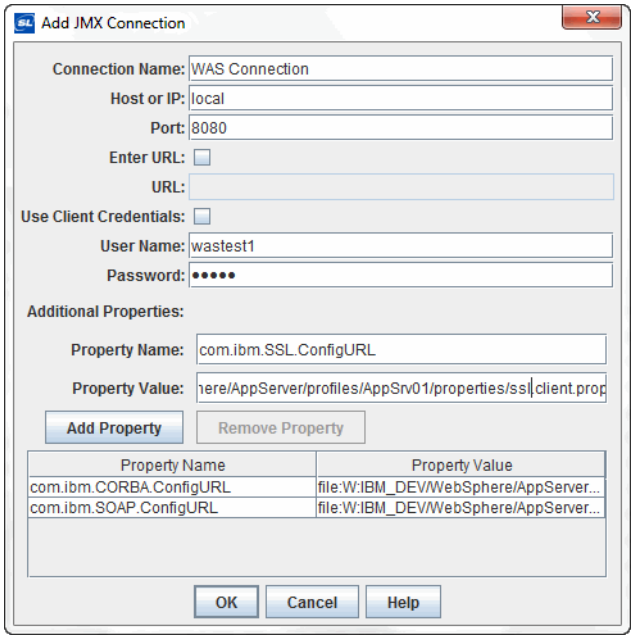
Note: The IBM JVM that is required for connecting to an IBM WebSphere Application Server does not support the use of more than one truststore in a session, which means that the first truststore loaded by RTView via a defined JMX connection is the only truststore it will see. Hence, this truststore will be

expected to accept the certificates of all JMX connections defined for your data server. If using a single valid truststore for all of your servers is not possible, then defining an RTVAgent configuration where each server has a dedicated agent data server is one possible solution. See "RTVAgent Data Source" for more information.

To create the connection:

1. Navigate to **Application Options**> **JMX**> **JMX Connections** tab> **Add Connection**.

   The **Add JMX Connection** dialog displays.



2. In the **Add JMX Connection** dialog:

   a. Enter **Connection Name**, **Host or IP**, and **Port**

   b. Enter Client Credentials (**User Name** and **Password**)

   c. Specify each **Property Name** and **Property Value** and click the **Add Property** button after defining each.

   d. Click **OK**.

3. Click **Save** on the **Application Options** dialog to save your changes.

   The changes are saved to the **JMXOPTIONS.ini** file:

```
jmxconn WebSpheretestConnection localhost 8080 URL:- WebSpheretester
013210129901317013500133501349013500133501348012 83 false
com.ibm.CORBA.ConfigURL file:W:/IBM_DEV/WebSphere/AppServer/profiles/AppSrv01/
properties/sas.client.props
com.ibm.SOAP.ConfigURL file:W:/IBM_DEV/WebSphere/AppServer/profiles/AppSrv01/
properties/soap.client.props
com.ibm.SSL.ConfigURL file:W:/IBM_DEV/WebSphere/AppServer/profiles/AppSrv01/
properties/ssl.client.props
jmxdefaultconn WebSpheretestConnection
jmx_mbeans_change_dynamically true
```

4. To run the instances defined in the property files simultaneously, you need to set the **jmx_use_multiple_threads** property to **true**. You can accomplish this in one of three different ways:

   a. Specify the property on the command line when starting the dataserver:

   ```
   run_dataserver -jmx_use_multiple_threads
   ```

   b. Add the property to the **sample.properties** file:

   ```
   jmx_use_multiple_threads=true
   ```

   c. Add the property to the **JMXOPTIONS.ini** file:

   ```
   jmxconn WebSpheretestConnection localhost 8080 URL:- WebSpheretester
   013210129901317013500133501349013500133501348012 83 false
   com.ibm.CORBA.ConfigURL file:W:/IBM_DEV/WebSphere/AppServer/profiles/AppSrv01/
   properties/sas.client.props
   com.ibm.SOAP.ConfigURL file:W:/IBM_DEV/WebSphere/AppServer/profiles/AppSrv01/
   properties/soap.client.props
   com.ibm.SSL.ConfigURL file:W:/IBM_DEV/WebSphere/AppServer/profiles/AppSrv01/
   properties/ssl.client.props
   jmxdefaultconn WebSpheretestConnection
   jmx_mbeans_change_dynamically true
   jmx_use_multiple_threads true
   ```

Connecting to Multiple Secure Tomcat Servers

If you need to connect to multiple secure Tomcat servers, you can create a JMX connection to a Tomcat server that, during setup, allows you to add the following Tomcat properties:

javax.net.ssl.trustStore (required)

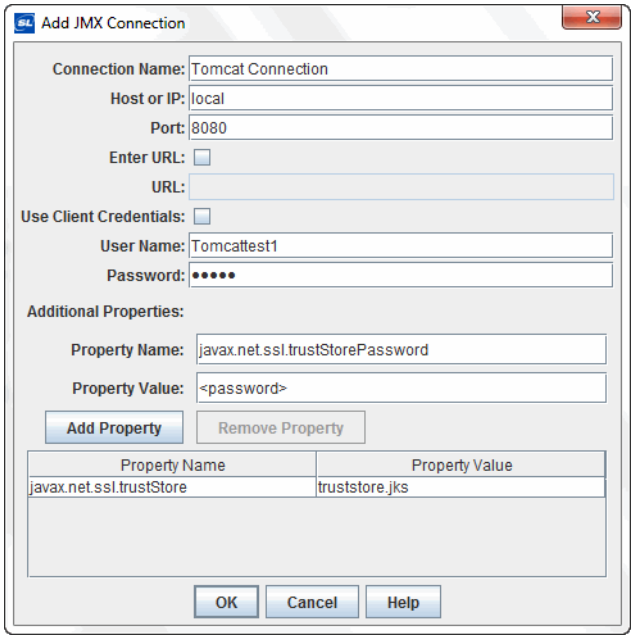
javax.net.ssl.trustStorePassword (required)

These properties point to Tomcat properties files that can be used, among other things, to connect to other Tomcat servers.

To create the connection:

1. Navigate to **Application Options**> **JMX**> **JMX Connections** tab> **Add Connection**.

   The **Add JMX Connection** dialog displays.



2. In the **Add JMX Connection** dialog:

   a. Enter **Connection Name**, **Host or IP**, and **Port**

   b. Enter Client Credentials (**User Name** and **Password**)

   c. Specify each **Property Name** and **Property Value** and click the **Add Property** button after defining each.

   d. Click **OK**.

3. Click **Save** on the **Application Options** dialog to save your changes.

   The following is added to the **JMXOPTIONS.ini** file:

```
jmxconn 'Tomcat Connection' local 8080 URL:- - - false
javax.net.ssl.trustStorePassword <password>  javax.net.ssl.trustStore
truststore.jks
```

4. To run the instances defined in the property files simultaneously, you need to set the **jmx_use_multiple_threads** property to **true**. You can accomplish this in one of three different ways:

   a. Specify the property on the command line when starting the dataserver:

```
run_dataserver -jmx_use_multiple_threads
```

   b. Add the property to the **sample.properties** file:

```
jmx_use_multiple_threads=true
```

   c. Add the property to the **JMXOPTIONS.ini** file:

```
jmx_metrics_period 5000
jmx_minreconnecttime 15
jmxconn rtvdisplay localhost 9998 URL:- - - false
jmxconn 'Tomcat Connection' local 8081 URL:- - - false
javax.net.ssl.trustStorePassword <password>  javax.net.ssl.trustStore
truststore.jks
```

```
jmxconn samplejmxapp localhost 9995 URL:- - - false
jmxconn rtvdata localhost 9997 URL:- - - false
jmx_mbeans_change_dynamically true
jmx_use_multiple_threads true
```

Connecting to Multiple Secure WebLogic Servers

If you need to connect to multiple secure WebLogic servers, you can create a JMX connection to a WebLogic server that, during setup, allows you to add the following WebLogic property:

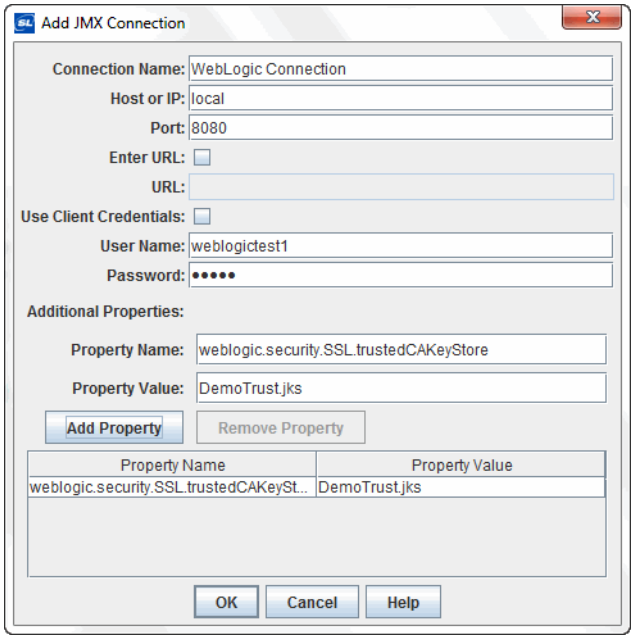
weblogic.security.SSL.trustedCAKeyStore (required)

This property points to WebLogic properties files that can be used, among other things, to connect to other WebLogic servers.

To create the connection:

1. Navigate to **Application Options**> **JMX**> **JMX Connections** tab> **Add Connection**.

   The **Add JMX Connection** dialog displays.



2. In the **Add JMX Connection** dialog:

   a. Enter **Connection Name**, **Host or IP**, and **Port**

   b. Enter Client Credentials (**User Name** and **Password**)

   c. Specify each **Property Name** and **Property Value** and click the **Add Property** button after defining each.

   d. Click **OK**.

3. Click **Save** on the **Application Options** dialog to save your changes.

   The following is added to the **JMXOPTIONS.ini** file:

```
jmxconn 'WebLogic Connection' local 8080 URL:- weblogictest1
013500133501349013500128 3 false weblogic.security.SSL.trustedCAKeyStore
DemoTrust.jks
```

4. To run the instances defined in the property files simultaneously, you need to set the **jmx_use_multiple_threads** property to **true**. You can accomplish this in one of three different ways:

   a. Specify the property on the command line when starting the dataserver:

   ```
   run_dataserver -jmx_use_multiple_threads
   ```

   b. Add the property to the **sample.properties** file:

   ```
   jmx_use_multiple_threads=true
   ```

   c. Add the property to the **JMXOPTIONS.ini** file:

   ```
   jmx_metrics_period 5000
   jmx_minreconnecttime 15
   jmxconn rtvdisplay localhost 9998 URL:- - - false
   jmxconn 'WebLogic Connection' local 8080 URL:- weblogictest1
   013500133501349013500128 3 false weblogic.security.SSL.trustedCAKeyStore
   DemoTrust.jks
   jmxconn samplejmxapp localhost 9995 URL:- - - false
   jmxconn rtvdata localhost 9997 URL:- - - false
   jmx_mbeans_change_dynamically true
   jmx_use_multiple_threads true
   ```

RTView Deployment - JMX

This page contains details about the deployment process that are specific to your data source. Please go to the **Deployment** section of this documentation for instructions on how to implement your RTView deployment option. Return to this page whenever you are instructed to refer to deployment information that is specific to your data source.

System Requirements and Setup

The JMX data source has no additional requirements.

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See “Application Options”, “Application Options - JMX”, and “RTV_JAVAOPTS” for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
JMXOPTIONS.ini	Contains data source options for JMX.

Note: Options specified using command line parameters override values set in these initialization files.

Demos - JMX

Except where noted, all demos can be run in three ways, as an application, or via rich or thin client in a browser.

Before You Begin

Start the Demo Server

Thin Client Demo only.

There is a thin client demo already installed on the "RTView Demo Server".

In an initialized command window (see "Initializing a Command Prompt or Terminal Window"):

Type **run_startup_demoserver**

Data Source Demo

The Data Source Demo is designed to illustrate each data source.

1. **Start the Simulators**

   Start the simulators for each data source you will be using. To run the "Sample JMX Application":

   In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory and:

   type run_samplejmxapp

2. **Run Demos - Application or Thin Client Browser**

   Application Demo

   1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
   2. To view the demo, type:

      run_viewer

   3. To edit the demo, type:

      run_builder

   Thin Client Browser Demo

   "Start the Demo Server" if it is not running.

   1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
   2. Start the Display Server by typing:

      run_displayserver

   3. Open a browser and navigate to **http://localhost:8068/dstutorial**.

JMX Monitor Demo

1. "Start the Simulators"
2. **Run Demos - Application or Thin Client Browser**

Application Demo

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/jmxmonitor** directory.
2. To view the demo, type:

   run_viewer

3. To edit the demo, type:

   run_builder

Thin Client Browser Demo

"Start the Demo Server" if it is not running.

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/jmxmonitor** directory.
2. Start the Display Server by typing:

   run_displayserver

3. Open a browser and navigate to **http://localhost:8068/jmxmonitor**.

Quick Start Tutorial: JMX

This Quick Start Tutorial provides you with the fundamentals on how to use RTView with a JMX Application. Once completed, you can swiftly apply this knowledge to building your own real-time dashboard displays for visual access to your JMX data.

Learn to:

- Animate graphic objects with JMX data
- Create a drill down display with JMX data
- Execute a JMX command

Note: The JMX data source may not be licensed in your RTView installation.

Get Started

This tutorial requires the following:

- Register for a license key. If you have not, you must do so before continuing. See "Registration" for more information.
- "Quick Start Tutorial" - This tutorial requires that you have a working knowledge of RTView. We recommend that you complete the Quick Start Tutorial before continuing.

Start the Sample JMX Application

In this exercise you start the sample JMX application which is the data source used in this tutorial. The sample JMX application is a JMX instrumented application that you can use to animate objects in the Display Builder and Display Viewer.

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), start the sample JMX application:

   Type **run_samplejmxapp &**

   The sample JMX application runs as a background process.

Note: You must follow this initialization process for each new terminal window you open. See the "Setup" section for more details about setting up your environment.

Start the Display Builder

If you are already logged onto the Display Builder, skip this section and go to the **Create a Display** section, below.

1. In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"):

   type **run_builder**

2. Login to the Display Builder. By default, the Display Builder does not require a login. Login can be enabled at setup to support "Role-based Security". The default user name and password are:

   User Name: **admin**

   Password: **admin**

   Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role.

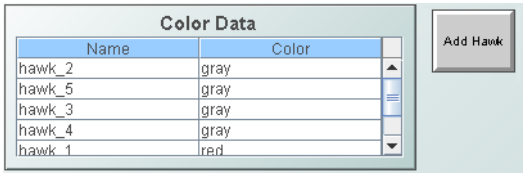
   You are now ready to create a display using the data source.

Create A Display

At this point you have:

- Registered for a license key. See "Registration" for more information.
  - Started the sample JMX application
  - Logged on to the Display Builder
  - Completed the "Quick Start Tutorial"

In this tutorial you use the sample JMX application as a data source to create a Color Data table that has two columns, one that lists hawk names and the other that lists their color status, as shown below. You will also create an Add Hawk button that adds another hawk to the sample JMX application which is then reflected in the Color Data table.



As you saw in the Quick Start Tutorial, the data structure of tables and graphs (tabular data) enables RTView to automatically create several data source specific, built-in Substitutions for you. You will see these built-in Substitutions used in the target display when you create the drill down. For more information on Substitutions, see "Substitutions".

In this exercise, you create a drill down using the previously created display, jmx_dd_qs.rtv, as the target display. First you will set the Color Data table to display two columns of data: the device name and the color. Then you will create a drill down that opens a bar graph that shows more detailed data for each device.

Setup Your JMX Connection

Setting up the JMX connection makes the data source available for animating graphic objects in your display.

1. In the Display Builder, select **Tools>Options** to open the **Application Options** dialog. See "Application Options - JMX" for more information.

2. Select the **JMX Connections** tab and click on **Add Connection** to open the Add Connection dialog.

3. In the **Add Connection** dialog:

   Connection Name - Enter **samplejmxapp**

   Host or IP - Enter **localhost**.

   Port - Enter **9995**. This is the port that the sample JMX application uses to expose its MBean methods.

4. Click **OK**.

5. Click **OK** to apply and close the Application Options dialog.

The **samplejmxapp** connection is now available for animating graphic objects in your display.

Display Data in a Table

In this exercise you add a table and then display data in the table by attaching it to the data source.

1. Click on the Add Table  button and click again in the Working Area to place the table.

2. In the Object Properties dialog:

   label (category: Label) - Change the name of the label to **Color Data**.

   autoResizeFlag (category: Column) - Click to select the check box.

   valueTable (category: Data) - Right-click in the Property Name field and select **Attach to Data>JMX**.

3. In the **Attach to JMX Data** dialog:

   Connection - Select **samplejmxapp**.

   MBean Name - Select **SampleJmxApp:name=Manager**.

   Method(s)/Attributes(s) - Select **AllData**.

   Field(s) - Click on the ... button to open the Select Columns dialog.

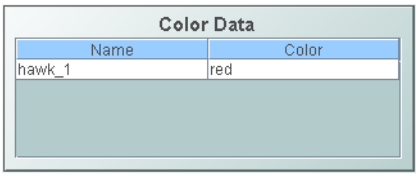
4. In the **Select Columns** dialog:

   Select **Name** in the Available Columns list and click **Add**.

   Select **Color** in the Available Columns list and click **Add**.

5. Click **OK** to close the Select Columns dialog.

6. Click **OK** to apply these values and close the Attach to JMX Data dialog.



The first hawk from **samplejmxapp** is displayed in the Color Data table. You now need to create a button that adds other hawks.

Create a Button that Executes a Command

In this exercise you create a button that adds additional hawks to the sample JMX application. You do this by adding the button object and defining it with a command. In this case, the command is **addHawk**.

1. Click the Add Rectangle Label button and click again in the Working Area to add a label to the display.

2. In the **Object Properties** dialog:

   command (category: Interaction) - Right-click in the Property Name field and select **Define Command>JMX**.

3. In the "Define JMX Command" dialog:

   Connection - Select **samplejmxapp**.

   MBean Name - Select **SampleJmxApp:name=Manager**.

   Method(s)/Attributes(s) - Select **addHawk**.

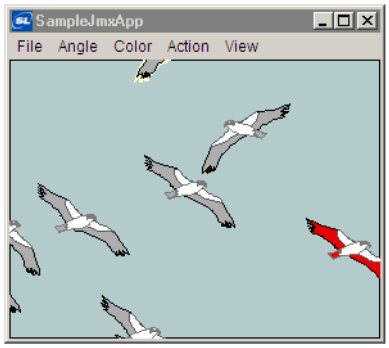
4. Click **OK**.

5. In the **Object Properties** dialog:

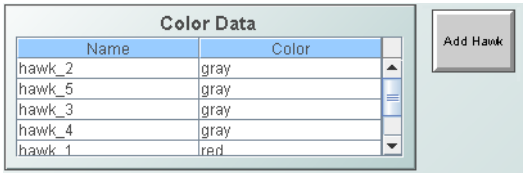
   label (category: Label) - Change to Add Hawk.

6. Double-click on the Add Hawk label. A new hawk appears in the sample JMX application.

7. Double-click again on the Add Hawk button. Each time you double-click the button another hawk is added to the sample JMX application.



Another hawk is then reflected in the table.



You are now ready to setup the drill down.

Create a Drill Down Target in the Table

In this exercise, you create the drill down using the previously created display, jmx_dd_qs.rtv, as the target.

1. Select the Color Data table.
2. In the **Object Properties** dialog:

   drillDownTarget (category: Interaction) - Double-click in the Property Name field to bring up the Drill Down Properties dialog.
3. In the "Drill Down Properties" dialog:

   Apply Drill Down To - Select **Named Window** from the drop down menu. This option lets you re-use the window when you drill down multiple times.

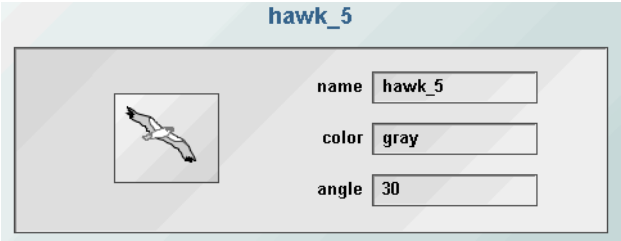
   Window Name - Enter **jmx**. This name should be unique unless the display is to open in an existing window.

   Drill Down Display Name - Select **dstutorial\jmx_dd_qs.rtv** from the drop down menu.
4. Click OK to attach the drill down target and close the dialog.

View the Drill Down Display

In this exercise, you drill down to the target display.

1. Double-click on any cell in the table to drill down to detailed data. The target display opens.



2. Double-click on another cell in the table and the same display is used to show different data based on the row you select.

3. Close the drill down window.

4. In the Display Builder select **File>Save**.

Go to the main "Quick Start Tutorial"

Sample JMX Application

A sample JMX Application is provided to enable customers to see how to monitor or execute commands on a JMX Application without using their environment. This application uses the generally accepted conventions for MBean architecture. It contains an agent that registers the MBean with the MBean server, an MBean interface that refines all of the methods, and a manager bean which implements all of the MBean methods. The sample JMX Application contains multiple MBeans as described below. All attributes can be queried and are also sent as notifications. Because the hawks are constantly moving, the location attribute notifications are not sent by default. They can be enabled using -notifyMovements: command line argument, specifying the number of milliseconds between notifications. For example -notifyMovements:5000 will send notifications every 5 seconds.

The sample JMX application contains one MBean named **SampleJmxApp:name=Manager** with the following methods:

Method / Attribute	Type	Description
AllData	TabularData	One row for each hawk, containing the angle, color and name of the hawk.
AllHawkFlapMode	boolean[]	Flap mode of each hawk (true = flap, false = soar).
AllHawkPos	TabularData	One row for each hawk, containing the flap mode, name, X and Y of the hawk.
AllHawkX	double[]	X location of each hawk.
AllHawkY	float[]	Y location of each hawk.
Angle	int	Angle of the selected hawk.
AngleList	int[]	List of all possible hawk angles.
Color	String	Color of the selected hawk.
ColorList	String[]	List of all possible hawk colors.
HawkPos	CompositeData	Flap mode, X and Y location of the selected hawk.

HawkX	double	X location of the selected hawk.
HawkY	float	Y location of the selected hawk.
Name	String	Name of the selected hawk.
SelectedData	CompositeData	Angle, color and name of the selected hawk.

The **SampleJmxApp:name=Manager** MBean also supports the following commands:

Method / Attribute	Description
accelerate	Accelerates all hawks.
addHawk	Adds a hawk.
causeException	Throws exception.
decelerate	Decelerates all hawks.
deleteHawk	Deletes the selected hawk.
resume	Resumes animation of all hawks (after suspend).
setAngle	Sets the angle of the selected hawk.
setColor	Sets the color of the selected hawk.
setName	Sets the name of the selected hawk.
suspend	Suspends animation of all hawks.

An additional MBean is created for each hawk, named **SampleJmxHawk_hawkName** with the following attributes:

Method / Attribute	Type	Description
AllData	TabularData	One row, containing the angle, color and name of the hawk.
Angle	int	Angle of the hawk.
Color	String	Color of the hawk.
HawkX	double	X location of the hawk.
HawkY	float	Y location of the hawk.
Name	String	Name of the hawk.

The **SampleJmxHawk_hawkName** also supports the following commands:

Method / Attribute	Description
flap	Sets the wing mode to flap.
setAngle	Sets the angle of the hawk.
setColor	Sets the color of the hawk.
soar	Sets the wing mode to soar.

An additional MBean is created for each gull, named **SampleJmxGull_gullName** with the following attributes:

Method / Attribute	Type	Description
AllData	TabularData	One row, containing the angle, color and name of the gull.
Angle	int	Angle of the gull.
GullX	double	X location of the gull.
GullY	float	Y location of the gull.
Name	String	Name of the gull.
Shape	String	Color of the gull.

The **SampleJmxGull_gullName** also supports the following commands:

Method / Attribute	Description
flap	Sets the wing mode to flap.
fly	Sets the wing mode to soar.
setAngle	Sets the angle of the gull.
setColor	Sets the color of the gull.

Running the Sample JMX Application

From an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"), type:

run_samplejmxapp

JMX Data Source Command Line Options

In addition to General Options, the following command line arguments are enabled with the JMX data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: **"-sub:$data:my Data"**).

Command Line Arguments

-jmxautodiscover

Automatically discover local MBean Servers once each poll interval.

Note: Poll interval is determined by either the **Default Poll Interval** or according to the **General Update Period** specified in **Application Options** on the "General Tab". See "Application Options - JMX" for more information.

Note: You must include the **tools.jar** file (from your JDK installation) in "RTV_USERPATH".

Example:

-jmxautodiscover

Filter auto-discovered connections (i.e. **-jmxautodiscover:XXX**, where **XXX** is the filter). The filter can be a single value or a ; (semicolon) delimited list of values.

Example:

-jmxautodiscover:com.sl;org.apache

This would limit auto-discovered connections to those that start with **com.sl** or **org.apache**.

-jmxconn

Define a JMX connection. Specify the connection name, host, port, user name and password.

Example:

-jmxconn:Myconnection host1 9998

-jmxdsCheckValidAttributeNames

Specifies to only use column names that the MBean recognizes. The given string array of column names is filtered by comparison against the most recent MBean list of valid column names. JMX MBeans are free to support or not support an attribute at any time.

Example:

-jmxdsCheckValidAttributeNames

Note: The **jmxdsCheckValidAttributeNames** option can also be specified in the initialization file **JMXPTIONS.ini** file. Options specified using command line arguments override values saved in initialization files.

-jmx_mbeans_change_dynamically:(true or false)

The **jmx_mbeans_change_dynamically** property works in conjunction with the **jmx_optimize_mbics** property (below) to optimize MBean Info Cache loading. The default setting is true.

MBean Info Caches are a set of local caches created by the RTView JMX data source. The JMX data source creates a cache for each MBean. Each cache contains data the MBean provides to users, as well as mapping information to corresponding Data Objects. The JMX data source uses these local caches to retrieve data updates.

Example:

-jmx_mbeans_change_dynamically:true

There are three possible configurations using these two properties:

1. **jmx_optimize_mbics = false, jmx_mbeans_change_dynamically = true**

If your JMX performance is sufficient, this configuration is appropriate. This is the default setting, and the method RTView has historically used. This method uses the Polling Data Mode to obtain data from the JMX Data Source and update MBean Info Caches. Updates occur as specified by the RTView Update Period.

There are two potential performance downsides in polling mode that the two properties can mitigate. With the polling mode: A) The JMX Data Source updates the MBean Info Caches even if it is not time to update the polled data. If the RTView Update Period is 10 seconds and the JMX Metrics Period is 30 seconds, there would be three updates for every polled data loading. The first two MBean Info Cache updates would happen and then the JMX data source would notice it was not time to update the polled data—thereby performing two unnecessary updates. The third update would coincide with the JMX Metrics Period and the MBean Info Caches would again be updated followed by a data update. And B) If there are no JMX Data Objects that need to be notified, the MBean Info Cache updates are unnecessary.

2. **jmx_optimize_mbics = true, jmx_mbeans_change_dynamically = true**

This configuration reduces update traffic by setting the MBean Info Cache updates to occur only just before the data is about to be updated. Use this setting if you see updates to the MBean Info Caches on the JMX Update period that do not get followed with a data update (Inefficiency A described above).

3. **jmx_optimize_mbics = false or true, jmx_mbeans_change_dynamically = false**

This provides the maximum in MBean Info Cache loading optimization as no periodic MBean Info Cache updates occur. Use this configuration if (a) your installation is not using Notified Data Objects or, (b) you are using Notified Data Objects but the beans are static. That is, the data on the beans may change but the beans themselves do not, thereby mitigating the need for MBean Info Cache updates (the code does initially fill the MBean Info Caches but their value remains static).

Note: When **jmx_mbeans_change_dynamically** is set to false, the setting for **jmx_optimize_mbics** is ignored. **jmx_mbeans_change_dynamically** overrides **jmx_optimize_mbics**.

Note: The **jmx_mbeans_change_dynamically** option can also be specified in the initialization file **JMXPTIONS.ini**. Options specified using command line arguments override values saved in initialization files.

-jmx_metrics_period

Enter the time in milliseconds to control how often MBean methods are executed. Default is 0, which executes MBean methods according to the update period specified on the Application Options General tab.

Note: Because the MBean Method Execution Interval is superseded by the General Update Period, the amount of time elapsed between method executions may be longer than the value entered. For example, if the General Update Period is 2000 milliseconds and the MBean Method Execution Interval is 5000 milliseconds, MBean methods will be executed every six seconds.

Example:

-jmx_metrics_period:5000

-jmx_minreconnecttime:(seconds)

Enter the minimum number of seconds that will elapse before attempting to reconnect. Default is 30.

Example:

-jmx_minreconnecttime:15

-jmx_optimize_mbics:(true or false)

The **jmx_optimize_mbics** property works in conjunction with the **jmx_mbeans_change_dynamically** property to optimize MBean Info Cache loading. The default setting is false. For details, see **jmx_mbeans_change_dynamically**.

Example:

-jmx_optimize_mbics:false

Note: The **jmx_optimize_mbics** option can also be specified in the initialization file **JMXPTIONS.ini**. Options specified using command line arguments override values saved in initialization files.

-jmx_use_multiple_threads:(true or false)

Enable multiple threads for commands and polled queries. Default is **false**. This option enhances performance in the case where some JMX servers are slow, thus preventing a single server (or servers) from delaying updates from all JMX servers.

Note: RTView will create one thread per connection, so this is not a practical option for cases where hundreds of connections are defined.

If true, then the thread that created the JMX connection will be used for all commands and polled queries on that connection. The polled queries will initiate based on the Poll Interval for the data attachment and each connection will update their listeners asynchronously when data becomes available.

If false, then polled queries are executed synchronously and the listener is not updated until the data from all connections in the data attachment has returned.

Example:

-jmx_use_multiple_threads:true

Note: The **jmx_use_multiple_threads** option can also be specified on the "JMX Administration Tab" of the **Application Options** dialog and saved to the **JMXOPTIONS.ini** file. Options specified using command line arguments override values saved in initialization files.

Apache log4j Data Source

Apache log4j is a widely used, open source, Java logging utility used to instrument Java applications. It can be configured to output selected information in a variety of forms and this output can be used to monitor application health, progress or problems. In addition to producing textual logs, Apache log4j can generate real time logging events.

With RTView, you can connect to a configured application instrumented with Apache log4j and receive generated logging events in real time. You can then use RTView to build sophisticated monitoring and management applications that allow you to:

- Present real-time business content included in Apache log4j logging events,
- Archive business metrics in the RTView Historian for trend analysis, and
- Provide a look at application health and troubleshooting by analyzing message content.

This section includes:

System Requirements - Apache log4j

In addition to basic "System Requirements", this data source requires Apache log4j. See the **README_sysreq.txt** file in your installation's home directory for the current version(s) supported.

Setup - Apache log4j

In addition to general environment variables (see "Setup"), this data source includes the **log4j.jar** file on the RTV_HOME classpath.

For an instrumented application to be monitored by the Apache log4j data source, the application must be configured (using the standard log4j configuration mechanism - typically a properties or XML file) to use the built in SocketHubAppender on a known port. RTView will then use a named connection to connect to the specified port for the SocketHubAppender.

Note: Due to the fact that communication to the log4 SocketHubAppender is via sockets on a known port, firewalls may have to be configured to allow access to that port.

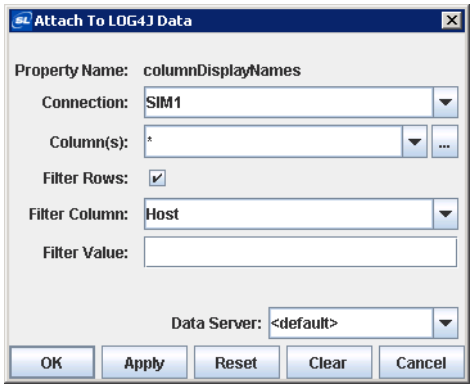
Attach to Log4j Data

Note: The Apache log4j data source may not be licensed in your RTView installation.

From the **Object Properties** window you can access the **Attach to LOG4J Data** dialog, which allows you to choose a connection that connects to a stream of log4j logging events, generated by a configured log4j instrumented Java application. The received logging events can be filtered (client side row and column filtering) using standard client side fields.

Once an object property has been attached to a log4j logging event, it receives continuous updates. It updates whenever a new logging event is received.

Right-click on the Property Name from the **Object Properties** window and select **Attach to Data>LOG4J** to display the **Attach to LOG4J Data** dialog. The **Attach to LOG4J Data** dialog provides several drop down menus that allow you to specify information. If the drop down menu does not contain the item you require, type your selection into the text field.



Field Name	Description
Connection	The connection name. You may define a connection on the "LOG4J Connections Tab" of the **Application Options** dialog. Enter * to aggregate log4j logging events from all defined connections.
Column(s)	Select which columns to display. To bring up the **Select Column(s)** dialog, click on the ellipses button ... in the **Column(s)** field or right-click in the **Column(s)** field and click **Select Column(s)**. This dialog should contain a list of **Available Column(s)** you can add to your table.
Filter Rows	Check box to indicate whether or not to filter rows in the log4j logging event. See "Row Filtering" for more information.
Filter Column	Name of the log4j logging event column to use as a filter. Multiple column names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col 3). If your column name contains a space or a semicolon, then the entire name must be enclosed in single quotes.
Filter Value	Value that the Filter Column must equal. Multiple filter values should be entered as a nested list, where values for a given column are separated by commas within a semicolon (;) delimited list (i.e. val1,val2;val3,val4;val5,val6). If your filter value contains a space or a semicolon, then the entire value must be enclosed in single quotes. When * is entered as a filter field value, data for all values in the specified filter column will be used to update the object property. When "*" is entered, only the literal comparative value will be used. These are only allowed for objects which display tabular data.

Data Server	Select to read data through your configured Data Server and not directly from the Apache log4j data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab". **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a Named Data Server that you configured in **Application Options>**"Data Server Tab". **Multi-Server Attachment** - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more **Named Data Servers** (e.g. ds101;ds102). Each name specified must correspond with a Named Data Server that you configured in **Application Options>Data Server**. It is also possible to specify **__default** and **__none** (e.g. __default;ds101;ds102). **Note:** The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**. When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received. A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways: **1.** The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows. **2.** The multi-server attachment can be applied as the **Table** argument of the RTView function named Combine Multi-Server Tables. See "Tabular Functions" for more information.

When an object property has been attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing values from the **Object Properties** window is no longer possible. To remove the data attachment and resume editing capabilities in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information entered into the dialog is validated against the selected server.

Note: Filters are not validated.

Blue	Unknown	Cannot validate entry.
White	Valid state	Entry is valid.
Red	Invalid state	Incomplete or invalid entry.

Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. A generic connection such as **$conn** is used instead of a specific connection. Later when the display is running, this generic value is defined by the actual name of a specific connection, such as **samplelog4japp**. In this way, a single display can be reused to show data from a number of different sources. Substitutions can be used in any field in this dialog. For more information on creating displays using substitution values, see "Substitutions".

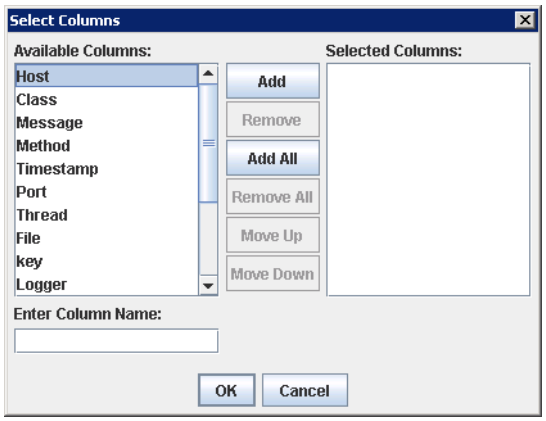
Select Column(s)

From the **Attach to Data** dialog you can specify which columns to search in log4j logging event, as well as which columns to display in your resulting data and in what order they will appear. To view the list of **Available Column(s)**, click on the ellipses button ... to open the **Select Column(s)** dialog.

To add a column, select an item from the **Available Column(s)** list and click on the **Add** button. If the item you require is not listed, type your selection into the **Enter Column** field. Click the **Remove** button to delete an item previously added to the **Selected Column(s)** list. You can control the order of fields in a table by arranging the items in the **Selected Column(s)** list with the Move Up and Move Down buttons.

Validation colors indicate whether selected columns are valid. However if even one column selected is invalid, the Column(s) field in the **Attach to LOG4J Data** dialog will register as an invalid entry.

If no data is available for a table row within a selected column, the table cell will display one of the following values: **N/A**, **false**, **0**, or **0.0**.



The following describes **Attach to LOG4J Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied.

Application Options - Apache Log4j

To access the **Application Options** dialog, in the Display Builder select **Tools>Options**. There are two Log4j options tabs: LOG4J Connections and LOG4J Options.

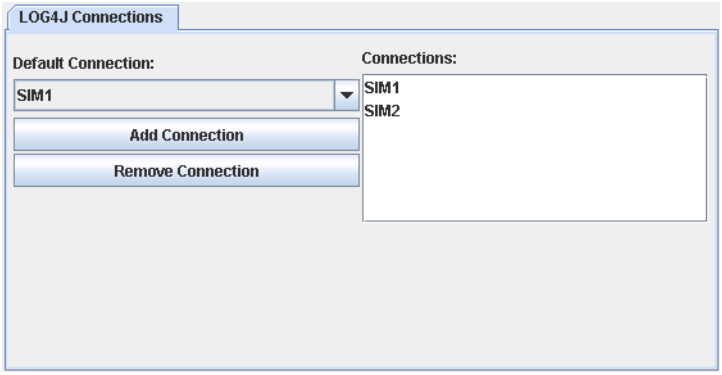
Options specified in the LOG4J options tabs can be saved in an initialization file (**LOG4JOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server and Historian to set initial values. If no directory has been specified for your initialization files and LOG4JOPTIONS.ini is not found in the directory where you started the application, then RTView will search under lib in your installation directory. See "RTV_JAVAOPTS" for more information.

LOG4J Connections Tab

This tab allows you to add or remove connections and set your default connection.

When you add an Apache log4j connection to the list it will be highlighted in yellow indicating that RTView has not connected to it. To attempt to connect, click **OK**, **Apply**, or **Save**. If the background remains yellow, then RTView was unable make a connection. Check that your log4j connection was setup correctly.

Note: Regardless of which tab you are currently working from in the Application Options dialog, RTView will attempt to connect to all unconnected connections each time you click **OK**, **Apply**, or **Save**.



Field Name	Description
Default Connection	Name of connection used as the default for data attachments. Select from drop down menu to change default setting.
Add Connection	Click to open the **Add LOG4J Connection** dialog. To edit, select a connection from the list and double-click. Connections that are updating objects in a current display cannot be renamed.

Connection Name - Name to use when referencing this Apache log4j connection in your data attachments.

Source - Event source of the connection. Choose SocketAppender or SocketHubAppender.

SocketAppender - Connect to port on local host.

SocketHubAppender - Connect to port on remote host.

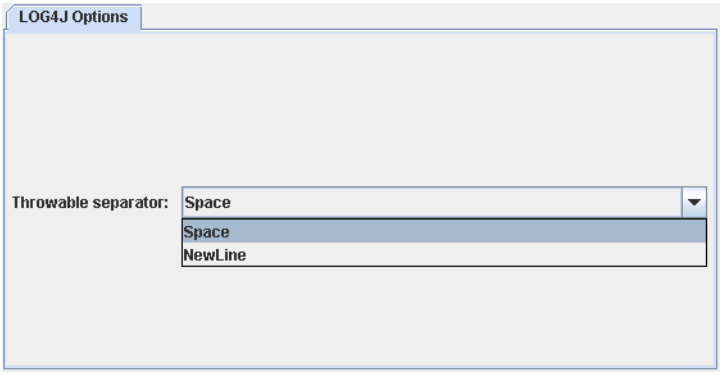
Host - Host name or IP address. **Note:** If the Source selected is SocketAppender, this field auto-populates with IP address of local host (127.0.0.1).

Port - Port number of the connection.

Remove Connection	Select a connection from the list and click Remove Connection to delete. Connections that are updating objects in a current display cannot be removed.

LOG4J Options Tab

This tab allows you to define a separator between multi-element Throwable information.



Field Name	Description
Throwable Separator	Choose a separator between elements: Space or NewLine. **Space** - Use a Space as a separator between elements. The resulting field will be a single line value. **NewLine** - Use a NewLine as a separator between elements. The resulting field will be a multi-line value.

Deployment - Apache Log4j

This page contains details about the deployment process that are specific to your data source. Please go to the Deployment section of this documentation for instructions on how to implement your RTView deployment option. Return to this page whenever you are instructed to refer to deployment information that is specific to your data source.

System Requirements and Setup

The Apache log4j data source has additional System Requirements and Setup. See "System Requirements - Apache log4j" and "Setup - Apache log4j" for more information.

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options", "Application Options - Apache Log4j", and "RTV_JAVAOPTS" for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
LOG4JOPTIONS.ini	Contains data source options for log4j.

Note: Options specified using command line parameters override values set in these initialization files.

Round Robin Database Data Source

Round Robin Database (RRD) is an Open Source, high-performance data logging system for time-series data. Many IT departments use RRD to log time-stamped information concerning the status of enterprise applications. The resulting data is often used to debug, determine alert conditions, and analyze performance management and capacity planning.

This section includes:

Round Robin Database (RRD) Requirements and Setup

System Requirements

In addition to basic "System Requirements", the RRD data source requires that you have RRDTool and a Round Robin Database installed. RRDTool is an open source application that you can download from *http://oss.oetiker.ch/rrdtool*. RTView supports RRDTool on Windows and Linux. See the **README_sysreq.txt** file in your installation's home directory for the current version(s) supported.

Attach to RRD Data

Note: The Round Robin Database data source may not be licensed in your RTView installation.

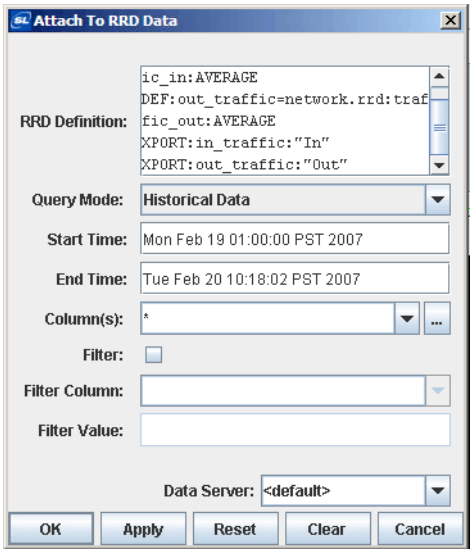
The **Attach to RRD Data** dialog, which can be accessed from the **Object Properties** window, is used to construct an RRDTool command to query your Round Robin Database. An object will receive continuous updates once it has been attached to your database.

When an object property is attached to data, the **Property Name** and **Value** in the **Object Properties** window will be displayed in green. This indicates that editing this value from the **Object Properties** window is no longer possible. To remove the data attachment and resume editing capabilities in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the database when the Property Name and Value are no longer green.

Right-click on the Property Name from the **Object Properties** window and select **Attach to Data>RRD** to display the **Attach to RRD Data** dialog. The **Attach to RRD Data** dialog provides drop down menus and an optional filter field that allow you to specify information that will be used to create a query for your RRDTool in the following format:

rrdtool xport --start <Start Time> --end <End Time> <RRD Definition>

To specify a step, add this as a command line argument in the **RRDTool Executable Name** in the **Application Options** tab. See "Application Options - Round Robin Database" for more information.



Field Name	Description
RRD Definition	An RRDTool definition that returns one or more columns. These columns can either be values from the database or calculated values. Refer to the rrdxport section of the RRDTool documentation for information on syntax. The definition text is not validated. See the rrdcreate section of the RRDTool documentation for information on creating data in your Round Robin Database.
Query Mode	Specifies the type of data to be retrieved. Select one of the three query options: **Note:** In all query modes, if a date is left blank the current date is used. For example, if the end date is left blank in a Historical Data query, data will be returned from the start date to the present. Because the query is for historical data only, the data is retrieved only once. To cause the data to update as new values are added, specify a Historical and Current Data query mode instead. **Historical Data --** Only historical data is returned, from the start date to the end date. **Historical and Current Data** -- Historical data from the start date to the present is returned, and new data is returned as it is added to the database. The end date is ignored for this query mode. **Current Data** -- The most current value for the point is returned from the Round Robin database. New values are returned as data is added to the database. Both the start and end dates are ignored for this query mode.
Start Time	Enter the start time for the query. Use either a formatted time string or a Unix standard time argument in seconds. The Unix standard time argument is the number of seconds since midnight January 1, 1970.
End Time	Enter the end time for the query. Use either a formatted time string or a Unix standard time argument in seconds. The Unix standard time argument is the number of seconds since midnight January 1, 1970.
Column(s)	Select which column(s) to display.

Filter	Check box to indicate whether or not to filter RRD data. See "Row Filtering" for more information.
Filter Column	Name of the column to use as a filter. Multiple column names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col 3). If your column name contains a space or a semicolon, then the entire name must be enclosed in single quotes.
Filter Value	Value that the Filter Column must equal. Multiple filter values should be entered as a nested list, where values for a given column are separated by commas within a semicolon (;) delimited list (i.e. val1,val2;val3,val4;val5,val6). If your filter value contains a space or a semicolon, then the entire value must be enclosed in single quotes. When * is entered as a filter field value, data for all values in the specified filter column will be used to update the object property. When "*" is entered, only the literal comparative value will be used. These are only allowed for objects which display tabular data.
Data Server	Select to read data through your configured Data Server and not directly from the RRD data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab". **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a Named Data Server that you configured in **Application Options>**"Data Server Tab". **Multi-Server Attachment** - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more Named Data Servers (e.g. **ds101**;**ds102**). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify __**default** and __**none** (e.g. __**default**;**ds101**;**ds102**). **Note:** The values __**default** and __**none** begin with two underscore characters. Alternatively, a value of ***** can be entered to specify all data servers, including __**default** and __**none**. When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received. A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways: **1.** The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows. **2.** The multi-server attachment can be applied as the **Table** argument of the RTView function named **Combine Multi-Server Tables**.

Substitutions

Substitutions allow you to build open-ended displays in which data attachments depend on values defined at the time the display is run. For example generic values, such as $start and $end, are used instead of specific values. Later when the display is running, these generic values are defined by the actual values, such as January 1, 2007. In this way, a single display can be reused to show data from a number of different queries. For more information on creating displays using substitution values, see "Substitutions".

Select Table Columns

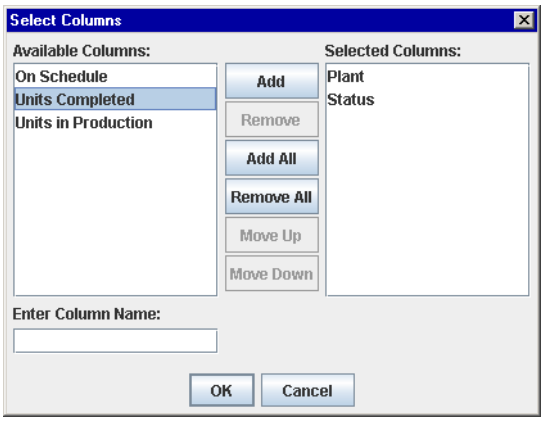
From the **Attach to RRD Data** dialog, you can specify which table columns to display and in what order they will appear. In order to populate the listing of available columns, you must first select a valid database and table.

To bring up the **Select Columns** dialog, click on the ellipsis button ... in the **Column(s)** field (or right-click in the **Column(s)** field and click on **Select Columns**). The dialog should contain a list of Available Columns that you can add to your table.

To add a column, select an item from the **Available Columns** list and click on the **Add** button. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of columns in a table by arranging the items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

Validation colors indicate whether selected columns are valid. However, if even one column selected is invalid the **Column(s)** field in the **Attach to RRD Data** dialog will register as an invalid entry.

Note: Invalid columns will not update.



The following describes the **Attach to RRD Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.

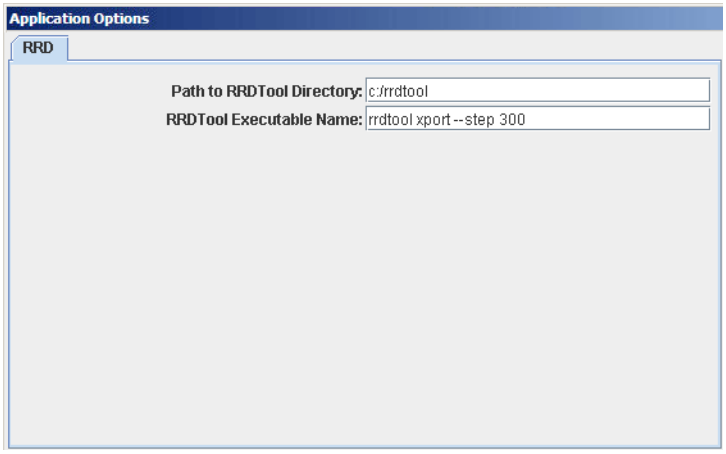
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from database (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Application Options - Round Robin Database

Select **Tools>Options** in the Display Builder to access the **Application Options** dialog.

Options specified in the **RRD Options** tab can be saved in an initialization file (**RRDOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server and Historian to set initial values. If no directory has been specified for your initialization files and **RRDOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

Note: Options specified using command line arguments will override values set in initialization files. See "RRD Data Source Command Line Options" for more information.



Field Name	Description
Path to RRDTool Directory	The path to the directory containing the RRDTool executable. If not specified, **c:\rrdtool** will be used on windows and **/usr/bin/rrdtool** will be used on Unix.
RRDTool Executable Name	The name of the RRDTool executable. This allows the use of customized and renamed RRDTool executables. In addition to specifying the executable name, RRDTool command line options can also be specified. If not specified, **rrdtool xport --step 300** will be used.

RTView Deployment - RRD

This section contains details about the deployment process that are specific to your data source. Please go to the Deployment section of this documentation for instructions on how to implement your RTView deployment option. Return to this page whenever you are instructed to refer to deployment information that is specific to your data source.

System Requirements and Setup

The RRD data source has additional System Requirements and Setup. See "Round Robin Database (RRD) Requirements and Setup" for more information.

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options", "Application Options - Round Robin Database", and "RTV_JAVAOPTS" for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
RRDOPTIONS.ini	Contains data source options for RRD.

Note: Options specified using command line parameters override values set in these initialization files.

Round Robin Database Demos

Except where noted, all demos can be run in three ways, as an application, or via rich or thin client in a browser.

Before You Begin

Start the Demo Server

Thin Client Demo only.

There is a thin client demo already installed on the "RTView Demo Server".

In an initialized command window (see "Initializing a Command Prompt or Terminal Window"):

Type **run_startup_demoserver**

Data Source Demo

The Data Source Demo is designed to illustrate each data source.

1. **Setup the Sample Database:**

Round Robin Databases are platform specific, so a sample database is provided for Windows and Linux i386 only. See the Round Robin Database documentation for information on how to port this database to another platform. On Windows, rename demos\dstutorial\network_win.rrd to network.rrd. On Linux i386, renamed demos/dstutorial/network_rhel5.rrd to network.rrd.

2. Run Demos - Application or Thin Client Browser

Application Demo

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
2. To view the demo, type:

   run_viewer
3. To edit the demo, type:

   run_builder

Thin Client Browser Demo

"Start the Demo Server" if it is not running.

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
2. Start the Display Server by typing:

   run_displayserver
3. Open a browser and navigate to **http://localhost:8068/dstutorial**.

RRD Data Source Command Line Options

In addition to General Options, the following command line arguments are enabled with the RRD data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: "**-sub:$data:my Data**").

Name	Description
-RRDds.rsh:(host)	Specify a Windows machine (host) on which to run the RRD data source remotely from Unix. **Note:** The host must have RRDTool installed and an rshd running. Example: **-RRDds.rsh:myrrdserver**

RTVAgent Data Source

Note: The RTVAgent data source may not be licensed in your RTView installation.

The RTVAgent data adapter accepts connections from remote agent applications and publishes that data to an RTView application (i.e. Display Builder, Data Server, Display Server, Historian or Display Viewer Application). Once a connection is established, objects (typically Cache objects) can be configured with data tables received from agents via the "Attach to RTVAgent Data" dialog.

The RTVAgent data adapter is ideal when there is a variable or unknown number of applications that wish to publish data to be consumed and processed by a central RTView Data Server. Once the RTVAgent data adapter input port is defined and enabled, the Data Server can begin receiving data from any number of RTVAgents. This data can be received via socket, http, or https, which means that RTVAgent applications can publish data from within or outside of the firewall.

It is possible to create a custom RTVAgent, an external Java application that will provide data to an RTView application. See "Customization - RTVAgent" for details.

This section includes:

System Requirements and Setup - RTVAgent

System Requirements

The RTVAgent data source has no additional "System Requirements".

Setup

The RTVAgent data source requires no additional "Setup".

Attach to RTVAgent Data

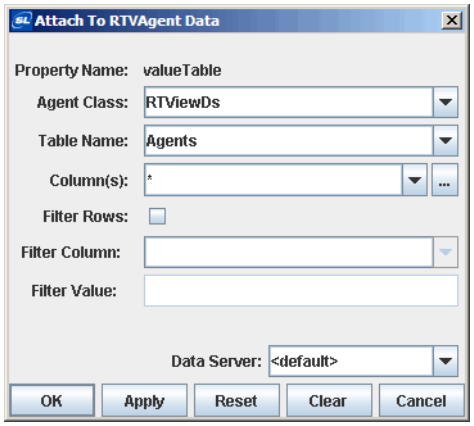
Note: The RTVAgent data source may not be licensed in your RTView installation.

The **Attach to RTVAgent Data** dialog, which is used to connect objects to tabular data sent from remote agent applications, can be accessed from the **Object Properties** window. The **valueTable** property of an object, typically a Cache object, will be attached to a data table received from the agent. See "Caches" for more information.

Note: The **AgentName** column in that table should be specified in the **indexColumnNames** property of the Cache object.

Right-click on the Property Name from the **Object Properties** window and select **Attach to Data>RTVAgent** to display the **Attach to RTVAgent Data** dialog. The **Attach to RTVAgent Data** dialog provides several drop down menus that allow you to specify information. The **Agent Class** drop down menu lists the agent class names of all of agents that have connected to the RTVAgent adapter. Drop down menus for **Table Name**, **Column(s)**, and **Filter Column** populate based on the selected agent Class. If the item you require is not listed, type your selection into the field. For information on enabling a connection, see "Application Options - RTVAgent".

Note: The **Attach to RTVAgent Data** dialog will only display agents currently connected and publishing data.



Field Name	Description
Agent Class	The class name of the agent from which you want content. Choose either a class name or "RTViewDs". You must enable a connection on the "RTVAgent Options Tab" of the **Application Options** dialog. **Note:** Since multiple agents can have the same class name, data attachments will receive information from all agents with the specified Agent Class and Table Name.
Table Name	Name of the table. **Note:** The agent assigns a name to the table when it sends it to the RTVAgent data adapter.

Column(s)

Select which column(s) to display. To display the **Select Columns** dialog, click on the ellipses button ... in the **Column(s)** field (or right-click in the **Column(s)** field and choose **Select Columns**). The dialog should contain a list of Available Columns that you can add to your table.

Filter Rows

Check box to indicate whether or not to filter the RTVAgent data. See "Row Filtering" for more information.

Filter Column

Name of the column to use as a filter. Multiple column names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col3). If your column name contains a space or a semicolon, then the entire name must be enclosed in single quotes.

Filter Value

Value that the **Filter Column** must equal. Multiple filter values should be entered as a nested list, where values for a given column are separated by commas within a semicolon (;) delimited list (i.e. val1,val2;val3,val4;val5,val6). If your filter value contains a space or a semicolon, then the entire value must be enclosed in single quotes.

When ***** is entered as a filter field value, data for all values in the specified filter column will be used to update the object property. When "*****" is entered, only the literal comparative value will be used.

Data Server

Select to read data through your configured Data Server and not directly from the RTVAgent data source.

Default - Select the default Data Server you configured in the **Application Options>**"Data Server Tab".

None - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source.

Named Data Servers - Select a **Named Data Server** that you configured in the **Application Options>**"Data Server Tab".

Multi-Server Attachment - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more **Named Data Servers** (e.g. ds101;ds102). Each name specified must correspond with a Named Data Server that you configured in the **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**).

Note: The values **__default** and **__none** begin with two underscore characters.

Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received.

A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways:

1. The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows.
2. The multi-server attachment can be applied as the **Table** argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

When an object property is attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing this value from the **Object Properties** window is no longer possible. Once a property has been attached to data, it receives continuous updates. To remove the data attachment, and resume editing capability in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information entered into the dialog is validated against data received from agents.

The following describes the significance of the Attach to RTVAgent Data validation colors:

Blue	Unknown	Cannot validate entry.
White	Valid state	Entry is valid.
Red	Invalid state	Incomplete or invalid entry.

*If an agent is validated as Unknown, RTView will attempt to read it when you click **OK** or **Apply**.

Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. A generic name such as $table is used instead of a specific table name. Later when the display is running, this generic value is defined by the actual name of a specific table, such as SalesTable. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

RTViewDs

There is a predefined agent Class named **RTViewDs** that defines a single table named **Agents**. The **RTViewDs.agents** table contains a row for each agent that is currently connected. The columns in the table are as follows:

RTViewDs		
Agents (RTViewDs.agents)	**Column Available**	**Description**
	AgentName	Name of agent.
	AgentClass	Class name of agent.
	LastReceiveTime	Time that a data table was most recently received from the agent.
	TotalRowsReceived	Cumulative count of table rows received from the agent.
	ClientID	Unique number assigned to the agent connection.

Select Table Columns

From the **Attach to RTVAgent Data** dialog you can specify which table columns to display and in what order they will appear. In order to populate the listing of available columns, you must first select a valid agent Class source and Table Name.

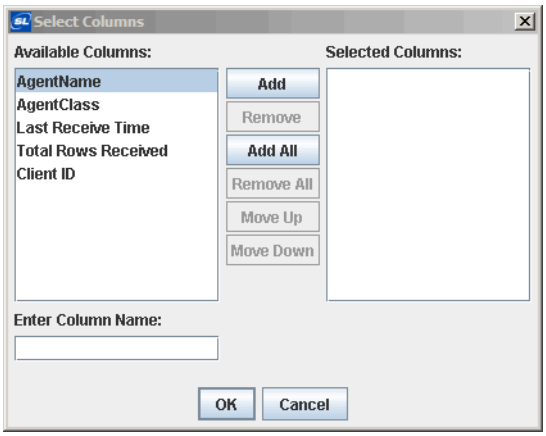
Click on the ellipses button ... in the **Column(s)** field (or right-click in the **Column(s)** field and choose Select Columns) to display the **Select Columns** dialog. The dialog should contain a list of Available Columns that you can add to your table.

To add a column, select an item from the **Available Columns** list and click on the **Add** button. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of columns in a table by arranging the items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

Validation colors indicate whether selected columns are valid. However, if even one column selected is invalid, the **Column(s)** field in the **Attach to RTVAgent Data** dialog will register as an invalid entry.

Note: Invalid columns will not update.

If no data is available for a table row within a selected column, the table cell will display one the following values: **N/A**, **false**, **0**, or **0.0**.



The following describes the **Attach to RTVAgent Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Application Options - RTVAgent

Select **Tools>Options** in the Display Builder to display the **Application Options** dialog.

Options specified in the **RTVAgent** tab can be saved in an initialization file (**RTVAGENTOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server, and Historian* to set initial values. If no directory has been specified for your initialization files and **RTVAGENTOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under lib in your installation directory. See "RTV_JAVAOPTS" for more information.

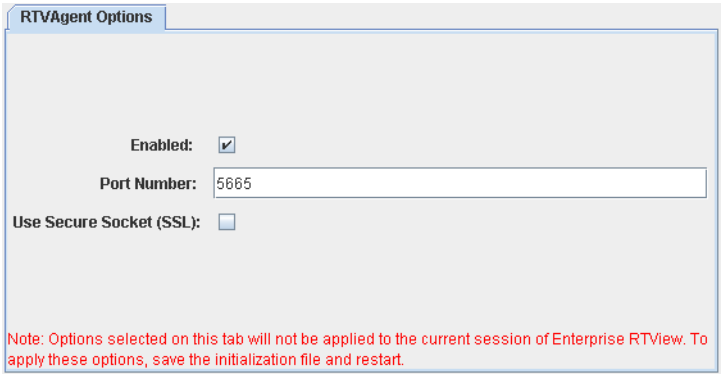
Note: Options specified using command line arguments will override values set in initialization files. See "Command Line Options - RTVAgent" for more information.

---

*If you are enabling RTVAgent options to utilize the Historian for Alert Persistence, then you must copy the **RTVAGENTOPTIONS.ini** file to the directory where you will be running the Historian.

RTVAgent Options Tab

This tab allows you to enable connections from remote agent applications.



Field Name	Description
Enabled	Select to enable connections from remote agent applications. By default, this option is disabled. Once enabled, agents can send tabular data and objects can be configured with data attachments using the "Attach to RTVAgent Data" dialog.
Port Number	Port on which the RTVAgent data adapter will accept agent connections. Default is **5665**. Valid port numbers are greater than **1024**. **Note:** Only one application at a time can accept connections on a given port.
Use Secure Socket (SSL)	If selected, a secure socket will be used for each agent that makes a direct socket connection to the RTVAgent data adapter. Data is encrypted before transmission over a secure socket. **Note:** This option does not apply to agents that connect indirectly via HTTP and the rtvagent servlet. For secure connections in that case, an HTTPS connection should be used from the agent to the rtvagent servlet.

RTView Deployment - RTVAgent

This section contains details about the deployment process that are specific to your data source. Please go to the Deployment section of this documentation for instructions on how to implement your RTView deployment option. Return to this page whenever you are instructed to refer to deployment information that is specific to your data source.

Note: If you plan to set up the Data Server to accept RTVAgent data via HTTP, then complete the section titled **Set up RTVAgent Servlet on Application Server.**

This section includes:

System Requirements and Setup

The XML data source has additional System Requirements and Setup. See "System Requirements and Setup - RTVAgent" for more information.

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options", "Application Options - RTVAgent", and "RTV_JAVAOPTS" for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
RTVAGENTOPTIONS.ini	Contains general options as well as data source options for RTVAgent.

Note: Options specified using command line parameters override values set in these initialization files.

Command Line Options - RTVAgent

In addition to General Options, the following command line arguments are enabled with the RTVAgent data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: "**-sub:$data:my Data**").

Name		Description
-rtvagentoption	**enabled**	Enable connections from remote agent applications. Default is **false**. Example: **-rtvagentoption:enabled=true**
	port	Specify port on which the RTVAgent data adapter will accept agent connections. Default is **5665**. Valid port numbers are greater than 1024. **Note:** Only one application at a time can accept connections on a given port. Example: **-rtvagentoption:port=5665**
	ssl	Select to encrypt data before transmission over a secure socket. Default is **false**. **Note:** This option does not apply to agents that connect indirectly via HTTP and the rtvagent servlet. For secure connections in that case, an HTTPS connection should be used from the agent to the rtvagent servlet. Example: **-rtvagentoption:ssl=true**

RTV HTTP Data Source

Note: The RTV HTTP data source may not be licensed in your RTView installation.

The RTView HTTP Data Adapter is an extensible data adapter that allows you to process data (that has been posted to a particular URL) using a defined handler. You enable and specify the root and handler URL via "Application Options - RTV HTTP", and then process the data using the handler defined in "Attach to RTVHttp Data".

You can use the default handlers, which can be added by copying the **RTVHTTPOPTIONS.ini** file (in the **custom/rtvhttp** directory) to your working directory or to the **lib** directory (which is located in your installation directory). See the **README** file in the **custom/rtvhttp** directory for more information on using the handlers provided with the RTView HTTP Data Adapter.

This section includes:

System Requirements and Setup - RTV HTTP

System Requirements

The RTV HTTP data source has no additional "System Requirements".

Setup

In order to use the pre-defined default handlers, perform the following steps:

1. Go to the **custom / rtvhttp** directory and copy the **RTVHTTPOPTIONS.ini** to your working directory (or to the **lib** directory (which is located in your installation directory) so that it will work from any directory in which you are working).

2. To verify that the defaults are displaying properly, run the display builder and select **Tools > Options**.

   The **Application Options** window displays.

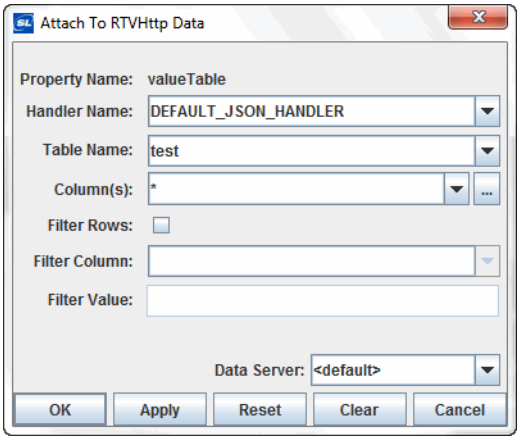
3. Select **RTVHttp** from the left menu.

   The **RTVHttp Handlers** tab displays by default with the **DEFAULT_CACHE_WRITER** and **DEFAULT_JSON_HANDLER** options listed in the **Handlers** list box. **DEFAULT_JSON_HANDLER** is listed as the **Default Handler**. See "Application Options - RTV HTTP" for more information.

See the **README** file in the **custom / rtvhttp** directory for more information. See "Setup" for the basic setup required.

Attach to RTVHttp Data

Right-click on the Property Name from the **Object Properties** window and select **Attach to Data> RTVHttp** to display the **Attach to RTVHttp Data** dialog. The **Attach to RTVHttp Data** dialog provides several drop down menus including the **Handler Name**, which is the name of the handler you want to use to process the data, and the **Table Name**, which is the table from which you want to collect data.



Field Name	Description
Handler Name	Select the handler that you want to use to process the data posted on the pre-defined URL (which was set up in "Application Options - RTV HTTP"). The pre-defined default handlers and any custom handlers you have created display in this drop down list.
Table Name	Select the table name from which you want the handler to collect data.
Column(s)	Name of the column within the table to use as a filter. Multiple column names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col3). If your column name contains a space or a semicolon, then the entire name must be enclosed in single quotes.
Filter Rows	Select this check box to enable the **Filter Column** and **Filter Value** fields.
Filter Column	The name(s) of the column(s) to which the **Filter Value** should be applied.
Filter Value	The filter value(s) to be applied to the filter column(s). Multiple filter values should be entered as a nested list, where values for a given column are separated by commas within a semicolon (;) delimited list (i.e. val1,val2;val3,val4;val5,val6). If your filter value contains a space or a semicolon, then the entire value must be enclosed in single quotes. When ***** is entered as a filter field value, data for all values in the specified filter column will be used to update the object property. When "*****" is entered, only the literal comparative value will be used.
Data Server	Select to read data through your configured Data Server and not directly from the RTV HTTP data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab" **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a Named Data Server that you configured in **Application Options>**"Data Server Tab". **Multi-Server Attachment** - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more Named Data Servers (e.g. ds101;ds102). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**). **Note:** The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**. When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received. A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways: **1.** The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows. **2.** The multi-server attachment can be applied as the **Table** argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

When an object property is attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing this value from the **Object Properties** window is no longer possible. Once a property has been attached to data, it receives continuous updates. To remove the data attachment and resume editing capability in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information entered into the dialog is validated against the data received.

The following describes the significance of the **Attach to RTVHttp Data** validation colors:

Blue	Unknown	Cannot validate entry.
White	Valid state	Entry is valid.
Red	Invalid state	Incomplete or invalid entry.

*If the entry is validated as Unknown, RTView will attempt to read it when you click **OK** or **Apply**.

Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. A generic name such as $table is used instead of a specific table name. Later, when the display is running, this generic value is defined by the actual name of a specific table, such as SalesTable. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

Select Table Columns

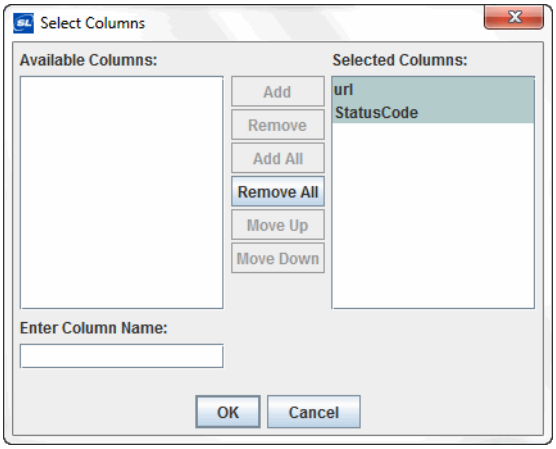
You can specify which table columns to display and in what order they will appear from the **Attach to RtvHttp Data** dialog. Since the incoming data can constantly change, the **Available Columns** region on the **Available Columns** window will not be populated automatically. You either need to know the column names in the incoming data or you need to view the incoming data beforehand to determine the column names.

Once you know the column names in the incoming data, there are two ways to specify the columns that you want to view. You can either enter the column names directly in the **Column(s)** field (using a semi-colon to separate the column names when specifying multiple columns), or you can click on the ellipses button  in the **Column(s)** field (or right-click in the **Column(s)** field and choose **Select Columns**) to display the **Select Columns** dialog. To add a column, type the name of the column you want to add in the **Enter Column Name** field and click the **Add** button. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of columns in a table by arranging the items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

Validation colors indicate whether selected columns are valid. However, if even one column selected is invalid, the **Column(s)** field in the **Attach to RtvHttp Data** dialog will register as an invalid entry.

Note: Invalid columns will not update.

If no data is available for a table row within a selected column, the table cell will be empty or will display one the following values: **N/A**, **false**, **0**, or **0.0**.



The following describes the **Attach to RtvHttp Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Application Options - RTV HTTP

Select **Tools > Options** in the Display Builder to display the **Application Options** dialog, and then select **RTVHttp** in the left menu. There are two tabs available:

- "RTVHttp Handlers Tab": This tab allows you to select, create, update, and remove handlers.
- "RTVHttp Options Tab": This tab allows you to enable the RTV HTTP Data Source, to specify the default port to use, to specify the root URL, and to require HTTPS to be used.

Options specified in the **RTVHttp** tab can be saved in an initialization file (**RTVHTTPOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server, and Historian* to set initial values. If no directory has been specified for your initialization files and **RTVHTTPOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory.

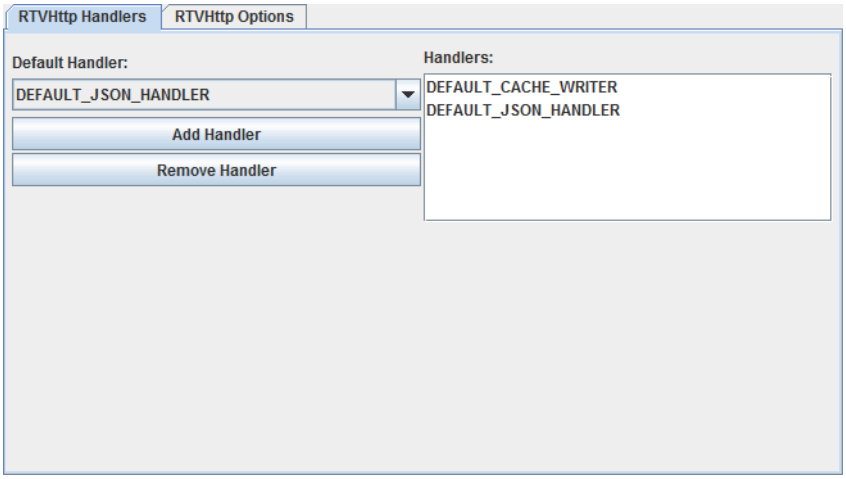
Note: Options specified using command line arguments will override values set in initialization files. See "Command Line Options - RTV HTTP" for more information.

---

*If you are enabling RTVHttp options to utilize the Historian for Alert Persistence, then you must copy the **RTVHTTPOPTIONS.ini** file to the directory where you will be running the Historian.

RTVHttp Handlers Tab

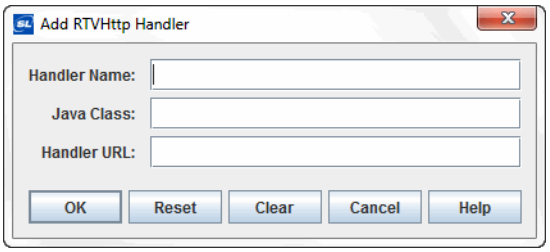
This tab allows you to select a default handler, add a new handler, edit an existing handler, and remove an existing handler.



Field Name	Description
Default Handler	Select the handler that you want to use as the default. The handler selected from this drop down will be the default handler that displays in the "Attach to RTVHttp Data" window. **Note:** This drop down list will default to the handler specified in the **rtvhttpdefaultconn __name** property in the **RTVHTTPOPTIONS.ini** file.

Add Handler

Clicking the **Add Handler** button displays the **Add RTVHttp Handler** window, which allows you to add a handler. Once added, the new handler displays in the **Handlers** display list. This window contains the following fields and buttons:



Handler Name: Specify the name for the new handler.

Java Class: Specify the Java class for the new handler.

Handler URL: Specify the URL for the new handler. This URL is appended to the **Root URL** defined in the "RTVHttp Options Tab" tab to define the complete URL from which the data is collected. For example, if the **Root URL** is defined as **/rtview** and this URL is defined as **/json/data**, then the entire URL would be defined as **/rtview/json/data**.

OK: Saves any changes made to the window.

Reset: Returns all fields to their original settings (when the window was first opened).

Clear: Clears any text in the fields.

Cancel: Closes the window without saving any changes made.

Help: Displays a pop-up window with definitions for the fields on the window.

Remove Handler

Selecting a handler from the **Handlers** display list and then clicking this button removes the handler from the display list.

Handlers

Lists the default handlers and any custom handlers that have been created. Double-clicking on one of the handlers in this display box opens the **Add RTVHttp Handler** window with the currently defined properties for the selected handler.

RTVHttp Options Tab

This tab allows you to enable the RTV HTTP Data Source, specify the port to be used, specify the root URL, and specify whether or not to require HTTPS to be used.

Field Name	Description
Enabled	Selecting this check box enables the RTV HTTP Data Source functionality and displays the **Port** field, the **Root URL** field, and the **Use Https** check box.
Port	Specify the port to which you want to connect to collect data.
Root URL	Specify the URL to be used as the root portion of the URL. The **Handler URL**, as defined on the "RTVHttp Handlers Tab", will be appended to this URL to form the complete URL.
Use Https	Selecting this check box requires that HTTPS be used for the defined URL.

RTView Deployment - RTV HTTP

This section contains details about the deployment process that are specific to your data source. Please go to the Deployment section of this documentation for instructions on how to implement your RTView deployment option. Return to this page whenever you are instructed to refer to deployment information that is specific to your data source.

Note: If the application(s) posting data to the HTTP Data Source cannot post directly to your RTView Application due to security or post access limitations, then complete the section titled **Set up RTVPost Servlet on Application Server**. Your application(s) can then post data to the RTVPost Servlet, which will forward the data to RTView. See "F: Set up RTVPost Servlet on Application Server (Optional)" for more information on application deployment, or "F: Set up RTVPost Servlet on Application Server (Optional)" for more information on browser deployment.

This section includes:

System Requirements and Setup

There are no additional setup steps required for this data source.

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options", "Application Options - RTV HTTP", and "RTV_JAVAOPTS" for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
RTVHTTPOPTIONS.ini	Contains defined handler names and the options specified in the "RTVHttp Options Tab" for the RtvHttp data source.

Note: Options specified using command line parameters override values set in these initialization files.

Command Line Options - RTV HTTP

In addition to General Options, the following command line arguments are enabled with the RtvHttp data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: "**-sub:$data:my Data**").

Name	Description
-rtvhttptrace:N	Enable data source specific tracing. N is an Integer and higher numbers yield more trace output. Default for this option is no tracing enabled (-rtvhttpresttrace:0). Example: -rtvhttptrace:6

RTV HTTP Rest Data Source

The RTView HTTP REST Data Adapter is an extensible data adapter that allows RTView to interact with REST APIs by means of HTTP requests. This data adapter provides an extensible framework with RTView, which uses the Apache HTTP Client to make HTTP requests. You can specify handlers to assist in the processing of an HTTP request.

The RTVHttpRest adapter supports data attachments that will execute an HTTP Request to get an HTTP Response, which can then be processed to extract a data table for use within RTView. The data attachments specify the handlers used for the various parts of making and processing an HTTP Request and Response. The data attachments allow the user to specify values that will be passed to handlers that implement the appropriate interfaces.

Default handlers are available, or you can implement custom handlers to extend functionality. There are examples of custom handlers in the **custom/rtvhttprest** directory of the RTView installation. See the **README** file in the **custom/rtvhttprest** directory for more information.

Note: The RTV HTTP Rest data source may not be licensed in your RTView installation.

This section includes:

System Requirements and Setup - RTV HTTP REST

System Requirements

The RTV HTTP REST data source has no additional "System Requirements".

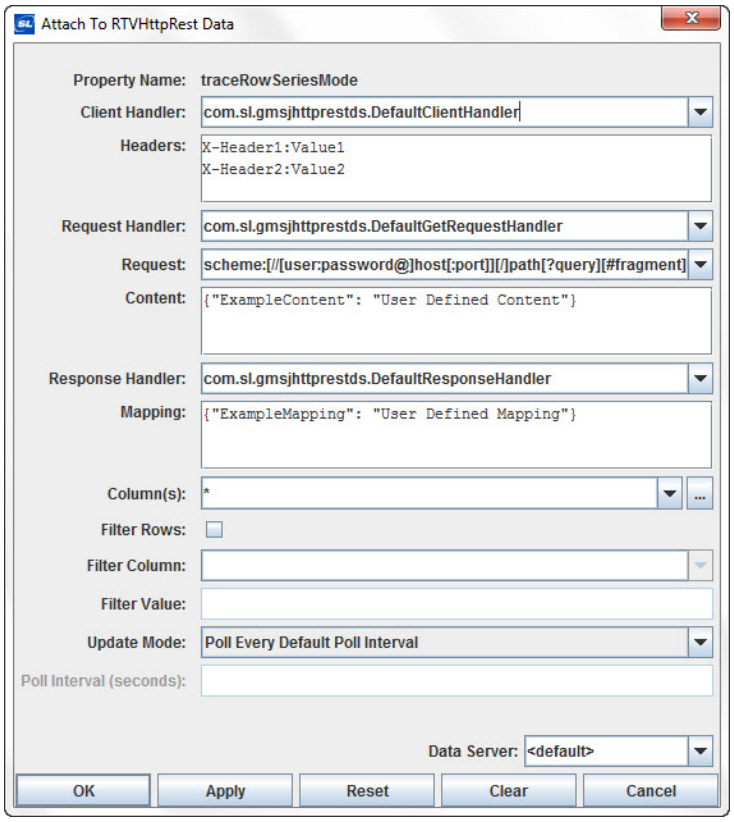
Setup

The RTV HTTP REST data source requires no additional "Setup".

Attach to RTVHttpRest Data

The **Attach to RTVHttpRest Data** dialog, which is used to define the HTTP request and the handlers used to process the request, can be accessed from the **Object Properties** window. The **valueTable** property of an object, typically a Cache object, will be attached to a data table received from the request. See "Caches" for more information.

Right-click on the Property Name from the **Object Properties** window and select **Attach to Data> RTVHttpRest** to display the **Attach to RTVHttpRest Data** dialog. The **Attach to RTVHttpRest Data** dialog provides several drop down menus that allow you to specify parameters of the request including the **Client Handler**, the **Request Handler**, and the **Response Handler** drop down menus, which list the names of the available handlers (default and custom).



Field Name	Description
Client Handler	The ClientHandler creates a client (CloseableHttpClient) that is used to execute a request (returned by the Request Handler) to return a response (processed by the ResponseHandler). A default client handler (com.sl.gmsjhttprestds.DefaultClientHandler) is provided, but you can create custom client handlers if needed. There are examples of custom handlers in the **custom / rtvhttprest** directory. See the **README** file in the **custom / rtvhttprest** directory for more information.
Headers	Define additional headers to be used by the ClientHandler to populate the HTTP headers in the generated request. This field allows you to add additional headers to your request rather than having to modify the original request. Headers are set on ClientHandlers that implement the Headers interface. See the **custom / rtvhttprest** directory for examples.
Request Handler	The RequestHandler creates a request (HttpUriRequest) that is executed by a client handler to return a response (processed by the ResponseHandler). A default request handler (com.sl.gmsjhttprestds.DefaultGetRequestHandler) is provided, but you can create custom request handlers if needed. See the **README** file in the **custom / rtvhttprest** directory for more information.

Request	Enter the request that you want the RequestHandler to execute. The Request Handler may use the request as is or process it to extract additional information used in the request. The format of a URI used by an HTTP request is (where scheme is usually HTTP or HTTPS): `scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]`
Content	Specify any additional content that you want the RequestHandler to include in the body/payload of the HTTP request. This field allows you to add additional content to the payload instead of having to modify the original request. Content is set on RequestHandlers that implement the Content interface. See the **custom/rtvhttprest** directory for examples.
Response Handler	The ResponseHandler returns the data (extracted by the RequestHandler) from the response. A default response handler (com.sl.gmsjhttprestds.DefaultResponseHandler) is provided, but you can create a custom response handler if necessary. See the **README** file in the **custom/rtvhttprest** directory for more information.
Mapping	Specify additional mapping information that the ResponseHandler can use in the generated response. Mapping is set on ResponseHandlers that implement the Mapping interface. See the **custom/rtvhttprest** directory for examples.
Column(s)	Name of the column to use as a filter. Multiple column names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col3). If your column name contains a space or a semicolon, then the entire name must be enclosed in single quotes.
Filter Rows	Select this check box to enable the **Filter Column** and **Filter Value** fields.
Filter Column	The name(s) of the column(s) to which the **Filter Value** should be applied.
Filter Value	The filter value(s) to be applied to the filter column(s). Multiple filter values should be entered as a nested list, where values for a given column are separated by commas within a semicolon (;) delimited list (i.e. val1,val2;val3,val4;val5,val6). If your filter value contains a space or a semicolon, then the entire value must be enclosed in single quotes. When ***** is entered as a filter field value, data for all values in the specified filter column will be used to update the object property. When "*****" is entered, only the literal comparative value will be used.
Update Mode	Select the desired polling interval from this drop down list. You can select from: **Poll Every Default Poll Interval**: refreshes data when the default polling interval occurs. **Every Poll Interval**: refreshes data when the value you defined in the Poll Interval field occurs. **Poll On Demand**: refreshes data when manually requested. **Poll Once (Static Data)**: populates data only once and never refreshes the data.
Poll Interval	Define the desired polling interval in which the data will be refreshed (if using the **Every Poll Interval** option).

Data Server

Select to read data through your configured Data Server and not directly from the RTV HTTP REST data source.

Default - Select the default Data Server you configured in **Application Options>**"Data Server Tab"

None - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source.

Named Data Servers - Select a Named Data Server that you configured in **Application Options>**"Data Server Tab".

Multi-Server Attachment - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more Named Data Servers (e.g. ds101;ds102). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**).

Note: The values **__default** and **__none** begin with two underscore characters.

Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received.

A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways:

1. The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows.

2. The multi-server attachment can be applied as the **Table** argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

When an object property is attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing this value from the **Object Properties** window is no longer possible. Once a property has been attached to data, it receives continuous updates. To remove the data attachment and resume editing capability in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information entered into the dialog is validated against the data received.

The following describes the significance of the **Attach to RTVHttpRest Data** validation colors:

Blue	Unknown	Cannot validate entry.
White	Valid state	Entry is valid.
Red	Invalid state	Incomplete or invalid entry.

*If the entry is validated as Unknown, RTView will attempt to read it when you click **OK** or **Apply**.

Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. A generic name such as $table is used instead of a specific table name. Later, when the display is running, this generic value is defined by the actual name of a specific table, such as SalesTable. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

Select Table Columns

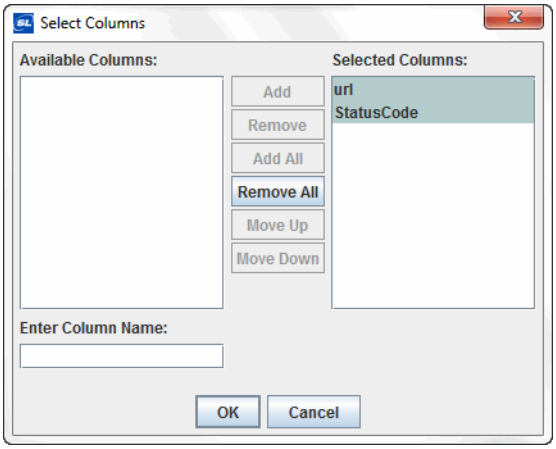
You can specify which table columns to display and in what order they will appear from the **Attach to RtvHttpRest Data** dialog. Since the incoming data can constantly change, the **Available Columns** region on the **Available Columns** window will not be populated automatically. You either need to know the column names in the incoming data or you need to view the incoming data beforehand to determine the column names.

Once you know the column names in the incoming data, there are two ways to specify the columns that you want to view. You can either enter the column names directly in the **Column(s)** field (using a semi-colon to separate the column names when specifying multiple columns), or you can click on the ellipses button  in the **Column(s)** field (or right-click in the **Column(s)** field and choose **Select Columns**) to display the **Select Columns** dialog. To add a column, type the name of the column you want to add in the **Enter Column Name** field and click the **Add** button. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of columns in a table by arranging the items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

Validation colors indicate whether selected columns are valid. However, if even one column selected is invalid, the **Column(s)** field in the **Attach to RtvHttpRest Data** dialog will register as an invalid entry.

Note: Invalid columns will not update.

If no data is available for a table row within a selected column, the table cell will be empty or will display one the following values: **N/A**, **false**, **0**, or **0.0**.



The following describes the **Attach to RtvHttpRest Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Application Options - RTV HTTP REST

Select **Tools>Options** in the Display Builder to display the **Application Options** dialog, Then select **RTVHttpRest** from the left menu. Options specified in the **RTVHttpRest** tab can be saved in an initialization file (**RTVHTTPRESTOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server, and Historian* to set initial values. If no directory has been specified for your initialization files and **RTVHTTPRESTOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

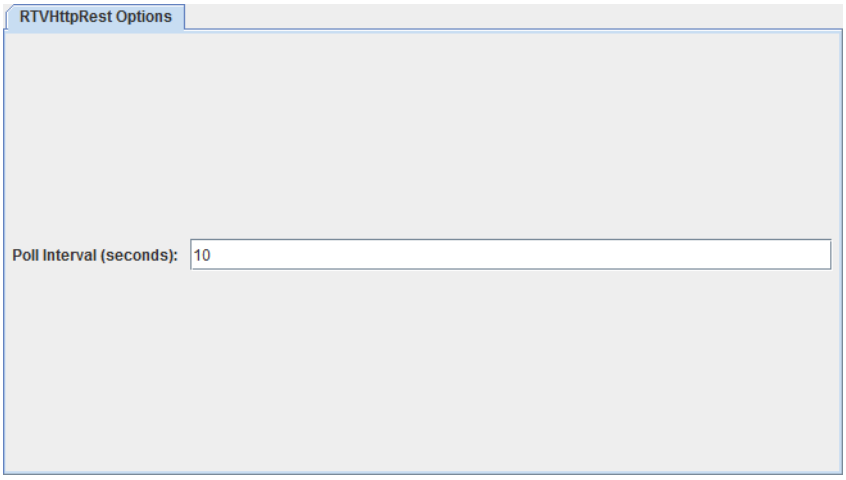
Note: Options specified using command line arguments will override values set in initialization files. See "Command Line Options - RtvHttpRest" for more information.

---

*If you are enabling RTVHttpRest options to utilize the Historian for Alert Persistence, then you must copy the **RTVHTTPRESTOPTIONS.ini** file to the directory where you will be running the Historian.

RTVHttpRest Options Tab

This tab allows you to specify the polling interval that is used to refresh the data in the display.



Field Name	Description
Poll Interval (seconds)	Define the polling interval (in seconds) used to refresh data in the display. By default, this option is set to **0**. Once set, all objects with **Update Mode** (on the "Attach to RTVHttpRest Data" window) set to **Poll Every Default Poll Interval** will use this setting to refresh their data.

RTView Deployment - RtvHttpRest

This section contains details about the deployment process that are specific to your data source. Please go to the Deployment section of this documentation for instructions on how to implement your RTView deployment option. Return to this page whenever you are instructed to refer to deployment information that is specific to your data source.

This section includes:

- "System Requirements and Setup" on page 572
- "Data Source Configuration File" on page 572

System Requirements and Setup

There are no additional setup steps required for this data source.

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options", "Application Options - RTV HTTP REST", and "RTV_JAVAOPTS" for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
RTVHTTPRESTOPTIONS.ini	Contains general options as well as data source options for RtvHttpRest.

Note: Options specified using command line parameters override values set in these initialization files.

Command Line Options - RtvHttpRest

In addition to General Options, the following command line arguments are enabled with the RtvHttpRest data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: "**-sub:$data:my Data**").

Name	Description
-rtvhttpresttrace:N	Enable data source specific tracing. N is an Integer and higher numbers yield more trace output. Default for this option is no tracing enabled (-rtvhttpresttrace:0). Example: -rtvhttpresttrace:6

RTVPipe Data Source

Note: The RTVPipe data source may not be licensed in your RTView installation.

The RTVPipe data adapter supports data attachments that launch an external process, reads the text output of that process and converts it into a string or a data table for use within RTView.

The external process can be any executable program or script that writes lines of text to its standard output stream. The connection between RTView and the output stream of the process is referred to as a "pipe". The RTVPipe data adapter uses a Java class known as a "pipe handler" to parse the output it reads from the process.

For more sophisticated conversions it is possible to create a custom RTVPipe Handler. See "Customization - RTVPipe Handler" for details.

This section includes:

System Requirements and Setup - RTVPipe

System Requirements

The RTVPipe data source has no additional "System Requirements".

Setup

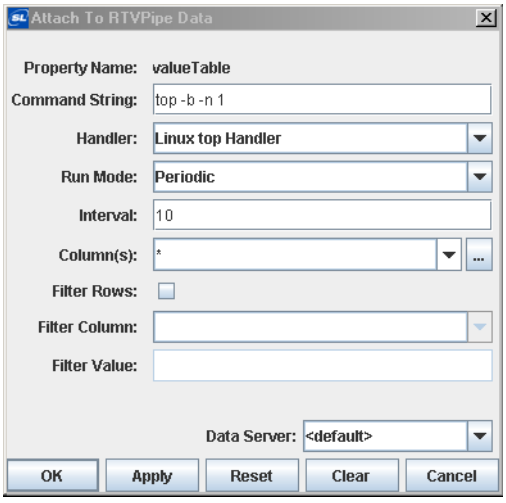
The RTVPipe data source requires no additional "Setup".

Attach to RTVPipe Data

Note: The RTVPipe data source may not be licensed in your RTView installation.

The **Attach to RTVPipe Data** dialog, which can be accessed via the **Object Properties** window, is used to specify the Command String to be executed by the pipe adapter and select the Handler to process the output.

Right-click on the Property Name from the **Object Properties** window and select **Attach to Data>RTVPipe** to display the **Attach to RTVPipe Data** dialog. The **Attach to RTVPipe Data** dialog provides several drop down menus that allow you to specify information. The RTVPipe adapter will connect to the output stream of the external process created by executing the specified Command String and send the lines of text from that stream to the selected Handler. Drop down menus for **Column(s)** and **Filter Column** populate based on the selected Handler. If the item you require is not listed, type your selection into the field.



Field Name	Description
Command String	Enter the Command String to be executed by the RTVPipe adapter. This value can contain substitutions. The specified command string must be executable on the host on which the pipe adapter is running. • On Windows, this means the command string must specify an .exe file (e.g. cmd.exe /c x.bat) to run the x.bat script file. • On UNIX, this means the command string must specify an executable program or shell script. In all cases the launched process must write lines of text to its output stream.
Handler	Select the name of a Handler. Two generic built-in handlers are available, as well as any custom handlers defined on the "RTVPipe Handlers Tab" of the **Application Options** dialog. **RTView Generic Text Handle**r - Converts process output into a multi-line text string. **RTView Generic Table Handler** - Converts process output into a single-column table with one row per line of output. **Note:** A unique instance of the handler class will be created for each unique pipe (i.e. each unique data attachment).
Handler Hints	Enter string of hints to be passed to the Handler. **Note:** The syntax of this string is determined by the selected Handler. If the selected Handler does not support hints this field is not visible. **Note:** The built-in generic handlers do not support hints.
Run Mode	Select from the following to specify what action the RTVPipe adapter should take once the pipe's process is complete. **Once** -- Do not re-execute command string. **Note:** This mode is useful for an on-demand reading of, for example, a complete error log file. **Periodic** -- Re-execute the command string after waiting the number of seconds specified in the Interval field. **Note:** This mode is useful for periodically reading an updating log file via a command such as cat or tail (without the -f/-F option). **Continuous** -- Re-execute the command string immediately. **Note:** This mode is useful for reading an updating log file via a continuous command such as **tail -F**.
Interval	Enter the duration (in seconds) to wait before re-executing the command string once the pipe's process is complete. **Note:** This field is only visible when the Run Mode selected is Periodic.
Timeout	Enter the duration (in seconds). If no output is read from the process in the specified timeout period, the process will be restarted. Default is **0**, indicating no timeout. **Note:** This field is only visible when the Run Mode selected is Continuous.
Column(s)	If the specified Handler returns tabular data, then select which column(s) to display. To bring up the **Select Columns** dialog, click on the ellipses button ... in the **Column(s)** field (or right-click in the **Column(s)** field and choose **Select Columns)**. The dialog should contain a list of Available Columns that you can add to your table.
Filter Rows	Check box to indicate whether or not to filter the RTVPipe data. See "Row Filtering" for more information.

Filter Column	Name of the column to use as a filter. Multiple column names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col 3). If your column name contains a space or a semicolon, then the entire name must be enclosed in single quotes.
Filter Value	Value that the Filter Column must equal. Multiple filter values should be entered as a nested list, where values for a given column are separated by commas within a semicolon (;) delimited list (i.e. val1,val2;val3,val4;val5,val6). If your filter value contains a space or a semicolon, then the entire value must be enclosed in single quotes. When * is entered as a filter field value, data for all values in the specified filter column will be used to update the object property. When "*" is entered, only the literal comparative value will be used.
Data Server	Select to read data through your configured Data Server and not directly from the RTVPipe data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab" **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a Named Data Server that you configured in **Application Options>**"Data Server Tab". **Multi-Server Attachment** - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more Named Data Servers (e.g. ds101;ds102). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**). **Note:** The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**. When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received. A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways: **1.** The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows. **2.** The multi-server attachment can be applied as the **Table** argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

When an object property is attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing this value from the **Object Properties** window is no longer possible. Once a property has been attached to data, it receives continuous updates. To remove the data attachment, and resume editing capability in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid.

The following describes the significance of the **Attach to RTVPipe Data** validation colors:

Blue	Unknown	Cannot validate entry.*
White	Valid state	Entry is valid.

*If a Handler is validated as Unknown (i.e. specified handler class cannot be loaded), RTView will attempt to validate it when you click **OK** or **Apply**.

Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. A generic name such as **$path** is used instead of a specific command string. Later when the display is running, this generic value is defined by the actual command string, such as **tail -f** $path. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

RTViewDs

If the value **RTViewDS.Pipes** is entered in the **Command String** field, the **Handler** and **Run Mode** fields will be hidden in the data dialog and an internal table of pipe process metrics will be applied to the listener. The **RTViewDS.Pipes** table will contain one row for each unique combination of Command String and Run Mode that are currently active in the pipe data source. The columns in the table are as follows:

	RTViewDs	
RTViewDs.Pipes	**Column Available**	**Description**
	Command String	Command string executed by the pipe.
	Mode	Run Mode for the command string. If the selected Run Mode is Periodic, the interval is also shown in this column.
	State	Current state of the external process, either **running** or **exited**.
	Runs	Total number of times command string has been executed.
	References	Number of handlers that are currently using this pipe.
	Total Lines Read	Total number of lines read from all runs of the command.
	Last Read Time	Last (most recent) time at which a line of input was read from the pipe.
	Last Line	Last line read from the pipe.

Select Table Columns

You can specify which table columns to display and in what order they will appear using the **Attach to RTVPipe Data** dialog.

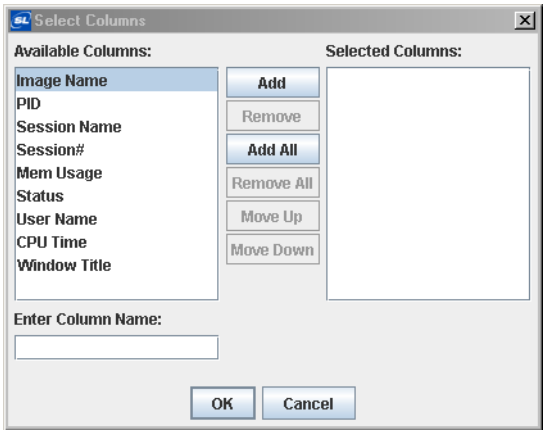
To display the **Select Columns** dialog, click on the ellipses button ... in the Column(s) field (or right-click in the **Column(s)** field and choose **Select Columns**). The dialog should contain a list of Available Columns that you can add to your table.

To add a column, select an item from the Available Columns list and click on the **Add** button. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of columns in a table by arranging the items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

Validation colors indicate whether selected columns are valid. However, if even one column selected is invalid, the **Column(s)** field in the **Attach to RTVPipe Data** dialog will register as an invalid entry.

Note: Invalid columns will not update.

If no data is available for a table row within a selected column, the table cell will display one the following values: **N/A**, **false**, **0**, or **0.0**.



The following describes the **Attach to RTVPipe Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Application Options - RTVPipe

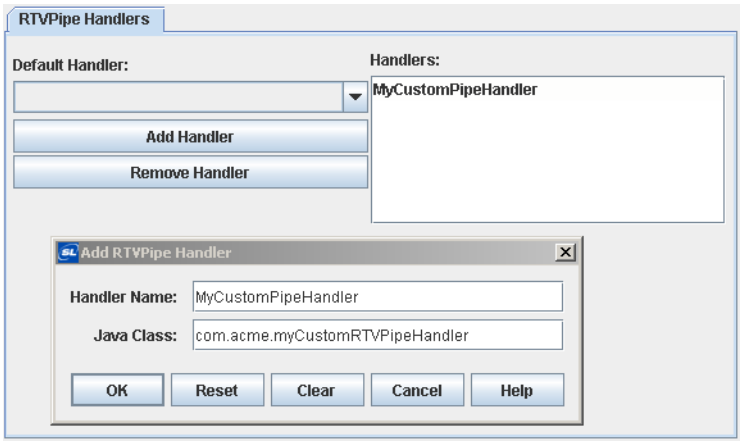
Select **Tools>Options** in the Display Builder to display the **Application Options** dialog.

Options specified in the **RTVPipe Handlers** tab can be saved in an initialization file (**RTVPIPEOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server and Historian to set initial values. If no directory has been specified for your initialization files and **RTVPIPEOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

Note: Options specified using command line arguments will override values set in initialization files. See "Command Line Options - RTVPipe" for more information.

RTVPipe Handlers Tab

The RTVPipe Handlers tab in the **Applications Options** dialog allows you to define custom pipe handlers created to parse the output of specific programs or scripts. See "Customization - RTVPipe Handler" for more information.



Field Name	Description
Default Handler	Select a default Handler from the drop down list.
Add Handler	Click to open the **Add RTVPipe Handler** dialog and define a custom handler. To edit an existing Handler, double-click on a name from the **Handlers** list. **Handler Name --** Enter unique name for this handler. This name will appear in the **Handler** drop down menu of the **Attach to RTVPipe Data** dialog. **Java Class --** Enter the fully-qualified name (Java package name + class name) for the class that implements this handler. You must add this Java class to the "RTV_USERPATH" environment variable.
Remove Handler	Select a name from list and click **Remove Handler** to delete.

RTView Deployment - RTVPipe

This section contains details about the deployment process that are specific to your data source. Please go to the "Deployment" section of this documentation for instructions on how to implement your RTView deployment option. Return to this page whenever you are instructed to refer to deployment information that is specific to your data source.

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options", "Application Options - RTVPipe", and "RTV_JAVAOPTS" for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
RTVPIPEOPTIONS.ini	Contains general options as well as data source options for RTVPipe.

Note: Options specified using command line parameters override values set in these initialization files.

Command Line Options - RTVPipe

In addition to General Options, the following command line arguments are enabled with the RTVPipe data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: "**-sub:$data:my Data**").

Name	Description
-rtvpipetrace	Set to **1** to enable message tracing. Default is **0** and tracing is disabled. Example: **-rtvpipetrace:1**

SNMP Data Source

Note: The SNMP data source may not be licensed in your RTView installation.

The SNMP data adapter allows you to query data or receive traps from SNMP-capable applications or devices.

This section includes:

System Requirements and Setup - SNMP

System Requirements

See the **README_sysreq.txt** file in your installation's home directory for the current SNMP version(s) supported.

Setup

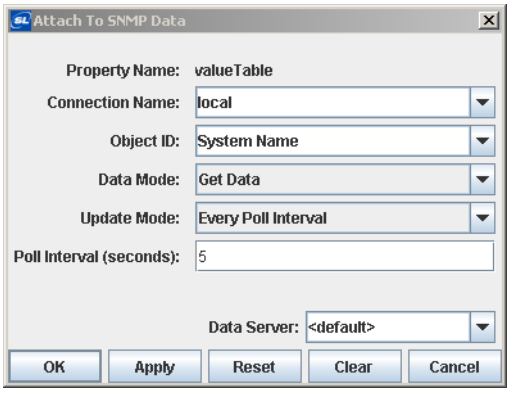
The SNMP data source requires no additional "Setup".

Attach to SNMP Data

Note: The SNMP data source may not be licensed in your RTView installation.

The **Attach to SNMP Data** dialog, which is used to access data in a local or remote SNMP agents, can be accessed via the **Object Properties** window.

Right-click on the Property Name from the **Object Properties** window and select **Attach to Data>SNMP** to display the **Attach to SNMP Data** dialog. Drop down menus for **Connection Name** and **Object ID** populate based on the information defined in the **Application Options** dialog.



Field Name	Description
Connection Name	Select a Connection Name. Connections can be defined on the "SNMP Connections Tab" of the **Application Options** dialog. If the selected Data Mode is **Receive Traps** and a value of ***** is specified, then traps from all connections will be displayed.
Object ID	Select an Object ID name. Names for Object IDs can be specified on the "SNMP Options Tab" of the **Application Options** dialog. If the selected Data Mode is **Receive Traps** and a value of ***** is specified, then traps from all Object IDs will be displayed.
Data Mode	**Get Data** -- Get data from SNMP servers. **Receive Traps** -- Receive traps from SNMP applications. The SNMP data adapter is set up to index incoming traps by their Connection Name, so in general it will display only the last trap of any type from a given connection. **Note:** The Cache data source may be used to keep and display a history of received traps.
Update Mode	Specify which mode to use to update connection: **Poll Every Default Poll Interval** -- Update connection each Default Poll Interval. See the "SNMP Options Tab" in the **Application Options** dialog for information on setting the **Default Poll Interval**. This is the default **Update Mode**. **Every Poll Interval** -- Update connection each Poll Interval. If this option is selected, you must specify a **Poll Interval**. **Poll On Demand** -- Update connection each time a display that uses this data attachment is opened and each time a substitution string that appears in the data attachment has changed. **Poll Once (Static Data)** -- Poll for data only once. Select if the data returned by this attribute or operation is static.
Poll Interval	Specify the interval (in seconds) to update connection. This option is only available if the Update Mode selected is Every Poll Interval. **Note:** Because the Poll Interval is superseded by the General Update Period, the amount of time elapsed between updates may be longer than the value entered. For example, if the General Update Period is 2 seconds and the Poll Interval is 5 seconds, the connection will be updated every six seconds. See "General Tab" for more information.
Data Server	Select to read data through your configured Data Server and not directly from the SNMP data source.

Default - Select the default Data Server you configured in the **Application Options>**"Data Server Tab".

None - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source.

Named Data Servers - Select a **Named Data Server** that you configured in the **Application Options>**"Data Server Tab".

Multi-Server Attachment - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more **Named Data Servers** (e.g. ds101;ds102). Each name specified must correspond with a Named Data Server that you configured in the **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**).

Note: The values **__default** and **__none** begin with two underscore characters.

Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received.

A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways:

1. The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows.
2. The multi-server attachment can be applied as the Table argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

When an object property is attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing this value from the **Object Properties** window is no longer possible. Once a property has been attached to data, it receives continuous updates. To remove the data attachment, and resume editing capability in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid.

The following describes the significance of the **Attach to SNMP Data** validation colors:

Red	Invalid state	Entry is not valid.
White	Valid state	Entry is valid.

Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. A generic name such as **$conn** is used instead of a specific SNMP connection. Later when the display is running, this generic value is defined by the actual name of a connection. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

RTViewDs

If the value RTViewDS is selected in the **Command String** menu, the **Object ID** menu will list two internal tables (Data or Traps) that contain metrics on SNMP requests completed and traps received per connection. The columns in these tables are as follows:

	RTViewDs	
Data	**Column**	**Description**
	Total Requests	Total number of data requests
	Failed Requests	Total number of failed data requests
	Last Time	Time elapsed (in milliseconds) between last request and response.
	Average Time	Average time elapsed (in milliseconds) between all requests and responses.
	Last Error	Most recent error message, if any.
Traps	**Column**	**Description**
	Traps Received	Number of traps received.
	Last Timestamp	Date of last timestamp.

The following describes the **Attach to SNMP Data** dialog commands:

Command	**Description**
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Application Options - SNMP

Select **Tools>Options** in the Display Builder to access the **Application Options** dialog.

Options specified on SNMP tabs can be saved in an initialization file (**SNMPOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server, and Historian to set initial values. If no directory has been specified for your initialization files and **SNMPOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

Note: Options specified using command line arguments will override values set in initialization files. See "Command Line Options - SNMP" for more information.

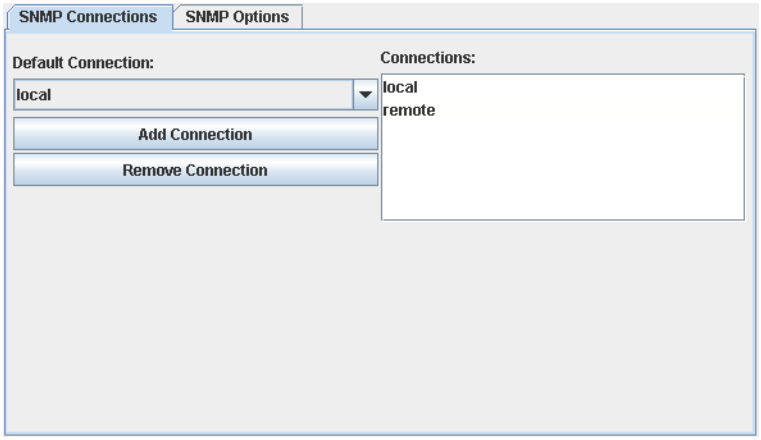
There are two Application Options tabs for SNMP: "SNMP Connections Tab" and "SNMP Options Tab".

SNMP Connections Tab

This tab allows you to add or remove connections and set your default connection.

When you add a SNMP connection to the list it will be highlighted in yellow indicating that RTView has not connected to it. To attempt to connect, click **OK**, **Apply**, or **Save**. If the background remains yellow, then RTView was unable make a connection. Check that your connection was setup correctly and that the server is running.

Note: Regardless of which tab you are currently working from in the **Application Options** dialog, each time you click **OK**, **Apply**, or **Save**, RTView will attempt to connect to all unconnected connections.



Field Name	Description
Default Connection	Select a default Connection from the drop down list.

Add Connection Click to open the **Add Connection** dialog and define a connection. To edit an existing connection, double-click on a name from the **Connections** list.

Connections that are updating objects in a current display cannot be renamed.

Connection Name - Enter unique name for this Connection. This name will appear in the **Connection Name** drop down menu of the **Attach to SNMP Data** dialog.

Hostname or IP address -- Name or IP address of SNMP server.

Data Port - Port used to receive SNMP data.

Trap Port - Port used to receive SNMP traps.

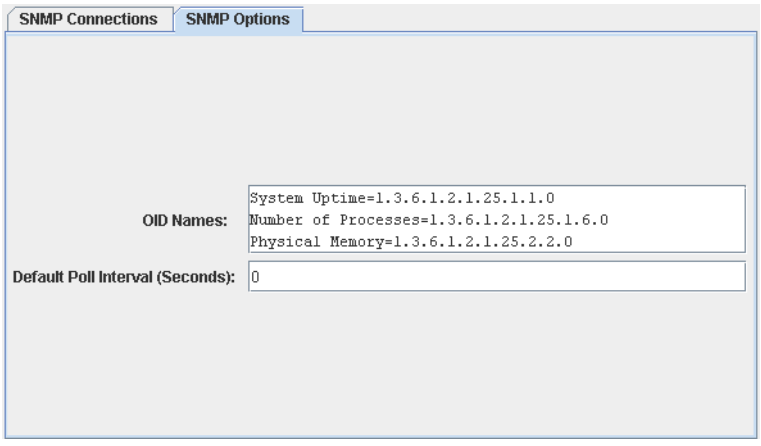
SNMP Version - Default is **v2c**.

Community - Default is **public**.

Remove Connection Select a name from list and click **Remove Connection** to delete.

SNMP Options Tab

The **SNMP Options** tab allows you to specify names for numeric OID values and set the Default Poll Interval. In the example below, names have been given to some common OID values from LAN Manager MIB II (RFC 1213).



Name **Description**

OID Names	Enter names for numeric OID values. Named OID values will appear in the **Object ID** drop down menu of the **Attach to SNMP Data** dialog.
Default Poll Interval	Enter the time in seconds to control how often the connection is updated or operations in data attachments are executed if no Poll Interval is specified in the **Attach to SNMP Data** dialog. Default is **0**, which updates connections and operations according to the **General Update Period** specified in **Application Options** on the "General Tab". **Note:** Because the **Default Poll Interval** is superseded by the **General Update Period**, the amount of time elapsed between updates may be longer than the value entered. For example, if the **General Update Period** is 2 seconds and the **Default Poll Interval** is 5 seconds, the connection will be updated every 6 seconds.

RTView Deployment - SNMP

This section contains details about the deployment process that are specific to your data source. Please go to the Deployment section of this documentation for instructions on how to implement your RTView deployment option. Return to this page whenever you are instructed to refer to deployment information that is specific to your data source.

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options", "Application Options - SNMP", and "RTV_JAVAOPTS" for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
SNMPOPTIONS.ini	Contains general options as well as data source options for SNMP.

Note: Options specified using command line parameters override values set in these initialization files.

Command Line Options - SNMP

In addition to General Options, the following command line arguments are enabled with the SNMP data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: "**-sub:$data:my Data**").

Name	Description
-snmptrace	Set to **1** to enable tracing. Default is **0** and tracing is disabled. Example: **-snmptrace:1**

Splunk Data Source

Splunk is a data source used in conjunction with RTView application monitoring solutions.

Splunk (*www.splunk.com*) offers the ability to perform high performance search queries on log files and then makes it easy to grab content out of a log file that is applicable to application performance or status. That information can then be fed into RTView dashboards, reports and alerts as any standard RTView data adapter.

The Splunk data source allows you to access data in local or remote Splunk servers and use that data in RTView displays. Within the Splunk Web interface you can define various filters, such as time range filters, embed them in the search and then copy and paste these search definitions directly into the "Attach to Splunk Data" dialog.

If you have a large amount of various log files containing complex information that might require specialized parsing to gather pertinent information, Splunk can greatly enhance access to and maintenance of this important source of application performance data.

This section includes:

Splunk System Requirements

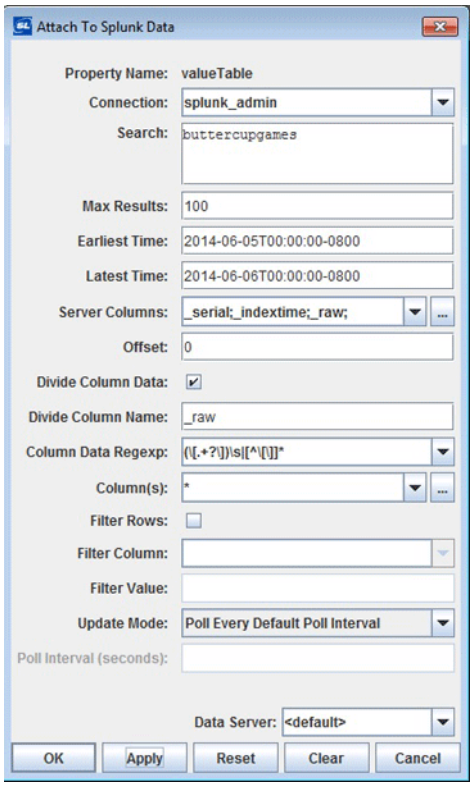
See the **README_sysreq.txt** file in your installation's home directory for the current Splunk version(s) supported.

Note: If you are using Windows, make sure **JAVA_HOME** system variable is set to the directory where the JDK is installed.

Attach to Splunk Data

Note: The Splunk data source may not be licensed in your RTView installation.

Right-click on the Property Name from the **Object Properties** window and select **Attach to Data>SPLUNK** to display the **Attach to Splunk Data** dialog, which is used to search the server in the defined Splunk connection and filter the results. The **Attach to Splunk Data** dialog provides several drop down menus that allow you to specify information. If the drop down menu does not contain the item you require, type your selection into the text field.



Field Name	Description
Connection	Connection name. You may define a connection on the "Splunk Connections Tab" of the **Application Options** dialog.
Search	Enter a search. **Note:** It is possible to copy and paste a search query directly from the search bar at the top of the Splunk Web dashboard into the **Attach to Splunk Data** dialog. Refer to your *Splunk Documentation* for further information on searches, including search syntax and how to define various filters and display them outside of the search.
Max Results	Set the maximum number of rows of data to be returned from the Splunk server. If the search results in fewer rows than specified, only those rows of data are returned. Default is **100**. **Note:** If value is set to **0**, then all available results are returned. This value should be used with caution as it is possible for a large amount of data to transferred from the Splunk server.

Earliest Time	Filters data by timestamp. Only objects that have a timestamp greater than or equal to earliest_time will be returned. If left blank, no limit will be set on the earliest object returned. Use in conjunction with **Latest Time** to return objects within a specific time range. **Note:** Time format is always ISO-8601 formatted. For example: **2014-06-05T00:00:00-0800** **2014-06-06T00:00:00-0800**
Latest Time	Filters data by timestamp. Only objects that have a timestamp less than latest_time will be returned. If left blank, no limit will be set on the latest object returned. Use in conjunction with **Earliest Time** to return objects within a specific time range. **Note:** Time format is always ISO-8601 formatted. For example: **2014-06-05T00:00:00-0800** **2014-06-06T00:00:00-0800**
Server Column(s)	Select which columns to search in the Splunk server. To bring up the **Select Server Column(s)** dialog, click on the ellipses button in the Server Column(s) field or right-click in the **Server Column(s)** field and click **Select Server Column(s)**. This dialog should contain a list of Available Column(s) you can add to your search. See "Select Server Column(s) and Select Column(s)" for more information. **Note:** If this field is left blank, all columns in the Search will be returned.
Offset	Starting offset of the first object in the list of data returned from the Splunk server. Use in conjunction with **Max Results** to parse a large set of returned data, one small section at a time. For example, with **Max Results** set to **100** and **Offset** set to **0**, you would get search results with a _serial column value of **0-99**. Or with **Offset** set to **10**, you would get search results with a _serial column value of **10-109**. In both cases you would get a subset of 100 results.
Divide Data Column	Select to divide up data in a returned column, via Java Regular Expression, into numbered columns. Typically this is used to extract data that has an implicit token or field based positional location in the raw data. For example, to break up raw log line/HHTP request into fields/columns.
Divide Column Name	Name of the column to split into into individual numbered columns that contain matched occurrences of the selected Java Regular Expression, where the new column name will be the number of the occurrence (e.g. "1", "2", "3", etc). In most cases, the Divide Column will be the raw data column (i.e.: _raw) returned from the Splunk server.
Column Data Regexp	Enter or select the Java Regular Expression (Regexp) desired to split the data in the specified Divide Column. The drop down menu contains a number of predefined regular expression literals or you can enter your own. **Note:** Normally Splunk will recognize and extract fields and make them available for searching on, but sometimes you need to split up the raw data to extract meaningful information.
Column(s)	Select which columns to display. To display the **Select Column(s)** dialog, click on the ellipses button in the **Column(s)** field or right-click in the **Column(s)** field and click **Select Column(s)**. This dialog should contain a list of Available Column(s) you can add to your table. See "Select Server Column(s) and Select Column(s)" for more information.

Filter Rows	Check box to indicate whether or not to filter rows. See "Row Filtering" for more information.
Filter Column	Name of the column to use as a filter. Multiple column names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col 3). If your column name contains a space or a semicolon, then the entire name must be enclosed in single quotes.
Filter Value	Value that the **Filter Column** must equal. Multiple filter values should be entered as a nested list, where values for a given column are separated by commas within a semicolon (;) delimited list (i.e. val1,val2;val3,val4;val5,val6). If your filter value contains a space or a semicolon, then the entire value must be enclosed in single quotes. When ***** is entered as a filter field value, data for all values in the specified filter column will be used to update the object property. When "*" is entered, only the literal comparative value will be used. These are only allowed for objects which display tabular data.
Update Mode	Specify which mode to use to update your Splunk connection: **Every Default Poll Interval** - Update connection each Default Poll Interval. See "Application Options - Splunk" for information on setting the Default Poll Interval. This is the default Update Mode. **Every Poll Interval** - Update connection each Poll Interval. If this option is selected, you must specify a Poll Interval. **Poll On Demand** - Update connection each time a display that uses this data attachment is opened and each time a substitution string that appears in the data attachment has changed. **Poll Once (Static Data)** - Poll for data only once. Select if the data returned by this attribute or operation is static.
Poll Interval	Specify the interval (in seconds) to update your Splunk connection. This option is only available if the **Update Mode** selected is **Poll Every Poll Interval**. See "General Tab" for more information. **Note:** Because the **Poll Interval** is superseded by the **General Update Period**, the amount of time elapsed between updates may be longer than the value entered. For example, if the **General Update Period** is 2 seconds and the **Poll Interval** is 5 seconds, the connection will be updated every six seconds.
Data Server	Select to read data through your configured Data Server and not directly from the Splunk data source. **Default** - Select the default Data Server you configured in the **Application Options>**"Data Server Tab". **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a Named Data Server that you configured in the **Application Options>**"Data Server Tab". **Multi-Server Attachment** - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more Named Data Servers (e.g. **ds101**;**ds102**). Each name specified must correspond with a Named Data Server that you configured in the **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**). **Note:** The values **__default** and **__none** begin with two underscore characters.

Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received.

A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways:

1. The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows.
2. The multi-server attachment can be applied as the Table argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

When an object property has been attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing values from the **Object Properties** window is no longer possible. To remove the data attachment and resume editing capabilities in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information entered into the dialog is validated against the selected server.

Note: Filters are not validated.

Blue	Unknown	Cannot validate entry.
White	Valid state	Entry is valid.
Red	Invalid state	Incomplete or invalid entry.

Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. For example, a generic substitution value such as **$MATCH** could be used in a search. Later, when the display is running, this generic value is defined and used to control which records are matched on the server. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

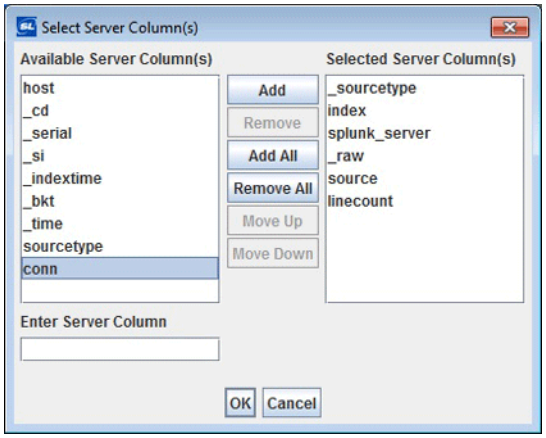
Select Server Column(s) and Select Column(s)

From the **Attach to Data** dialog you can specify which columns to search in the Splunk server as well as which columns to display in your resulting data and in what order they will appear. To view the list of Available Column(s), click on the ellipsis button ... to open the **Select Server Column(s)** dialog the or **Select Column(s)** dialog.

To add a column, select an item from the **Available Column(s)** list and click on the **Add** button. If the item you require is not listed, type your selection into the **Enter Column** field. Click the **Remove** button to delete an item previously added to the **Selected Column(s)** list. You can control the order of fields in a table by arranging the items in the **Selected Column(s)** list with the **Move Up** and **Move Down** buttons.

Validation colors indicate whether selected columns are valid. However if even one column selected is invalid, the **Server Column(s)** or **Column(s)** field in the **Attach to Splunk Data** dialog will register as an invalid entry.

If no data is available for a table row within a selected column, the table cell will display one of the following values: **N/A**, **false**, **0**, or **0.0**.



The following describes **Attach to Splunk Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

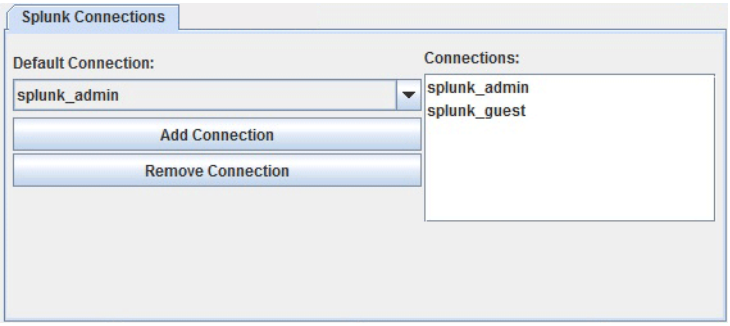
Application Options - Splunk

Select **Tools>Options** in the Display Builder to access the **Application Options** dialog. Options specified in the **Splunk Connections** tab can be saved in an initialization file (**SPLUNKOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server, and Historian to set initial values. If no directory has been specified for your initialization files and **SPLUNKOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

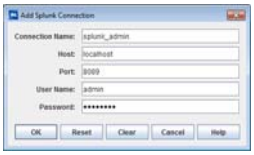
Splunk Connections Tab

This tab allows you to add or remove connections and set your default connection. When you add a Splunk connection to the list it will be highlighted in yellow indicating that RTView has not connected to it. To attempt to connect to a Splunk connection, click **OK**, **Apply**, or **Save**. If the background remains yellow, then RTView was unable make a connection. Check that your Splunk connection was setup correctly and that the Splunk server is running.

Note: Regardless of which tab you are currently working from in the **Application Options** dialog, each time you click **OK**, **Apply**, or **Save**, RTView will attempt to connect to all unconnected connections.



Field Name	Description
Default Connection	Name of connection used as the default for data attachments. Select from drop down menu to change default setting.
Add Connection	Click to open the **Add Splunk Connection** dialog. To edit, select a connection from the list and double-click. Connections that are updating objects in a current display cannot be renamed.

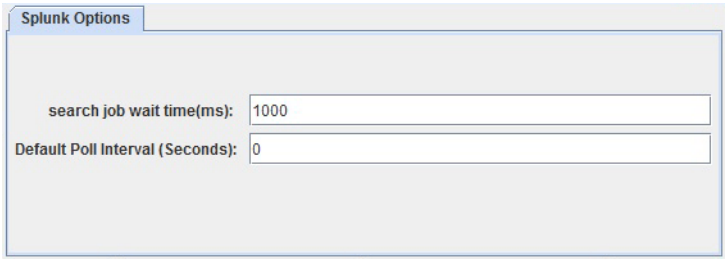


Connection Name - Name to use when referencing this Splunk connection in your data attachments. This should be a unique name used to identify the connection to your Spunk server (splunkd) along with associated account details:

	Host - Host name or IP address **Port** - Port number **User Name** - User name **Password** - Password. If you need to provide an encrypted password (rather than expose server password names in a clear text file, use the encode_string command line option with the following syntax: **encode_string type mypassword** where **type** is the key for the data source and **mypassword** is your plain text password. **Note:** The type argument is only required when you encrypt a string for a data source. For example, enter the following in an initialized command window (see "Initializing a Command Prompt or Terminal Window"): **encode_string splunk mypassword** and you will receive an encrypted password: **encrypted value: 013430135501346013310134901353013450134801334** Copy the encrypted value, paste it into the password field and click Save to save this value to the initialization (***.ini**) file. Or, if necessary, manually edit the (***.ini**) file to include the encrypted value. **Note:** If you need to manually edit a configuration (*.ini) file, contact SL Technical Support at support@sl.com for information about supported syntax.
Remove Connection	Select a connection from the list and click **Remove Connection** to delete. Connections that are updating objects in a current display cannot be removed.

Splunk Options Tab

This tab allows you to set a wait time for polling search requests and a default poll interval to update your Splunk connection.



Field Name	Description
search job wait time (ms)	Enter the time in milliseconds to control the wait time between polls for search data until a search job is complete. Default is **1000**.

Default Poll Interval (Seconds)	Enter the time in seconds to control how often your Splunk connection is updated or operations in data attachments are executed if no Poll Interval is specified in the data attachment. Default is **0**, which updates connections and operations according to the **General Update Period** specified in **Application Options** on the "General Tab". **Note:** Because the **Default Poll Interval** is superseded by the **General Update Period**, the amount of time elapsed between updates may be longer than the value entered. For example, if the **General Update Period** is **2** seconds and the **Default Poll Interval** is **5** seconds, the connection will be updated every six seconds.

RTView Deployment - Splunk

This section contains details about the deployment process that are specific to your data source. Please go to the Deployment section of this documentation for instructions on how to implement your RTView deployment option. Return to this page whenever you are instructed to refer to deployment information that is specific to your data source.

System Requirements and Setup

The Splunk data source has additional System Requirements and Setup. See "Splunk System Requirements" for more information.

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under lib in your installation directory. See "Application Options", "Application Options - Splunk", and "RTV_JAVAOPTS" for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
SPLUNK.ini	Contains data source options for Splunk.

Note: Options specified using command line parameters override values set in these initialization files.

Rich Client Browser Deployment Setup for Direct Data Connection

This deployment is supported by your data source. No additional client setup is required.

SQL Data Source

The SQL data source provides access to JDBC enabled databases. The **Attach to SQL Data** dialog makes it easy to browse, select data tables, filter information and institute query policies with a simple user interface. For those familiar with SQL, it is also possible to enter SQL commands to make database queries.

This section includes:

SQL System Requirements and Setup

System Requirements

In addition to basic "System Requirements", the SQL data source requires a database with a JDBC driver. See "SQL Database Connection Setup" for information.

Setup

In addition to general environment variables (see "Setup"), you must include the following if you will be using a direct JDBC connection to communicate with your database(s):

Name	Description	Example
RTV_USERPATH	Location of your SQL database JDBC jar. **Note:** If **RTV_USERPATH** already exists, append the JDBC jar to it.	**c:\sql\mydriver.jar;c:oracle\oracledriver**

Attach to SQL Data

Note: The SQL data source may not be licensed in your RTView installation.

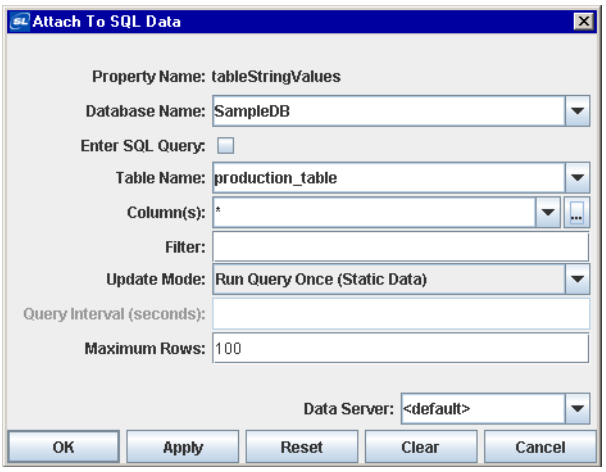
The **Attach to SQL Data** dialog, which is used to connect an object to your database using an SQL query, can be accessed via the **Object Properties** window. Once an object has been attached to your database it receives continuous updates.

When an object property is attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing this value from the **Object Properties** window is no longer possible. To remove the data attachment and resume editing capabilities in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the database when the Property Name and Value are no longer green.

Right-click on the Property Name from the **Object Properties** window and select **Attach to Data>SQL** to display the **Attach to SQL Data** dialog. The **Attach to SQL Data** dialog provides drop down menus and an optional filter field that allow you to specify information that will be used to create an SQL query for the selected database. Alternatively, select the **Enter SQL Query** check box to enter an advanced query.

Use the **-sqlquote** command line parameter options to enclose all table and column names specified in the **Attach to SQL Data** dialog in quotes when an SQL query is run. This is useful when attaching to databases that support quoted case-sensitive table and column names. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: If a case-sensitive table or column name is used in the **Filter** field, or you are entering an advanced query in the **SQL Query** field, they must be entered in quotes even if the **-sqlquote** option is specified.



Field Name	Description
Database Name	Name of database to query or "RTViewDs". You may specify a database connection on the "SQL Tab" of the **Application Options** dialog.
Enter SQL Query	Select to enter an advanced query. When selected, a SQL Query text field replaces the Table Name, Column(s) and Filter fields. **Note:** This option is for advanced users, SQL syntax will not be validated or checked for errors. You can also specify a timeout or ID for your query. For details, see "Specifying a Timeout or ID For a Query".
Table Name	Name of table in database to query.
Column(s)	Columns in table to display.
Filter	Optional SQL filter to apply to query.

Update Mode	Select one of the following: **Run Query Once** - Select this check box if the data returned by this query is static. If selected, RTView will run this query only once. This is the default setting. **Run Query Every Update Period** - Select to run this query each update period. See **Application Options>**"General Tab" for information on setting the update period. **Run Query Every Query Interval** - Select to run this query once every Query Interval. **Run Query On Demand** - Select to run this query each time a display that uses the query is opened and each time a substitution string that appears in the query string has changed. **Run Query On Schedule** - When selected, the **Schedule Name** field displays allowing you to select the name of a user-defined schedule. This option provides more control over when user-defined queries are run. See the **Schedule Name** field (below) for more information.
Schedule Name	This field displays when the **Run Query On Schedule** option is selected from the **Update Mode** drop down list and allows you to select a pre-defined schedule that allows users to define when specific queries are run. See "SQL Scheduler" for more information.
Query Interval (seconds)	Controls how often RTView will run this query. Enter the time in seconds or a substitution value (e.g. **$intervalQueryPeriod**). The **Update Mode** must be set to **Run Query Every Query Interval**. **Note:** The query interval is evaluated during each update pass, so the amount of time elapsed between queries may be longer than the value entered. For example, if the update period is 2 seconds and the query interval is 5 seconds, the query will get run every six seconds.
Maximum Rows	Enter the maximum number of rows to return from this query. **Note:** On some objects an additional property may further reduce the number of data points displayed. For example, the **maxNumberOfRows** property on the table or the **maxPointsPerTrace** property on the trend graph.
Data Server	Select to read data through your configured Data Server and not directly from the SQL data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab". **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a Named Data Server that you configured in **Application Options>**"Data Server Tab". **Multi-Server Attachment** - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more Named Data Servers (e.g. **ds101**;**ds102**). Each name specified must correspond with a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**). The values **__default** and **__none** begin with two underscore characters.

Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received.

A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways:

1. The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows.
2. The multi-server attachment can be applied as the Table argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

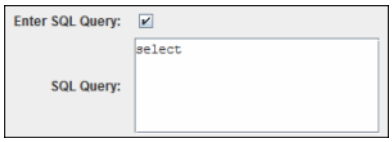
The **Database Name** drop down menu lists all available databases. The **Database Name** field automatically displays the name of the default database. If the item you require is not listed, type your selection into the field. A Database Repository file can be used to populate the initial values of drop down menus for Table Name and Column(s). See "Application Options - SQL" for information on how to create a Database Repository file. You can also create a file to exclude tables from the **Table Name** drop down menu. See "SQL Database Connection Setup" for details. Otherwise, drop down menus populate based on databases added from the **Application Options** dialog or those typed directly into the **Database Name** field. The **Filter** field is optional and uses standard SQL syntax.

Note: If you are using a direct JDBC connection, you will need to add your database in "Application Options - SQL".

Specifying a Timeout or ID For a Query

When you select **Enter SQL Query**, you can optionally specify a timeout, an ID, or both for your query.

Note: This option is for advanced users, SQL syntax will not be validated or checked for errors.



Specify Timeout For Query

To specify a timeout for the SQL query, prepend the following text to the query string:

rtvTimeout=NN;

where **NN** specifies the timeout interval in seconds.

For example: **rtvTimeout=20; select * from MyTable**

A semicolon must immediately follow the timeout value. You can also specify a substitution string.

For example: **rtvTimeout=$timeout; select * from MyTable**

When a query with a timeout is invoked by the SQL data adapter, on each subsequent update cycle (2 seconds, by default) the data adapter checks if the query has completed. If the query has not completed it checks if the timeout has expired. If the timeout has expired, the query is canceled and an error message is printed to the RTView console. For example:

ERROR: SQL query timeout after 20 seconds; db=MyDatabase; query=<select * from MyTable>

The status of the query is then set to "timeout" and an empty table is applied to the objects that are attached to the query result. In addition, the query is not executed again for at least the length of the timeout interval. For example, if the query is configured to run every 60 seconds and the timeout is 120 seconds, then after a timeout the next query occurs after 120 seconds, not 60 seconds.

Specify ID For Query

To specify an ID for an SQL query, terminate the ID string with a semicolon and prepend the following text to the query string:

rtvID=S;

where **S** specifies the ID string.

For example: **rtvID=MyTable Query 1; select * from MyTable**

The ID string cannot contain a semicolon (;) or equals (=) character. You can also specify a substitution string.

For example: **rtvID=$table Query; select * from $table**

The ID string has no effect on the execution of the query. The ID string appears as the value of the **Query ID** column in the **Keys** table of the "RTViewDs" database. The ID string also appears in any error messages issued by the SQL data adapter that involve the query.

Specify Timeout and ID For Query

To specify both a timeout and a query ID for the query, use the syntax described above and a semicolon after each parameter.

For example: **rtvTimeout=120;rtvID=MyTable Query 1; select * from MyTable**

The timeout and ID parameters can be specified in either order.

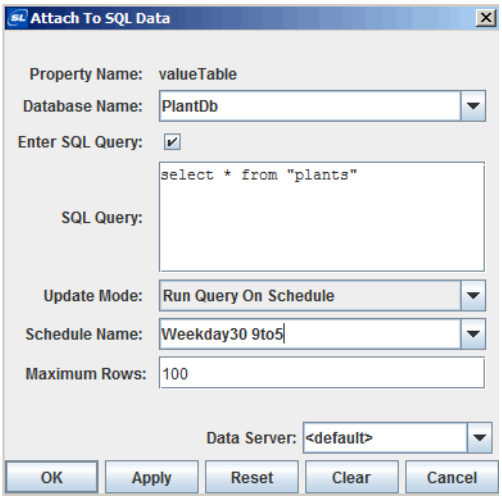
SQL Scheduler

The SQL Scheduler functionality allows you to define at what time of day and at what frequency your user-defined SQL queries are run.

Enabling SQL Scheduler

1. Set up the Rule table and Schedule table. See "Scheduling Functionality" for more information.

2. In the "Attach to SQL Data" dialog, enter your query, select **Run Query On Schedule** from the **Update Mode** drop down, and select the desired schedule from the **Schedule Name** drop down.



Instead of selecting a schedule name from the **Schedule Name** drop down, there are two other ways of selecting schedules/rules to use:

- You can enter a substitution string (for example, **$mySchedule**, which could have a value of weekday30 9to5) in the **Schedule Name** drop down. See "Substitutions" for more information.
- You can enter a list of rule names separated by commas (rule1,rule2,rule3) in the **Schedule Name** drop down. This option might be useful when there is no schedule defined with the desired set of rules.

Note: If schedules and rules are added or modified at runtime, then the next update of the data attachment used as the input to the **RtvScheduleTable** and **RtvScheduleRuleTable** will apply those changes to all SQL queries using those schedules and rules.

Example Configuration

An example configuration of the SQL Scheduler is available in the **custom/sql_scheduler** directory, which includes a schedule table, rule table, and scheduled queries. See the **README.txt** file in the **custom/sql_scheduler** directory for more information.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information entered into the dialog is validated against the selected database or the Database Repository file. See "Application Options - SQL" for information on how to create a Database Repository file.

Note: Filters and advanced SQL queries are not validated.

The following describes the significance of the **Attach to SQL Data** validation colors:

Blue	Unknown	Entry does not match any known database (or you have not attempted a connection*).
Yellow	Offline	Not connected to database.
White	Valid state	Entry is valid.
Red	Invalid state	Database is valid, but Table or Column(s) selected are not.

*If your database is validated as Unknown, see **Application Options>**"SQL Tab" for information on how to add a database.

Substitutions

Substitutions allow you to build open-ended displays in which data attachments depend on values defined at the time the display is run. Generic names, such as $table1 and $table2, are used instead of specific values. Later when the display is running, these generic values are defined by the actual names, such as production_table and system_table. In this way, a single display can be reused to show data from a number of different databases. For more information on creating displays using substitution values, see "Substitutions".

RTViewDs

The RTViewDs database contains two tables that contain metrics about SQL database connections and queries. In the **Attach to SQL Data** dialog, select **RTViewDs** from the **Database Name** drop down menu and then select either **Connections** or **Keys** from the **Table Name** drop down menu.

Note: RTViewDs tables do not support SQL queries, row filters, custom query intervals or maximum row settings.

The selected **Update Mode** affects the **Keys** table as follows:

Run Query Once	Static SQL data attachments are never removed from the Keys table.
Run Query Every Update Period or Query Interval	Periodic SQL data attachments are removed from the Keys table when the object with the attachment is deleted (e.g., when the display containing the object is closed).
Run Query On Demand	SQL data attachments are removed from the Keys table when they have completed.
Run Query On Schedule	SQL data attachments are updated according to the schedule.

RTViewDs Table Name	Column Available	Description
Connections	**Database**	Name of database connection.
	Connected	**True / False**. Indicates whether connected.

	Concurrent	**True / False**. Indicates whether Run Queries Concurrently option is selected for this connection.
	Query Object Count	Number of queries currently defined on this connection.
	Total Execution Count	Number of queries executed on this connection since startup.
	Listener Count	Number of data attachments using this connection. **Note:** In a Data Server deployment, this count indicates the total number of unique queries using the connection.
	Enabled	**True / False**. Indicates whether enabled.
Keys	**Key**	SQL data attachment string. This string begins with the database name, followed by various query parameters and finally the query string itself.
	Active	**True / False**. Indicates that query was either executing or waiting to be executed when the **Keys** table was last updated. A query may be waiting to execute if the database connection it uses is busy performing other queries. **Note:** For queries that are scheduled to run every General Update Period, Active and Running columns may always be checked since those queries will always be executing at the same time the Keys table is updated. See "General Tab" for more information.
	Running	**True / False**. Indicates that query was executing when the Keys table was last updated. **Note:** For queries that are scheduled to run every General Update Period, Active and Running columns may always be checked since those queries will always be executing at the same time the Keys table is updated. See "General Tab" for more information.
	Execution Count	Number of times query has been executed.
	Last Query Time	Time at which the most recent execution of query completed.
	Last Execution Time	Duration (in seconds) of the most recent query execution.
	Last Row Count	Number of rows in the most recent query result. Otherwise, -1 if the query failed.
	Last Process Time	Elapsed time between when query was scheduled and when it completed. This may be longer than the Last Execution Time if other queries use the same database connection and this query was delayed waiting for other queries to complete.

Last Query Status	Status of most recently completed execution of query. **OK** -- Query executed successfully. **no query** -- Query string is blank. **error** -- Error occurred while executing the query **connection failed** -- Connection to database was lost. **no connection** -- Database connection undefined or invalid. **timeout** -- Query timed out. **<blank>** -- Query has never been executed.
Avg Execution Time	Average of **Last Execution Time** for this query.
Avg Process Time	Average of the **Last Process Time** for this query.
Listener Count	Number of data attachments using this query result.
Query ID	ID string assigned to this query, if any.
Database	Name of the database used by this query.
Schedule	Name of the schedule, if any, configured for the query.
Next Query Time	The next time the query will be run according to its schedule or, if the query does not have a schedule, the next time it will be run according to its configured query interval. For Static and On Demand queries, this field will be populated with 1/1/1970 GMT.

Select Table Columns

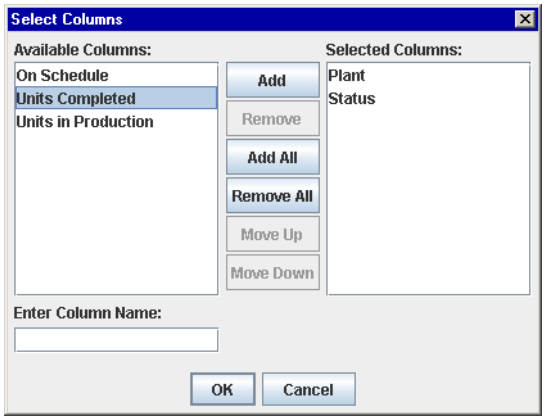
From the **Attach to SQL Data** dialog you can specify which table columns to display and in what order they will appear. In order to populate the listing of available columns, you must first select a valid database and table.

To bring up the **Select Columns** dialog, click on the ellipses button  in the **Column(s)** field (or right-click in the **Column(s)** field and click on **Select Column(s)**. The dialog should contain a list of Available Columns that you can add to your table.

To add a column, select an item from the **Available Columns** list and click on the **Add** button. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of columns in a table by arranging the items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

Validation colors indicate whether selected columns are valid. However, if even one column selected is invalid the **Column(s)** field in the **Attach to SQL Data** dialog will register as an invalid entry.

Note: Invalid columns will not update.



The following describes the **Attach to SQL Data** dialog commands:

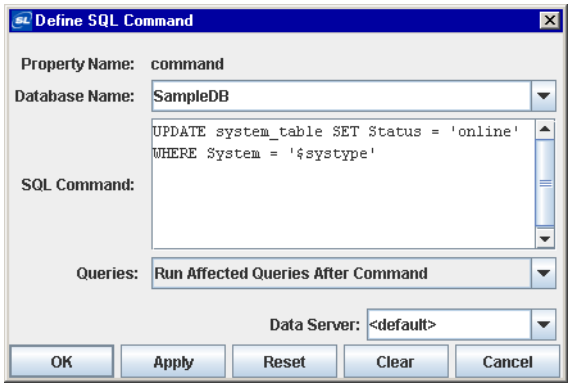
Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from database (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied.

Define SQL Command

Note: The SQL data source may not be licensed in your RTView installation.

The **Define SQL Command** dialog, which is used to assign SQL commands that allow you to issue commands from within an RTView display, can be accessed via the **Object Properties** window. If you execute a SQL command from a Thin Client with Direct Data Connection or any Served Data deployment, the command will execute on the server.

Right-click on the appropriate command property in the **Object Properties** window and select **Define Command>SQL** to display the **Define SQL Command** dialog. The **Define SQL Command** dialog provides a drop down menu with available databases and a field to enter a SQL statement. See the "Define/Execute Command" section for information on how to execute a command.



Field Name	Description
Database Name	Name of database to query. To attach the Database Name to data, right-click and choose **Attach to Data** or double-click in the field.
SQL Command	Enter a SQL statement to execute using standard SQL syntax. **Note:** This option is for advanced users, SQL syntax will not be validated or checked for errors. To attach the SQL Command to data, right-click and choose **Attach to Data** or double-click in the field. See the "Special Values" section for details on SQL commands to use to disable/enable a database. For example, if a database server needs to be shutdown for maintenance, that database can be disabled to avoid repeated connection attempts/failures from RTView. Or disable a database connection that has become unreliable, and then enable again to make a new connection.
Queries	If **Run Affected Queries After Command** is selected, RTView immediately runs queries (except for those queries with **Update Mode** set to **Run Query On Demand**) that use the database table modified by the command, which results in table changes being displayed immediately rather than waiting for the next scheduled query update. This option is only supported for update, insert, and delete operations in which the name of the database table to be modified is specified explicitly. If a command performs another SQL operation (such as running a stored procedure which modifies tables), the results of the operation will not be displayed until the next scheduled update of each affected query. Display of the modified data may be delayed for other reasons, for example, if the database does not commit the results immediately and instead returns the old data on the next query.

Data Server	Select to read data through your configured Data Server and not directly from the SQL data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab". **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". **Multi-Server Command** - When multiple data servers are specified, the command will be executed on each data server in the list. To configure multiple data servers, enter a semicolon (;) delimited list containing two or more **Named Data Servers** (e.g. ds101;ds102). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**). **Note:** The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

The **Database Name** drop down menu lists all available databases. The **Database Name** field automatically displays the name of the default database. If the item you require is not listed, type your selection into the field. Drop down menus populate based on databases added from the **Application Options** dialog or those typed directly into the **Database Name** field.

Note: If you are using a direct JDBC connection, you need to add your database in "Application Options - SQL".

Validation Colors

The Database Name field changes colors according to the information entered. These colors indicate whether or not information is valid. Information entered into the dialog is validated against the selected database or the Database Repository file. See "Application Options - SQL" for information on how to create a Database Repository file.

Note: The SQL Command field is not validated.

The following describes the significance of the **Define SQL Command** validation colors:

Blue	Unknown	Entry does not match any known database (or you have not attempted a connection*).
Yellow	Offline	Not connected to specified database.
White	Valid state	Database name is valid.

*If your database is validated as Unknown, see "Application Options - SQL" for information on how to add a database.

Note: If you are using a direct JDBC connection, you will need to add your database in **Application Options**.

Substitutions

Substitutions allow you to build open-ended displays in which data attachments depend on values defined at the time the display is run. Generic names, such as **$table1** and **$table2**, are used instead of specific values. Later when the display is running, these generic values are defined by the actual names, such as **production_table** and **system_table**. In this way, a single display can be reused to execute commands on a number of different databases. For more information on creating displays using substitution values, see "Substitutions".

Special Values

rtvEnableDB=false	Disable database. In the **SQL Command** field enter: **rtvEnableDB=false**. **Note:** This string must be entered exactly as shown, with no embedded spaces. When a database is disabled, RTView will not attempt to make any queries on that database or attempt to reconnect until the database is enabled again.
rtvEnableDB=true	Enable database. In the **SQL Command** field enter: **rtvEnableDB=true**. **Note:** This string must be entered exactly as shown, with no embedded spaces.
rtvEnableDB=false; rtvEnableDB=true	Disable database and immediately enable it again. In the **SQL Command** field enter: **rtvEnableDB=false;rtvEnableDB=true**. **Note:** This string must be entered exactly as shown, with no embedded spaces.
$value	When an **actionCommand** is executed **$value** is replaced with the value from the control. This value may be used in any field in the **Define SQL Command** dialog. **Note:** This value may only be used for Action Commands. See "Define/ Execute Command" for more information.

The following describes **Define SQL Command** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from assigned command (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Application Options - SQL

Select **Tools>Options** in the Display Builder to access the **Application Options** dialog.

Options specified in the SQL tab can be saved in an initialization file (**OPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server, and Historian to set initial values. If no directory has been specified for your initialization files and **OPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

Note: Options specified using command line arguments will override values set in initialization files. See "SQL Data Source - Command Line Options" for more information.

SQL Tab

This tab allows you to add or remove your databases and set the default database. In order for RTView to communicate with your databases, you must set up a direct JDBC connection. See "SQL Database Connection Setup" for more information.

When you add a database to the list, it will be highlighted in yellow indicating that it is not connected. To attempt to connect to a database, click **OK**, **Apply**, or **Save**. If the background remains yellow, then RTView was unable to make a connection to your database.

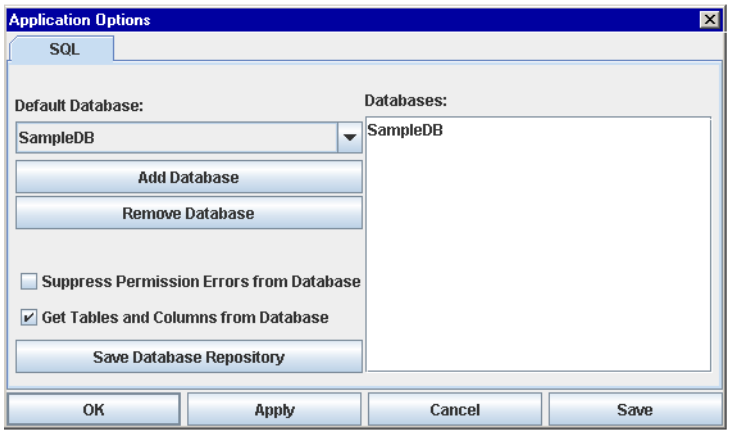
Note: Databases that have been setup to **Use Client Credentials** will not connect unless you are logged in and you have objects in your display that are using that connection.

Check your database connection and see "SQL Database Connection Setup" for information on how to set up your driver correctly.

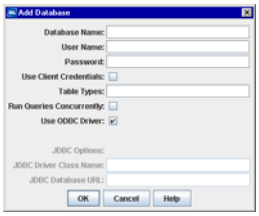
If the connection is successful, and the **Get Tables and Columns from Database** check box is selected, RTView will use information from this database to populate drop down menus in the "Attach to SQL Data" dialog with available tables and columns. If a database repository is found, information from your database will be merged with data from the repository file. If you deselect the **Get Tables and Columns from Database** check box, RTView will no longer query your database for this information but the database repository will still be used to populate drop down menus. Using a database repository to populate drop down menus makes it possible to specify which tables and columns from your database will be listed in the **Attach to Data** dialog and gives you the ability to build displays while databases are offline.

If you are using a direct JDBC connection, you must click **Save** in order to record your options in **OPTIONS.ini**. This will allow RTView to reconnect with your database the next time you run the Display Builder or the Display Viewer.

Note: Regardless of which tab you are currently working from in the **Application Options** dialog, each time you click **OK**, **Apply**, or **Save**, RTView will attempt to connect to all unconnected databases except those that have **Use Client Credentials** checked.



Field Name	Description
Default Database	Name of database used as the default for data attachments. Select from drop down menu to change default setting.
Add Database	Click to open the **Add Database** dialog. To edit, select a database from the list and double-click. Databases that are updating objects in a current display cannot be renamed.



Database Name -- Name to use when referencing this database connection in your data attachments.

User Name -- User name to pass into this database when making a connection. This parameter is optional.

Password -- The password to pass into this database when making a connection. This parameter is optional.

If you need to provide an encrypted password (rather than expose server password names in a clear text file, use the **encode_string** command line option with the following syntax:

encode_string type mypassword

where **type** is the key for the data source and **mypassword** is your plain text password.

Note: The **type** argument is only required when you encrypt a string for a data source.

For example, enter the following in an initialized command window (see "Initializing a Command Prompt or Terminal Window"):

encode_string sql mypassword

and you will receive an encrypted password:

encrypted value:
0134301355013460133101349013530134501348013 34

Copy the encrypted value, paste it into the password field and click **Save** to save this value to the initialization (***.ini**) file. Or, if necessary, manually edit the (***.ini**) file to include the encrypted value.

Note: If you need to manually edit a configuration (***.ini**) file, contact SL Technical Support at support@sl.com for information about supported syntax.

Use Client Credentials -- If selected, the User Name and Password from the RTView login will be used instead of the User Name and Password entered in the **Add Database** dialog. Connections to this database will only be made when you are running with login enabled and a display is opened that accesses this database. See "Login" for more information.

As a result, this connection will not be made when you click **OK** or **Apply** in the **Application Options** dialog and will remain yellow. If you will be using the Data Server or the Display Server with a database connection that has this option enabled, you must enable **Use Client Credentials for Database Login** in these applications.

Table Types -- Specify the types of tables to retrieve when querying the database for available tables. Refer to your database manual for a list of valid table types. This parameter is optional. Table types are entered as a comma delimited list, e.g., TABLE, VIEW.

Run Queries Concurrently -- If selected, each query on the connection is run on its own execution thread. The default is disabled.

Note: This option should be used with caution since it may cause SQL errors when used with some database configurations and may degrade performance due to additional database server overhead. See your database documentation to see whether it supports concurrent queries on multiple threads.

JDBC Options

JDBC Driver Class Name -- Fully qualified name of the JDBC driver class to use when connecting to this database. The path to this driver must be included in the **RTV_USERPATH** environment variable.

JDBC Database URL -- Full database URL to use when connecting to this database using the specified JDBC driver. Consult your JDBC driver documentation if you do not know the database URL syntax for your driver.

Note: If a value of **__none** (two underscores before the n) is entered for the JDBC Driver Class Name or JDBC Database URL, then the SQL data source will not attempt to connect to the database and ignore any SQL data attachments that specify that database. The **RTViewDs.Connections** table will contain a row for the database, but the Connected and Enabled columns will always show a value of false. If the **-sqltrace** command line option is used, then you'll see the following message appear once in the console:

<dbname>: ignoring connection with driver/url=__none

Remove Database Select a database from the list and click **Remove Database** to delete. Databases that are updating objects in a current display cannot be removed.

Suppress Permission Errors From Database If selected, SQL errors with the word "permission" in them will not be printed to the console. This is helpful if you have selected the **Use Client Credentials** option for a database. In this case, your login does not allow access for some data in their display, you will not see any errors.

Get Tables and Columns from Database	If selected, information from your database will automatically populate drop down menus in the **Attach to Data** dialog and you will be able to select from available tables and columns in your database. **Note:** If a database repository is found, information from your database will be merged with data from the repository file.
Save Database Repository	Click to save a file that records available tables and columns in your database and applies values to drop down menus in the **Attach to Data** dialog.

Database Repository

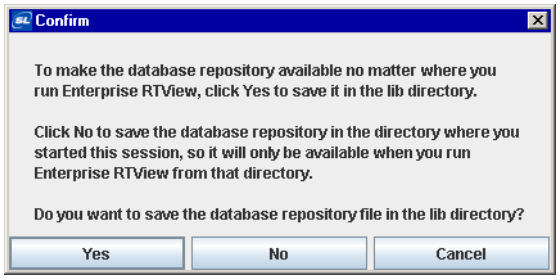
Click **Save Database Repository** to save a file that contains available information for tables and columns in your database. Before saving a database repository, you must add the database(s) from which the file will retain information.

Note: If RTView does not make a connection with your database, then information from that database cannot be saved to the database repository file.

Information stored in the database repository file will be used to populate the initial values of drop down menus in the **Attach to Data** dialog.

Note: The saved file will be named **sqlrepository.xml**. If the name of the database repository file is changed, RTView will not be able to locate the file. As a result, drop down menus will populate based on databases added from the **Application Options** dialog or those typed directly into the **Attach to Data** dialog.

When you click **Save Database Repository**, a confirmation dialog will appear to verify in which directory you would like to save the database repository file. If you specified a directory for your initialization files, all repository files will be saved to, and read from, that directory. If you select the **lib** directory, the repository file will be available from any directory where you run RTView. If you do not select the **lib** directory, the repository file will be saved in the directory where you started the current session and will only be available when you run RTView from that particular directory. See "RTV_JAVAOPTS" for more information.



See "SQL Database Connection Setup" for details on editing an existing database repository file.

Excluding Tables From The Attach To SQL Data Dialog

To exclude tables from the **Attach to SQL Data** dialog, see "SQL Database Connection Setup" for details.

RTView Deployment - SQL

This section contains details about the deployment process that are specific to your data source. Please go to the Deployment section of this documentation for instructions on how to implement your RTView deployment option. Return to this page whenever you are instructed to refer to deployment information that is specific to your data source.

System Requirements and Setup

The SQL data source has additional System Requirements and Setup. See "SQL System Requirements and Setup" for more information.

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options", "Application Options - SQL", and "RTV_JAVAOPTS" for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
OPTIONS.ini	Contains general options as well as data source options for SQL.

Note: Options specified using command line parameters override values set in initialization files.

SQL Demos

Except where noted, all demos can be run in three ways, as an application, or via rich or thin client in a browser.

Before You Begin

Start the Demo Server

Thin Client Demo only.

There is a thin client demo already installed on the "RTView Demo Server".

In an initialized command window (see "Initializing a Command Prompt or Terminal Window"):

Type **run_startup_demoserver**

Data Source Demo

The Data Source Demo is designed to illustrate each data source.

1. **Setup Sample SQL Database.** See "SQL Database Connection Setup" for more information.

2. **Run Demos - Application or Thin Client Browser**

   Application Demo

   1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
   2. To view the demo, type:

      run_viewer

   3. To edit the demo, type:

      run_builder

   Thin Client Browser Demo

   "Start the Demo Server" if it is not running.

   1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
   2. Start the Display Server by typing:

      run_displayserver

   3. Open a browser and navigate to **http://localhost:8068/dstutorial**.

ElectroSphere Demo

The ElecroSphere demo, a Business Information Application, allows you to navigate through multiple types of business data.

Note: This demo requires Windows and Microsoft Access.

---

1. **Run Demos - Application or Thin Client Browser**

Application Demo

On Windows:

In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/esphere** directory.

1. To view the demo, type:

   run_viewer

2. To edit the demo, type:

   run_builder

Thin Client Browser Demo

"Start the Demo Server" if it is not running.

On Windows:

In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/esphere** directory.

1. Start the Display Server by typing:

   run_displayserver

2. To view the demo open a browser and navigate to **http://localhost:8068/esphere**.

Quick Start Tutorial - SQL

This Quick Start Tutorial provides you with the fundamentals on how to use RTView with the SQL data source. Once completed, you can swiftly apply this knowledge to building your own dashboard displays for visual access to your SQL data.

Learn to:

- Animate graphic objects with SQL data
- Create a drill down display with SQL data

Get Started

This tutorial requires the following:

- Microsoft Access 97+.
- Setup the sample database - If you have not, you must setup the sample database used in this tutorial. See and "SQL Database Connection Setup" for information.
- Register for a license key. If you have not, you must do so before continuing. See "Registration" for more information.
- "Quick Start Tutorial" - This tutorial requires that you have a working knowledge of RTView. We recommend that you complete the Quick Start Tutorial before continuing.

Start the Display Builder

If you are already logged onto the Display Builder, skip this section and go to "Create a Display".

1. In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"), type:

   run_builder

2. Login to the Display Builder. By default, the Display Builder does not require a login. "Login" can be enabled at setup to support "Role-based Security". The default user name and password are:

   User Name: **admin**

Password: **admin**

Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role. See "Role-based Security" for more information.

You are now ready to create a display using the SQL data source.

Create a Display

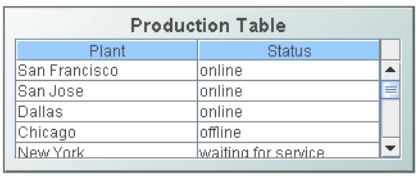
At this point you have:

- Registered for a license key. See "Registration" for more information.
- Logged on to the Display Builder
- Setup the SQL sample database.
- Completed the "Quick Start Tutorial".

In this tutorial you use the sample SQL database as a data source to create a Production Table that displays production numbers per plant, as seen below.

As you saw in the "Quick Start Tutorial", the data structure of tables and graphs (tabular data) enables RTView to automatically create several data source specific, built-in Substitutions for you. You will see these built-in Substitutions used in the target display when you create the drill down. For more information on Substitutions, see "Substitutions".

In this exercise, you create a drill down using the previously created display, **sql_dd_qs.rtv**, as the target display. First you will set the Production Table to display production status per plant. Then you will create a drill down that opens a bar graph that shows more detailed data for each plant.



Add the SQL Database To List of Available Databases

Adding the sample SQL database to the Database List makes it available for animating graphic objects in your display.

1. In the Display Builder, select **Tools>Options** to open the **Application Options** dialog.
2. Select the **SQL** tab and click on **Add Database** to open the **Add Database** dialog.
3. In the **Add Database** dialog:

   Database Name - Enter **SampleDB**
4. Click **OK** to close **Add Database** dialog.
5. Click **OK** to apply and close the **Application Options** dialog.

The sample SQL database is now available for animating graphic objects in your display.

Display Data in a Table

In this exercise you add a table and then display data in the table by attaching it to the data source.

1. Click on the Add Table button and click again in the Working Area to place the table.
2. In the **Object Properties** dialog:

   autoResizeFlag (category: Column) - Click to select the check box.

   label (category: Label) - Change the name of the label to **Production Table**. Press <Enter>.

   valueTable (category: Data) - Right-click in the Property Name field and select **Attach to Data>SQL**.
3. In the "Attach to SQL Data" dialog:

   Database Name - **SampleDB** should already be selected.

   Table Name - Select **production_table**.

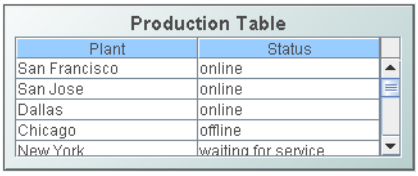
   Column(s) - Select the ... button to open the **Select Columns** dialog.
4. In the **Select Columns** dialog:

   Select **Plant** in the **Available Columns** list and click **Add**.

   Select **Status** in the **Available Columns** list and click **Add**.

   Click **OK** to close the **Select Columns** dialog.
5. In the "Attach to SQL Data" dialog leave the **Filter** field blank.
6. Click **OK** to apply these values and close the **Attach to SQL Data** dialog.

The table populates with values from **production_table** in the sample SQL database.



7. In the Display Builder, select **File>Save**.

You are now ready to create the drill down.

Create a Drill Down Target in the Table

In this exercise, you create the drill down using the previously created display, **sql_dd_qs.rtv**, as the target.

1. In the **Object Properties** dialog:

   drillDownTarget (category: Interaction) - Double-click in the **Property Name** field to bring up the **Drill Down Properties** dialog.

2. In the "Drill Down Properties" dialog:

   Apply Drill Down To - Select **Named Window** from the drop down menu. This option lets you re-use the window when you drill down multiple times.

   Window Name - Enter **sql**. This name should be unique unless the display is to open in an existing window.

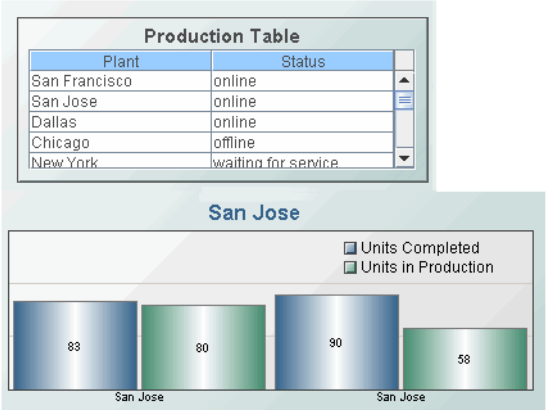
   Drill Down Display Name - Select **dstutorial\sql_dd_qs.rtv** from the drop down menu.

3. Click OK to set the drill down target and close the **Drill Down Properties** dialog.

View the Drill Down Display

In this exercise, you drill down to the target display.

1. Double-click on any cell in the table to drill down to detailed data. The target display opens.



2. Double-click on another row in the table and the same display is used to show different data based on the row you select.

3. Close the drill down window.

4. In the Display Builder select **File>Save**.

Go to the "Quick Start Tutorial"

SQL Database Connection Setup

RTView communicates with your database using either a direct JDBC connection. Connections require some setup before RTView can communicate with your database.

Once you have setup your database connection, you will need to add your database in the Display Builder from the **Application Options** dialog on the "SQL Tab". RTView will attempt to connect to your database. If RTView is unable to connect to your database, this means that either the driver is not setup correctly or that you do not have permission to access the database.

Note: Databases that have been setup to **Use Client Credentials** will not connect unless you are logged in and you have objects in your display that are using that connection. See **Application Options>**"SQL Tab".

If the connection is successful, and the **Get Tables and Columns from Database** check box is selected in the **Application Options** dialog, RTView will use information from this database to populate drop down menus in the "Attach to SQL Data" dialog with available tables and columns. If a Database Repository is found, information from your database will be merged with data from the repository file. If you deselect the **Get Tables and Columns from Database** check box, RTView will no longer query your database for this information, but the Database Repository will still be used to populate drop down menus. Using a Database Repository to populate drop down menus makes it possible to specify which tables and columns from your database will be listed in the **Attach to Data** dialog and gives you the ability to build displays while databases are offline.

Direct JDBC Connection

In order for RTView to communicate with your database using a straight JDBC connection, you must have a JDBC driver for your database. JDBC drivers are available from most database vendors. To make your database driver available to RTView, locate the driver on your machine and define an environment variable named **RTV_USERPATH** so that it includes the path to the driver class or the jar that contains the driver class. The **RTV_USERPATH** variable is included in the classpath for RTView. It may contain paths to multiple driver classes. You must know the fully qualified class name for the driver class and the database URL required to connect to your database when you add the database in the **Application Options** dialog. The database URL typically contains the protocol and sub-protocol strings for your database as well as the path to the database and a list of properties. If you do not know the syntax for your database URL, consult the documentation for your JDBC driver.

Database Repository File

A Database Repository file may be used to populate the initial values of drop down menus in the "Attach to SQL Data" dialog. See "Application Options - SQL" for information on how to create a Database Repository file.

It is possible to edit an existing Database Repository file, however the file name **sqlrepository.xml** cannot be modified. If **sqlrepository.xml** is not found in the specified directory or your current working directory, RTView will look in the **lib** directory. If the Database Repository file is not found, drop down menus will remain empty until databases are added from the **Application Options** dialog or typed directly into the **Attach to SQL Data** dialog. See "RTV_JAVAOPTS" for more information.

To edit an existing Database Repository file, supported tags and attributes are as follows:

Tag	Attribute		Description
sqlrepository	xmlns		Top level tag that includes the namespace attribute **xmlns**, which must be defined as www.sl.com (**xmlns="www.sl.com"**)
db	name		Database name
	table	name	Table name
		column	Possible column value

An example Database Repository file:

```
<?xml version="1.0"?>
<sqlrepository xmlns="www.sl.com" version="3.0">
        <db name="SampleDB">
                <table name="production_table">
                        <col>Plant</col>
                        <col>Units in Production</col>
                        <col>Units Completed</col>
                        <col>Status</col>
                        <col>On Schedule</col>
                </table>
                <table name="system_table">
                        <col>System</col>
                        <col>Status</col>
                        <col>%Free Space</col>
                        <col>CPU Usage</col>
                       <col>On Site</col>
                </table>
                <table name="trade_table">
                        <col>Customer</col>
                        <col>Symbol</col>
                        <col>Shares</col>
                        <col>Purchase Price</col>
                        <col>Current</col>
                        <col>High</col>
                        <col>Low</col>
                </table>
        </db>
</sqlrepository>
```

Excluding Tables From The Attach To SQL Data Dialog

To exclude tables from the **Attach to SQL Data** dialog, copy the **sqlrepository.xml** file to a new **sqlexcludedtables.xml** file and remove the table references that you want to include in the **Attach to SQL Data** dialog drop down menus. See "Database Repository File" for more information. For example:

```
<?xml version="1.0"?>
<sqlrepository xmlns="www.sl.com" version="3.0">
        <db name="SampleDB">
                <table name="production_table">
                        <col>Plant</col>
                        <col>Units in Production</col>
                        <col>Units Completed</col>
```

```
                    <col>Status</col>
                    <col>On Schedule</col>
                </table>
        </db>
</sqlrepository>
```

Save the **sqlexcludedtables.xml** file to your **RTV_HOME\lib** directory. Table information stored in **sqlexcludedtables.xml** will now be excluded from the initial values of **Table Name** drop down menus in the **Attach to SQL Data** dialog.

Note: It is not necessary to create a Database Repository file in order to use **sqlexcludedtables.xml**. RTView will still use the **sqlexcludedtables.xml** file to exclude tables from the **Attach to SQL Data** dialog. If you have an **sqlexcludedtables.xml** file and you click the **Save Database Repository** button, the new **sqlrepository.xml** file will not contain any of the tables listed in your **sqlexcludedtables.xml** file.

To create your own **sqlexcludedtables.xml** file without creating a Database Repository File, supported tags and attributes are as follows:

Tag	Attribute		Descriptions
sqlexcludedtables	xmlns		Top level tag that includes the namespace attribute xmlns, which must be defined as www.sl.com (**xmlns="www.sl.com"**)
db	name		Database name
	table	name	Table name to exclude

Note: The file name **sqlexcludedtables.xml** cannot be modified.

SQL Data Source - Command Line Options

In addition to General Options, the following command line arguments are enabled with the SQL data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: "**-sub:$data:my Data**").

Name	Description
-dbfailedlimit:(number of failed queries)	Specify the number of consecutive failed SQL queries after which to close this database connection and attempt to reconnect. Default is **-1**, which disables this feature. Example: **-dbfailedlimit:20**

-dbretry:(milliseconds)	Specify the interval (in milliseconds) to retry connecting to a database after an attempt to connect fails. Default is **-1**, which disables this feature. Example: **-dbretry:30000**
-nopermerrors	SQL errors with the word "permission" in them will not be printed to the console. This is helpful if you have selected the **Use Client Credentials** option for a database. In this case, your login does not allow access for some data in their display, you will not see any errors. See "Application Options - SQL" for more information. Example: **-nopermerrors**
-noquerydbinfo	RTView will no longer query your database for available tables and columns in your database. If a Database Repository file is found, it will be used to populate drop down menus in the **Attach to SQL Data** dialog. Example: **-noquerydbinfo**
-smallintbooleans	Set to interpret the SQL SMALLINT data type as a BOOLEAN, so that tables display a check box for columns of this type. Example: **-smallintbooleans**
-sqlquerytimeout:(seconds)	Specify the default timeout (in seconds) for all SQL queries. Example: **-sqlquerytimeout:90**

-sqlquote	Encloses all table and column names specified in the **Attach to SQL Data** dialog in quotes when an SQL query is run. This is useful when attaching to databases that support quoted case-sensitive table and column names. **Note:** If a case-sensitive table or column name is used in the **Filter** field, or you are entering an advanced query in the **SQL Query** field, they must be entered in quotes, even if the **-sqlquote** option is specified. Example: **-sqlquote**
-sqltryodbc: (true/false)	Specifies whether or not an ODBC connection attempt will be made when a SQL query or command is executed using a database that is not defined in **OPTIONS.ini**. This option is not recommended and is intended only for backward compatibility with older RTView releases. This option is **false** by default and, if an undefined database name is encountered, then no connection attempt is made and the following message appears in the console: **Undefined SQL database XYZ** where **XYZ** is the name of the undefined database. Set this option to **true** to force the SQL data source to attempt an ODBC connection when an undefined database is referenced. **OPTIONS.ini Example:** **sqltryodbc true** **Properties File Example:** **sl.rtview.sql.sqltryodbc=true** **Command Line Example:** **-sqltryodbc:true** Note that in Java 1.8 and newer, the ODBC driver is not supported and RTView will not make any ODBC connection attempts, regardless of the option.

StreamBase Data Source

StreamBase® Event Stream Processing is a CEP (Complex Event Processing) platform that empowers organizations to process, analyze and act on real-time streaming data within milliseconds of its arrival. CEP solutions are capable of reading, creating, transforming, correlating or abstracting complex data events. A CEP platform provides the necessary performance for solutions that demand high-speed correlation of streaming data (i.e.: financial services, fraud detection, IT quality of service [SLAs], eCommerce monitoring and location based services).

With RTView you can connect directly to StreamBase® event streams and create real-time business activity monitoring applications, such as:

- a financial trading application to monitor real-time trade activity;
- a fraud detection dashboard and alerting system that would monitor on-line financial services; or
- a web-based monitoring system that would show business activity and analyze behavior on an eCommerce web site.

In typical solutions, users define CEP using the StreamBase® tool set and then utilize RTView to:

- visualize the resulting information via dashboards;
- archive information for historical trend analysis; and/or
- activate automated user behavior and create reports.

The StreamBase® data source also provides metrics from the StreamBase® platform, so the status and current operating levels of the platform can be monitored for capacity planning.

This section includes:

StreamBase System Requirements and Setup

System Requirements

In addition to basic "System Requirements", the StreamBase data source requires StreamBase and is only available if you include your StreamBase client jar in **RTV_USERPATH**. See the README_sysreq.txt file in your installation's home directory for the current version(s) supported.

Setup

In addition to general environment variables (see "Setup"), you must include the StreamBase client jar in **RTV_USERPATH**:

Name	Description	Example
RTV_USERPATH	Path for the StreamBase **sbclient.jar**. **Note:** If **RTV_USERPATH** already exists, append **sbclient.jar** to it.	**C:\Program Files\StreamBase Systems\StreamBase\lib\sbclient.jar**

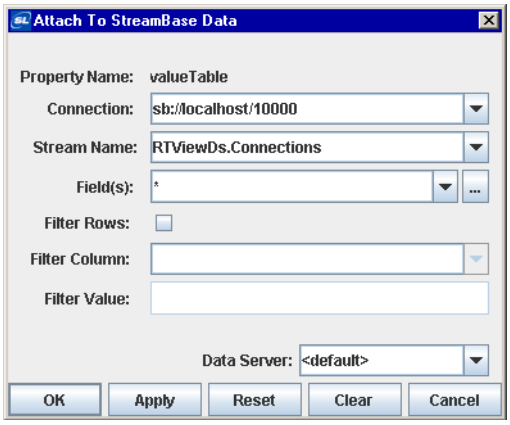
See StreamBase provider documentation for more information on the location of this jar.

Attach to StreamBase Data

Note: The StreamBase data source may not be licensed in your RTView installation.

Right-click on the Property Name (e.g., the value property or the valueTable property) from the **Object Properties** window and select **Attach to Data>SB** to display the **Attach to StreamBase Data** dialog. This dialog is used to attach an object property to StreamBase monitoring metrics or stream content and display real-time results in an RTView display. Once an object property has been attached it receives continuous updates.

The **Attach to StreamBase Data** dialog provides several drop down menus that allow you to specify information regarding the tuples you retrieve. If the drop down menu does not contain the item you require, type your selection into the text field.



Field Name	Description
Connection	The StreamBase connection name. You may specify a connection on the "StreamBase Connections Tab" of the **Application Options** dialog. **Note:** If ***** is entered, data will be retrieved from all defined connections.
Stream Name	The name of the stream from which you want content. Choose either a Stream Name or "RTViewDs". Available Stream Names are based on the Connection specified in the **Application Options** dialog. **Note:** All returned rows of data are identified by both Connection and Stream Name. In the returned table, Connection names will appear in a column named conn added to each incoming tuple. **Streams** -- Returns a table with a single row containing the latest tuple data. **History** -- Returns a table with one row for each tuple received. This table continuously grows to the maximum specified for History Depth in the **Application Options** dialog. Example: **streamname_history** **History2** -- Returns a table containing one row for each tuple received up to that point in time. Does not continue updating as new tuples come in. Example: **streamname_history2**
Field(s)	If **Stream**, **History**, or **History2** is selected in the **Stream Name** field, the **Field(s)** field dropdown menu is populated with the following options. For RTViewDs options, see the "RTViewDs" table.
Filter Rows	Check box to indicate whether or not to filter the rows.
Filter Column	Name of the column field to use as a filter.
Filter Value	Value that the filter field must equal.

Data Server

Select to read data through your configured Data Server and not directly from the StreamBase data source.

Default - Select the default Data Server you configured in **Application Options>**"Data Server Tab".

None - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source.

Named Data Servers - Select a **Named Data Server** that you configured in **Application Options>**"Data Server Tab".

Multi-Server Attachment - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more **Named Data Servers** (e.g. ds101;ds102). Each name specified must correspond with a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**).

Note: The values **__default** and **__none** begin with two underscore characters.

Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received.

A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways:

1. The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows.

2. The multi-server attachment can be applied as the **Table** argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions"

Drop down menus populate based on connections added from the **Application Options** dialog or those typed directly into the **Attach to StreamBase Data** dialog.

When an object property has been attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing values from the **Object Properties** window is no longer possible. To remove the data attachment and resume editing capabilities in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green

Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. A generic stream name such as $stream is used instead of a specific stream name. Later when the display is running, this generic value is defined by the actual name of a specific stream, such as SalesInfo. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

RTViewDs

The following special values can be entered for the **Stream Name** field:

RTViewDs	Column Available	Description
RTViewDs.Connections	Connected	True / False. Indicates whether connected.
	Status	Status information for this connection.
	StatusFull	Status information for this connection.
	Stream Count	Count of active streams in a connection.
	URI	The connection name.
	Version	StreamBase client software version currently running.
RTViewDs.Streams	Connected	True if the connection for this stream is connected.
	Last Tuple Time	The time RTView received the latest tuple on this stream.
	Stream	The name of the stream.
	Time Since Last Tuple	The amount of time in seconds that has passed since RTView received a tuple on this stream.
	Tuples Received	The total number of tuples received on this stream since RTView started. This does not include history.
	Tuples Total	The total number of tuples received on this stream since RTView started plus the total number of tuples from history.
	URI	The URI for the connection for this stream.
RTViewDs.OperatorInfo	Timestamp	The time that the Snapshot for this data was taken.
	URI	The URI for the connection for this monitor information.
	Name	The name of the operator.
	InputPortCount	The number of input ports on this operator.
	InputTuplesDequeuedDelta	The number of input tuples consumed by this operator since the previous Snapshot. If this number is less than the number of tuples enqueued to the input stream, then tuples are being queued up faster than the operator can consume them.
	InputTuplesDequeuedTotal	The total number of input tuples consumed by this operator since the StreamBase Server was started. If this number is less than the number of tuples enqueued to the input stream, then tuples are being queued up faster than the operator can consume them.

	InputTuplesEnqueuedDelta	The number of tuples enqueued on this operator's input streams since the previous Snapshot.
	InputTuplesEnqueuedTotal	The total number of input tuples enqueued on this operator's input streams since the StreamBase Server was started.
	OutputPortCount	The number of output ports on this operator.
	OutputTuplesEnqueuedDelta	The number of output tuples produced by this operator since the previous Snapshot.
	OutputTuplesEnqueuedTotal	The total number of output tuples produced by this operator since the StreamBase Server was started.
	PercentUsage	The number of profiler hits divided by the total profiler hits.
	ProfilerHits	The number of miniticks used by the operator, as defined by profiler samples.
	Size	The size of the operator. The definition of "size" varies by operator.
	Status	The status of the operator. When blank, this returns an empty string.
	TotalProfilerHits	The number of miniticks used by all operators in the contained ProfileState. The percentage usage is then profiler hits/total profiler hits.
RTViewDs.ThreadInfo	Timestamp	The time that the Snapshot for this data was taken.
	URI	The URI for the connection for this monitor information.
	Name	The name of the thread.
	PercentCPU	The percent of CPU used by this thread, in microseconds.
	PercentSystem	The percent of system CPU time used by this thread, in microseconds.
	PercentUser	The percent of user CPU time used by this thread, in microseconds.
	SystemTimeDelta	The system time spent by thread since the last Snapshot, in microseconds.
	SystemTimeTotal	The system time spent by this thread since the StreamBase Server started, in microseconds.
	UserTimeDelta	The user time spent since the last Snapshot, in microseconds.
	UserTimeTotal	The user time spent by thread since the StreamBase Server started, in microseconds.
RTViewDs.SystemInfo	DequeueClientCount	The current number of dequeueing clients.

FreeMemory	The amount of free memory in the StreamBase Server process, in bytes.
MaxMemory	The maximum amount of memory which the StreamBase Server process is allowed to allocate, in bytes.
Timestamp	The time that the Snapshot for this data was taken.
URI	The URI for the connection for this monitor information.
ProcessSize	The amount of memory used the StreamBase Server process, in Megabytes.
TimeRunning	The amount of time that the StreamBase Server has been running, in seconds.
TotalKB	The total amount of memory, in kilobytes, available for use in the StreamBase server process.
TotalMemory	The total amount of memory, in bytes, available for use in the StreamBase Server process.
FreeKB	The amount of free memory, in kilobytes, in the StreamBase server process.
maxKB	The maximum amount of memory, in kilobytes, which the StreamBase server process is allowed to allocate.
numDequeue	The current number of dequeuing clients.
UsedMemory	The total amount of memory currently in use in the StreamBase Server process, in bytes.

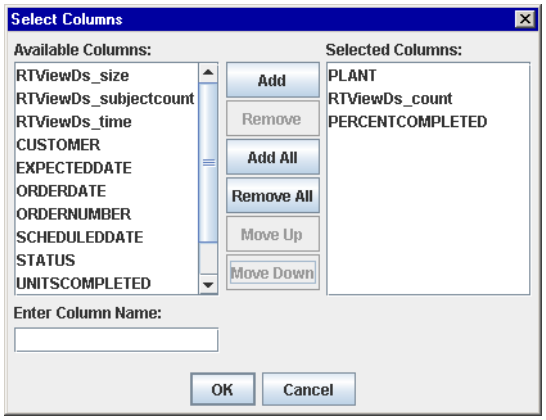
Select Columns Fields (Tables Only)

From the **Attach to Data** dialog you can specify which tuple fields to display as columns in a table and in what order they will appear. In order to populate the listing of available tuple fields, you must first select a valid stream name.

Click on the ellipses button  in the **Field(s)** field (or right-click in the **Field(s)** field and click **Select Columns**) to display the **Select Columns** dialog. The dialog contains a list of Available Columns that you can add to your table.

To add a field, select an item from the **Available Columns** list and click on the **Add** button. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of fields in a table by arranging the items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

If no data is available for a table row within a selected column, the table cell will display one of the following values: **N/A**, **false**, **0**, or **0.0**.



The following describes **Attach to StreamBase Data** dialog commands:

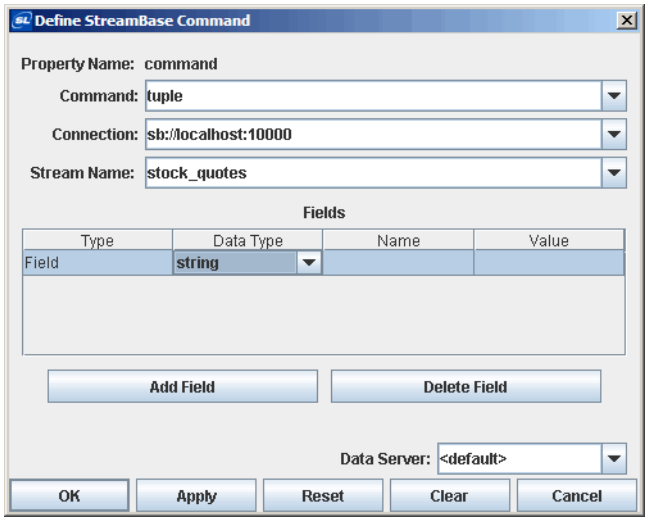
Field Name	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Define StreamBase Command

Note: The StreamBase data source may not be licensed in your RTView installation.

You can access the **Define StreamBase Command** dialog from the **Object Properties** window. This dialog is used to assign a StreamBase stream to an object's command property, giving you the ability to send tuples from within an RTView display. If you execute a StreamBase command from a Thin Client with Direct Data Connection or any Served Data deployment, the command will execute on the server.

To open the **Define StreamBase Command** dialog, right-click on the **command** property in the **Object Properties** window and select **Define Command>SB**. The information supplied assigns a stream to the command property. See the "Define/Execute Command" section for information on how to execute a command.



Field Name	Description
Command	**tuple** -- Sends a tuple.
Connection	Enter a connection name. You may define a connection on the "StreamBase Connections Tab" of the **Application Options** dialog.
Stream Name	The name of the stream you want content from. Available stream names are based on the connection specified.
Fields	**Type** - Select Field to define a message field. **Data Type** - Select the data type for this field or property. **Name** - Specify the name for this field or property. **Value** - Specify the value for this field or property.
Add Field / Property	Add a field.
Delete Field / Property	Delete the selected field or property.
Data Server	Select to read data through your configured Data Server and not directly from the StreamBase data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab"**.** **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". **Multi-Server Command** - When multiple data servers are specified, the command will be executed on each data server in the list.

	To configure multiple data servers, enter a semicolon (;) delimited list containing two or more **Named Data Servers** (e.g. ds101;ds102). Each name specified must correspond with a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**). **Note:** The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

Drop down menus populate based on connections added from the **Application Options** dialog or those typed directly into the "Attach to StreamBase Data" dialog.

Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. A generic stream name such as **$stream** is used instead of a specific stream name. Later when the display is running, this generic value is defined by the actual name of a specific stream, such as SalesInfo. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

Special Values

$value	When an **actionCommand** is executed, **$value** is replaced with the value from the control. This value may be used in any field in the **Define StreamBase Command** dialog. **Note:** This value may only be used for **Action Commands**. See "Define/Execute Command" for more information.

The following describes **Define StreamBase Command** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from assigned message subject (once **Apply** or **OK** is selected)
Cancel	Closes the dialog with last values applied.

Application Options - Streambase

Select **Tools>Options** in the Display Builder to access the **Application Options** dialog.

Options specified in StreamBase tabs can be saved in an initialization file (**SBOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server, and Historian to set initial values. If no directory has been specified for your initialization files and **SBOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

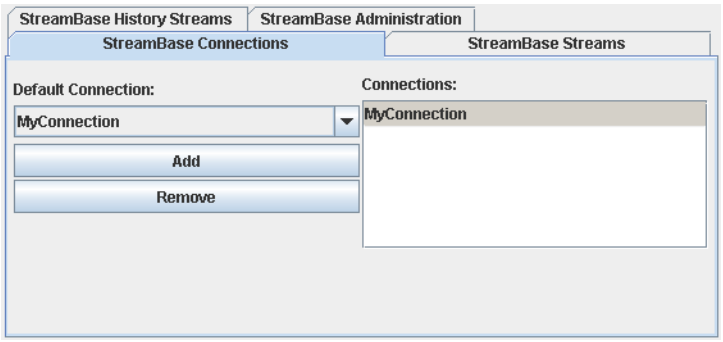
Note: Options specified using command line arguments will override values set in initialization files.

There are four Options tabs: "StreamBase Connections Tab", "StreamBase Streams Tab", "StreamBase History Streams Tab", and "StreamBase Administration Tab".

StreamBase Connections Tab

This tab allows you to add or remove connections and set your default connection. When you add a StreamBase connection to the list it will be highlighted in yellow indicating that RTView has not connected to it. To attempt to connect to a StreamBase connection, click **OK**, **Apply**, or **Save**. If the background remains yellow, then RTView was unable make a connection. Check that your StreamBase connection was setup correctly and that the StreamBase message server is running.

Note: Regardless of which tab you are currently working from in the **Application Options** dialog, RTView will attempt to connect to all unconnected connections each time you click **OK**, **Apply**, or **Save**.



Field Name	Description
Default Connection	Name of connection used as the default for data attachments. Select from drop down menu to change default setting.
Add Connection	Click to open the **Add Connection** dialog. To edit, select a connection from the list and double-click. Connections that are updating objects in a current display cannot be renamed.

Connection Name - The name to use when referencing this StreamBase connection in your data attachments.

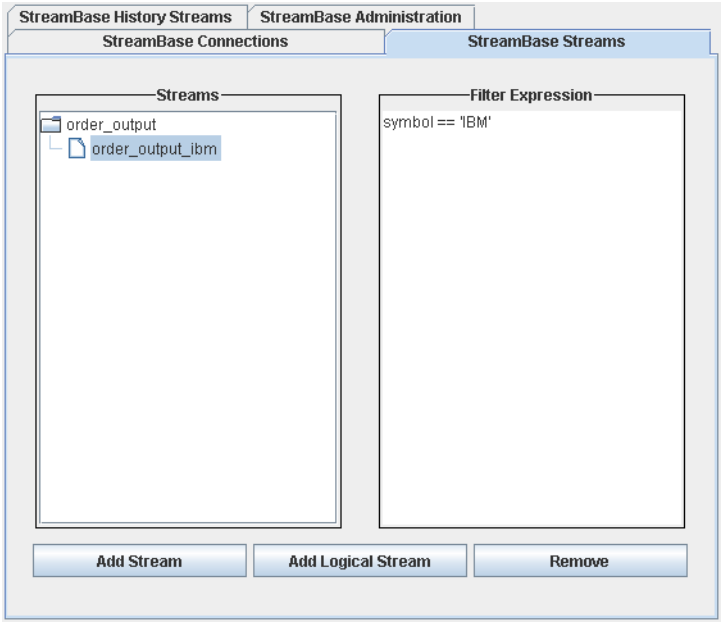
URIs - The complete URI of your StreamBase server. Multiple URIs can be entered, one per line, to connect to a StreamBase application running on multiple servers.

Dequeue Interceptor - Enter the name of a Java class. This class name must be added to the definition for the **RTV_USERPATH** environment variable. The Java class entered should include the processResult() method, which will be called by the StreamBase application in the course of dequeuing each tuple. Within the processResult() method you can make decisions about the dequeue process. For example, in an application where two servers are both dequeueing tuples but only one is designated the leader, you may choose to suppress the tuples from the non-leader server. For more information on StreamBase High-Availability features, go to *http://streambase.com/developers/docs/latest/admin/highavail.html*.

Remove Connection Select a connection from the list and click **Remove Connection** to delete. Connections that are updating objects in a current display cannot be removed.

StreamBase Streams Tab

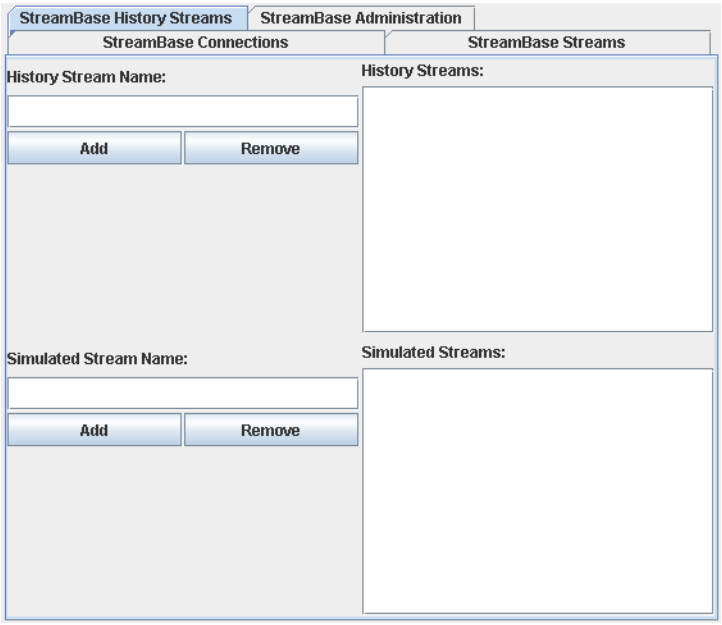
This tab allows you to configure streams, including any number of filtered logical streams. When RTView connects to a StreamBase server, it retrieves the list of available streams and then uses the Streams list you specify to control which subscriptions to make. Filtered logical streams will be applied on the server to each tuple before it is sent to the client and only tuples that pass through the filter will be sent on those logical streams, thus reducing the amount of data sent to the client.



Field Name	Description
Streams	Specify any number of logical streams for a given stream. When a logical stream is added there is no subscription to the physical stream, but rather to the logical stream with the filter expressions as specified. If no streams are specified, then RTView subscribes to all streams on each specified connection (URI). **Note:** If it is necessary to subscribe directly to the physical stream, then there are no performance benefits gained by creating logical streams. To filter the data, create one physical stream and use row filtering in the "Attach to StreamBase Data" dialog. **Add Stream** -- Click **Add Stream** and enter the name of a stream. Press **Enter** to add to the **Streams** list. **Add Logical Stream** -- Select a stream from the **Streams** list and click **Add Logical Stream**. Enter a name for the logical stream and press **Enter** to add to the **Streams** list. **Remove** -- Select a stream or logical stream from the list and click **Remove**.
Filter Expression	Select a logical stream from the list and enter filter expression. "Substitutions" may be used in filter expressions. **Note:** For a complete description of filter expression syntax, refer to your StreamBase documentation.

StreamBase History Streams Tab

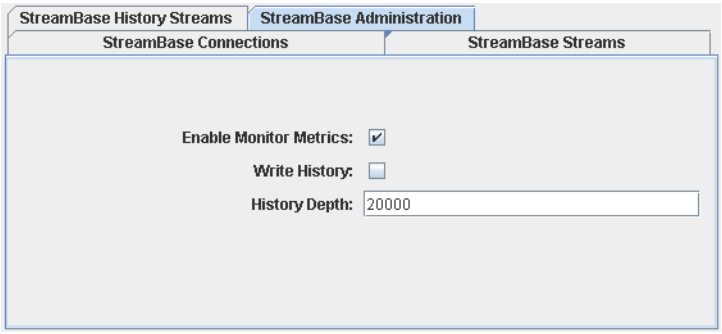
This tab allows you to configure history stream and simulated stream options.



Field Name	Description
History Streams	Enables the reading of the CSV file for the specified stream. Select **Write History** (on the "StreamBase Administration Tab") to create a CSV file. **Note:** In order for historical data to be available, the **DATA** subdirectory (located in the directory from which you launched RTView) must contain a CSV file for the stream. **Add** -- Enter the name of the stream for which you want to read a CSV file and click **Add** to include it in the **History Streams** list. **Remove** -- Select a stream from the list and click **Remove**.
Simulated Streams	For each stream name specified, the DATA subdirectory (located in the directory from which you launched RTView) must contain a CSV file named **stream_name.csv**. RTView will look for **stream_name.csv** on each update. If found, and the timestamp on the file has changed since the last time it was read, RTView will use the data in the file to simulate data for that stream. **Note:** The data in the CSV file must match the stream format defined in your **.sbapps** file. **Add** -- Enter the name of the stream that you want to simulate data for and click **Add** to include it in the **Simulated Streams** list. **Remove** -- Select a stream from the list and click **Remove**.

StreamBase Administration Tab

This tab allows you to setup administration options for the StreamBase data source.



Field Name	Description
Enable Monitor Metrics	Select to enable event driven statistics. These statistics will be polled every ten (10) seconds. Default is **enabled**. **Note:** Setting this parameter to true will increase traffic on your streams as the monitor information must be sent from StreamBase to RTView.

Write History	Select to enable the writing of a CSV file containing all the data for all tuples received for the streams specified in History Streams. CSV files are written to the **DATA** subdirectory, located in the directory from which you launched RTView. History Streams must be enabled for **Write History** to execute.
History Depth	The number of rows RTView keeps in the history table for each stream. Default is **20,000**.

RTView Deployment - StreamBase

This section contains details about the deployment process that are specific to your data source. Please go to the Deployment section of this documentation for instructions on how to implement your RTView deployment option. Return to this section whenever you are instructed to refer to deployment information that is specific to your data source.

System Requirements and Setup

The Streambase data source has additional "System Requirements" and "Setup".

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options", "Application Options - Streambase", and "RTV_JAVAOPTS" for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
SBOPTIONS.ini	Contains data source options for StreamBase.

Note: Options specified using command line override values set in these initialization files.

StreamBase Demos

Except where noted, all demos can be run in three ways, as an application, or via rich or thin client in a browser.

Before You Begin

Start the Demo Server

Thin Client Demo only.

There is a thin client demo already installed on the "RTView Demo Server".

In an initialized command window (see "Initializing a Command Prompt or Terminal Window"):

Type **run_startup_demoserver**

Data Source Demo

The Data Source Demo is designed to illustrate each data source.

1. **Start the Sample StreamBase Application**

   In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), navigate to the **demos/dstutorial** directory and type:

   run_sbsimapp

2. **Start the StreamBase Data Simulator**

   Start the simulators for each data source you will be using. To run the StreamBase Data Simulator:

   In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), navigate to the **demos/dstutorial** directory and type:

   run_sbsimdata

3. **Run Demos - Application or Thin Client Browser**

   Application Demo

   1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
   2. To view the demo, type:

      run_viewer

   3. To edit the demo, type:

      run_builder

   Thin Client Browser Demo

   "Start the Demo Server" if it is not running.

   1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
   2. Start the Display Server by typing:

      run_displayserver

   3. Open a browser and navigate to **http://localhost:8068/dstutorial**.

StreamBase Data Source Command Line Options

In addition to general options, the following command line arguments are enabled with the StreamBase data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: "**-sub:$data:my Data**").

Name	Description
-sbds:history.depth:	The number of rows RTView keeps in the history table for each stream. Default is **20,000**. Use for short term storage. For longer term storage, use the Historian. Example: **-sbds:history.depth:5000**
-sbds:loadhistory:	Enables the reading of the CSV file, created if **Write History** is selected, for the specified stream. In order for historical data to be available, the **DATA** subdirectory, located in the directory from which you launched RTView, must contain a CSV file for the stream. Enter the name of the stream for which you want to read a CSV file and click Add to include it in the **History Streams** list. To remove history streams from the list, select a history stream on the list and click **Remove**. See "Application Options - Streambase" for more information. Example: **-sbds:loadhistory streamname**
-sbds:monitor:(true or false)	Select to enable event driven statistics. These statistics will be polled every ten (10) seconds. Default is enabled. **Note:** Setting this parameter to true will increase traffic on your streams as the monitor information must be sent from StreamBase to RTView. Example: **-sbds:monitor:false**
-sbds:sim:	Enter the name of the stream that you want to simulate data for and click Add to include it in the **Simulated Streams** list. To remove streams from the list, select a stream on the list and click **Remove**. For each stream name specified, the DATA subdirectory, located in the directory from which you launched RTView, must contain a CSV file named stream_name.csv. RTView will look for the CSV file on each update. If found, and the timestamp on the file has changed since the last time it was read, RTView will use the data in the file to simulate data for that stream. The data in the CSV file must match the stream format defined in your .sbapps file. Example: **-sbds:sim:streamname**

-sbds:streams:	Enter the name of a stream to subscribe to. If no streams are specified, RTView subscribes to all streams on each specified connection (URI). If this option is used, RTView will only subscribe to the streams specified. Example: **-sbds:streams:streamname**
-sbds:writehistory	Select to enable the writing of a CSV file containing all the data for all tuples received for the streams specified in History Streams. CSV files are written to the **DATA** subdirectory, located in the directory from which you launched RTView. History Streams must be enabled for Write History to execute. Example: **-sbds:writehistory**

TIBCO EMS Administration Data Source

To properly create applications that monitor and manage the health of a JMS implementation you need to know the status of the JMS servers. Realizing the status of JMS servers involves metrics analysis (i.e.: number of consumers, producers, topics and queues), as well as message information (i.e.: rate and queue depth). Access to JMS server metrics is not part of the JMS standard, so a special data source must be provided with each commercial JMS offering.

The TIBCO® EMS Administration data source connects to TIBCO® Enterprise Message Service™ servers. Place raw metrics into RTView functions to calculate SLAs and then present data in dashboards, reports and automated alerts or archive data in the Enterprise RTView Historian for trend analysis on application performance.

In addition to metrics this data source also provides the capability to perform actions on the server, which includes the ability to create and delete topics, queues, routes and durables. It is also possible to automatically discover TIBCO® Enterprise Message Service™ servers by using defined server routes.

This section includes:

Note: The TIBCO EMS Manager application, which is no longer supported and has been removed from RTView Core, has been replaced by the RTView® TIBCO® EMS Monitor application. Contact SL Corporation for more information regarding RTView® TIBCO® EMS Monitor.

TIBCO EMS Administration System Requirements and Setup

System Requirements

In addition to basic "System Requirements", the TIBCO EMS Administration data source requires TIBCO EMS. See the **README_sysreq.txt** file in your installation's home directory for the current version(s) supported.

Setup

In addition to general environment variables (see "Setup"), you must set **TIBJMS_ROOT**. If you will be using SSL, you must set both **TIBJMS_ROOT** and **RTV_USERPATH**:

Name	Description	Example
TIBJMS_ROOT	TIBCO EMS installation directory. If you installed RTView using the Windows installer, this variable may already be set globally on your system.	**C:\TIBCO\ems**
RTV_USERPATH	TIBCO EMS SSL support jars. **Note:** If this variable already exists, append the TIBCO EMS jars to it.	If you are using SSL, also include the following, which are located in the **EMS_HOME\lib** directory: **tibcrypt.jar** **slf4j-api-1.4.2.jar** **slf4j-simple-1.4.2.jar**

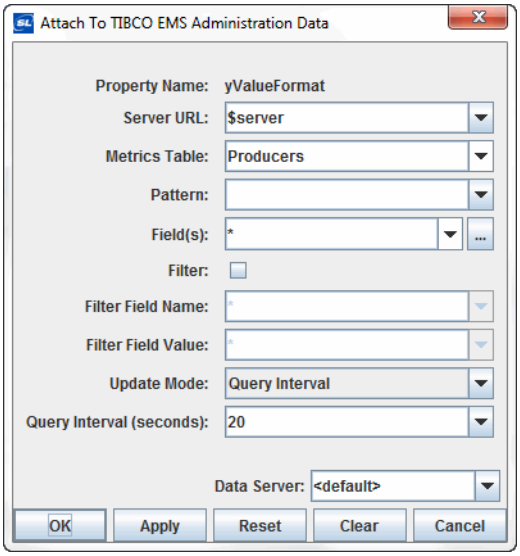
Attach to TIBCO EMS Administration Data

Note: The TIBCO EMS Administration data source may not be licensed in your RTView installation.

The **Attach to TIBCO EMS Administration Data** dialog, which is used to connect an object property to a TIBCO EMS Administration metric (see "TIBCO EMS Metrics"), can be accessed from the **Object Properties** window. Once a property has been attached to a metric, it receives continuous updates.

When an object property is attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing this value from the **Object Properties** window is no longer possible. To remove the data attachment, and resume editing capability in the **Object Properties** window, right-click on the **Property Name** and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Right-click on the Property Name from the **Object Properties** window and select **Attach to Data>JMSADM** to display the **Attach to TIBCO EMS Administration Data** dialog. The **Attach to TIBCO EMS Administration Data** dialog provides drop down menus that allow you to specify information regarding the EMS Administration metric to display.



Field Name	Description
Server URL	Select the name of a Server URL from the drop down menu of all available EMS servers. When ***** is entered, the table returned will contain rows with information from metrics tables for each EMS Server that is being monitored. The **Server URL** drop down menu lists all available EMS servers. Drop down menus for **Metrics Table**, **Field(s)**, and **Filter Name** populate based on the selected EMS Server. If the item you require is not listed, type your selection into the field. For information on adding EMS Servers, see "Application Options - TIBCO EMS".
Metrics Table	Select a **Metric** from the drop down menu. See "TIBCO EMS Metrics" for more information.
Pattern	This field is only available if **Queues**, **Topics**, **Producers**, or **Consumers** is selected from the **Metrics Table** drop down list. If the **Pattern** field is left empty, all queues, topics, producers, or consumers are returned. If a string is entered in this field, only those queues, topics, producers, or consumers that match the pattern are returned. The pattern may contain the wildcards * and **>**. For example, a pattern of **test1.>** or **test1.*.test**. See the TIBCO documentation for more information on how wildcards work in topic names.
Field(s)	Select the **Field(s)** to update the attached object.
Filter	Select to filter the fields of the selected metric.
Filter Field Name	Name of the field to use as a filter.
Filter Field Value	Value that the filter field must equal. Single or multiple values may be listed. Enter ***** to display all rows in the table. Enter **"*"** to use ***** as a literal comparative value. To list multiple values, separate with a semicolon. For example: **value1**;**value2**;**value3**. If your value contains a semicolon, enclose it in single quotes.

Update Mode

This drop down list contains two options:

Metrics Period: (default) queries data attachments at the same interval as defined by the **collector.sl.rtview.jmsadm.metrics_period** property in the **emsmon/conf/rtvapm.emsmon.properties** file. The default value is 15000 milliseconds (15 seconds).

Update Query: Selecting this option enables the **Query Interval** field (see below).

Query Interval (seconds)

Allows you to specify a query interval for the field that is different from the default interval (**metrics_period**).

The **Query Interval** is evaluated during each update pass, so the amount of time elapsed between queries may be longer than the value entered. Typically, the **Query Interval** will be set to an even multiple of the **metrics_period** but, for explanation purposes, let's say that the **metrics_period** is set to 15 seconds and **Query Interval** is set to 20 seconds:

metrics_period (seconds):...15...30...45...60

Query Interval (seconds): ....20....40....60

At 15 seconds, no update will occur since **Query Interval** is set to 20 (it has not been more than 20 seconds since the last update). When **metrics_period** reaches 30 seconds, the update occurs because it has been more than 20 seconds since the last update. When **metrics_period** reaches 45 seconds, the update will not occur because only 15 seconds have passed since the last update. When **metrics_period** reaches 60 seconds, it will have been 30 seconds since the last update (and, hence, greater than the defined 20 second **Query Interval**) and the update will occur.

If **Query Interval** was set as a multiple of the **metrics_period** in the above example, say 60 seconds, then updates would be skipped at 15 seconds, 30 seconds, and 45 seconds, but would be run at 60 seconds. Then, updates would be skipped at 75 seconds, 90 seconds, and 105 seconds, but would be run at 120 seconds since it would have been 60 seconds since the last update.

Data Server

Select to read data through your configured Data Server and not directly from the data source.

Default - Select the default Data Server you configured in **Application Options>**"Data Server Tab".

None - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source.

Named Data Servers - Select a Named Data Server that you configured in **Application Options>**"Data Server Tab".

Multi-Server Attachment - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more Named Data Servers (e.g. ds101;ds102). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**).

The values **__default** and **__none** begin with two underscore characters.

Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received.

A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways:

1. The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows.

The multi-server attachment can be applied as the **Table** argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

Substitutions

Substitutions allow you to build open-ended displays in which data attachments depend on values defined at the time the display is run. Generic names, such as **$server1** and **$server2**, are used instead of values for specific servers, metrics tables, or fields. Later when the display is running, these generic values are defined by the actual names. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

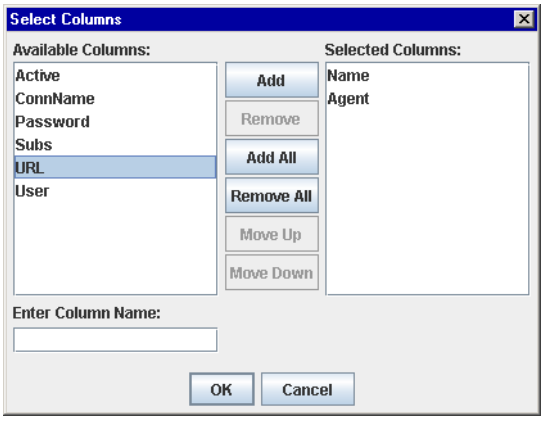
Select Table Columns

From the Attach to **TIBCO EMS Administration Data** dialog you can specify which table columns to display and in what order they will appear. In order to populate the listing of available columns, you must first select a valid **Server URL** and **Metrics Table.**

To bring up the **Select Columns** dialog, click on the ellipses button  in the **Field(s)** field (or right-click in the **Field(s)** field and click on **Select Columns**). The dialog should contain a list of Available Columns that you can add to your table.

To add a column, select an item from the **Available Columns** list and click on the **Add** button. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of columns in a table by arranging the items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

If no data is available for a table row within a selected column, the table cell will display one the following values: **N/A**, **false**, **0**, or **0.0**.



The following describes the **Attach to TIBCO EMS Administration Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied.

TIBCO EMS Metrics

RTView maintains server metrics tables for each EMS Server that is being monitored. The following pre-defined substitutions are recognized:

$server - Name of the selected server

$conn - Name of the admin connection to the selected server

$agent - Name of the TIBCO Hawk Agent associated with the selected server

The metrics for each EMS Server includes a table named ServerTable that contains information (e.g. Name, URL, Agent Name, etc.) for each server being monitored, in addition to a table for each of the following com.tibco.tibjms.admin classes:

Table Name	TibjmsAdmin Class
ServerInfo	com.tibco.tibjms.admin.ServerInfo
Users	com.tibco.tibjms.admin.UserInfo
ListenPorts	com.tibco.tibjms.admin.ServerInfo getListenPorts( ) method
Connections	com.tibco.tibjms.admin.ConnectionInfo
Consumers	com.tibco.tibjms.admin.ConsumerInfo
Producers	com.tibco.tibjms.admin.ProducerInfo
Durables	com.tibco.tibjms.admin.DurableInfo
Queues	com.tibco.tibjms.admin.QueueInfo
Topics	com.tibco.tibjms.admin.TopicInfo
Routes	com.tibco.tibjms.admin.RouteInfo

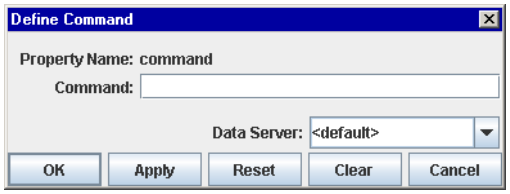
Note: When * is entered in the Server URL field of the "Attach to TIBCO EMS Administration Data" dialog, the table returned will contain rows with information from the metrics tables listed above for each EMS Server that is being monitored.

Define TIBCO EMS Administration Command

Note: The TIBCO EMS Administration data source may not be licensed in your RTView installation.

You can access the Define TIBCO EMS Administration dialog from the **Object Properties** window. This dialog is used to assign a TIBCO EMS Administration command to an object's command property, giving you the ability to execute administrative commands on your EMS Server from within an RTView display. If you execute a TIBCO EMS Administration command from a Thin Client with Direct Data Connection or any Served Data deployment, the command will execute on the server.

Right-click on the appropriate command property in the **Object Properties** window and select **Define Command>JMSADM** to display the **Define TIBCO EMS Administration** dialog. See the "Define/Execute Command" section for information on how to execute a command.



Field Name	Description
Command	Enter a command.
Data Server	Select to read data through your configured Data Server and not directly from the TIBCO EMS Administration data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab". **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a **Named Data Server** that you configured in **Application Options>**"Data Server Tab"**.** **Multi-Server Command** - When multiple data servers are specified, the command will be executed on each data server in the list. To configure multiple data servers, enter a semicolon (;) delimited list containing two or more **Named Data Servers** (e.g. ds101;ds102). Each name specified must correspond with a **Named Data Server** that you configured in **Application Options>Data Server**. It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**). **Note:** The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

Command Format

TIBCO EMS Administration uses the following command format.

Format	Example / Description
Server_URL Command Argument1 Argument2 Argument3 Argument4	**www.sl.com create_topic 'Sales 2005' true true 3200** • The first token in the command is always the server URL. • Depending on the command, one to six arguments may be used (the example above has four). • Single quotes must be used if a token contains a space. For example, 'Sales 2005'.

Command line parameters for TIBCO EMS Administration include:

Command	Description	Argument	Description	Example
delete_connection	Deletes a connection.	**Server URL**	URL of your local EMS server.	**tcp://emsserver1:7222 delete_connection 555**
		Connection ID	ID of connection to delete.	
create_topic	Creates a topic.	**Topic name**	Name of the topic to create.	**tcp://emsserver1:7222 create_topic Q4Results true true 3200**
		Global flag	Set global property of this topic. True / false.	
		Failsafe	Set failsafe property of this topic. True / false.	
		Maximum bytes	A numeric value for the maximum topic size.	
delete_topic	Deletes a topic.	**Topic name**	Name of the topic to delete.	**tcp://emsserver1:7222 delete_topic Q4Results**
purge_topic	Purges a topic.	**Topic name**	Name of the topic to purge.	**tcp://emsserver1:7222 purge_topic Q4Results**
create_queue	Creates a queue.	**Queue name**	Name of the queue to create.	**tcp://emsserver1:7222 create_queue Sales true true 3200**
		Global flag	Set global property of this queue. True / false.	
		Failsafe	Set failsafe property of this queue. True / false.	
		Maximum bytes	A numeric value for the maximum queue size.	
delete_queue	Deletes a queue.	**Queue Name**	Name of the queue to delete.	**tcp://emsserver1:7222 delete_queue Sales**

purge_queue	Purges a queue.	**Queue Name**	Name of the queue to purge.	**tcp://emsserver1:7222 purge_queue Sales**
create_durable	Creates a durable.	**Durable Name**	Name of the durable to create.	**tcp://emsserver1:7222 create_durable Durable1 Sales jsmith selector1 true true**
		Topic Name	Name of topic on which to create durable.	
		Client ID	Client ID associated with the durable.	
		Selector	The selector associated with the durable.	
		No local	Prevents reception of messages sent on this session. True / False	
		Route	Designates this as a durable for another server. True / False	
delete_durable	Deletes a durable.	**Durable Name**	Name of the durable to delete.	**tcp://emsserver1:7222 delete_durable Durable1 jsmith**
		Client ID	Client ID associated with the durable (can be null).	
purge_durable	Purges a durable.	**Durable Name**	Name of the durable to purge.	**tcp://emsserver1:7222 purge_durable Durable1 jsmith**
		Client ID	Client ID associated with the durable (can be null).	
create_route	Creates a route to another server.	**Route Name**	Name of the route to create. This must be the same as the destination server URL.	**tcp://emsserver1:7222 create_route tcp://emsserver2:7222 tcp://emsserver2:7222 zone1 MHOP**
		Destination Server	URL of the destination server.	
		Zone Name	Name of the zone to create this route in.	
		Zone Type	The type of zone to create this route in.	
delete_route	Deletes a route.	**Route Name**	Name of the route to delete.	**tcp://emsserver1:7222 delete_route tcp://emsserver2:7222**

Substitutions

The Substitutions feature allows you to build open-ended displays in which commands depend on values defined at the time the display is run. Substitutions can be used in place of any token in a command. For example, a generic server URL such as $server is used instead of a specific server URL. Later when the display is running, this generic value is defined by the actual name of a specific server. In this way, a single display can be reused to execute commands on a number of different servers. For more information on creating displays using substitution values, see "Substitutions".

Special Values

$value	When an **actionCommand** is executed **$value** is replaced with the value from the control. This value may be used in any token in the **Define TIBCO EMS Administration** dialog. **Note:** This value may only be used for **Action Commands**. See "Define/Execute Command" for more information.

The following describes **Define TIBCO EMS Administration dialog** commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from assigned message subject (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Application Options - TIBCO EMS

Select **Tools>Options** in the Display Builder to access the **Application Options** dialog.

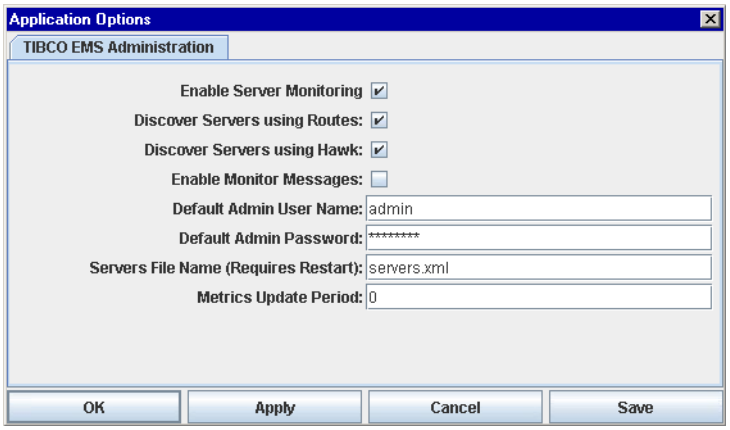
Options specified in TIBCO EMS tabs can be saved in an initialization file (**JMSADMOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server and Historian to set initial values. If no directory has been specified for your initialization files and **JMSADMOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

Note: Options specified using command line arguments will override values set in initialization files. See "TIBCO EMS Administration Data Source Command Line Options" for more information.

There are two Application Options tabs for TIBCO EMS: "TIBCO EMS Administration Tab" and "TIBCO EMS Servers Tab".

TIBCO EMS Administration Tab

This tab allows you to administer connection settings for your TIBCO EMS servers. After configuring TIBCO EMS server options, click **OK**, **Apply**, or **Save**.



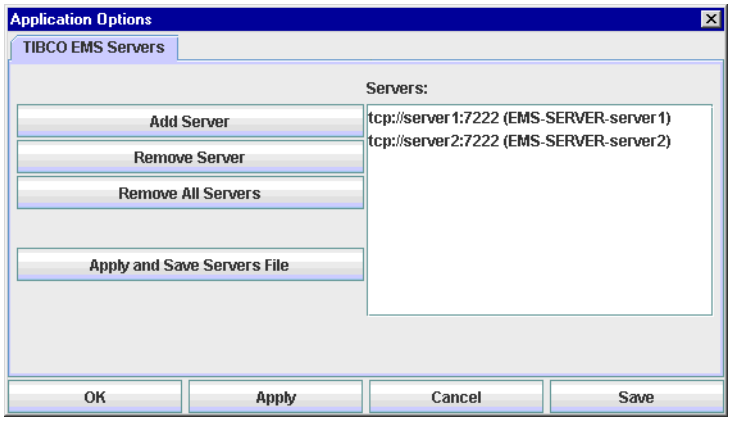
Field Name	Description
Enable Server Monitoring	Enables the maintenance of TIBCO EMS Administration metrics for each server listed in the specified Servers File. This option is enabled by default. See "TIBCO EMS Metrics" for more information.
Discover Servers using Routes	Uses the Routes server metrics table ("TIBCO EMS Metrics") to enable automatic discovery of EMS servers. A new server is added to the table only if the route to that server is active and connected. This option is enabled by default. To connect to only those servers listed in the specified Servers File, you must deselect both this option and Discover Servers using TIBCO Hawk.
Discover Servers using TIBCO Hawk	Uses the TIBCO Hawk JMS_Controller microagent to enable automatic discovery of EMS servers. This option is enabled by default. To connect to only those servers listed in the specified Servers File, you must deselect both this option and Discover Servers using Routes.
Enable Monitor Messages	Enables the creation of an admin connection to each EMS Server that will be used to receive **Monitor Messages**. **Note:** Because the TIBCO EMS server publishes message monitor topics when events (connect, disconnect, produce, consume, etc.) occur, enabling **Monitor Messages** in production systems may have an adverse effect on system performance. If it will be used in production, the Monitor Messages display should be modified to narrow down the number of messages. Refer to Monitoring Server Events in TIBCO's EMS documentation for more information.
Default Admin User Name	Specify a user name to connect to TIBCO EMS servers. Default is admin.

Default Admin Password	Specify a password to connect to TIBCO EMS servers. Default is **no password**. If you need to provide an encrypted password (rather than expose server password names in a clear text file, use the encode_string command line option with the following syntax: **encode_string type mypassword** where **type** is the key for the data source and **mypassword** is your plain text password. **Note:** The type argument is only required when you encrypt a string for a data source. For example, enter the following in an initialized command window (see "Initializing a Command Prompt or Terminal Window"): **encode_string jmsadm mypassword** and you will receive an encrypted password: **encrypted value: 013430135501346013310134901353013450134801334** Copy the encrypted value, paste it into the password field and click **Save** to save this value to the initialization (***.ini**) file. Or, if necessary, manually edit the (***.ini**) file to include the encrypted value. **Note:** If you need to manually edit a configuration (***.ini**) file, contact SL Technical Support at support@sl.com for information about supported syntax.
Servers File Name	Enter the name of the file (including path, if necessary) that RTView will read to determine which servers to monitor. Default file name is **servers.xml**, which is located in the directory where you started RTView. **Note:** The Servers File is created when you click **Apply and Save Servers File** on the **TIBCO EMS Servers** tab. To only monitor those servers specified in the Servers File and disable automatic discovery, deselect both **Discover Servers** check boxes.
Metrics Update Period	Enter the time in milliseconds to control how often EMS Server metrics will be queried. Default is **0**, which queries according to the update period specified on the **Application Options General** tab. **Note:** Because the **Metrics Update Period** is superseded by the **General Update Period**, the amount of time elapsed between queries may be longer than the value entered. For example, if the **General Update Period** is 2000 milliseconds and the **Metrics Update Period** is 5000 milliseconds, EMS Server metrics will be queried every six seconds.

TIBCO EMS Servers Tab

This tab allows you to add and remove EMS servers. After adding or removing EMS servers, or applying and saving an EMS server file, click **OK**, **Apply**, or **Save**.

Note: Additional setup is required to connect to your EMS Server using SSL. See "TIBCO EMS Administration SSL Parameters" for more information.



Field Name	Description
Add Server	Click to open the **Add EMS Server** dialog. If TIBCO Hawk is available but the JMS_controller microagent is not running, the Agent is used to tell RTView the name of the TIBCO Hawk agent that is running on the host machine that contains the server. The information in the Host Details page will be updated using data collected from that TIBCO Hawk microagent. If both the TIBCO Hawk agent and JMS_controller microagent are running on the server's host machine, then TIBCO Hawk will discover the server running on that machine. In this case, the name of the agent does not need to be provided in the servers file as RTView will make the association automatically.  **Server URL** - The complete URL for your EMS server. **Server Name** - The name of your EMS server. **Agent Name** - Name of the agent containing the microagent method to subscribe. This field is optional. If TIBCO Hawk is available but the JMS_controller microagent is not running, this value is used to tell RTView the name of the TIBCO Hawk agent that is running on the host machine that contains the server. The information in the **Host Details** page will be updated using data collected from that TIBCO Hawk microagent. If both the TIBCO Hawk agent and JMS_controller microagent are running on the server's host machine, then TIBCO Hawk will discover the server running on that machine. In this case, the name of the agent does not need to be provided here as RTView will make the association automatically. **User Name** - The user name to use when creating this connection. This field is optional. **Password** - The password to use when creating this connection. This field is optional.
Remove Server	Click to remove an EMS server selected in the **Servers** field.

Remove All Servers	Click to remove all EMS servers from the **Servers** field. **Note:** You must deselect both **Discover Servers** check boxes on the **TIBCO EMS Administration** tab or the **Servers** field will immediately repopulate. The **Discover Servers** check boxes are selected by default.
Apply and Save Servers File	Click to apply all options, from all tabs, and save your EMS servers file.

TIBCO EMS Administration SSL Parameters

Note: This section assumes you have a working knowledge of writing, compiling and deploying Java classes.

To use SSL with your TIBCO EMS Administration connections, you will need to create a Java class named **MyTibJmsSSLHandler** that extends the **com.sl.gmsjtibjmsadmin.GmsRtViewTibJmsSSLHandler** class.

In **MyTibJmsSSLHandler**, define the following method:

```
public Map getSSLParams (String serverUrl)
```

This method will get called to retrieve the list of SSL parameters to pass in when RTView creates each TIBCO EMS Server Administration connection. See the TIBCO EMS documentation for information on creating a Map of SSL parameters suitable to pass into the TIBCO EMS Server Administration connection.

Add **gmsjtibjmsadmin.jar**, located in the **lib** directory (found in your installation directory), to your classpath when you compile **MyTibJmsSSLHandler**. The compiled **MyTibJmsSSLHandler** class must be included in the RTView classpath by adding it to the definition for the **RTV_USERPATH** environment variable.

The following is an example of **MyTibJmsSSLHandler**. To execute this example you must include the tibjms.jar in your classpath when compiling.

```
import java.util.*;
 import com.tibco.tibjms.TibjmsSSL;
 import com.sl.gmsjtibjmsadmin.GmsRtViewTibJmsSSLHandler;

 public class MyTibJmsSSLHandler extends GmsRtViewTibJmsSSLHandler
 {

 /** This method will get called once per connection attempt for each
 * URL. It should return a Map of the SSL parameters needed to
 * connect to the specified serverUrl or null if no SSL parameters
 * are needed.
 * @param serverUrl The String form of the URL RTView is connecting to.
 */

 public Map getSSLParams (String serverUrl)
 {

System.out.println("trying to connect to:"+serverUrl);

Hashtable<String, String> h = new Hashtable<String, String>();
 // Add values common to all servers to the Hashtable.
 // Modify these to contain the correct values for your server
 // and add any additional SSL parameters that are requred by your servers
```

```
 // or remove any SSL parameters that are not required by your servers.
 // If any of these are not common to all servers, move them into
 // the server specific if/else statements below.
 // See com.tibco.tibjms.TibjmsSSL in the TIBCO EMS Javadocs
 // for a full list of SSL parameters.

 h.put(com.tibco.tibjms.TibjmsSSL.TRACE,"true");
 h.put(com.tibco.tibjms.TibjmsSSL.DEBUG_TRACE, "true");
 h.put(com.tibco.tibjms.TibjmsSSL.ENABLE_VERIFY_HOST_NAME, "false");
 h.put(com.tibco.tibjms.TibjmsSSL.ENABLE_VERIFY_HOST, "false");
 h.put(com.tibco.tibjms.TibjmsSSL.IDENTITY,"C:/tibco/config/tibco/cfgmgmt/ems/data/
certs/client_identity.p12");
 h.put(com.tibco.tibjms.TibjmsSSL.PASSWORD, "password");
 h.put(com.tibco.tibjms.TibjmsSSL.TRUSTED_CERTIFICATES,"C:/tibco/config/tibco/cfgmgmt/
ems/data/certs/client_root.cert.pem");

 // add server specific values to the Hashtable
 if (serverUrl.equals("ssl://myserver1:7243")) {
 h.put(com.tibco.tibjms.TibjmsSSL.EXPECTED_HOST_NAME,"myserver1");

 } else if (serverUrl.equals("ssl://myserver2:7243")) {
 h.put(com.tibco.tibjms.TibjmsSSL.EXPECTED_HOST_NAME,"myserver2");
 }

 return h;
 }

 }
```

RTView Deployment - TIBCO EMS Administration

This section contains details about the deployment process that are specific to your data source. Please go to the Deployment section of this documentation for instructions on how to implement your RTView deployment option. Return to this section whenever you are instructed to refer to deployment information that is specific to your data source.

System Requirements and Setup

The TIBCO EMS Administration data source has additional System Requirements and Setup. See "TIBCO EMS Administration System Requirements and Setup" for more information.

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options""Application Options - TIBCO EMS", and "RTV_JAVAOPTS" for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
JMSADMOPTIONS.ini	Contains data source options for TIBCO EMS Administration.

Note: Options specified using command line parameters override values set in these initialization files.

TIBCO EMS Administration Data Source Command Line Options

In addition to General Options, the following command line arguments are enabled with the TIBCO EMS Administration data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: "**-sub:$data:my Data**").

Name	Description
-disableFTDiscovery	By default, when this option is not used, RTView queries each EMS Server (each server that is defined in the servers.xml or discovered using Hawk or Routes) for its fault tolerant URL. The URL returned by the EMS Server is the value used in the ft_active setting in the EMS Server configuration. If RTView does not already have this URL in the ServerTable, it will be added and a connection to it will be made. This can cause duplicate entries in the ServerTable in the case where the server was specified in servers.xml using a different URL than was used in the ft_active field. For example, the URL in ft_active uses a host name but the servers.xml entry for the same server uses an IP address. Or, the servers.xml uses a DNS alias, but the ft_active uses a host name or ip address. When the **-disableFTDiscovery** option is used, RTView will not query each EMS Server for its fault tolerant URL. Instead, it expects that all fault tolerant EMS Server pairs are added to servers.xml as compound URLs (ex. **tcp://myhost:7222,tcp://myhost2:7224**). For each server in the compound URL, the fault tolerant URL will be set to the other server in the compound URL. No error checking is done to confirm that the two URLs listed are actually a fault tolerant pair. It is up to the user to be sure that the servers.xml information is correct.
-jmsadm_metrics_period: (milliseconds)	Specify the rate at which the TIBCO EMS Administration metrics should be queried. Default is **2000**. See "TIBCO EMS Metrics" for more information. Example: **-jmsadm_metrics_period:2000**

-jmsadm_maxmetricsrowcount: (number of rows)	Specify the maximum number of rows in a server metrics table. If the table size exceeds this limit, no rows will be read. Default is **1000**. Example: **-jmsadm_maxmetricsrowcount:1000**
-jms_minreconnecttime: (seconds)	Enter the minimum number of seconds that will elapse before attempting to reconnect to the server. Default is **30**. Example: **-jms_minreconnecttime:30**
-servers:(filename)	Specify the path and file name for your servers file. Default is **servers.xml**. Example: **-servers:c:\myservers.xml**

TIBCO Hawk Data Source

TIBCO Hawk® is a tool for monitoring and managing distributed applications and systems that use TIBCO Rendezvous® or TIBCO Enterprise Message Service™ as transports. It exposes metrics to a variety of infrastructure data (i.e.: SNMP, CPU utilization and network statistics), as well as providing information about other TIBCO® middleware such as TIBCO BusinessWorks™.

RTView for TIBCO Hawk® is primarily used to create applications that monitor application and infrastructure health. These applications can provide:

- Drill-down capability to detailed agent and microagent information
- Side-by-side comparisons of information from multiple agents
- Health and status visualization of AMI instrumented applications
- Summarized views of application status and alerts
- TIBCO Hawk® commands to manage applications (i.e.: restart an application)

RTView ships with a standard set of TIBCO Hawk® monitoring displays that can be used as-is or as a starting point for a more sophisticated application.

This section includes:

TIBCO Hawk - System Requirements and Setup

System Requirements

In addition to basic "System Requirements", the TIBCO Hawk data source requires TIBCO Rendezvous and TIBCO Hawk. See the **README_sysreq.txt** file in your installation's home directory for the current version(s) supported.

By default, all RTView applications connect to TIBCO Hawk using the TIBCO Rendezvous Daemon (rvd) transport.

Setup

In addition to general environment variables (see "Setup"), you must set both **RV_ROOT** and **HAWK_ROOT** and have TIBCO Rendezvous and TIBCO Hawk installed:

Name	Description	Example
RV_ROOT TIBCO	TIBCO Rendezvous installation directory. If you installed RTView using the Windows installer, this variable will already be set globally on your system.	**c:\TIBCO\TIBRV**
HAWK_ROOT	TIBCO Hawk installation directory. This variable must point to a Hawk 4.x or Hawk 5.1+ installation. If you installed RTView using the Windows installer, this variable will already be set globally on your system.	**c:\TIBCO\HAWK**
TIBJMS_ROOT	TIBCO EMS installation directory. This is only required if you are using an EMS transport for your TIBCO Hawk agents. If you installed RTView using the Windows installer, this variable may already be set globally on your system.	**C:\TIBCO\ems**

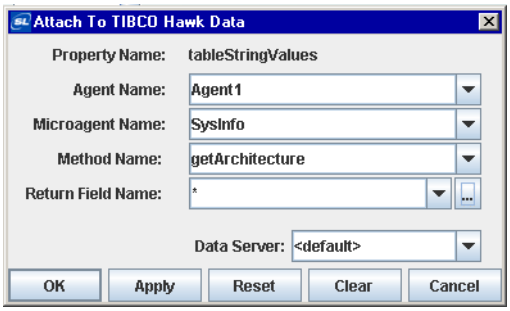
Note: The **PATH** environment variable should include the bin subdirectory for **HAWK_ROOT** and **RV_ROOT** variables (e.g.: **c:\TIBCO\HAWK\bin** and/or **c:\TIBCO\TIBRV\bin**).

Attach to TIBCO Hawk Data

The **Attach to TIBCO Hawk Data** dialog, which is used to subscribe to a TIBCO Hawk Agent's microagent method, can be accessed from the **Object Properties** window. In the **Attach to TIBCO Hawk Data** dialog, you will specify which return field from the method will be used to update an object's dynamic properties. Once a property has been attached to a microagent method subscription, it receives continuous updates.

When an object property is attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing values from the **Object Properties** window is no longer possible. To remove the data attachment and resume editing capabilities in the **Object Properties** window, right-click on the Property Name and select **Detach from TIBCO Hawk Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Right-click on the Property Name from the **Object Properties** window and select **Attach to Data>HAWK** to display the **Attach to TIBCO Hawk Data** dialog. The **Attach to TIBCO Hawk Data** dialog provides several drop down menus that allow you to specify information regarding a method subscription.



Field Name	Description
Agent Name	Name of the agent containing the microagent method to subscribe. To specify an agent on a specific connection, append the connection name enclosed in parentheses. For example: **Agent1 - Agent1 on the default connection.** **Agent1(ConnectionA) - Agent1 on ConnectionA.** **Agent1(ConnectionB) - Agent1 on ConnectionB.** To subscribe to all agents with the same name on all connections, append (*****) to the Agent Name value. For example: Agent1(*) - Agent1 on all connections. To specify all agents for a particular connection, use the TIBCO Hawk Agent Group that is automatically created for that connection. For example: ***:ConnectionB - All agents from ConnectionB.** The "TIBCO Hawk Agent and Microagent Groups Tab" allows you to define Agent Groups and add them to the drop down menu in the **Attach to Data** dialog. See Special Values for more supported values for this field.
Microagent Name	Name of the microagent containing the method to subscribe. The "TIBCO Hawk Agent and Microagent Groups Tab" allows you to define Microagent Groups and add them to the drop down menu in the **Attach to Data** dialog. See "Special Values" for more supported values for this field.
Method Name	Name of the method to subscribe.
Return Field Name	Name of the field in the method return data containing the value to use to update the object. See **Special Values** for more supported values for this field.
Method Argument(s)	Method argument fields (e.g., **Return Field Name**, **TimeInterval**, **RTVPollInterval**) depend on which microagent and method are selected.

Data Server

Select to read data through your configured Data Server and not directly from the TIBCO Hawk data source.

Default - Select the default Data Server you configured in **Application Options>**"Data Server Tab".

None - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source.

Named Data Servers - Select a **Named Data Server** that you configured in **Application Options>**"Data Server Tab".

Multi-Server Attachment - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more Named Data Servers (e.g. ds101;ds102). Each name specified must correspond with a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**).

Note: The values **__default** and **__none** begin with two underscore characters.

Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received.

A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways:

1. The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows.
2. The multi-server attachment can be applied as the **Table** argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

A Method Repository file is used to populate the initial values in the drop down menus for all fields when agents and microagents are not available. This makes it easy to create displays for agents and microagents when they are not online. See "Application Options - TIBCO Hawk" for information on how to create a Method Repository file.

As new agents and microagents come online, drop down menus are automatically updated. If a drop down menu does not contain the item you require, type your selection into the field.

The method argument fields that appear depend on the microagent and method that you select. Argument fields are labeled according to the selected method's requirements. If the method is not recognized, five generic argument fields are provided.

Note: The argument **RTVPollInterval** is provided so that you can set a time interval for synchronous methods.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information entered into the dialog is validated against the contents of the Method Repository, as well as information contained in live agents and microagents.

Note: Agent, Microagent and Method values must be valid in order to create a subscription.

The following describes the significance of the Attach to TIBCO Hawk Data validation colors:

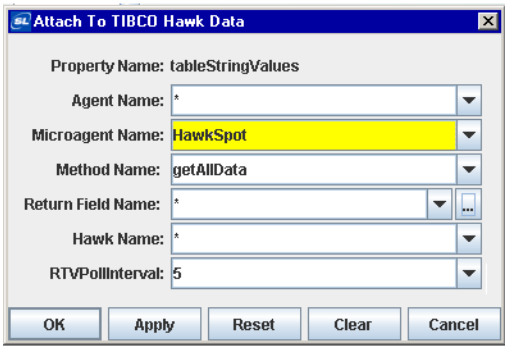
Blue	Unknown	Entry was not found in the Method Repository, nor can it be confirmed against a live agent/microagent. When a microagent is unknown, the Method, Return Field, and Argument(s) fields are also unknown.
Yellow	Offline	Entry was either found in the Method Repository or was previously available, but is not currently online. This applies to agents and microagents.
White	Valid state	Entry is valid and was either found in the Method Repository or could be confirmed against a live agent/microagent.
Red	Invalid state	Entry is not valid.
Gray	Not Required	Field does not require a value. This applies to the Return Field for methods that do not have any returns.

Substitutions

Substitutions allow you to build open-ended displays in which data attachments depend on values defined at the time the display is run. Generic names for agents, such as **$agent1** and **$agent2**, are used instead of values for specific agents. Later when the display is running, these generic values are defined by the actual names of specific agents, such as **computer6** and **computer11**. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

Special Values

Several special values can be entered for agents, microagents, and return fields.

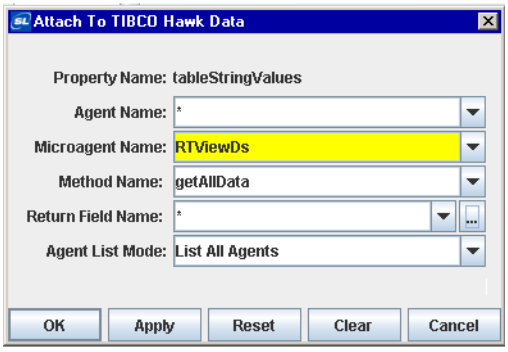


Field Name	Description	
Agent Name	*	When ***** is entered, the return fields from all specified microagents on all agents will be used to update the object property. This value is only useful for objects which display tabular data.
	Agent (Connection)	When the connection is specified after the agent name, the agent on the specified connection will be used. **Agent1** - Agent1 on the default connection. **Agent1(ConnectionA)** - Agent1 on ConnectionA. **Agent1(ConnectionB)** - Agent1 on ConnectionB. To subscribe to all agents with the same name on all connections, append (*) to the Agent Name value. For example: **Agent1(*) - Agent1 on all connections.** To specify all agents for a particular connection, use the TIBCO Hawk Agent Group that is automatically created for that connection. For example: ***:ConnectionB - All agents from ConnectionB.**
	:groupname (:group1)	When ***:groupname** is entered, the return fields from all specified microagents on all agents in the specified Agent Group will be used to update the object property. This value is only useful for objects which display tabular data. See "Application Options - TIBCO Hawk" for more information.
Microagent Name	name (HawkSpot) name* (HawkSpot*)	When only the name of a microagent is listed, the first microagent found on the specified agent is the one that will be used to update the object property. You can also use a wildcard character at the end of the Microagent Name. Data attachments will use the first instance found for all microagents that start with the specified name. If multiple microagents are found for a data attachment, the data for each will be returned.
	name:index (HawkSpot:1) name*:index (HawkSpot*:1)	When an index is specified after the microagent name, the microagent instance for that index will be used to update the object property. You can also use a wildcard character at the end of the Microagent Name. Data attachments will use the instance for the specified index for all microagents that start with the specified name. If multiple microagents are found for a data attachment, the data for each will be returned.
	name:* (HawkSpot:*) name*:* (HawkSpot*:*)	When ***** is entered as an index, all instances of the specified microagent will be used to update the object property. This is only useful for objects which display tabular data. You can also use a wildcard character at the end of the Microagent Name. Data attachments will use all active instances for all microagents that start with the specified name. If multiple microagents are found for a data attachment, the data for each will be returned.

	:groupname (:group1)	When ***:groupname** is entered, Method Name, Return Field Name and Argument fields will validate and populate against the first microagent in the Microagent Group. This value is only useful for objects which display tabular data. See "Application Options - TIBCO Hawk" for more information.
Return Field	*	When * is entered, values from all return fields will be used to update the object property. This is only allowed for objects which display tabular data.

RTViewDs

RTViewDs is a simulated microagent that allows you to call the following methods from RTView to get information that is not available via a microagent method regarding an agent. RTViewDs methods take one argument, **Agent List Mode**, which indicates whether to return information regarding agents that are offline.



The following describes RTViewDs methods:

getName	Returns name of specified agent or the value UNKNOWN if agent is inactive. This method has only one return field: **Name**.
getStatus	Returns status of specified agent (values: **1=agent** is active, **0=agent** is inactive). This method has only one return field: **Status**
getCluster	Returns cluster name of specified agent or the value UNKNOWN if agent is inactive. This method has only one return field: **Cluster**
getConnectionName	Returns the name of the connection for this agent or the value UNKNOWN if the agent is inactive. This method has only one return field: **Connection**. See "Application Options - TIBCO Hawk" for more information.
getConnectionTable	Returns one row for each defined connection with the following columns: **Name** - the connection name **Connected** - **true** if connected, **false** otherwise This method ignores the **Disable Data Caching** application option and caches all of the data regardless of how that flag is configured. See "TIBCO Hawk Methods and Alerts Tab" for more information.
getDsName	Returns name of specified agent including the connection name if it is not on the default connection, or the value UNKNOWN if agent is inactive. This method has only one return field: **DsName**

getAlert	Returns highest alert (values: **1**,**2**,**3**,**4**=correspond to TIBCO Hawk Console API, **0**=agent is inactive). This method has only one return field: **Alert**
getAlertString	Returns highest alert according to TIBCO Hawk Console API or the value UNKNOWN if agent is inactive. This method has only one return field: **AlertString**
getAlertData	Returns information about an alert when the alert is posted. By default, this method does not retrieve alerts posted prior to the data attachment or before the view is displayed. For information on displaying existing alerts, see Application Options. Return fields to choose from: **Agent** - Name of the agent that posted the alert **AlertID** - ID of the alert **AlertState** - Rulebase state of the alert (values: 1,2,3,4= correspond to the TIBCO Hawk Console API). **AlertString** - Rulebase state of the alert (values: ALERT-LOW, ALERT-MEDIUM, ALERT-HIGH correspond to the TIBCO Hawk Console API). **RuleBase** - Rulebase that posted the alert **Cleared** - Returns false if alert is active, returns true if alert is cleared **Time** - Date and time alert was generated **AlertText** - Alert text See "Application Options - TIBCO Hawk" for more information.
getPlatform	Returns the platform the agent is running on or the value UNKNOWN if the agent is inactive. Return fields to choose from: **Agent** - Name of the agent **Architecture** - Platform architecture **Name** - Name of the platform **Version** - Version of the platform If no return field is selected, the method will return the platform as follows: **Architecture:Name:Version**
getIPAddress	Returns the IP address for the agent or UNKNOWN if the agent is inactive. This method has only one return field: **IPAddress**
getAllData	Returns all RTViewDs data. Return fields correspond to the methods listed above. Return fields to choose from: **Alert** **AlertString** **Cluster** **Connection Name** **DsName** **IPAddress** **Name** **Platform** **Status**

getGroupData	Returns group information about a particular Agent. This is useful when caches are used to store data for applications that organize or aggregate Hawk Agent information using groups. Return fields to choose from: **Agent** - Name of the agent **GroupName** - Name of the group containing the agent **ConnectionName** - Name of the connection for this agent **DSName** - Name of the agent including the connection name if it is not on the default connection **Note:** Since an Agent may be in more than one group, a single Agent may show up multiple times (once for each group that contains it).
getAgentStatus	Used only for the purpose of debugging agent and subscription problems. **Note:** Returns the following columns containing information about the subscription: **Display Name** - Display name of the agent. **Status** - Alive indicates RTView received an onAgentAlive event for this agent that is currently active; Expired indicates RTView received an onAgentExpired event and has not yet received another onAgentAlive event; Not Discovered indicates RTView is configured for this agent but has not yet received an onAgentAlive event. **onAgentAlive Time** - Timestamp of the most recent onAgentAlive event for this agent. **onAgentExpired Time** - Timestamp of the most recent onAgentExpired event for this agent. **onAgentExpired Count** - Number of onAgentExpired events RTView has received for this agent since startup. **Data Received Time** - Last time RTView received data from this agent. This may be subscription data or alert data. To see values over time (rather than just the latest value) attach the method to a cache, set the **indexColumnNames** on the cache to Display Name and deselect the **allowDuplicatesInHistoryFlag** on the cache. This method ignores the **Disable Data Caching** application option and caches all of the data regardless of how that flag is configured. See "TIBCO Hawk Methods and Alerts Tab" for more information.
getSubscriptionStatus	Used only for the purpose of debugging agent and subscription problems. **Note:** Because this method can cause performance issues, it must not be used in standard user displays, nor should a cache containing this information be deployed in a production environment. Returns the following columns containing information about the subscription: **Agent** - Display name of the agent. **Key** - Key used internally to track this subscription, containing the agent, microagent, method and listener information. **Status** - Pending indicates RTView queued but not yet processed the subscription request; **Active** indicates RTView processed the subscription; **Cancelled** indicates RTView cancelled the subscription. **Request Time** - Time this subscription was requested. **Create Time** - Time this subscription was created. **Create Length** - Amount of time, in milliseconds, it took the Hawk agent to respond to the subscription request.

Queue Length - Amount of time, in milliseconds, this subscription waited in the queue before RTView requested it from the agent.

Data Received Time - Last time RTView received data for this subscription.

Cancel Time - Time this subscription was cancelled.

Cancel Reason - Reason this subscription was cancelled. No Listeners indicates there are no more listeners for this subscription; Microagent Removed indicates the microagent is no longer available; Agent Connection indicates RTView received an **onAgentExpired** event for the agent.

onData Length - Amount of time, in milliseconds, that the last onData call took to process the subscription data

To see values over time (rather than just the latest value) attach the method to a cache, set the **indexColumnNames** on the cache to Key and deselect the **allowDuplicatesInHistoryFlag** on the cache.

This method ignores the **Disable Data Caching** application option and caches all of the data regardless of how that flag is configured. See "TIBCO Hawk Methods and Alerts Tab" for more information.

Select Columns

From the **Attach to TIBCO Hawk Data** dialog you can specify which method return columns to display in a table and in what order they will appear. To populate the listing of available method return columns, you must first select a valid microagent and a method.

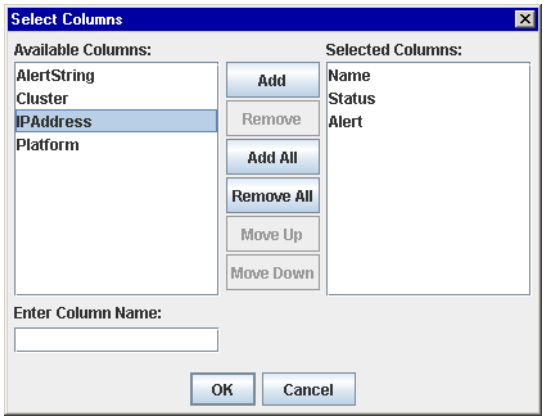
To display the **Select Columns** dialog, click on the ellipsis button in the **Return Field Name** field (or right-click in the **Return Field Name** field and choose **Select Columns**). The dialog should contain a list of **Available Columns** that you can add to your table.

To add a column, select an item from the **Available Columns** list and click on the **Add** button. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of columns in a table by arranging the items in the **Selected Columns** list with the Move Up and Move Down buttons.

Validation colors indicate whether selected method return columns are valid. However if even one selected column is invalid, the **Return Field Name** field in the **Attach to TIBCO Hawk Data** dialog will register as an invalid entry.

Note: Agent, **Microagent**, and **Method** values must be valid in order to create a subscription.

If no data is available for a table row within a selected column, then the table cell will display one the following values: **N/A**, **false**, **0**, or **0.0**.



The following describes **Attach to TIBCO Hawk Data** dialog commands:

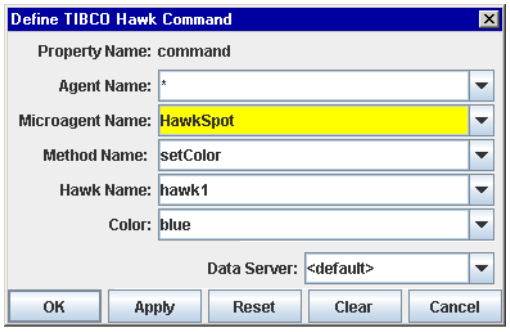
Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied.

Define TIBCO Hawk Command

Note: The TIBCO Hawk data source may not be licensed in your RTView installation.

You can access the Define TIBCO Hawk Command dialog from the **Object Properties** window. This dialog is used to assign a TIBCO Hawk Agent's microagent method to an object's command property, thus giving you the ability to invoke methods from within a display. If you execute a TIBCO Hawk command from a Thin Client with Direct Data Connection or any Served Data deployment, the command will execute on the server.

Right-click on the appropriate command property in the **Object Properties** window and select **Define Command>HAWK** to display the **Define TIBCO Hawk Command** dialog. The **Define TIBCO Hawk Command** dialog provides several drop down menus to specify information regarding assigning a method to an object. The information supplied assigns a microagent method as a command. See the "Define/Execute Command" section for information on how to execute a command.



Field Name	Description
Agent Name	Name of the agent containing the microagent method to invoke. To specify an agent on a specific connection, append the connection name enclosed in parentheses. For example: **Agent1** - Agent1 on the default connection. **Agent1(ConnectionA)** - Agent1 on ConnectionA. **Agent1(ConnectionB)** - Agent1 on ConnectionB. To invoke a method on all agents with the same name on all connections, append (*) to the Agent Name value. For example: **Agent1(*)** - Agent1 on all connections. To specify all agents for a particular connection, use the TIBCO Hawk Agent Group that is automatically created for that connection. For example: ***:ConnectionB** - All agents from ConnectionB. The "TIBCO Hawk Agent and Microagent Groups Tab" allows you to define Agent Groups and add them to the drop down menu in the **Define Command** dialog. See "Special Values" for more supported values for this field.
Microagent Name	Name of the microagent containing the method to invoke. The "TIBCO Hawk Agent and Microagent Groups Tab" allows you to define Microagent Groups and add them to the drop down menu in the **Define Command** dialog. See "Special Values" for more supported values for this field.
Method Name	Name of the method to invoke.

Method Argument(s)	Method argument fields (e.g. Color) depend on which microagent and method are selected. To attach a method argument to data, right-click and choose **Attach to Data** or double-click in the field.
Data Server	Select to read data through your configured Data Server and not directly from the TIBCO Hawk data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab". **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a Named Data Server that you configured in **Application Options>**"Data Server Tab". **Multi-Server Command** - When multiple data servers are specified, the command will be executed on each data server in the list. To configure multiple data servers, enter a semicolon (;) delimited list containing two or more **Named Data Servers** (e.g. **ds101**;**ds102**). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**). The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

A Method Repository file is used to populate the initial values in the drop down menus for all fields when agents and microagents are not available. This makes it easy to create displays for agents and microagents when they are not online. See "Application Options - TIBCO Hawk" for information on how to create a Method Repository file.

As new agents and microagents come online, drop down menus are automatically updated. If a drop down menu does not contain the item you require, type your selection into the field.

The method argument fields that appear depend on the microagent and method that you select. Argument fields are labeled according to the selected method's requirements. If the method is not recognized, five generic argument fields are provided.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information entered into the dialog is validated against the contents of the Method Repository, as well as information contained in live agents and microagents. Only methods that are not asynchronous and have an impact value equal to **IMPACT_ACTION** or **IMPACT_ACTION_INFO** are considered to be valid.

Note: The assigned method will be invoked only if all entries are valid.

The following describes the significance of the **Define TIBCO Hawk Command** dialog validation colors:

Blue	Unknown	Entry was not found in the Method Repository, nor can it be confirmed against a live agent/microagent. When a microagent is unknown, the **Method** and **Argument(s)** fields are all also unknown.
Yellow	Offline	Entry was either found in the Method Repository or was previously available, but is not currently online. This applies to agents and microagents.

White	Valid state	Entry is valid and was either found in the Method Repository or can be confirmed against a live agent/microagent. For the Method Name field, this indicates entry is not asynchronous and has an impact value equal to **IMPACT_ACTION** or **IMPACT_ACTION_INFO**.
Red	Invalid state	Entry is not valid. For the Method Name field, this indicates entry is asynchronous and does not have an impact value equal to **IMPACT_ACTION** or **IMPACT_ACTION_INFO**.
Gray	Not Required	Field does not require a value.

Substitutions

Substitutions allow you to build open-ended displays in which commands depend on values defined at the time the display is run. Generic names for agents, such as **$agent1** and **$agent2**, are used instead of values for specific agents. Later when the display is running, these generic values are defined by the actual names of specific agents, such as **computer6** and **computer11**. In this way, a single display can be reused to execute commands for a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

Special Values

Agent Name	*	This indicates that the method of the specified microagent will be executed on all agents.
	Agent(Connection)	When the connection is specified after the agent name, the agent on the specified connection will be used. **Agent1** - Agent1 on the default connection. **Agent1(ConnectionA)** - Agent1 on ConnectionA. **Agent1(ConnectionB)** - Agent1 on ConnectionB. To invoke a method on all agents with the same name on all connections, append (*****) to the Agent Name value. For example: **Agent1(*)** - Agent1 on all connections. To specify all agents for a particular connection, use the TIBCO Hawk Agent Group that is automatically created for that connection. For example: ***:ConnectionB** - All agents from ConnectionB
	:groupname (:group1)	When ***:groupname** is entered, the return fields from all specified microagents on all agents in the specified Agent Group will be used to update the object property. This value is only useful for objects which display tabular data. See "Application Options - TIBCO Hawk" for more information.

Microagent Name	Name (HawkSpot) name* (HawkSpot*)	When only the name of a microagent is listed, the first microagent found on the specified agent is the one on which the method will execute. You can also use a wildcard character at the end of the Microagent Name. Commands will use the first instance found for all microagents that start with the specified name. If multiple microagents are found for a command, the command will execute on each microagent.
	name:index (HawkSpot:1)	When :index is added after the microagent name, the microagent with the index indicated is the one on which the method will execute. You can also use a wildcard character at the end of the Microagent Name. Commands will use the instance for the the specified index for all microagents that start with the specified name. If multiple microagents are found for a command, the command will execute on each microagent.
	name:* (HawkSpot:*)	When * is used as the index for a microagent, the method will be executed on all instances of the specified microagent. You can also use a wildcard character at the end of the Microagent Name. Commands will use all active instances for all microagents that start with the specified name. If multiple microagents are found for a command, the command will execute on each microagent.
	:groupname (:group1)	When ***:groupname** is entered, **Method Name**, **Return Field Name**, and **Argument** fields will validate and populate against the first microagent in the Microagent Group. This value is only useful for objects which display tabular data. See "Application Options - TIBCO Hawk" for more information.
This value may be used in any field in the Define TIBCO Hawk Command dialog.	$value	When an **actionCommand** is executed $value is replaced with the value from the control. **Note:** This value may only be used for Action Commands. See "Define/Execute Command" for more information.

The following describes **Define TIBCO Hawk Command** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from assigned method (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied

TIBCO Hawk Data Source Substitutions

In addition to standard built-in substitutions (see "Substitutions"), this data source also sets the following drill down substitutions:

Substitution Value	Definition
$agent	Agent from the selected row or object.
$ma	Microagent from the selected row or object.

Application Options - TIBCO Hawk

Select **Tools>Options** in the Display Builder to display the **Application Options** dialog.

Options specified in TIBCO Hawk tabs can be saved in an initialization file (**HAWKOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server and Historian to set initial values. If no directory has been specified for your initialization files and **HAWKOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

Note: Options specified using command line arguments will override values set in initialization files. See "TIBCO Hawk - Command Line Options"

There are two Application Options tabs for TIBCO Hawk: "TIBCO Hawk Communication Tab" and "TIBCO Hawk Methods and Alerts Tab".

TIBCO Hawk Communication Tab

This tab allows you to add and remove TIBCO Hawk connections and set your default connection. RTView supports multiple connections to TIBCO Hawk.

If you are using only one connection, select default in the **Connection** list, edit it, and click **Add**. If you are using more than one connection, type in the configurations including a unique connection name, then click **Add**. The new connection appears in the **Conection** list. To edit a connection, select it in the **Connection** list, edit it, and click **Add**. To remove a connection, select it in the **Connection** list and click **Remove**. **Apply**, **OK**, and **Save** applies all changes. Select **Cancel** to not save all the recently made connections. If RTView is unable to make a connection, an error message will appear on the console when you click **Apply**.

A unique name must be specified for each connection. This name is used to create an Agent Group by the same name containing all agents on that connection. A default connection is automatically created that can be modified but not removed. By default, TIBCO Hawk uses **rvd**. Different configurations apply to **rva** and EMS, as described below. See "TIBCO Hawk Agent and Microagent Groups Tab" for more information.

Note: Connections should have unique parameters or some agents may not be added to the appropriate group.

See "Communicating with TIBCO Hawk" for a detailed explanation of **rva** and **rvd**.

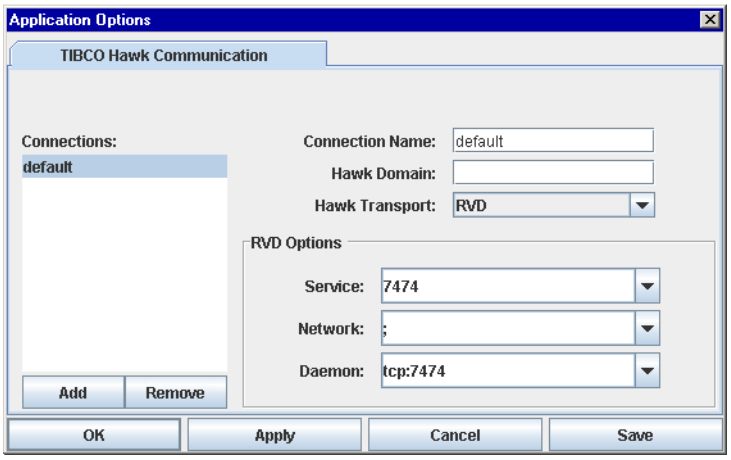
TIBCO Hawk Communication Settings

The following settings apply to **rvd**, **rva**, and **EMS**.

Field	Description
Connection Name	Unique name for the connection.
Hawk Domain	TIBCO Hawk Domain. The default is null.
Hawk Transport	The transport to use when connecting to TIBCO Hawk. The default is **rvd**.

RVD

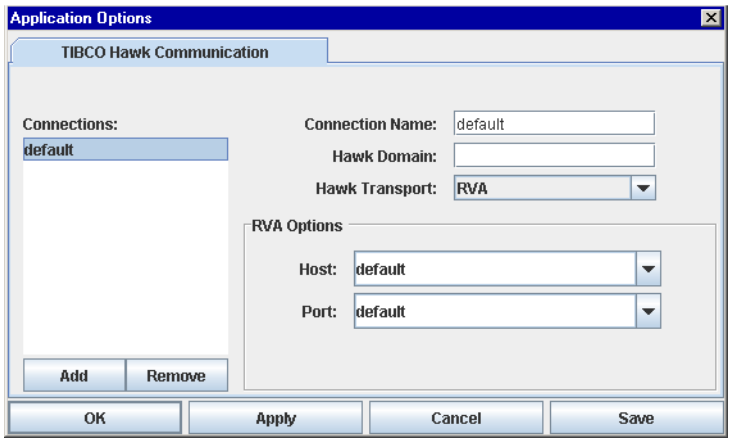
When using **rvd** (the default), the Service, Network, and Daemon settings apply.



Field Name	Description
Service	TIBCO Rendezvous Session Service. The default is 7474.
Network	TIBCO Rendezvous Session Network. The default is ;.
Daemon	TIBCO Rendezvous Session Daemon. The default is tcp:7474.

RVA

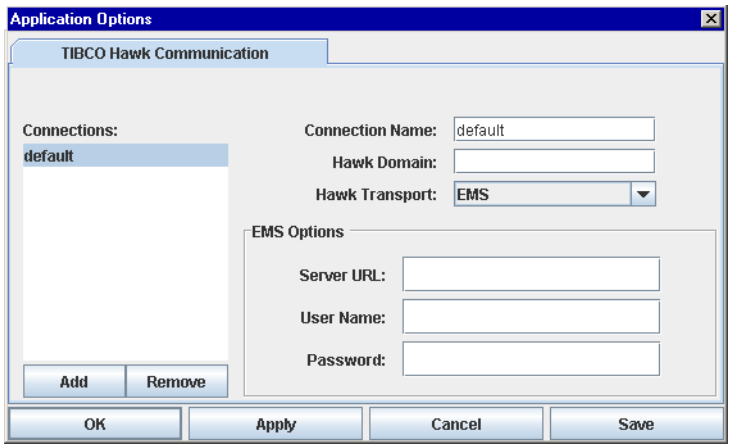
When using **rva**, Host and Port parameters apply. If the Host is set to default, the local host will be used. If the Port is set to default, the TIBCO Rendezvous default port will be used.



Field Name	Description
Host	Host running rva. The default is the local host. This option is only used if running in rva mode.
Port	Port on which rva is running. The default is the rva default. This option is only used if running in rva mode.

EMS

When using EMS, Server URL, User Name, and Password settings apply.



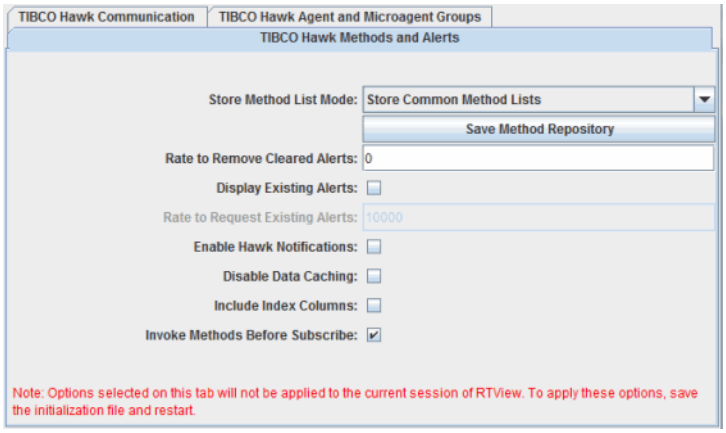
Field Name	Description
Server URL	The complete URL for your EMS server. This must match the settings on the **Transport** tab of the **TIBCO Hawk Configuration Utility**. If nothing is specified, use the default: **tcp://localhost:7222**.
User Name	The user name for your EMS server. This must match the settings on the **Transport** tab of the **TIBCO Hawk Configuration Utility**.
Password	The password. This must match the settings on the **Transport** tab of the **TIBCO Hawk Configuration Utility**. If you need to provide an encrypted password (rather than expose server password names in a clear text file, use the **encode_string** command line option with the following syntax: **encode_string type mypassword** where **type** is the key for the data source and **mypassword** is your plain text password. **Note:** The type argument is only required when you encrypt a string for a data source. For example, enter the following in an initialized command window (see "Initializing a Command Prompt or Terminal Window"): **encode_string hawk mypassword** and you will receive an encrypted password: **encrypted value: 013430135501346013310134901353013450134801334** Copy the encrypted value, paste it into the password field and click **Save** to save this value to the initialization (***.ini**) file. Or, if necessary, manually edit the (*.ini) file to include the encrypted value. **Note:** If you need to manually edit a configuration (***.ini**) file, contact SL Technical Support at support@sl.com for information about supported syntax.

Note: Additional setup is required if you are using SSL with EMS Transport. See "TIBCO Hawk SSL Parameters" for more information.

TIBCO Hawk Methods and Alerts Tab

The **TIBCO Hawk Methods and Alerts** tab allows you to select the **Store Method List Mode**, save a Method Repository file and set the **Rate to Remove Cleared Alerts**, or **Display Existing Alerts**.

Note: Options selected on the **TIBCO Hawk Methods and Alerts** tab are not applied to the current session of RTView. Click the **Save** button to record all TIBCO Hawk application options to the **HAWKOPTIONS.ini** initialization file and restart RTView to apply these changes.

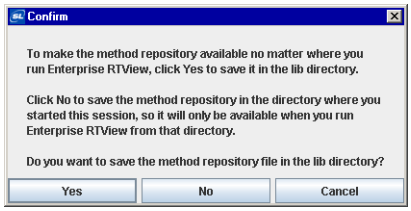


Field Name	Description
Store Method List Mode	The **Store Method List Mode** option allows you to control how method information is stored in RTView. When you change the **Store Method List Mode**, click the **Save** button to record this change in the initialization file. Because settings on this tab cannot be applied to the current session of RTView, you must restart in order to enable the selected method list mode and then save a corresponding method repository file. **Store Common Method Lists** - Stores a single method list for multiple microagents running on the same platform and version of Hawk. This option is memory efficient and the best choice for most users. **Store All Method Lists** - Stores a method list for each microagent running. This options uses more memory, but enables data attachments to multiple microagents (on a single platform) that contain different method definitions for the same method name. **Note:** RTView will output an error message in your command window to alert you that the current method repository file does not correspond to the selected method list mode. If you continue to use RTView with a method repository file that is incorrectly formatted, no method descriptions from that file will be used.
Save Method Repository	Click **Save Method Repository** to create a file that records the current listing of agents and microagents that are online at the time the file is saved. The information stored in this file is then used to populate the initial values of drop down menus in the **Attach to Data** and **Define Command** dialogs. This makes it possible to build displays when agents and microagents are offline. Depending on your Hawk Agent settings, it may take up to a few minutes for a complete listing of agents and microagents to come online. The saved file will be named **hawkmethodrepository.xml**. See "Attach to TIBCO Hawk Data" and "Define TIBCO Hawk Command" for more information. **Note:** If the name of the method repository file is changed, RTView will not be able to locate the file. As a result, drop down menus will remain empty until agents and microagents begin to come online. The format of the method repository file must correspond to the **Store Method List Mode**. The Display Builder must be started with the selected method list mode enabled in order for you to save a method repository file that is formatted correctly. When you save a new **Store Method List Mode**, you must restart RTView (in order for the change to take effect) and then save a method repository file.

Note: RTView will output an error message in your command window to alert you that the current method repository file does not correspond to the selected method list mode. If you continue to use RTView with a method repository file that is incorrectly formatted, no method descriptions from that file will be used.

When you click **Save Method Repository**, a confirmation dialog will appear to verify the directory in which you would like to save the method repository file. If you specified a directory for your initialization files, all repository files will be saved to, and read from, that directory. If you select the **lib** directory, the repository file will be available from any directory where you run RTView. If you do not select the **lib** directory, the repository file will be saved in the directory where you started the current session and will only be available when you run RTView from that particular directory. See "RTV_JAVAOPTS" for more information.

See the **TIBCO Hawk** "Method Repository" section for information on how to edit an existing Method Repository file.



Rate to Remove Cleared Alerts

Enter the time (in milliseconds) to control how often cleared alerts should be removed from RTView. For example, if you set the rate to **300000**, then cleared alerts will be removed every 5 minutes. This setting is useful to optimize performance if you are running displays over a long period of time on a system that generates a large number of alerts. If you do not enter a rate, cleared alerts will never be removed.

Display Existing Alerts

By default, only alerts that occur after you have started RTView are visible in your displays. If you select this check box, RTView will begin to request alerts as soon as a display is opened that contains data attachments to the "RTViewDs" **getAlertData** method. Existing alerts will be requested, one agent at a time, for each agent referenced in these data attachments.

Note: TIBCO does not immediately return alerts, so the first time a display is opened there may be a delay before existing alerts are displayed.

Alerts are requested for one agent every 10 seconds (10000 milliseconds) until existing alerts are displayed for all agents referenced in data attachments to the "RTViewDs" **getAlertData** method. TIBCO documentation warns applications not to request all existing alerts at the same time as this may cause an unacceptably large number of simultaneous network messages.

Rate to Request Existing Alerts

Enter the time (in milliseconds) to control how often RTView will request alerts. The default rate is **10000 milliseconds** (10 seconds). In determining the request rate, you should consider the average number of alerts outstanding on the entire network when a display is opened that contains data attachments with the "RTViewDs" **getAlertData** method.

Enable Hawk Notifications

Select this check box activate the display of TIBCO Hawk notification events. The **AlertString** for these notifications will be **NO_ALERT**. By default, TIBCO Hawk notifications are disabled.

Disable Data Caching

Select this check box to disable the caching of data in the TIBCO Hawk data source. When selected the initial update on data attachments to multiple agents returns a table with a row for each agent, but subsequent updates will include only the rows that have changed. This option is useful when attaching TIBCO Hawk data as input to the Cache data source or the Historian.

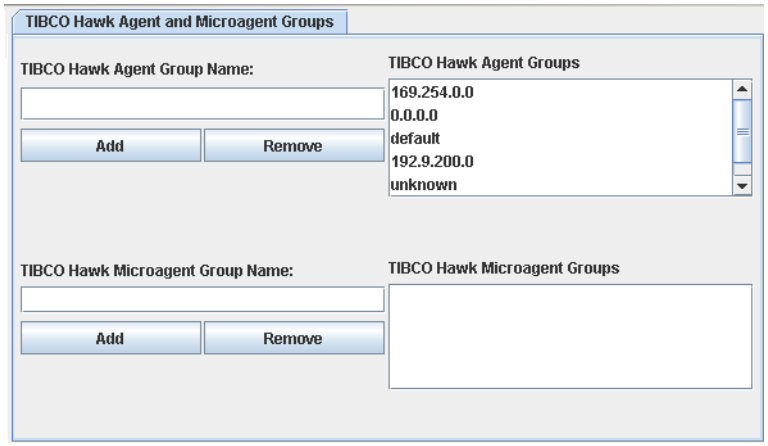
Include Index Columns	If selected, the following three additional columns will be available in the Return Field menu of the **Attach to TIBCO Hawk Data** dialog. Note that if you select the **RTViewDs microagent**, then these additional columns will not be available. **AgentName** **MicroAgentName** **MicroAgentInstance** **Note:** If ***** is entered for the Return Field value in the "Attach to TIBCO Hawk Data" dialog, then values from all available return fields will be used to update the object.
Invoke Methods Before Subscribe	If selected, RTView invokes synchronous microagent methods before subscribing to them for data attachments. This makes the data for these data attachments available immediately after the data attachment is made instead of waiting for the first update from the subscription. By default, this option is selected. You can disable this option from the **TIBCO Hawk Methods and Alerts** tab in the **Application Options** dialog, using a command line argument. The command line argument is: **Command Line: -hawknoinvoke** Data attachments to asynchronous methods are not affected by this option. See the TIBCO Hawk documentation for your microagent method to determine whether or not it is synchronous.

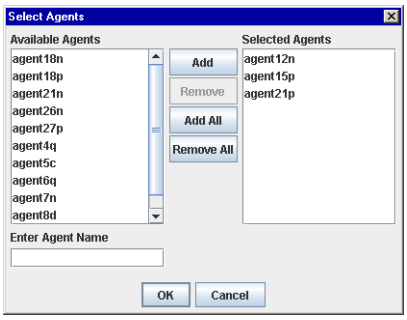
TIBCO Hawk Agent and Microagent Groups Tab

The **TIBCO Hawk Agent and Microagent Groups** tab allows you to define Agent and Microagent Groups and add them to the drop down menu in the **Attach to Data** and **Define Command** dialogs. As soon as this information is saved to **HAWKOPTIONS.ini** it will be used to populate dialog menus each time you run RTView.

TIBCO Hawk Microagent groups are useful when you want to attach to multiple microagents that have different names, but the same methods. This is often the case with TIBCO adapter microagents running on different domains.

Note: Agent Groups are automatically created for each TIBCO Hawk cluster and each Hawk Connection defined on the TIBCO Hawk Communication tab. It is not possible to modify or delete Agent Groups that are automatically created for TIBCO Hawk clusters.

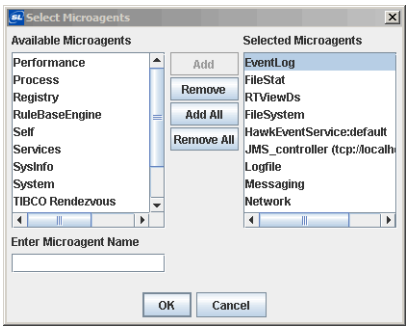


Field Name	Description
TIBCO Hawk Agent Group Name	Enter a name for the Agent Group **Add** -- Click **Add** to insert the Agent Group into the listing. You must click **Save** in order for Agent Groups you've added to be listed the next time you run RTView. **Remove** -- Select an Agent Group from the list and click the **Remove** button to delete. It is not possible to permanently remove Agent Groups that are automatically created for TIBCO Hawk clusters. **Define TIBCO Hawk Agent Groups** To select which agents to add to a group, double-click on the name of a group from the TIBCO Hawk Agent Groups listing. The **Select Agents** dialog should contain a list of Available Agents that you can add. **Note:** It is not possible to modify or permanently remove Agent Groups that are automatically created for TIBCO Hawk clusters. **Note:** n object. Validation colors indicate whether selected agents are valid.  **Add** -- Select from the Available Agents list and click **Add** or click **Add All**. **Enter Agent Name** -- If the agent you require is not listed, type your selection into the Enter Agent Name field. **Note:** For agents that are not on the default connection, specify the connection in parentheses after the agent name. For example, to specify Agent1 on Connection1, use Agent1(Connection1). **Remove** -- Select from the Selected Agents list and click **Remove** or click **Remove All**. It is not possible to add or remove agents while the selected group is currently updating a
TIBCO Hawk Microagent Group Name	Enter a name for the Microagent Group **Add** -- Click **Add** to insert the Microagent Group into the listing. You must click Save in order for Microagent Groups you've added to be listed the next time you run RTView. **Remove** -- Select an Microagent Group from the list and click the **Remove** button to delete.

Define TIBCO Hawk Microagent Groups

To select which Microagents to add to a group, double-click on the name of a group from the TIBCO Hawk Microagent Groups listing. The **Select Microagents** dialog should contain a list of Available Microagents that you can add.

Note: In order for a Microagent Group to be useful, the microagents in the group must have common methods.



Add -- Select from the Available Microagents list and click **Add** or click **Add All**.

Enter Microagent Name -- If the microagent you require is not listed, type your selection into the **Enter Microagent Name** field.

Note: Microagent names can include *****.

Remove -- Select from the **Selected Microagents** list and click **Remove** or click **Remove All**.

Note: It is not possible to add or remove microagents while the selected group is currently updating an object. Validation colors indicate whether selected microagents are valid.

TIBCO Hawk SSL Parameters

Note: This section assumes you have a working knowledge of writing, compiling and deploying Java classes.

To use SSL with EMS Transport for TIBCO Hawk (version 4.6+), you will need to create a Java class named **MyHawkSSLHandler** that extends the **GmsRtViewHawkCustomSSLHandler** class.

In **MyHawkSSLHandler.java**, define the following method:

```
public Hashtable getSSLParams ()
```

This method will get called to retrieve the list of SSL parameters to pass in when RTView creates the TIBCO Hawk Console. See TIBCO Hawk documentation for information on creating a Hashtable of SSL parameters suitable to pass into the TIBCO Hawk Console.

Add **gmsjhawkds.jar**, located in the **lib** directory (found in your installation directory), to your classpath when you compile **MyHawkSSLHandler**. The compiled **MyHawkSSLHandler** class must be included in the RTView classpath by adding it to the definition for the **RTV_USERPATH** environment variable.

The following is an example of **MyHawkSSLHandler**:

```
import java.util.Hashtable;
import com.tibco.tibjms.TibjmsSSL;
import com.sl.gmsjhawkds.GmsRtViewHawkCustomSSLHandler;

public class MyHawkSSLHandler extends GmsRtViewHawkCustomSSLHandler
```

```
{

public Hashtable getSSLParams ()
{
        System.out.println("==> getSSLParams");

        Hashtable sslParameters = new Hashtable();
        sslParameters.put(com.tibco.tibjms.TibjmsSSL.TRACE, new Boolean(true));
        sslParameters.put(com.tibco.tibjms.TibjmsSSL.DEBUG_TRACE,
                                                    new Boolean(true));
        sslParameters.put(com.tibco.tibjms.TibjmsSSL.VENDOR, "j2se");
        return sslParameters;

}
}
```

RTView Deployment - TIBCO Hawk

This section contains details about the deployment process that are specific to your data source. Please go to the Deployment section of this documentation for instructions on how to implement your RTView deployment option. Return to this section whenever you are instructed to refer to deployment information that is specific to your data source.

System Requirements and Setup

The TIBCO Hawk data source has additional System Requirements and Setup. See "TIBCO Hawk - System Requirements and Setup" for more information.

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options", "Application Options - TIBCO Hawk", and "RTV_JAVAOPTS" for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
HAWKOPTIONS.ini	Contains data source options for TIBCO Hawk.

Note: Options specified using command line parameters override values set in initialization files.

TIBCO Hawk Demos

Except where noted, all demos can be run in three ways: as an application, or via a rich client or a thin client in a browser.

Before You Begin

Start the Demo Server

Thin Client Demo only.

There is a thin client demo already installed on the "RTView Demo Server".

In an initialized command window (see "Initializing a Command Prompt or Terminal Window"):

Type **run_startup_demoserver**

Data Source Demo

The Data Source Demo is designed to illustrate each data source.

1. Start the Simulators

Start the simulators for each data source you will be using. To run the "Sample TIBCO Hawk Microagent":

In an initialized command window, go to the **demos/dstutorial** directory and:

type **run_hawkspot**

2. Run Demos - Application or Thin Client Browser

Application Demo

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
2. To view the demo, type:

   run_viewer
3. To edit the demo, type:

   run_builder

Thin Client Browser Demo

"Start the Demo Server" if it is not running.

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
2. Start the Display Server by typing:

   run_displayserver
3. Open a browser and navigate to **http://localhost:8068/dstutorial**.

TIBCO Hawk Monitor Demo

Displays in this demo are designed for monitoring TIBCO Hawk Agents.

1. Run Demos - Application or Thin Client Browser

Application Demo

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/hawkmonitor** directory.
2. To view the demo, type:

   run_viewer
3. To edit the demo, type:

   run_builder

Thin Client Browser Demo

"Start the Demo Server" if it is not running.

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/hawkmonitor** directory.
2. Start the Display Server by typing:

   run_displayserver
3. Open a browser and navigate to **http://localhost:8068/hawkmonitor/panels.jsp**.

To include history displays do the following:

1. In an initialized command window, go to the **demos/hawkmonitor** directory and:

   type **run_historian -sub:$agent:<your agent>** where **<your agent>** is the agent for which you want to gather information.

GI Demo

This demo illustrates how to integrate the Display Server with TIBCO GI.

1. "Start the Simulators"
2. **Run Demo**

   Thin Client Browser Demo

   "Start the Demo Server" if it is not running.

   1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/hawkmonitor** directory.
   2. Start the Display Server by typing:

      run_displayserver
   3. In a browser, navigate to:

      http://localhost:8068/gi/SLHawkMonitor.html

The source code for this demo is in your RTView installation under **demos\gi**. Refer to the **README.txt** file in **demos\gi\JSXAPPS\SLHawkMonitor** for information on how this demo was built.

Quick Start Tutorial - TIBCO Hawk

This Quick Start Tutorial provides you with the fundamentals on how to use RTView with a TIBCO Hawk data source. Once completed, you can swiftly apply this knowledge to building your own real-time dashboard displays for visual access to your TIBCO Hawk data.

Learn to:

- Animate graphic objects with TIBCO Hawk data
- Create a drill down display with TIBCO Hawk data

Note: The TIBCO Hawk data source may not be licensed in your RTView installation.

Get Started

This tutorial requires the following:

- Register for a license key. If you have not, you must do so before continuing. See "Registration" for information.
- "Quick Start Tutorial" - This tutorial requires that you have a working knowledge of RTView. We recommend that you complete the Quick Start Tutorial before continuing.

Start the Display Builder

If you are already logged onto the Display Builder, skip this section and go to "Create a Display".

1. In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"):

   type **run_builder**

2. Login to the Display Builder. By default, the Display Builder does not require a login. "Login" can be enabled at setup to support "Role-based Security". The default user name and password are:

   User Name: admin

   Password: admin

Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role. See "Role-based Security" for more information.

You are now ready to create a display using the TIBCO Hawk data source.

Note: You must follow this initialization process for each new terminal window you open. See the "Setup" section for more details about setting up your environment.

Create a Display

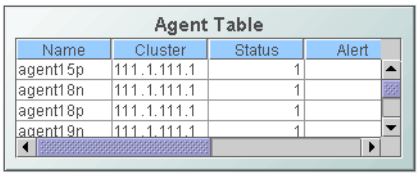
At this point you have:

- Registered for a license key. See "Registration" for more information.
- Logged on to the Display Builder
- Completed the "Quick Start Tutorial"

In this tutorial you create a table that displays live data for your TIBCO Hawk agents, as seen below.

As you saw in the Quick Start Tutorial, the data structure of tables and graphs (tabular data) enables RTView to automatically create several data source specific, built-in Substitutions for you. You will see these built-in Substitutions used in the target display when you create the drill down. For more information on Substitutions, see "Substitutions".

In this exercise, you create a drill down using the previously created display, **hawk_dd_qs.rtv**, as the target display. First you will set the table to display data for each agent. Then you will create a drill down that will open a bar graph that shows more detailed data for each agent.



Setup TIBCO Hawk Connection

By default, RTView communicates with TIBCO Hawk using the TIBCO Rendezvous Daemon (**rvd**) and listens for the TIBCO Rendezvous session 7474;tcp:7474. If you will use this default setup, skip this section. If you will not use the default communication settings, do the following:

1. In the Display Builder, select **Tools>Options** to open the **Application Options** dialog. See "Application Options - TIBCO Hawk"
2. Select the **TIBCO Hawk Communication** tab.
3. In the **TIBCO Hawk Communication** dialog:

   Select **default** from the Connection List.

   Hawk Domain - Enter the Hawk domain name.

   Hawk Transport - Select the Hawk transport type and fill in the remaining fields that apply.

   Click **Add** when you are finished.
4. Click **OK** to close the **TIBCO Hawk Communication** dialog.

Display Data in a Table

In this exercise you add a table and then display data in the table by attaching it to the data source.

1. Click on the Add Table button  and click again in the Working Area to place the table.
2. In the **Object Properties** dialog:

   label (category: Label) - Change the name of the label to Agent Table.

   valueTable (category: Data) - Right-click in the Property Name field and select **Attach to Data>HAWK**.

3. In the "Attach to TIBCO Hawk Data" dialog:

   Agent Name - Enter ***

   Microagent Name - Select **RTViewDs**.

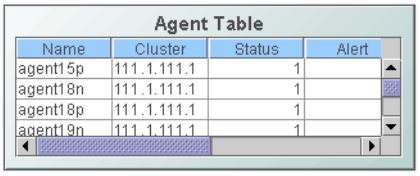
   Method Name - Select **getAllData**.

   Return Field Name - Enter ***

   Agent List Mode - Select **List All Agents**.

4. Click OK to apply these values and close the Attach to TIBCO Hawk Data dialog.

The table populates with values from TIBCO Hawk Agents.



5. Select **File>Save** in the Display Builder.

You are now ready to set up the drill down.

Create a Drill Down Target in the Table

In this exercise, you create the drill down using the previously created display, **hawk_agent_detail.rtv**, as the target.

1. In the **Object Properties** dialog:

   drillDownTarget (category: Interaction) - Double-click in the **Property Name** field to bring up the **Drill Down Properties** dialog.

2. In the "Drill Down Properties" dialog:

   Apply Drill Down To - Select **Named Window** from the drop down menu. This option lets you re-use the window when you drill down multiple times.

   Window Name - Enter hawk. This name should be unique unless the display is to open in an existing window.

   Drill Down Display Name - Select **dstutorial\hawk_agent_detail.rtv** from the drop down menu.

Note: The drill down display dstutorial\hawk_agent_detail.rtv was created to display information from Windows TIBCO Hawk Agents. Not all methods are available on all agents.

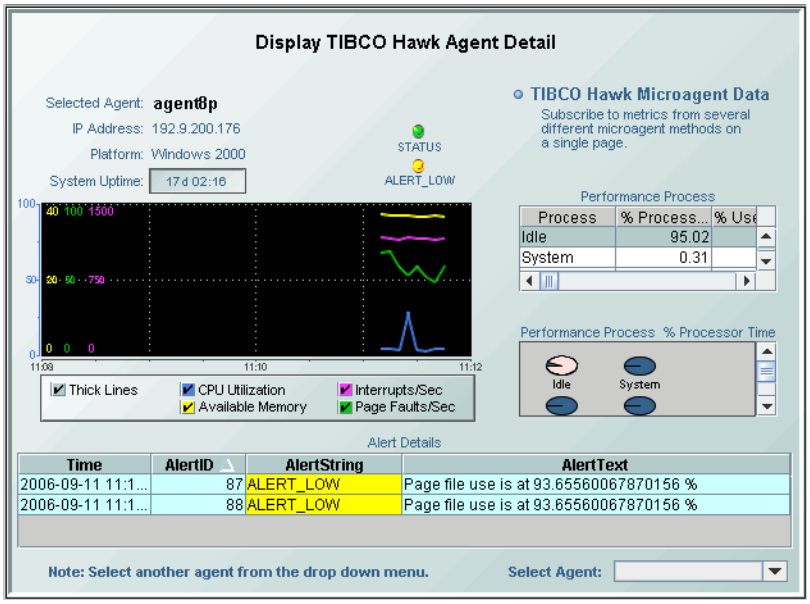
---

3. Click OK to attach the drill down target and close the dialog.

View the Drill Down Display

In this exercise, you drill down to the target display.

1. Double-click on any cell in the table to drill down to detailed data. The target display opens.



2. Double-click on another row in the table and the same display is used to show different data based on the row you select.

3. Close the drill down window.

4. In the Display Builder select **File>Save**.

Go to the main "Quick Start Tutorial"

Sample TIBCO Hawk Microagent

A sample Hawk Spot microagent is provided to enable customers to see how to monitor or execute commands on a microagent without using their environment. Hawk Spot supports the following methods:

- accelerate
- addHawk
- decelerate
- deleteHawk
- getAllData
- getAngle
- getColor
- getColorAssignmentCount
- getMaxThreads
- getName
- getReleaseVersion
- getTraceLevel
- getTraceParameters
- resume
- setAngle
- setColor
- setMaxThreads
- setTraceLevel
- setTraceParameters
- suspend

Running Hawk Spot

From an initialized terminal window (see "Initializing a Command Prompt or Terminal Window") type:

run_hawkspot

Method Repository

A Method Repository file is used to populate the initial values of drop down menus in the "Attach to TIBCO Hawk Data" and "Define TIBCO Hawk Command" dialogs. This makes it easy to create displays for agents and microagents when they are not online. See "Application Options - TIBCO Hawk" for more information on creating a Method Repository file.

It is possible to edit an existing Method Repository file, however, the file name **hawkmethodrepository.xml** cannot be modified. If **hawkmethodrepository.xml** is not found in the specified directory or your current working directory, RTView will look in the **lib** directory. If the Method Repository file is not found, dialog drop down menus will remain empty until agents and microagents begin to come online.

To edit an existing Method Repository file, supported tags and attributes are as follows:

Tag	Attribute			Description
hawkmethodrepository	xmlns			Top level tag that includes the namespace attribute xmlns, which must be defined as www.sl.com (xmlns="www.sl.com")
	storeMethodListMode			Format of Method Repository changes according to the value of storeMethodListMode (values: 0=store common method lists 1=store all method lists)
methodlist	name			Method list name
	method	name		Method name
		async		Method asynchronous value
		impact		Method impact value
		returnfield		Return field name
		argument	name	Argument name
			type	Argument type
			choice	Choice for argument
agent	name			Agent name
	connection name			Connection name
	microagent	name		Microagent name
		key		Corresponds to name of method list

An example Method Repository file:

```
<?xml version="1.0"?>
<hawkmethodrepository xmlns="www.sl.com" storeMethodListMode="0">
        <methodlist name="Self_x86:Windows NT:4.0_4.0.0">
                <method name="describe"
                        async="false"
                        impact="0">
                        <argument name="Instance"
                                type="java.lang.String"/>
                        <argument name="Name"
                                type="java.lang.String"/>
                </method>
                <method name="getComponentInfo"
                        async="false"
                        impact="0"
                        returnfield="Date"
                        returnfield="Name"
                        returnfield="Version">
                        <argument name="Component"
                                type="java.lang.String"/>
                </method>
                <method name="getMicroAgentInfo"
                        async="false"
```

```
                        impact="0"
                        returnfield="Count"
                        returnfield="Help"
                        returnfield="Name">
                        <argument name="Name"
                                type="java.lang.String"/>
                </method>
                <method name="getReleaseVersion"
                        async="false"
                        impact="0"
                        returnfield="Date"
                        returnfield="Major"
                        returnfield="Minor"
                        returnfield="Name"
                        returnfield="Update"
                        returnfield="Version">
                </method>
                <method name="getSecurityInfo"
                        async="false"
                        impact="0"
                        returnfield="Description"
                        returnfield="Policy Class">
                </method>
                <method name="getUptime"
                        async="false"
                        impact="0"
                        returnfield="Total days"
                        returnfield="Total hours"
                        returnfield="Total millisec"
                        returnfield="Uptime">
                </method>
                <method name="turnDiagnosticsOff"
                        async="false"
                        impact="1">
                </method>
                <method name="turnDiagnosticsOn"
                        async="false"
                        impact="1">
                </method>
        </methodlist>
        <agent name="MyAgent1" connection="default">
                <microagent name="Self" key="Self_x86:Windows NT:4.0_4.0.0"/>
        </agent>
        <agent name="MyAgent2" connection="default">
                <microagent name="Self" key="Self_x86:Windows NT:4.0_4.0.0"/>
        </agent>
</hawkmethodrepository>
```

TIBCO Hawk - Command Line Options

In addition to General Options, the following command line arguments are enabled with the TIBCO Hawk data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: "**-sub:$data:my Data**").

Name	Description
-hawkalertcleartime:(milliseconds)	Set the rate at which cleared alerts are removed from RTView. If you set the rate to **300000**, cleared alerts will be removed every 5 minutes. Default is **0** (cleared alerts are never removed). Example: **-hawkalertcleartime:300000**
-enablenotifications	Enable the display of TIBCO Hawk notification events. The **AlertString** for these notifications will be **NO_ALERT**. By default, TIBCO Hawk notifications are disabled. Example: **-enablenotifications**
-displayexistingalerts:(milliseconds)	Set the rate at which alerts are requested by RTView. If you set the rate to **30000**, cleared alerts will be requested every 30 seconds. Example: **-displayexistingalerts:30000**
-hawkdaemon:	TIBCO Rendezvous Session Daemon. Default is **tcp:7474**. Example: **-hawkdaemon:7475**
-hawkdomain:	TIBCO Hawk Domain. Default is **null**. Example: **-hawkdomain:mydomain**
-hawkemsserverurl:	The complete URL for your EMS server. This must match the settings on the **Transport** tab of the TIBCO Hawk Configuration Utility. Example: **-hawkemsserverurl:tcp://localhost:7222**
-hawkemsusername:	The user name for your EMS server. This must match the settings on the **Transport** tab of the TIBCO Hawk Configuration Utility. Example: **-hawkemsusername:myuser**
-hawkemspassword:	The password for your EMS server. This must match the settings on the **Transport** tab of the TIBCO Hawk Configuration Utility. Example: **-hawkemspassword:mypassword**

-hawkmaxreconnect:	This command line option attempts to reconnect when the initial connection to a transport fails. By default, this option is set to 60, which means that reconnection attempts will happen once a minute for 60 attempts (minutes). To disable reconnection attempts, set this option to 0. The example below will attempt to reconnect once a minute for 3 attempts/minutes.
	Example:
	-hawkmaxreconnect:3
	Every time a reconnect is attempted, the following displays in the console:
	`016-03-07 13:00:50.282 re-attempt connection to transport <test_ems>: 1 of 60`
	`016-03-07 13:00:52.911 re-attempt connection to transport <test_ems>: succeeded/failed`

-hawknetwork:	TIBCO Rendezvous Session Network. Default is **;**. Example: **"-hawknetwork:;"**
-hawknoinvoke	Disables the **Invoke Methods Before Subscribe** option. By default, this option is enabled. See "TIBCO Hawk Methods and Alerts Tab" for more information. Example: **-hawknoinvoke**
-hawkservice:	TIBCO Rendezvous Session Service. Default is **7474**. Example: **-hawkservice:7475**
-hawksecurity:(security policy class name)	The fully qualified name of the security policy class. If you are not using one of the security classes included in the TIBCO Hawk jar, security.jar, the jar containing this class must be in your classpath. Example: **-hawksecurity:COM.TIBCO.hawk.security.trusted.Trusted**
-hawktransportmode:	Set transport to use when connecting to TIBCO Hawk. Default is 0. Values: **0 = rvd** **1 = rva** **2 = EMS** Example: -hawktransportmode:0
-hawknocache	Initial update on data attachments to multiple agents returns a table with a row for each agent, but subsequent updates include only the rows that have changed. This option is useful when using Hawk data attachments as input to the Cache data source or the Historian. **Note:** If you specify this option on the command line for the Display Builder or Configuration Utility it will be included in **HAWKOPTIONS.ini** when application options are saved.

-storeMethodListMode:(mode value)	Control how method information is stored. Default is **0**. Values: **1=store all method lists,** **0=store common method lists** Example: **-storeMethodListMode:**0
-rvahost:	Host running rva. Default is the local host. This option is only used if running in rva mode. Example: **-rvahost:computer6**
-rvaport:	Port running rva. Default is the rva default. This option is only used if running in rva mode. Example: **-rvaport:7600**

Communicating with TIBCO Hawk

Note: The TIBCO Hawk data source may not be licensed in your RTView installation.

By default, the Display Builder and Display Viewer Application communicate with TIBCO Hawk using the TIBCO Rendezvous Daemon (**rvd**) process. The **rvd** process is automatically started by TIBCO Hawk whenever it is needed and runs as a background service.

In order to facilitate testing, you may run the Display Builder and Display Viewer Application using rva. The **rva** process must already be running before the Display Builder or Display Viewer Application are started in **rva** mode.

Running the TIBCO Rendezvous Agent Process

The **rva** process must be running for the Display Builder or Display Viewer Application to communicate with TIBCO Hawk if they are running in **rva** mode. RTView assumes that **rva** is running on your local machine using the default port. If **rva** is running on a different host or port, you will need to specify both the host and the port either in the **Application Options** dialog or on the command line. See "Application Options - TIBCO Hawk" and "TIBCO Hawk - Command Line Options" for more information.

Review the appropriate section below to determine what host and port parameters you will need to specify. If you are not sure which section applies to you, contact your TIBCO Hawk system administrator.

If you have never run **rva**, you must configure **rva** with your daemon connection.

If rva is already running on your local machine

You do not need to specify a host. You do not need to specify a port if **rva** is running on the default port. Otherwise, you will need to specify a port either in the **Application Options** dialog or on the command line. Please refer to TIBCO Rendezvous documentation for more information on running **rva**. See "Application Options - TIBCO Hawk" and "TIBCO Hawk - Command Line Options" for more information.

If you have not configured the daemon connection, you will need to configure the TIBCO Rendezvous Agent Process in order to communicate with TIBCO Hawk. See "Configuring the TIBCO Rendezvous Agent Process" for more information.

If rva is running on another machine on your network

You need to specify the name of the host that is running **rva**. You do not need to specify a port if rva is running on the default port. Otherwise, you will need to specify both the host and the port either in the Application Options dialog or on the command line. Please refer to TIBCO Rendezvous documentation for more information on running **rva**. See "Application Options - TIBCO Hawk" and "TIBCO Hawk - Command Line Options" for more information.

If you have not configured the daemon connection, you will need to configure the TIBCO Rendezvous Agent Process in order to communicate with TIBCO Hawk. See "Configuring the TIBCO Rendezvous Agent Process" for more information.

If rva is not already running

You need to start rva on your local machine:

1. Open a Command Prompt window, go to **<rendezvous installation>\bin** and type

   rva -store rvafilename

where **rvafilename** is the name of the file where your **rva** settings are stored. If you have never run **rva**, this file will not exist. In this case, **rva** will create a file by the name you specify and save the **rva** settings to that file.

2. This will start **rva** and print some information regarding **rva** to the console. Note the following information:

   RVA: Http interface:

   RVA Listen:

   RVA Host:

RVA Listen will be the port and **RVA Host** will be the host that you will use to run RTView. The **RVA Http interface** (e.g.: http://hostname:7680/) is necessary to configure **rva** to work with messages sent by the data simulator.

3. If the **RVA Listen** is 7600 and **RVA Host** is the name of the machine you are running on, you will not need to specify either parameter. Otherwise, you will need to specify the host and/or the port either in the **Application Options** dialog or on the command line. Please refer to TIBCO Rendezvous documentation for more information on running **rva**. See "Application Options - TIBCO Hawk" and "TIBCO Hawk - Command Line Options" for more information.

4. Since you have never run **rva**, you must configure the TIBCO Rendezvous Agent Process for you daemon connection. See "Configuring the TIBCO Rendezvous Agent Process" for more information

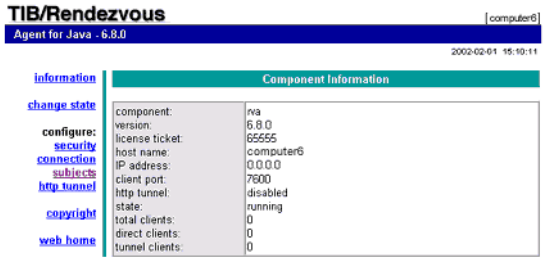
Configuring the TIBCO Rendezvous Agent Process

To configure the TIBCO Rendezvous Agent Process (**rva**) for your daemon connection:

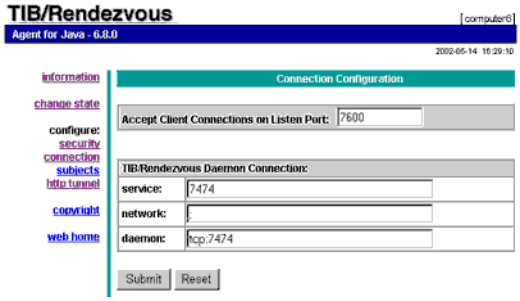
1. Open a browser to the **rva** http interface address. If you do not know the **rva** http interface, the default is:

http://host:7680/

where host is the name of your machine. If this does not work, contact your TIBCO Hawk system administrator. Your setup may be running with a different http interface or you may not have permission to change the **rva** settings. The browser interface should look similar to the following:



2. Click on the **connection** link to go to the **Connection Configuration** screen.



Type your service, network and daemon settings. If you do not know your settings, use the default settings or contact your TIBCO Hawk system administrator. The default settings are

service: 7474

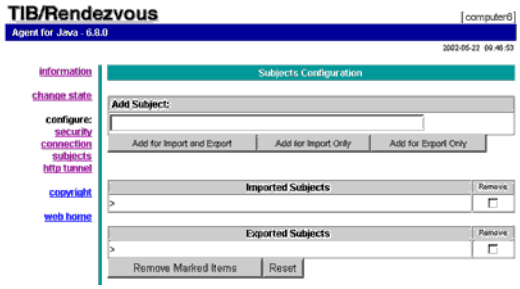
network: ;

daemon: tcp:7474

3. Click the **Submit** button.

4. Click on the **subjects** link to go to the **Subjects Configuration** screen.

Type the following character into the Add Subject field: >



5. Click the **Add for Import** and **Export** button.

6. Click on the **change state** link.

7. Click the button marked **Change state** to idle. Click the button marked **Change state** to running.

Once **rva** has been setup, the daemon configuration information will be stored in the specified settings file. Please refer to your TIBCO Rendezvous documentation for more information regarding configuring **rva**.

TIBCO Rendezvous Data Source

TIBCO Rendezvous® is a high-performance proprietary publish/subscribe, request/reply, messaging middleware transport. RTView for TIBCO Rendezvous® subscribes to real-time messages and extracts the content for use in dashboards, reports and alerts. To improve application monitoring, RTView can provide metrics about a TIBCO Rendezvous® implementation. A set of standard TIBCO Rendezvous® dashboards is provided that can be used as-is or modified for advanced monitoring and control.

This section includes:

TIBCO Rendezvous System Requirements and Setup

System Requirements

In addition to basic "System Requirements", the TIBCO Rendezvous data source requires TIBCO Rendezvous. See the README_sysreq.txt file in your installation's home directory for the current version(s) supported

You may need to modify your Java security setting to include the following permission:

```
permission java.util.PropertyPermission "sun.arch.data.model", "read, write";
```

By default, all RTView applications connect to TIBCO Rendezvous using the TIBCO Rendezvous Daemon (**rvd**) transport.

Setup

In addition to general environment variables (see "Setup"), you must set the **RV_ROOT** variable and have TIBCO Rendezvous installed:

Name	Description	Example
RV_ROOT	TIBCO Rendezvous installation directory. If you installed RTView using the Windows installer, this variable will already be set globally on your system.	**C:\TIBCO\TIBRV**

Note: The PATH environment variable should include the **bin** subdirectory for **RV_ROOT** variable (e.g., **c:\TIBCO\TIBRV\bin**).

Attach to TIBCO Rendezvous Data

Note: The TIBCO Rendezvous data source may not be licensed in your RTView installation.

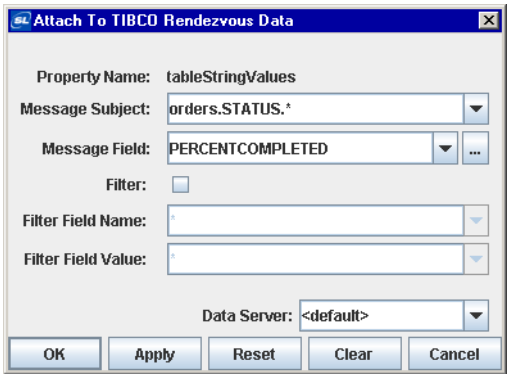
The **Attach to TIBCO Rendezvous Data** dialog, which is used to register an RTView object as a listener for a TIBCO Rendezvous message, can be accessed from the **Object Properties** window. In the **Attach to TIBCO Rendezvous Data** dialog, enter the name of a message subject and specify which message field should be used to update an object property. Once an object property has been attached to a message, it receives continuous updates. It is also possible to set up a filter based on a particular field in the message.

For example, a message with the subject **orders.STATUS.London** has two possible message fields: **PERCENTCOMPLETED** and **PLANT**. The first field, **PERCENTCOMPLETED**, tracks the status of an order. The second field, **PLANT**, indicates which London plant (L1 or L2) the message is describing.

Setting the Message Field to **PERCENTCOMPLETED** will set the attached object to indicate the real-time status of an order. Without a filter, the attached object would receive simultaneous updates from both London plants. To receive updates for a specific plant, you would set up a filter for the **PLANT** field and enter a Filter Field Value of either L1 or L2 to indicate which London plant will update the attached object.

In the **RVALIAS.ini** file, you may create an alias for (top level or nested) messages, message fields and XML data embedded in messages. Once **RVALIAS.ini** is saved and RTView is restarted, you will be able to access these aliases from the **Message Subject** drop down menu. See "Create TIBCO Rendezvous Message Alias" for more information.

Right-click on the Property Name from the **Object Properties** window and select **Attach to Data>RV** to display the **Attach to TIBCO Rendezvous Data** dialog. The **Attach to TIBCO Rendezvous Data** dialog provides several drop down menus that allow you to specify information regarding a message. If the drop down menu does not contain the item you require, type your selection into the text field.



Field Name	Description
Message Subject	Message subject or alias name. Enter a specific subject name or use ***** as a wild card character. For example ***.*.*** or **orders.STATUS.***.
Message Field	Message field chosen to update the attached object.
Filter	Check box to indicate whether or not to filter the message.
Filter Field Name	Name of the message field to use as a filter.
Filter Field Value	Value that the filter field must equal.
Data Server	Select to read data through your configured Data Server and not directly from the TIBCO Rendezvous data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab". **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". **Multi-Server Attachment** - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more **Named Data Servers** (e.g. ds101;ds102). Each name specified must correspond with a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**). **Note:** The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received.

A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways:

1. The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows.
2. The multi-server attachment can be applied as the **Table** argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

A Message Repository file can be used to populate the initial values of drop down menus for all fields. See "Application Options - TIBCO Rendezvous" for information on how to create a Message Repository file. Otherwise, drop down menus populate based on message subjects added from the **Application Options** dialog or those typed directly into the **Attach to TIBCO Rendezvous Data** dialog. Message subjects will not be added to drop down menus until at least one TIBCO Rendezvous message with that subject has been received by RTView.

When an object property has been attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing values from the **Object Properties** window is no longer possible. To remove the data attachment and resume editing capabilities in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information is validated against the Message Repository file, message subjects added from **Application Options** dialog, or those typed directly into the **Attach to TIBCO Rendezvous Data** dialog. Message subjects will not be validated until at least one TIBCO Rendezvous message with that subject has been received by RTView.

Note: Some subjects using wild card characters are not validated at this time.

The following describes the significance of the validation colors:

Blue	Unknown	Entry is not recognized. When a Subject is unknown, the Message Field, Filter Field Name, and Filter Field Value are also unknown.
White	Valid state	Entry is valid.
Red	Invalid state	Entry is not valid.

Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. A generic Message Subject such as **$subject** is used instead of a specific message. Later when the display is running, this generic value is defined by the actual name of a specific message, such as **orders.STATUS.London**. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

Special Values

The following special values can be entered for message and filter fields:

Message Field	*	When ***** is entered as a message field, values from all message fields will be used to update the object property. This is only allowed for objects that display tabular data.
Filter Field Value	*	When ***** is entered as a filter field value, data for all values in the specified filter field will be used to update the object property. When "*****" is entered, only the literal comparative value will be used. These are only allowed for objects that display tabular data.

TIBCO Rendezvous RTViewDs Fields

Subject Fields

RTViewDs subject fields contain TIBCO Rendezvous monitoring information. A sample display file using RTViewDs fields (**rv_rvtrace_subject_tables.rtv**) can be found in your installation directory under **demos\rvmonitor**.

Several monitoring metrics tables are available in the **Subject** field. If these do not appear in the **Subject** field, do the following. In the Display Builder, select **Tools>Options** to open the **Application Options** dialog. Select the **TIBCO Rendezvous Monitoring** tab, and enable **Show Monitor Tables in Data Dialog** by selecting the check box. See "Application Options - TIBCO Rendezvous" for more information.

RTViewDs.HostStatusTotal	The latest data for all hosts obtained from the TIBCO Rendezvous information message: _**RV.INFO.SYSTEM.HOST.STATUS**. **Note:** If selected, the **Disable Cache Data** option will apply to this table. See "TIBCO Rendezvous Cache Tab" for more information.
RTViewDs.HostStatusCurrent	The same as **RTViewDs.HostStatusTotal**, except each numeric value is replaced with a delta from the previous message. **Note:** If selected, the **Disable Cache Data** option will apply to this table. See "TIBCO Rendezvous Cache Tab" for more information.
RTViewDs.HostStatusHistory	A running history of all the **RTViewDs.HostStatusCurrent** rows however it only sets a property on an object when referenced for the first time. Used for trend graph history.
RTViewDs.HostStatusHistory2	The same as **RTViewDs.HostStatusHistory**, except it sets a property on an object every time a new row is added. Used in history table.

RTViewDs.ConfigServices	The output from **tibrvcfg getServices** method. To attach to this table, specify a host using the filter in your data attachment. The **Filter Field Name** must be an HTTP URL and the **Filter Field Value** must be the HTTP URL from which the host will get configuration information. **Note:** You must set the **TIBRV_HOME** environment variable to your Rendezvous installation directory (i.e. same as **RV_ROOT** variable) in order to attach to this table. See "TIBCO Rendezvous System Requirements and Setup" for more information.
RTViewDs.ConfigClientTransports	The output from **tibrvcfg getClientTransports** method.To attach to this table, specify a host using the filter in your data attachment. The **Filter Field Name** must be an HTTP URL and the **Filter Field Value** must be the HTTP URL from which the host will get configuration information. **Note:** You must set the **TIBRV_HOME** environment variable to your Rendezvous installation directory (i.e. same as **RV_ROOT** variable) in order to attach to this table. See "TIBCO Rendezvous System Requirements and Setup" for more information.
RTViewDs.MultiCast*	The six tables showing all output from rvtrace: PacketSource, PacketDest, PacketTotal SubjectSource, SubjectDest, SubjectTotal. See "TIBCO Rendezvous Monitoring Tab" for more information.

Message Fields

RTViewDs message fields contain message tracking information. A sample display file using RTViewDs fields (**rv_msgmetrics.rtv**) can be found in your installation directory under **demos\dstutorial**.

Several RTViewDs message fields are available:

RTViewDS_count	Number of messages received for data attachment.
RTViewDS_size	Size of last message received for data attachment.
RTViewDS_subjectcount	Number of message subjects received for data attachment.
RTViewDS_time	Time of last message received for data attachment.

Select Columns

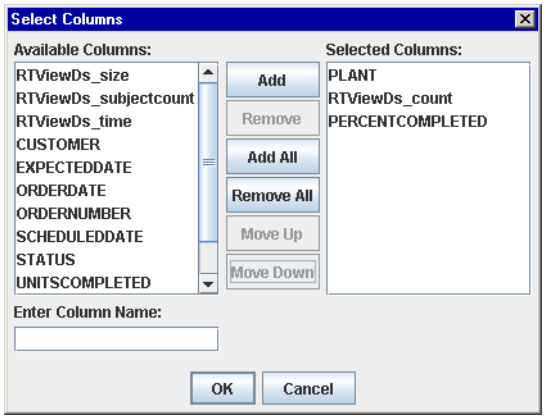
From the **Attach to Data** dialog you can specify which message fields to display as columns in a table and in what order they will appear. In order to populate the listing of available message fields, you must first select a valid subject.

To display the **Select Columns** dialog, click on the ellipses button  in the **Message Field** field (or right-click in the **Message Field** field and click **Select Columns**). The dialog should contain a list of Available Columns that you can add to your table.

To add a field, select an item from the **Available Columns** list and click the **Add** button. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of fields in a table by arranging the items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

Validation colors indicate whether selected message fields are valid. However, if even one field selected is invalid the **Message Field** field in the **Attach to TIBCO Rendezvous Data** dialog will register as an invalid entry.

If no data is available for a table row within a selected column, the table cell will display one of the following values: **N/A**, **false**, **0**, or **0.0**.



The following describes **Attach to TIBCO Rendezvous Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied.

Create TIBCO Rendezvous Message Alias

Note: The TIBCO Rendezvous data source may not be licensed in your RTView installation.

In the **RVALIAS.ini** file, you can create aliases for (top level or nested) messages, message fields and XML data embedded in TIBCO Rendezvous messages. Once **RVALIAS.ini** is saved and RTView is restarted, you will be able to access these aliases from the **Message Subject** drop down menu in the "Attach to TIBCO Rendezvous Data" dialog.

It is possible to specify a directory for your initialization files. If no directory has been specified for initialization files and **RVALIAS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

To define an alias in **RVALIAS.ini** use the following syntax:

```
aliasName subject partialUpdatesOK fieldNameList
```

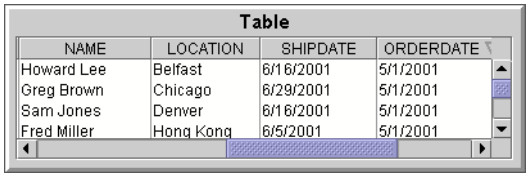
aliasName	Create a name to use when selecting this alias from the **Attach to TIBCO Rendezvous Data** dialog. **Note:** Alias names cannot contain spaces.
subject	TIBCO Rendezvous message subject. It is possible to use * as a wild card character (e.g.: **orders.STATUS.*** or ***.*.***). **Note:** If a message subject contains a space or a colon, then the subject must be enclosed in single quotes.
partialUpdatesOK	This keyword is optional. If present, new row data will be merged into existing row data when a new message comes in with the same subject. For example, if a message comes in which contains fields **FieldA**, **FieldB**, and **FieldC**, then another message with the same subject comes in which contains fields **FieldC**, **FieldD**, and **FieldE**, the new row will contain fields **FieldA** and **FieldB** from the previous message and **FieldC**, **FieldD**, and **FieldE** from the current message.
fieldNameList	Message field names listed in hierarchical order separated by a :: (e.g., **fieldName1::fieldName2::fieldName3** ... etc.) ending in the name of the field that contains the data to display. If no **fieldNameList** is specified, top level message fields will be displayed. If a message field in this list contains XML data, then add the **$xml:** prefix to the field name that contains the XML data. (e.g., **fieldName1::$xml:fieldName2::fieldName3** ... etc.) The name of the message field containing XML data may be followed by a list of XML tags in hierarchical order that ends in an XML tag with the specific data you would like to display. If the last **fieldName** listed contains more than one level of XML tags, then all subsequent tags will be converted to column names using the _ symbol. **Note:** If a field name contains a space or a colon, then the entire **fieldNameList** must be enclosed in single quotes.

Nested Message Field Aliases

In the alias defined below named **allOrders**, a message with the subject **orders.STATUS.*** contains the **customer_info** message field, which is nested within the sales message field.

```
allOrders orders.STATUS.* sales::customer_info
```

A table attached to **allOrders** will contain a column for each field in the **customer_info** message field.



It is also possible to create an alias for a specific field in the customer_info message field.

```
allNames orders.STATUS.* sales::customer_info::NAME
```

A table attached to **allNames** would contain one column for the **NAME** field.

If the message that the alias resolves to contains multiple fields with the same name, a row will be displayed for each field.

Note: Hierarchical message field names are converted to column names using the _ symbol (e.g.: Customer_Name, Customer_CID, etc.).

Embedded XML Data Aliases

To display XML data embedded within a TIBCO Rendezvous message, add the **$xml:** prefix to the field name that contains the XML data. In the alias defined below named **OrderInfo**, the **OrderData** message field (containing XML data) is nested within the **Production** message field.

```
OrderInfo orders.STATUS.* Production::$xml:OrderDatab
```

There are several ways that the following XML values and attributes (contained in the message field named **OrderData**) can be defined in an alias.

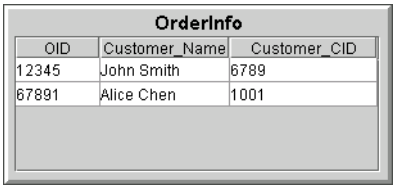
```
<Orders>
        <Order Date="March 1, 2004" Time="12:00:00">
                <OID>12345</OID>
                <Customer>
                        <Name>John Smith</Name>
                        <CID>6789</CID>
                </Customer>
        </Order>
        <Order Date="March 1, 2004" Time="12:00:00">
                <OID>67891</OID>
                <Customer>
                        <Name>Alice Chen</Name>
                        <CID>1001</CID>
                </Customer>
        </Order>
</Orders>
```

Displaying XML Values

To display all XML values contained within **Order** tags of the **OrderData** message field, you would use the following alias. Since the **Orders** tag contains two **Order** tags, 2 rows will be displayed in the table.

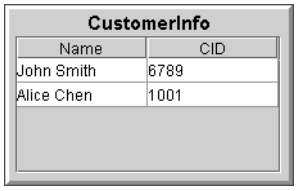
Note: Hierarchical XML tags are converted to column names using the _ symbol (e.g.: **Customer_Name**, **Customer_CID**, etc.).

```
OrderInfo orders.STATUS.* $xml:OrderData::Orders::Order
```



You can display specific information contained within **Customer** tags of the **OrderData** message field using the following alias:

```
CustomerInfo orders.STATUS.* $xml:OrderData::Orders::Order::Customer
```



CustomerInfo

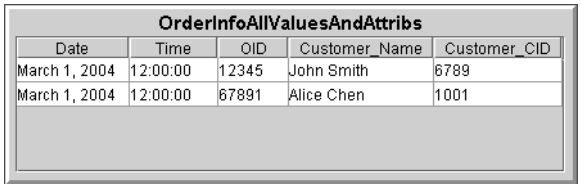
Name	CID
John Smith	6789
Alice Chen	1001

Displaying XML Values and Attributes

To display all XML values and attributes contained within **Order** tags of the **OrderData** message field, you would use the following alias.

Note: Hierarchical XML tags are converted to column names using the _ symbol (e.g.: **Customer_Name**, **Customer_CID**, etc.).

```
OrderInfoAllValuesAndAttribs orders.STATUS.*
$xml:OrderData::Orders::Order:$attrib=**
```



OrderInfoAllValuesAndAttribs

Date	Time	OID	Customer_Name	Customer_CID
March 1, 2004	12:00:00	12345	John Smith	6789
March 1, 2004	12:00:00	67891	Alice Chen	1001

Displaying XML Attributes Only

You can display all of the XML attributes contained within **Order** tags of the **OrderData** message field using the following alias:

```
OrderInfoAllAttribs orders.STATUS.* $xml:OrderData::Orders::Order:$attrib=*
```

OrderInfoAllAttribs

Date	Time
March 1, 2004	12:00:00
March 1, 2004	12:00:00

It is also possible to display only a particular attribute (Date) contained within the Order tags.

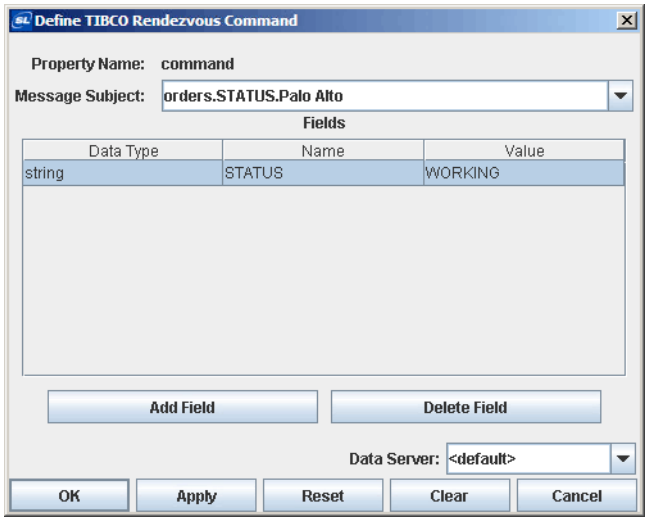
```
OrderInfoDate orders.STATUS.* $xml:OrderData::Orders::Order:$attrib=Date
```

Define TIBCO Rendezvous Command

Note: The TIBCO Rendezvous data source may not be licensed in your RTView installation.

The **Define TIBCO Rendezvous Command** dialog can be accessed from the **Object Properties** window. This dialog is used to assign a TIBCO Rendezvous message to an object's command property, giving you the ability to send messages from within an RTView display. If you execute a TIBCO Rendezvous command from a Thin Client with Direct Data Connection or any Served Data deployment, the command will execute on the server.

To open the **Define TIBCO Rendezvous Command** dialog, right-click on the appropriate command property in the **Object Properties** window and select **Define Command>RV**. The information supplied assigns a message to the command property. See the "Define/Execute Command" section for information on how to execute a command.



Field Name	Description
Message Subject	Enter a **Message Subject** name. To attach the **Message Subject** to data, right-click and choose **Attach to Data** or double-click in the field.

Fields	**Data Type** - Select a data type for this field. Data types are converted to **TibrvMsg** data types as follows: **string** - TibrvMsg.STRING **integer** - TibrvMsg.I32 **long** - TibrvMsg.I64 **float** - TibrvMsg.F32 **double** - TibrvMsg.F64 **boolean** - TibrvMsg.BOOL **Name** - Specify a name for this field. **Value** - Specify a value for this field. To attach the Value to data, right-click and choose **Attach to Data** or double-click in the field.
Add Field	Add a field.
Delete Field	Delete the selected field.
Data Server	Select to read data through your configured Data Server and not directly from the TIBCO Rendezvous data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab". **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". **Multi-Server Command** - When multiple data servers are specified, the command will be executed on each data server in the list. To configure multiple data servers, enter a semicolon (;) delimited list containing two or more **Named Data Servers** (e.g. ds101;ds102). Each name specified must correspond with a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**). **Note:** The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

A Message Repository file can be used to populate the initial values of drop down menus for all fields. See "Application Options - TIBCO Rendezvous" for information on how to create a Message Repository file. Otherwise, drop down menus populate based on message subjects added from the **Application Options** dialog or those typed directly into the **Attach to Data** dialog. Message subjects will not be added to drop down menus until at least one TIBCO Rendezvous message with that subject has been received by RTView.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information is validated against the Message Repository file, message subjects added from **Application Options** dialog, or those typed directly into the **Attach to Data** dialog. Message subjects will not be validated until at least one TIBCO Rendezvous message with that subject has been received by RTView.

Note: Some subjects using wild card characters are not validated at this time.

The following describes the significance of validation colors:

Blue	Unknown	Entry is not recognized. When a Subject is unknown, the Message Field, Filter Field Name, and Filter Field Value are also unknown.
White	Valid state	Entry is valid and was found.
Red	Invalid state	Entry is not valid.

Once all of the required fields are complete, click **OK** to close the dialog and set a value for the object's command property.

Substitutions

The Substitutions feature allows you to build open-ended displays in which commands depend on values defined at the time the display is run. A generic Message Subject such as **$subject** is used instead of a specific message. Later when the display is running, this generic value is defined by the actual name of a specific message, such as **orders.STATUS.London**. In this way, a single display can be reused to send a number of different messages. For more information on creating displays using substitution values, see "Substitutions".

Special Values

$value	When an **actionCommand** is executed, **$value** is replaced with the value from the control. This value may be used in any field in the **Define TIBCO Rendezvous Command** dialog. **Note:** This value may only be used for Action Commands. See "Define/ Execute Command" for more information.

The following describes **Define TIBCO Rendezvous Command** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from assigned message subject (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied.

TIBCO Rendezvous Data Source Substitutions

In addition to standard built-in "Substitutions", this data source also sets the following drill down substitutions:

Substitution Value	Definition
$subject	Message subject from the selected row or object.

$filterfield	Filter field name from the selected row or object.
$filtervalue	Filter field value from the selected row or object.

Application Options - TIBCO Rendezvous

Select **Tools>Options** in the Display Builder to access the **Application Options** dialog.

Options specified in TIBCO Rendezvous tabs can be saved in an initialization file (**RVOPTIONS.ini**). If no directory has been specified for your initialization files and **RVOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

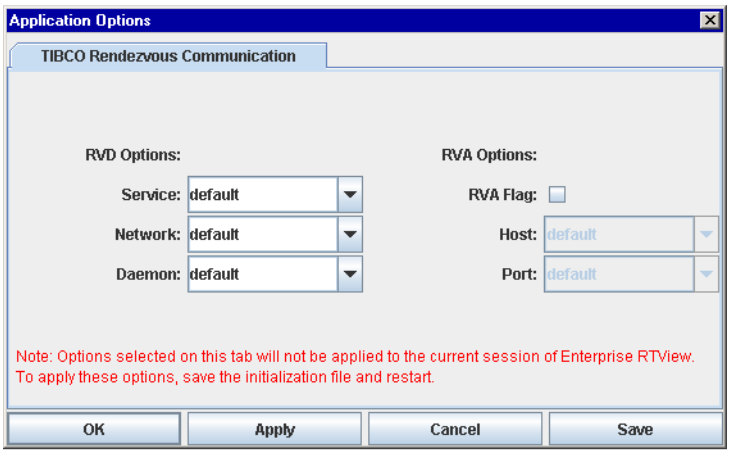
Note: Options specified using command line arguments will override values set in initialization files.

There are four Application Options tabs for TIBCO Hawk: "TIBCO Rendezvous Communication Tab", "TIBCO Rendezvous Monitoring Tab", "TIBCO Rendezvous Messages Tab", and "TIBCO Rendezvous Cache Tab".

TIBCO Rendezvous Communication Tab

These values reflect TIBCO Rendezvous communication settings. When using **rvd** (the default), the Service, Network, and Daemon settings apply. Default values for **rvd** settings are defined in the table below. When using **rva**, Host and Port parameters apply. If the Host is set to default, the local host will be used. If the Port is set to default, the TIBCO Rendezvous default port will be used. See "Communicating with TIBCO Rendezvous" for a detailed explanation of **rva** and **rvd**.

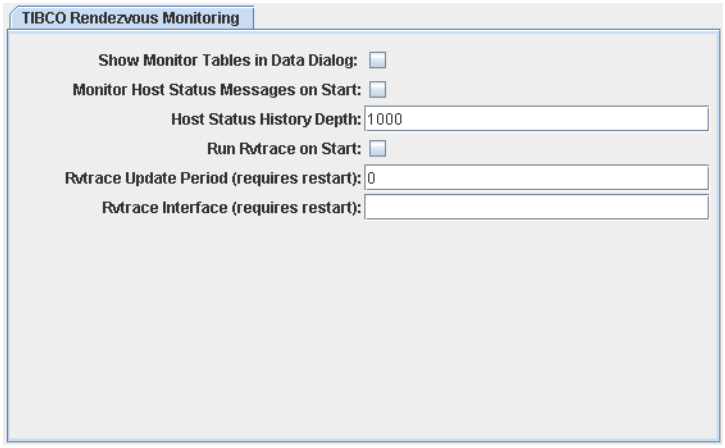
Note: Options selected on the TIBCO Rendezvous Communication tab are not applied to the current session of RTView. Click the **Save** button to record all TIBCO Rendezvous application options to the **RVOPTIONS.ini** initialization file and restart RTView to apply these changes.



Field Name	Description
Service	Rendezvous Session Service. Default is TIBCO Rendezvous service default.
Network	Rendezvous Session Network. Default is TIBCO Rendezvous network default (primary network interface for the host computer).
Daemon	Rendezvous Session Daemon. Default is TIBCO Rendezvous daemon default (local daemon on TCP socket 7500).
RVA Flag	Run in **rva** mode.
Host	Host running **rva**. Default is the local host. This option is only used if running in **rva** mode.
Port	Port on which **rva** is running. Default is the **rva** default. This option is only used if running in **rva** mode.

TIBCO Rendezvous Monitoring Tab

This tab allows you to configure your TIBCO Rendezvous monitoring settings.



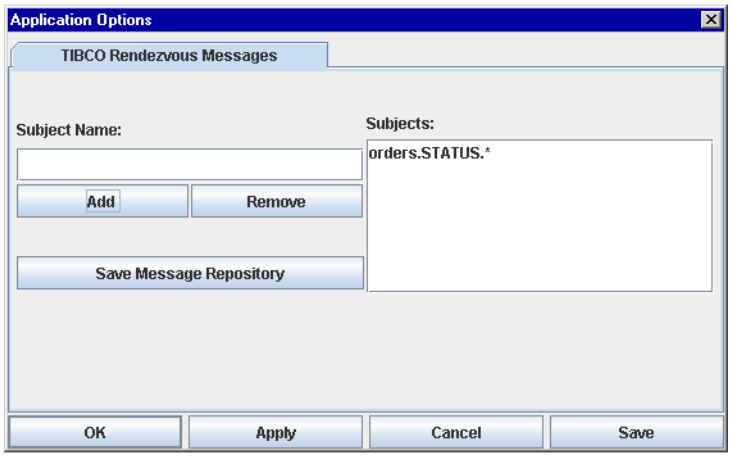
Field Name	Description
Show Monitor Tables in Data Dialog	Select whether RTViewDs monitoring tables are displayed in the Subject list in the "Attach to TIBCO Rendezvous Data" dialog. The default is disabled.
Monitor Host Status Messages on Start	Select to begin listening for TIBCO Rendezvous information messages on startup rather than waiting until a display references **RTViewDs.HostStatus** tables. The default is disabled. See "Attach to TIBCO Rendezvous Data" for more information.
Host Status History Depth	Set the number of rows of data to keep for each host in the **RTViewDs.HostStatus** tables. The default is **1000**. A value of **0** sets it to keep no rows. A value of **-1** sets it to keep an unlimited number of rows. See "Attach to TIBCO Rendezvous Data" for more information.
Run Rvtrace on Start	Select to begin running **rvtrace** on startup rather than waiting until a reference is made to a **RTViewDs.Multicast** table. The default is disabled. See "Attach to TIBCO Rendezvous Data" for more information.
Rvtrace Period	Set the **rvtrace** update period. The default is **10**. A value of **0** sets it to use the default.
Rvtrace Interface	Specify the Network Interface to use when **rvtrace** is started. Additional information about determining the Network Interface can be found in the TIBCO Rendezvous Administration manual. Refer to Chapter 11, "Protocol Monitor (rvtrace)" and, in particular, to the document titled "Selecting the Network Interface".

Note: RTView runs **rvtrace** to collect TIBCO Rendezvous monitoring information. The **rvtrace** application is part of your TIBCO Rendezvous installation and uses a **pcap** facility to capture network packets. Before using **rvtrace**, you must first ensure that the **pcap** facility is properly installed. It is strongly recommended that you run your TIBCO Rendezvous daemon with the **-reliability 0** option when using **rvtrace** to monitor TIBCO Rendezvous. This will prevent the daemon from requesting retransmissions and contributing to busy network problems. See your TIBCO Rendezvous documentation for more information.

TIBCO Rendezvous Messages Tab

Message subjects listed on the **TIBCO Rendezvous Message** tab are used to populate drop down menus in the "Attach to TIBCO Rendezvous Data" and "Define TIBCO Rendezvous Command" dialogs. RTView starts listening for new messages added from the **Subject Name** field after you click **OK**, **Apply**, or **Save**.

The "Message Repository" saves message field information for all Message subjects and uses that information to populate the initial values of drop down menus. Message subjects will not be added to drop down menus until RTView has received at least one TIBCO Rendezvous message with that subject.



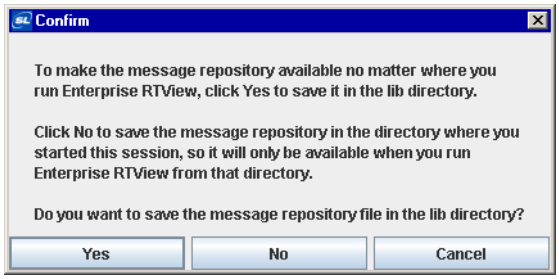
Field Name	Description
Subject Name	Enter the name of a TIBCO Rendezvous Message Subject.
Add	Add Subject Name to listing.
Remove	Select **Subject** from listing and click **Remove** to delete.
Save Message Repository	Click to save file that records current Subjects and applies values to drop down menus.

Message Repository

Click **Save Message Repository** to create a file that saves message field information for all Message Subjects. In order for a subject to be saved in the Message Repository, RTView must have received at least one TIBCO Rendezvous message with that subject. Information stored in the Message Repository file will be used to populate the initial values of drop down menus.

Note: The saved file will be named **rvrepository.xml**. If the name of the Message Repository file is changed, RTView will not be able to locate the file. As a result, drop down menus will populate based on message subjects added from **Application Options** dialog or those typed directly into the **Attach to TIBCO Rendezvous Data** dialog.

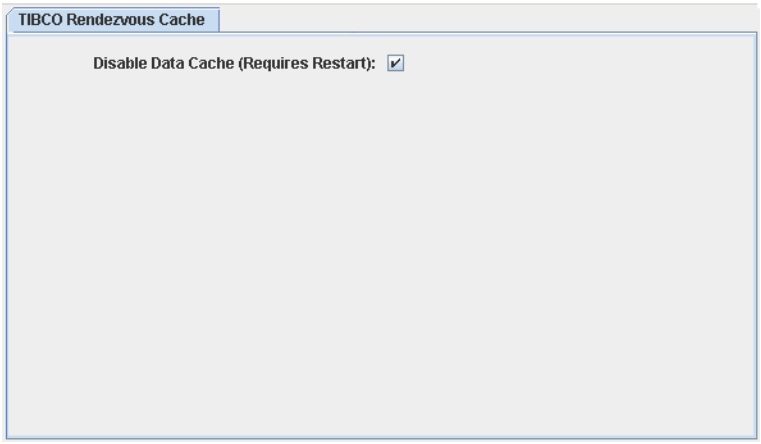
When you click **Save Message Repository**, a confirmation dialog will appear to verify the directory in which you would like to save the Message Repository file. If you specified a directory for your initialization files (see "RTV_JAVAOPTS"), all repository files will be saved to, and read from, that directory. If you select the lib directory, the repository file will be available from any directory where you run RTView. If you do not select the **lib** directory, the repository file will be saved in the directory where you started the current session and will only be available when you run RTView from that particular directory.



See the TIBCO Rendezvous "Message Repository" section for technical details on editing an existing Message Repository file.

TIBCO Rendezvous Cache Tab

The **Disable Data Cache** option allows you to disable the caching of data in the TIBCO Rendezvous Data Source.



Field Name	Description
Disable Cache Data	If selected, the initial update on data attachments to multiple subjects (e.g. **orders.STATUS.***) will return a table with a row for each message, but subsequent updates will include only the rows that have changed. **Note:** You must **Save** and restart for this option to take effect. This option is useful when using TIBCO Rendezvous data attachments as input to the Cache data source or the Historian. If selected, the **Disable Cache Data** option will apply to the following RTViewDs tables: • **RTViewDs.HostStatusCurrent** • **RTViewDs.HostStatusTotal** See "TIBCO Rendezvous RTViewDs Fields" for more information.

RTView Deployment - TIBCO Rendezvous

This section contains details about the deployment process that are specific to your data source. Please go to the Deployment section of this documentation for instructions on how to implement your RTView deployment option. Return to this section whenever you are instructed to refer to deployment information that is specific to your data source.

System Requirements and Setup

The TIBCO Rendezvous data source has additional System Requirements and Setup. See "TIBCO Hawk - System Requirements and Setup" for more information.

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options", "Application Options - TIBCO Rendezvous", and "RTV_JAVAOPTS" for more information.

Include the following initialization files when you deploy RTView with this data source:

File Name	Description
RVOPTIONS.ini	Contains data source options for TIBCO Rendezvous.
RVALIAS.ini	Contains TIBCO Rendezvous alias definitions. See "Create TIBCO Rendezvous Message Alias" for more information. **Note:** This file is optional.

Note: Options specified using command line parameters override values set in initialization files.

TIBCO Rendezvous Demos

Except where noted, all demos can be run in three ways, as an application, or via rich or thin client in a browser.

Before You Begin

Start the Demo Server

Thin Client Demo only.

There is a thin client demo already installed on the "RTView Demo Server".

In an initialized command window (see "Initializing a Command Prompt or Terminal Window"):

Type **run_startup_demoserver**

Data Source Demo

The Data Source Demo is designed to illustrate each data source.

1. **Start the Simulators**

   Start the simulators for each data source you will be using. To run the "TIBCO Rendezvous Data Simulator":

   In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory and type:

   run_rvsimdata

2. **Run Demos - Application or Thin Client Browser**

   Application Demo

   1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
   2. To view the demo, type:

      run_viewer

   3. To edit the demo, type:

      run_builder

   Thin Client Browser Demo

   "Start the Demo Server" if it is not running.

   1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
   2. Start the Display Server by typing:

      run_displayserver

   3. Open a browser and navigate to **http://localhost:8068/dstutorial**.

TIBCO Rendezvous Monitor Demo

Displays in this demo are designed for monitoring TIBCO Rendezvous.

1. **Start the Simulators.** See "Data Source Demo" for more information.

2. **Run Demos - Application or Thin Client Browser**

   Application Demo

   1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/rvmonitor** directory.
   2. To view the demo, type:

      run_viewer

   3. To edit the demo, type:

      run_builder

   Thin Client Browser Demo

   "Start the Demo Server" if it is not running.

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/rvmonitor** directory.
2. Start the Display Server by typing:

   run_displayserver

3. Open a browser and navigate to **http://localhost:8068/rvmonitor**.

Quick Start Tutorial: TIBCO Rendezvous

This Quick Start Tutorial provides you with the fundamentals on how to use RTView with a TIBCO Rendezvous data source. Once completed, you can swiftly apply this knowledge to building your own real-time dashboard displays for visual access to your TIBCO Rendezvous data.

Learn to:

- Animate graphic objects with TIBCO Rendezvous data
- Create a drill down display with TIBCO Rendezvous data

Note: The TIBCO Rendezvous data source may not be licensed in your RTView installation.

Get Started

This tutorial requires the following:

- Register for a license key. If you have not, you must do so before continuing. See "Registration" for more information.
- "Quick Start Tutorial". This tutorial requires that you have a working knowledge of RTView. We recommend that you complete the Quick Start Tutorial before continuing.

Note: By default, RTView and the TIBCO Rendezvous data simulator send and receive TIBCO Rendezvous messages using the TIBCO Rendezvous Daemon (**rvd**) with the default TIBCO Rendezvous service, the primary network interface for the host computer, tcp:7500. This tutorial uses these settings. If your application will use different settings, see "Application Options - TIBCO Rendezvous" for setup information.

Start the TIBCO Rendezvous Data Simulator

In this exercise you start the TIBCO Rendezvous simulator which is the data source used in this tutorial. The simulator sends TIBCO Rendezvous messages that are used to animate objects in your display.

In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"), go from your installation directory to the **demos** directory and type:

On Windows: **start run_rvsimdata**

On UNIX: **run_rvsimdata &**

The TIBCO Rendezvous data simulator runs as a background process and is ready when dots appear across the screen.

Note: You must follow this initialization process for each new terminal window you open. See the "Setup" section for more details about setting up your environment.

Start the Display Builder

If you are already logged onto the Display Builder, skip this section and go to "Create a Display".

1. In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window") type:

   run_builder

2. Login to the Display Builder. By default, the Display Builder does not require a login. "Login" can be enabled at setup to support "Role-based Security". The default user name and password are:

   User Name: admin

   Password: admin

Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role. See "Role-based Security" for more information.

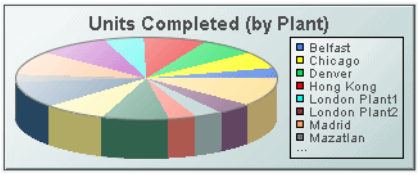
You are now ready to create a display using the TIBCO Rendezvous data source.

Create a Display

At this point you have:

- Registered for a license key. See "Registration" for more information.
- Logged on to the Display Builder
- Setup the TIBCO Rendezvous Data Source
- Completed the "Quick Start Tutorial"

In this tutorial you use the TIBCO Rendezvous data simulator as your data source to create an animated pie graph that displays production numbers per plant, as seen below.



As you saw in the "Quick Start Tutorial", the data structure of tables and graphs (tabular data) enables RTView to automatically create several data source specific, built-in Substitutions for you. You will see these built-in Substitutions used in the target display when you create the drill down. For more information on Substitutions, see "Substitutions".

In this exercise, you create a drill down using the previously created display, **rv_dd_qs.rtv**, as the target display. First you will set the pie graph to display units completed per plant. Then you will create a drill down that will open a bar graph that shows more detailed data for each plant.

Add the TIBCO Rendezvous Subject to List of Available Subjects

Adding the TIBCO Rendezvous subject makes it available for animating graphic objects in your display.

1. In the Display Builder, select **Tools>Options** to open the **Application Options** dialog.
2. Select the "TIBCO Rendezvous Messages Tab" and enter the name of a message subject.

   Subject Name - Enter **orders.STATUS.*** A series of messages with subject names that begin **orders.STATUS** are generated by the TIBCO Rendezvous data simulator.

   Click **Add**.

   Click **OK** to apply and close the **Application Options** dialog.

The TIBCO Rendezvous subject is now available for animating graphic objects in your display.

Display Data in a Pie Graph

In this exercise you add a pie graph and then display the data in the pie graph by attaching it to the data source.

1. Click  and click again in the Working Area to place the pie graph.
2. In the **Object Properties** dialog:

   label (category: Label) - Change the name of the label to **Units Completed (by Plant)**.

   legendWidthPercent (category: Legend) - Increase to **50**.

   valueTable (category: Data) - Right-click in the **Property Name** field and select **Attach to Data>RV**.
3. In the "Attach to TIBCO Rendezvous Data" dialog:

   Message Subject - Enter **orders.STATUS.***

   Message Field - Click the ellipsis button  to open the **Select Columns** dialog.
4. In the **Select Columns** dialog:

   Select **PLANT** in the **Available Columns** list and click **Add**.

   Select **UNITSCOMPLETED** in the **Available Columns** list and click **Add**.
5. Click **OK** to close the **Select Columns** dialog.
6. In the **Attach to TIBCO Rendezvous Data** dialog:

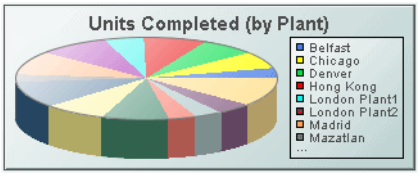
   Filter - Click to select the check box.

   Filter Field Name - Select **PLANT**.

Filter Field Value - ***** should already be selected.

7. Click **OK** to apply these values and close the **Attach to TIBCO Rendezvous Data** dialog.

The pie graph is now animated by the TIBCO Rendezvous data. Since the values in the **Units Completed** column are numeric, this data is graphed in the pie. Since the values in the **Plant** column contain text, they are shown in the legend.



You are now ready to create the drill down.

Create a Drill Down Target in the Pie Graph

In this exercise, you create a drill down in the pie graph using the previously created display, **rv_dd_qs.rtv**, as the target.

1. In the **Object Properties** window:

   drillDownTarget (category: Interaction) - Double-click in the **Property Name** field to bring up the **Drill Down Properties** dialog.

2. In the "Drill Down Properties" dialog:

   Apply Drill Down To - Select **Named Window** from the drop down menu.

   Window Name - Enter **rv**

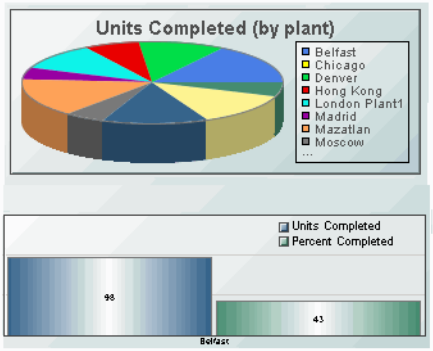
   Drill Down Display Name - Select **dstutorial\rv_dd_qs.rtv** from the drop down menu.

3. Click **OK** to attach the drill down target and close the dialog.

View the Drill Down Display

In this exercise, you drill down to the target display.

1. Double-click on any wedge in the pie graph to drill down to detailed data. The target display opens.



Note: You must select the top of a wedge for the drill down to open.

2. Double-click on another wedge in the pie and the same display is used to show different data based on the wedge you select.
3. Close the drill down window.
4. In the Display Builder select **File>Save**.

Go to the main "Quick Start Tutorial"

TIBCO Rendezvous Data Simulator

A TIBCO Rendezvous data simulator is provided to allow customers to work with RTView without setting up their own messages. The simulator creates, updates, and sends out messages with the following subjects:

- orders.STATUS.Belfast
- orders.STATUS.Chicago
- orders.STATUS.Denver
- orders.STATUS.HongKong
- orders.STATUS.London
- orders.STATUS.Madrid
- orders.STATUS.Mazatlan
- orders.STATUS.Moscow
- orders.STATUS.Palo Alto
- orders.STATUS.Seattle
- orders.STATUS.Tokyo
- WEATHER.REGIONAL.WEST
- WEATHER.REGIONAL.EAST

Running the Simulator

From an initialized terminal window (see "Initializing a Command Prompt or Terminal Window") type:

run_rvsimdata

By default, the simulator runs in **rvd** mode. You may run the simulator in **rva** mode by using the **-rva** command line parameter.

Note: The TIBCO Rendezvous Agent Process (**rva**) must be running before you start the simulator in **rva** mode. The **rva**/**rvd** mode of the simulator must match the mode of RTView in order to work correctly. By default, the Display Builder and Display Viewer Application run in **rvd** mode. See "Communicating with TIBCO Rendezvous" for more information on **rva** and **rvd**.

Note: You must run the TIBCO Rendezvous simulator from a command prompt or terminal window to use command line parameters.

Command line parameters for the simulator include:

Name	Description
-u (milliseconds)	Set update rate in milliseconds. Default is 2000.
-rvservice:	Rendezvous Session Service. Default is TIBCO Rendezvous service default.
-rvnetwork:	Rendezvous Session Network. Default is TIBCO Rendezvous network default (primary network interface for the host computer).
-rvdaemon:	Rendezvous Session Daemon. Default is TIBCO Rendezvous daemon default (local daemon on TCP socket 7500).
-rva	Run in **rva** mode.
-rvahost:	Name of host running **rva**. The default is the local host. This option is only used if running in **rva** mode.
-rvaport:	Port on which **rva** is running. The default is **rva** default. This option is only used if running in **rva** mode.

The data simulator messages for subject **orders.STATUS.*** contain the following fields:

Field Name	Field Type	Field Description
ORDERNUMBER	TibrvMsg.STRING	Order number
PERCENTCOMPLETED	TibrvMsg.F32	Percent of the order that is completed
PLANT	TibrvMsg.STRING	Plant name
X	TibrvMsg.F32	X location coordinate
Y	TibrvMsg.F32	Y location coordinate
UNITSCOMPLETED	TibrvMsg.F32	Number of units completed for this order
ORDERDATE	TibrvMsg.STRING	Date of the order

SCHEDULEDDATE	TibrvMsg.STRING	Date the order is scheduled for completion
EXPECTEDDATE	TibrvMsg.STRING	Expected date of completion
CUSTOMER	TibrvMsg.STRING	Customer name
STATUS	TibrvMsg.STRING	Order status: WORKING, BROKEN, COMPLETED

The data simulator messages for subject **WEATHER.REGIONAL.*** contain the following fields and properties:

Field Name	Field Type	Field Description
Name	String	WEST or EAST
nowPlus1Hr	Date	Current time plus one hour
NumberReports	int	Number of weather reports
RegionalData	XML String	XML string containing "Weather Data"
DateStringTimeString	String	Current time

Message Repository

A Message Repository file is used to populate the initial values of drop down menus in the "Attach to TIBCO Rendezvous Data" and "Define TIBCO Rendezvous Command" dialogs. See "Application Options - TIBCO Rendezvous" for information on how to create a Message Repository file.

A sample file, **rvrepository.xml**, is provided for use with the TIBCO Rendezvous data simulator. In order to work without a Message Repository, move this file out of the lib directory (which is located in your installation directory).

It is possible to edit an existing Message Repository file, however, the file name **rvrepository.xml** cannot be modified. If **rvrepository.xml** is not found in the specified directory or your current working directory, RTView will look in the **lib** directory. If the Message Repository file is not found, drop down menus will remain empty until message subjects are added from the **Application Options** dialog or typed directly into the **Attach to Data** dialog. See "RTV_JAVAOPTS" for more information.

To edit an existing Message Repository file, supported tags and attributes are as follows:

Tag	Attribute		Description
rvrepository	xmlns		Top level tag that includes the **namespace** attribute **xmlns**, which must be defined as www.sl.com (**xmlns="www.sl.com"**)
subject	name		Subject name
	field	name	Field name
		choice	Possible field value

An example Message Repository file:

```
<?xml version="1.0"?>
    <rvrepository xmlns="www.sl.com">
```

```
    <subject name="orders.STATUS.London">
        <field name="ORDERNUMBER" type="TibrvMsg.STRING"/>
        <field name="PERCENTCOMPLETED" type="TibrvMsg.F32"/>
        <field name="PLANT" type="TibrvMsg.STRING"
            choice="London Plant1"
            choice="London Plant2"/>
        <field name="X" type="TibrvMsg.F32"/>
    <field name="Y" type="TibrvMsg.F32"/>
    <field name="UNITSCOMPLETED" type="TibrvMsg.F32"/>
    <field name="ORDERDATE" type="TibrvMsg.STRING"/>
    <field name="SCHEDULEDDATE" type="TibrvMsg.STRING"/>
    <field name="EXPECTEDDATE" type="TibrvMsg.STRING"/>
        <field name="CUSTOMER" type="TibrvMsg.STRING"/>
         <field name="STATUS" type="TibrvMsg.STRING"
    choice="WORKING"
    choice="BROKEN"
    choice="COMPLETED"/>
     </subject>
</rvrepository>
```

TIBCO Rendezvous Data Source Command Line Options

In addition to General Options, the following command line arguments are enabled with the TIBCO Rendezvous data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: **"-sub:$data:my Data"**).

Name	Description
-rvds:showindialog:	Sets whether the RTViewDs monitoring tables are displayed in the **Subject** list in the "Attach to TIBCO Rendezvous Data" dialog. The default is disabled. Set to **true** or **false**. Example: **-rvds:showindialog:true**
-rvds:rtviewds.hostatus.onstart:	Sets RTView to begin monitoring TIBCO Rendezvous on startup rather than waiting until a display references a **HostStatus** table. The default is disabled. Set to **true** or **false**. Example: **-rvds:rtviewds.hostatus.onstart:true**
-rvds:rtviewds.hoststatus.history:	Sets the number of rows of data to keep for each host. The default is **1000**. A value of 0 sets it to keep no rows. A value of **-1** sets it to keep an unlimited number of rows. Example: **-rvds:rtviewds.hoststatus.history:-1**

-rvds:rtviewds.rvtrace.onstart:	Sets **rvtrace** to begin running on startup rather than waiting until a reference is made to a **Multicast** table. The default is disabled. Set to **true** or **false**. Example: **-rvds:rtviewds.rvtrace.onstart:false**
-rvds:rtviewds.rvtrace.period:	Sets the **rvtrace** update period. The default is **10**. A value of **0** sets it to use the default. Example: **-rvds:rtviewds.rvtrace.period:0**
-rvds:rvtrace.interface	Specify the Network Interface to use when rvtrace is started. Example: **run_viewer -rvds:rvtrace.interface:\Device\NPF_{FBE746DE-94E5-4116-92A3-578924520ADE}** This is equivalent to running **rvtrace** using the -i switch. Example: **rvtrace -i \Device\NPF_{FBE746DE-94E5-4116-92A3-578924520ADE}** **Note:** Additional information about determining the Network Interface can be found in the TIBCO Rendezvous Administration manual. Refer to Chapter 11, "Protocol Monitor (rvtrace)" and, in particular, to the document titled "Selecting the Network Interface".
-rvservice:	TIBCO Rendezvous Session Service. Default is TIBCO Rendezvous service default. Example: **-rvservice:7475**
-rvnetwork:	TIBCO Rendezvous Session Network. Default is TIBCO Rendezvous network default (primary network interface for the host computer). Example: **"-rvnetwork:;"**
-rvdaemon:	TIBCO Rendezvous Session Daemon. Default is TIBCO Rendezvous daemon default (local daemon on TCP socket 7500). Example: **-rvdaemon:7475**
-rvsubject:	Add a message subject that RTView will listen for and use to populate dialog drop down menus. Example: **-rvsubject:mySubject.*.***
-rvmsgparent:(fieldname)	Name of parent field for nested messages. Example: **-rvmsgparent:^data^**
-rva	Run in rva mode. Example: **-rva**

-rvahost:	Host running **rva**. Default is the local host. This option is only used if running in **rva** mode. Example: **-rvahost:computer6**
-rvaport:	Port running **rva**. Default is the **rva** default. This option is only used if running in **rva** mode. Example: **-rvaport:7600**

Communicating with TIBCO Rendezvous

Note: The TIBCO Rendezvous data source may not be licensed in your RTView installation.

By default, the Display Builder and Display Viewer Application communicate with TIBCO Rendezvous using the TIBCO Rendezvous Daemon (**rvd**) process. The **rvd** process is automatically started by TIBCO Rendezvous whenever it is needed and runs as a background service.

In order to facilitate testing, you may run the Display Builder and Display Viewer Application using **rva**. The **rva** process must already be running before the Display Builder or Display Viewer Application are started in **rva** mode.

Running the TIBCO Rendezvous Agent Process

The TIBCO Rendezvous Agent Process (**rva**) must be running in order for RTView to send and receive messages. RTView assumes that **rva** is running on your local machine using the default port. If **rva** is running on a different host or port, you will need to specify both the host and the port either in the **Application Options** dialog or on the command line. See "Application Options - TIBCO Rendezvous" and "TIBCO Rendezvous Data Source Command Line Options" for more information.

Review the appropriate section below to determine what host and port parameters you will need to specify. If you are not sure which section applies to you, contact your TIBCO Rendezvous system administrator.

If you have never run **rva**, you must configure **rva** to import and export your message subjects. You will also need to configure **rva** in order to utilize the RTView data simulator, which sends TIBCO Rendezvous messages.

If rva is already running on your local machine

You do not need to specify a host. You do not need to specify a port if rva is running on the default port. Otherwise, you will need to specify a port in the Application Options dialog or on the command line. See "Application Options - TIBCO Rendezvous" and "TIBCO Rendezvous Data Source Command Line Options" for more information. Please refer to TIBCO Rendezvous documentation for more information on running **rva**.

You will need to configure the TIBCO Rendezvous Agent Process in order to utilize the data simulator. The data simulator, which sends TIBCO Rendezvous messages, is necessary to work through the examples in the documentation. See "Configuring the TIBCO Rendezvous Agent Process" for more information.

If rva is running on another machine on your network

You need to specify the name of the host that is running **rva**. You do not need to specify a port if **rva** is running on the default port. Otherwise, you will need to specify both the host and the port in the **Application Options** dialog or on the command line. See "Application Options - TIBCO Rendezvous" and "TIBCO Rendezvous Data Source Command Line Options" for more information. Please refer to TIBCO Rendezvous documentation for more information on running **rva**.

You will need to configure the TIBCO Rendezvous Agent Process in order to utilize the data simulator. The data simulator, which sends TIBCO Rendezvous messages, is necessary to work through the examples in the documentation. See "Configuring the TIBCO Rendezvous Agent Process" for more information.

If rva is not already running

You need to start **rva** on your local machine:

1. Open a Command Prompt window, go to **<rendezvous installation>\bin** and type:

   rva -store rvafilename

where **rvafilename** is the name of the file where your **rva** settings are stored. If you have never run **rva** this file will not exist. In this case, **rva** will create a file by the name you specify and save the **rva** settings to that file.

2. This will start **rva** and print some information regarding **rva** to the console. Note the following information:

```
RVA: Http interface:
RVA Listen:
RVA Host:
```

RVA Listen will be the port and **RVA Host** will be the host that you will use to run RTView. The **RVA Http interface** (e.g.: http://hostname:7680/) is necessary to configure **rva** to work with messages sent by the data simulator.

3. If the RVA Listen is 7600 and RVA Host is the name of the machine you are running on, you will not need to specify either parameter. Otherwise, you will need to specify the host and/or the port in the **Application Options** dialog or on the command line. See "Application Options - TIBCO Rendezvous" and "TIBCO Rendezvous Data Source Command Line Options" for more information. Please refer to TIBCO Rendezvous documentation for more information on running **rva**.

4. Since you have never run **rva** you must configure the TIBCO Rendezvous Agent Process to import and export your message subjects. You will also need to configure **rva** in order to utilize the data simulator. The data simulator, which sends TIBCO Rendezvous messages, is necessary to work through the examples in the documentation. See "Configuring the TIBCO Rendezvous Agent Process" for more information.

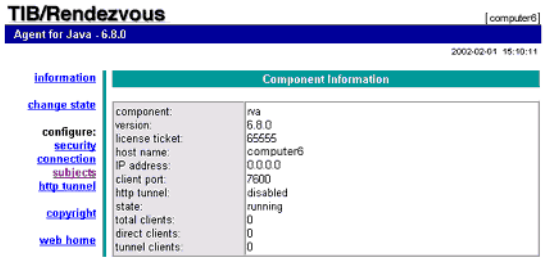
Configuring the TIBCO Rendezvous Agent Process

To configure the TIBCO Rendezvous Agent Process (**rva**) to import and export message subjects:

1. Open a browser to the **rva** http interface address. If you do not know the rva http interface, the default is:

http://host:7680/

where host is the name of your machine. If this does not work, contact your TIBCO Rendezvous system administrator. Your setup may be running with a different http interface or you may not have permission to change the **rva** settings. The browser interface should look similar to the following:



2. Click on the **subjects** link. The screen should look similar to the following:

Type the message subject into the **Add Subject** field. The subject to add for the data simulator is

orders.STATUS.*

3. Click on the button marked **Add for Import and Export**.

Once **rva** has been setup, the subject configuration information will be stored in the specified settings file. See the TIBCO Rendezvous documentation for more information regarding configuring **rva**.

XML Data Source

The XML data source allows data to be delivered via the standard RTView XML schema. In many cases, this is used in prototyping since data can be created via a simple XML text editor. In other cases, customers use the XML format as a quick means for new data sources to provide data to RTView.

An API is provided for creating custom data adapters that can be integrated into the Display Builder environment and provide unique dialogs for data attachment. However, in some cases it is quicker and sufficient to provide the data via XML and use the standard XML data attachment dialog.

This section includes:

XML System Requirements and Setup

System Requirements

The XML data source has no additional "System Requirements".

Setup

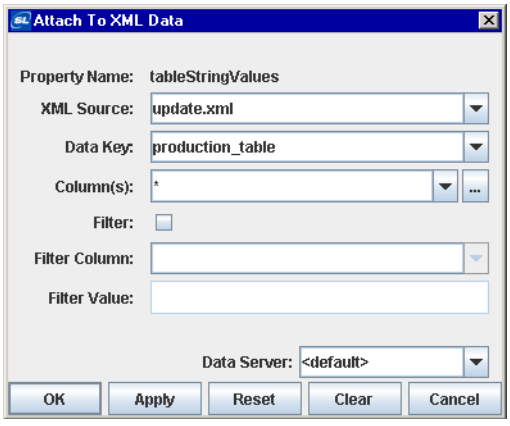
The XML data source requires no additional "Setup".

Attach to XML Data

The **Attach to XML Data** dialog, which is used to connect an object property to an element in your XML data, can be accessed from the **Object Properties** window. Once a property has been attached to an XML element, it receives continuous updates.

When an object property is attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing this value from the **Object Properties** window is no longer possible. To remove the data attachment, and resume editing capability in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Right-click on the Property Name from the **Object Properties** window and select **Attach to Data>XML** to display the **Attach to XML Data** dialog. The **Attach to XML Data** dialog provides three drop down menus that allow you to specify information regarding an XML element. See "Creating XML Sources" for technical details on creating and formatting an XML source.



Field Name	Description
XML Source	Name of XML source containing XML element.
Data Key	Key corresponding to XML element.
Column(s)	If XML element is tabular, you can select which column(s) to display.

Filter	Check box to indicate whether or not to filter the XML element. Filters can only be used for tabular data. See "Row Filtering"
Filter Column	Name of the column to use as a filter. Multiple column names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col 3). If your column name contains a space or a semicolon, then the entire name must be enclosed in single quotes.
Filter Value	Value that the **Filter Column** must equal. Multiple filter values should be entered as a nested list, where values for a given column are separated by commas within a semicolon (;) delimited list (i.e. val1,val2;val3,val4;val5,val6). If your filter value contains a space or a semicolon, then the entire value must be enclosed in single quotes. When ***** is entered as a filter field value, data for all values in the specified filter column will be used to update the object property. When **"*"** is entered, only the literal comparative value will be used. These are only allowed for objects which display tabular data.
Data Server	Select to read data through your configured Data Server and not directly from the XML data source. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab". **None** - Bypass data being redirected through the specified Data Server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". **Multi-Server Attachment** - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more Named Data Servers (e.g. ds101;ds102). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify __**default** and __**none** (e.g. __**default**;**ds101**;**ds102**). **Note:** The values __**default** and __**none** begin with two underscore characters. Alternatively, a value of ***** can be entered to specify all data servers, including __**default** and __**none**. When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received. A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways: **1.** The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows. **2.** The multi-server attachment can be applied as the **Table** argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

The XML Source drop down menu lists all available XML sources. Drop down menus for **Data Key**, **Column(s)**, and **Filter Column** populate based on the selected XML source. The **XML Source** field automatically displays the name of the default XML source. The **Column(s)** and **Filter Column** drop down menus will only contain options if the selected Data Key is a tabular element that has been included in an XML update. If the item you require is not listed, type your selection into the field. For information on adding XML sources or selecting a default XML source, see "Application Options - XML".

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information entered into the dialog is validated against elements included in an XML update.

The following describes the significance of the **Attach to XML Data** validation colors:

Blue	Unknown	Entry does not match any known XML source*.
White	Valid state	Entry is valid.
Red	Invalid state	XML source is valid, but Data Key, Column(s) or Filter Column selected are not.
Gray	Not Required	Field does not require a value. This applies to the Column(s), Filter Column or Filter Value field for data keys that are not tabular elements.

*If an XML source is validated as Unknown, RTView will attempt to read it when you click **OK** or **Apply**.

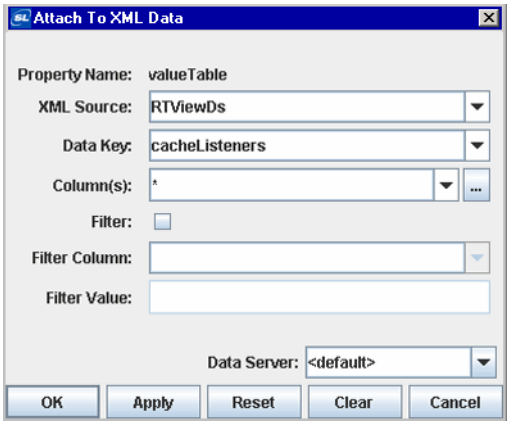
Substitutions

Substitutions allow you to build open-ended displays in which data attachments depend on values defined at the time the display is run. Generic names, such as **$data1** and **$data2**, are used instead of values for specific data keys. Later when the display is running, these generic values are defined by the actual names of specific keys, such as **element1_data** and **element2_data**. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

RTViewDs

The RTViewDs source provides information about the listeners currently active in the XML data source. The **RTViewDs.Server** source provides information about all listeners that a data server is currently maintaining for all of its clients. That is, while the **RTViewDs.Server** source provides information from a data server about all listeners from all of its clients, the RTViewDs source only provides information about listeners that the current client (e.g. Display Builder, Display Viewer) has redirected to data servers.

In the **Attach to XML Data** dialog, manually enter **RTViewDs** or **RTViewDs.Server** in the XML Source field. These options are not available from the drop down list.



XML Source	Data Key	Description	Column Name	Description
RTViewDs	listeners	Returns a table that contains a row for each unique data attachment to another XML source, and a row for each unique data attachment that has been redirected to a Data Server.	Source	The name of the XML source (file), or the name of the Data Server.
			Local	When checked, indicates that the source is an XML source. When unchecked, indicates that the source is a Data Server.
			Adapter	The name of the data adapter, which is xml for local XML attachments and cache, sql, and so forth, for redirected listeners.
			Key	The entire data attachment string.
			Listeners	The total number of listeners for the attachment string.
			Updated	Specifies the time that the data was most recently applied to the listener.

	cacheListeners	Returns a table that contains a row for each unique cache data attachment that has been redirected to a Data Server. **Note:** The **cacheListeners** table does NOT show local cache listeners but rather, only cache listeners that have been redirected to a Data Server. The Local and Adapter columns are not specified in this table because only remote listeners for the cache data adapter are shown.	Source	The name of the Data Server to which the cache attachment has been redirected.
			Key	The entire data attachment string.
			Listeners	The total number of listeners for the attachment string.
			Updated	Specifies the time that the data was most recently applied to the listener.
			Cache Name	The name of the cache in the cache data attachment.
			Table Name	The name of the table in the cache data attachment.
			Columns	The name of the table column in the cache data attachment.
			Filter Columns	The names of columns that have a filter applied to the.
			Filter Values	The value of the filters applied to the columns (listed in Filter Columns).
RTViewDs.Server	listeners	Returns a table that contains one row for each unique data attachment for each client of the data server. Client ID Numeric ID of the client.	Address	IP address of the client.
			Host	Hostname of the client, if available, or http if the client is connected via the rtvdata servlet.

		Adapter	The name of the data adapter, which is xml for local XML attachments and cache, sql, and so forth, for redirected listeners.
		Key	The entire data attachment string.
		Listeners	Total number of listeners for the attachment string.
		Pushed	Time data was most recently pushed from the server to the client.
		Seq	Sequence number assigned to most recent data push.
cacheListeners	Returns a table that contains one row for each unique cache data attachment for each client of the data server. Client ID Numeric ID of the client.	Address	IP address of the client.
		Host	Hostname of the client, if available, or http if the client is connected via the rtvdata servlet.
		Key	The entire data attachment string.
		Listeners	Total number of listeners for the attachment string.
		Pushed	Time data was most recently pushed from the server to the client.
		Seq	Sequence number assigned to most recent data push.
		Cache Name	The name of the cache in the cache data attachment.
		Table Name	The name of the table in the cache data attachment.
		Columns	The name of the table column in the cache data attachment.
		Filter Columns	The names of columns that have a filter applied to them.
		Filter Values	The value of the filters applied to the columns (listed in **Filter Columns**).

Select Table Columns

From the **Attach to XML Data** dialog you can specify which table columns to display and in what order they will appear. In order to populate the listing of available columns, you must first select a valid XML source and data key.

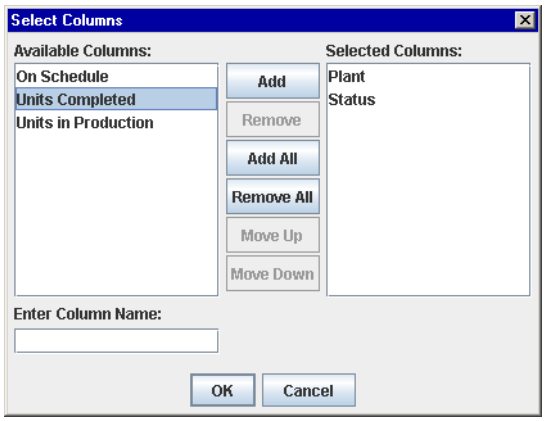
Click on the ellipses button in the **Column(s)** field (or right-click in the **Column(s)** field and choose **Select Columns)** to display the **Select Columns** dialog. The dialog should contain a list of Available Columns that you can add to your table.

To add a column, select an item from the **Available Columns** list and click the **Add** button. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of columns in a table by arranging the items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

Validation colors indicate whether selected columns are valid. However, if even one column selected is invalid, the **Column(s)** field in the **Attach to XML Data** dialog will register as an invalid entry.

Note: Invalid columns will not update.

If no data is available for a table row within a selected column, the table cell will display one the following values: **N/A**, **false**, **0**, or **0.0**.



The following describes the **Attach to XML Data dialog** commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied.

XML Data Source Substitutions

In addition to standard built-in "Substitutions", this data source also sets the following drill down substitutions:

Substitution Value	Definition
$XrowName (where X is the # of the rowname)	Data from a selected row or object. A substitution is defined for each part of the rowname. For example, if the rowname was **plant1:load:station7:machine8** the substitutions would be: **$1rowName:plant1** **$2rowName:load** **$3rowName:station7** **$4rowName:machine8**
$filterfield	Filter field name from the selected row or object.
$filtervalue	Filter field value from the selected row or object.

Application Options - XML

Select **Tools>Options** in the Display Builder to access the **Application Options** dialog.

Options specified in the XML tab can be saved in an initialization file (**OPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server and Historian to set initial values. If no directory has been specified for your initialization files and **OPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

Note: Options specified using command line arguments will override values set in initialization files.

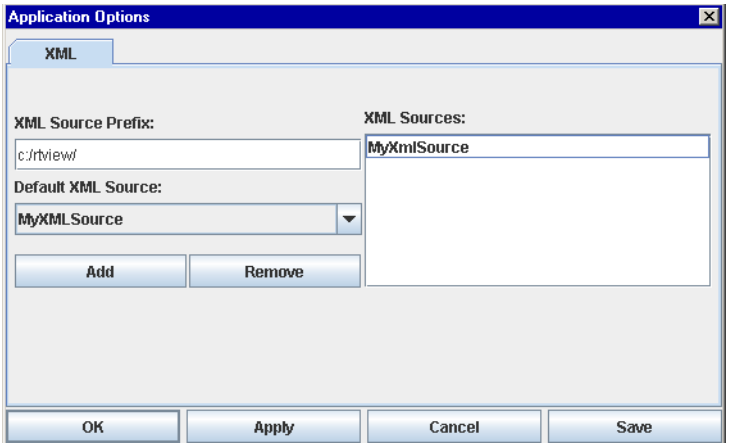
XML Tab

This tab allows you to allows you to add or remove XML sources, set the default XML source, and enter an XML source prefix.

Adding an XML source allows you to associate a name, which you will use when creating data attachments, with the full path to the XML source file used to update objects in your display. See "Creating XML Sources" for technical details on creating and formatting your own XML source file.

Entering an XML source prefix allows you to use the same path for several XML sources. The XML source prefix will be affixed to the beginning of all XML sources with the **Use XML Source Prefix** check box selected in the "Edit XML Source" dialog.

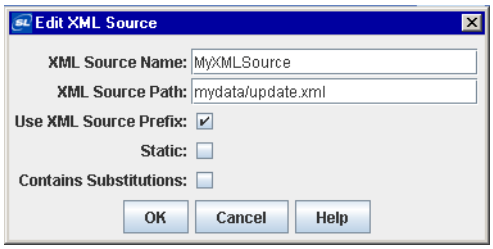
Once you add an XML source, it will be highlighted in yellow indicating that it has not been read. When you click **OK**, **Apply**, or **Save**, RTView will attempt to read it and populate menu items in the **Attach to XML Data** dialog unless the XML source definition has the **Contains Substitutions** check box selected. If the XML source remains yellow, the file was not found. When this information is saved to the initialization file, it will be used to populate **Attach to XML Data** dialog menus each time you run RTView.



Field Name	Description
XML Source Prefix	Directory path or URL base affixed to XML sources you add with the Use **XML Source Prefix** check box selected. This allows you to use the same path for several XML sources. **Note:** The XML source prefix will not be applied to paths that begin with http://.
Default XML Source	Name of XML source used as the default for data attachments. Select from drop down menu to change default setting.
Add	Click to open the **Edit XML Source** dialog. To edit an existing XML source, double-click on a name from the XML Sources list.
Remove	Select an XML source from the list and click the **Remove** button to delete. It is not possible to remove an XML source that is currently updating an object.

Edit XML Source

When you add an XML source, the name you create (e.g., MyXMLSource) will be associated with the full path to the XML source file (e.g., **mydata/update.xml**) being used to update objects in your display.



Field Name	Description
XML Source Name	Create a name to use when referencing the XML source file that will be used to update objects in your display. The XML source name will appear in the "Attach to XML Data" dialog when you create data attachments in the Display Builder.

XML Source Path	Full or relative path to the XML source file (e.g., mydata/update.xml) that is being used to update objects in your display. If your XML data source is a socket, specify with the following format: **remote://host:port/filename**
Use XML Source Prefix	If selected, the XML source prefix you entered will be affixed to the XML source path. This allows you to enter a path once and use it for several XML sources. For example, if the XML source prefix is c:/rtview/ and the XML source path is mydata/update.xml, then MyXMLSource will be defined as c:/rtview/mydata/update.xml. **Note:** This check box is only enabled if you filled in the **XML Source Prefix** field in the **Application Options** dialog.
Static	Select this option if your XML source file exclusively contains static information. If selected, RTView will read this file only once.
Contains Substitutions	Select this option if your XML Source Path contains substitutions. If selected, RTView will not read this file until the substitutions in the path have been defined. This applies to substitutions set in the **Application Options** dialog as well as those set in drill down displays.

RTView Deployment - XML

This section contains details about the deployment process that are specific to your data source. Please go to the "Deployment" section of this documentation for instructions on how to implement your RTView deployment option. Return to this page whenever you are instructed to refer to deployment information that is specific to your data source.

System Requirements and Setup

The XML data source has additional System Requirements and Setup. See "XML System Requirements and Setup" for more information.

Data Source Configuration File

RTView saves general application settings as well as data source configuration options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options", "Application Options - XML", and "RTV_JAVAOPTS" for more information.

Include the following initialization file when you deploy RTView with this data source:

File Name	Description
OPTIONS.ini	Contains general options as well as data source options for XML.

Note: Options specified using command line parameters override values set in these initialization files.

Setup Client

No additional client setup is required for this data source.

XML Demos

Except where noted, all demos can be run in three ways, as an application, or via rich or thin client in a browser.

Before You Begin

Start the Demo Server

Thin Client Demo only.

There is a thin client demo already installed on the "RTView Demo Server".

In an initialized command window (see "Initializing a Command Prompt or Terminal Window"):

Type **run_startup_demoserver**

Data Source Demo

The Data Source Demo is designed to illustrate each data source.

1. **Start the Simulators**

   Start the simulators for each data source you will be using. To run the XML "Data Simulator":

   In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory and:

   type **run_simdata**

2. **Run Demos - Application or Thin Client Browser**

   Application Demo

   1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
   2. To view the demo, type:

      run_viewer

   3. To edit the demo, type:

      run_builder

   Thin Client Browser Demo

   "Start the Demo Server" if it is not running.

   1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
   2. Start the Display Server by typing:

      run_displayserver

   3. Open a browser and navigate to **http://localhost:8068/dstutorial**.

Features Demo

Presents an overview of the many features of RTView.

Application Demo

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/features** directory.
2. Start the simulator by typing:

   Windows: **start run_simdata**

   UNIX: **run_simdata &**
3. To view the demo, type:

   run_viewer
4. To edit the demo, type:

   run_builder

Thin Client Browser Demo

"Start the Demo Server" if it is not running.

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/features** directory.
2. Start the simulator by typing:

   start run_simdata
3. Start the Display Server by typing:

   run_displayserver &
4. Open a browser and navigate to **http://localhost:8068/features**.

Geothermal Demo

A real-time operational application that allows you to drill down to various levels of detail.

Application Demo

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/geothermal** directory.
2. Start the simulator by typing:

   Windows: **start run_simdata**

   UNIX: **run_simdata &**
3. To view the demo, type:

   run_viewer

4. To edit the demo, type:

   run_builder

Thin Client Browser Demo

Start the Demo Server if it is not running.

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/geothermal** directory.

2. Start the simulator by typing:

   Windows: **start run_simdata**

   UNIX: **run_simdata &**

3. Start the Display Server by typing:

   run_displayserver

4. Open a browser and navigate to **http://localhost:8068/geothermal/panels.jsp**.

Creating XML Sources

Creating and Formatting XML Data

To attach an object property to XML data element, you will need to specify the XML source, data key and, if the element is tabular, column(s). The XML source is the name, including path, of the XML source file that will be used to update the object.

Note: XML source file names cannot contain spaces. RTView does not limit the number of XML source files you may create. Within each XML source file, you must format your data using the "Supported Tags and Attributes" in the table below.

In order to display your application data in RTView, have your application create an XML source file using the supported tags. Each time you would like to update an XML data element, your application must output an updated copy of the XML source file.

Note: Do not directly edit XML source files. This will cause errors if you are writing to the file while RTView is attempting to read it. When creating or updating XML source files, create a temporary file, write your XML data into it, and then rename the temporary file to the XML source file name. If your XML source file contains non-English characters, you must use UTF-8 encoding.

When an XML source file is added, either from **Application Options** or directly in the "Attach to XML Data" dialog, RTView will populate **Attach to Data** drop down menus based on that XML source file. If you specify (or modify) an XML source using the **Application Options** dialog, you may specify whether that XML source is static. Static XML sources are only read once each time you run RTView.

Data Elements: Scalar and Tabular

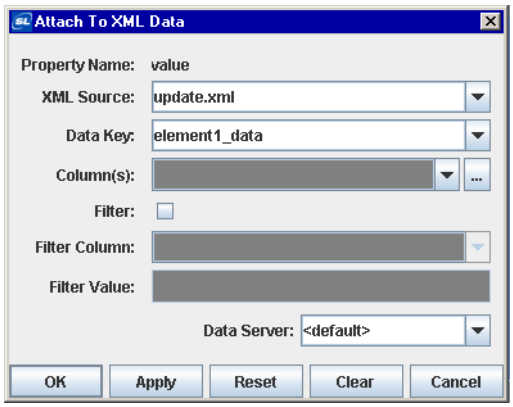
There are two types of data elements that you may include in an XML source: Scalar and Tabular.

A Scalar data element is a single value element. The data value can be a string, an integer or a double. Here is an example of how to format a Scalar data element in your XML source:

```
<data key="element1_data" value="10" />
```

data	Indicates that this is a scalar data element.
key	Assigns a key for this element. You will select this key in the **Attach to Data** dialog to attach an object property to this element. Key names cannot contain spaces.
value	Defines the value of this data element.

If this example was included in an XML source file named update.xml, you would select the following options from the Attach To Data drop down menus to connect an object property to this data element:



Once you **Apply** this data attachment, the attached object will update with the value "**10**".

A Tabular data element contains multiple columns and rows of data. The value for each field can be a string, an integer, a double, or a boolean. Tabular data elements are useful for updating Table and Graph objects in RTView. Each time you update tabular data in your XML source, you must include all of the column and row information. Here is an example of how to format a Tabular data element in your XML source:

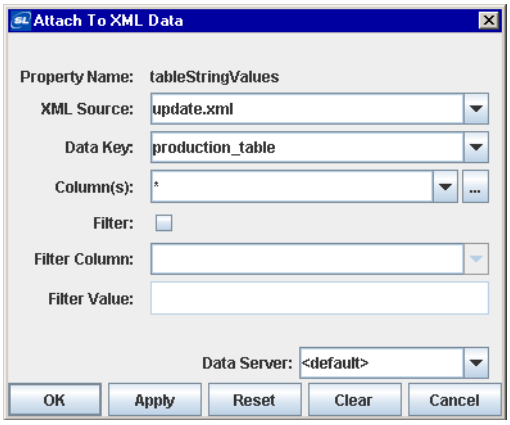
```
<table key="production_table">
        <tc name="Plant"
                type="string"
                index="true" />
        <tc name="Units in Production"
                type="int"
                index="false" />
<tr name="PID 0">
        <td>San Francisco</td>
        <td>99</td>
</tr>
<tr name="PID 1">
        <td=San Jose</td>
        <td=75</td>
```

```
</tr>
</table>
```

table	Indicates that this is a tabular data element.
key	Assigns key for this element. You will select this key in **Attach to Data** dialog to attach a property to this element. Key names cannot contain spaces.
tc	Indicates that you are defining a table column.
name	Name of the column. Each table column must have unique name.
type	Indicates what type of data will be included in the column. This field is optional. Valid values are: string, double, int, boolean or date. If the type is not specified, RTView will treat the values in the column as strings.
index	This field reserved for future use.
tr	Indicates that you are defining a table row.
name	Name of the row. Each table row must have unique name.
td	Defines the value of this field in the table.

All of the table columns must be specified before you specify any row data. The order of the table data elements in each row should correspond to the order of your columns. For example, if the first column you specified was a boolean, then the first field you add to each row must contain a boolean value.

If this example was included in an XML source file named **update.xml**, you would select the following options from the **Attach To Data** drop down menus to connect an object property to the Plant column in this data element:



Once you **Apply** this data attachment, the attached object will update with the value **San Francisco**.

Supported Tags and Attributes

The following tags and attributes in XML data files are supported:

Tag	Attribute		Description
dataset	xmlns		Top level tag that includes the namespace attribute xmlns, which must be defined as www.sl.com, and the attribute version, which must be defined as 1.0. (xmlns="www.sl.com" version="1.0")
	version		
data	key		Key unique to this data element. Key names cannot contain spaces.
	value		Value of this data element
table	key		Key unique to this tabular element. Key names cannot contain spaces.
	tc	name	This is a required field and must contain a unique column name. All columns must be specified before any rows are specified. Columns must be specified in the order they will be displayed in a table.
		type	Type of data contained in this column. This will be used for table sorting. If this field is not included or is invalid, the column will be assigned the type of string. Valid values: string, integer, double, boolean, date.
		index	This field reserved for future use.
	tr	name	This is a required field and must contain a unique row name.
		td	Value for a cell in this row. Values must be specified in the order they will be displayed in a table (i.e.: The first value will be displayed in the first column of the row, etc.) and must be of the type specified for the corresponding column.

An example XML data file:

```
<?xml version="1.0"?>
<dataset xmlns="www.sl.com" version="1.0">
    <data key="element1_status" value="completed" />
    <data key="element2_status" value="broken" />
    <data key="element1_load" value="0.0" />
    <data key="element2_load" value="1102.0" />
    <table key="trade_table">
            <tc name="Customer"
                    type="string"
                    index="true" />
            <tc name="Symbol"
                    type="string"
                    index="false" />
            <tc name="Shares"
                    type="int"
                    index="false" />
            <tc name="Purchase Price"
                    type="double"
                    index="false" />
            <tc name="Current"
                    type="double"
                    index="false" />
            <tc name="High"
                    type="double"
                    index="false" />
```

```
            <tc name="Low"
                     type="double"
                     index="false" />
            <tr name="TID 0">
                    <td>Alice Chen</td>
                    <td>IBM</td>
                    <td>18</td>
                    <td>44.7</td>
                    <td>27.5</td>
                    <td>177.5</td>
                    <td>17.5</td>
            </tr>
            <tr name="TID 1">
                    <td>Betty Jones</td>
                    <td>CVTX</td>
                    <td>19</td>
                    <td>32.0</td>
                    <td>10.8</td>
                    <td>160.8</td>
                    <td>0.8</td>
            </tr>
           <tr name="TID 2">
                    <td>Chris Smith</td>
                    <td>AWE</td>
                    <td>61</td>
                    <td>98.6</td>
                    <td>4.1</td>
                    <td>154.1</td>
                    <td>0.1</td>
            </tr>
    </table>
</dataset>
```

Data Simulator

A sample data simulator is provided to allow you to work with RTView without setting up your own XML data. The simulator creates and updates a group of data elements according to information specified in a data file. These data elements are given initial values and output to the **init.xml** initialization file. Once every update period the values for these data elements are updated and output to the **update.xml** update file.

From an initialized terminal window (see "Initializing a Command Prompt or Terminal Window") type:

run_simdata

By default, the update period is set to 2000 milliseconds (2 seconds) to match to default update period for RTView. To change the update period, you must run the XML simulator from a command prompt or terminal window and specify the desired rate on the command line using -u. For example, the following will set the update period to 1 second:

run_simdata -u1000

The data simulator outputs the following values:

Data Key	Data Type	Data Description
element1_load	double	Value starts at 20 and ranges between 0 and 100, updating by 5.
element1_status	string	Value iterates through the following values: working, broken, waiting, completed
element1_data	int	Value starts at 15 and ranges between 0 and 100, updating by 10.
element2_load	double	Value starts at 10 and ranges between 0 and 100, updating by 7.
element2_status	string	Value iterates through the following values: waiting, completed, working, broken
element2_data	int	Value starts at 5 and ranges between 0 and 100, updating by 12.
production_table	tabular	Table containing 15 rows of 5 columns: Column name="Plant" type="string" (does not change) Column name="Units in Production" type="int" (changes each update period) Column name="Units Completed" type="int" (changes each update period) Column name="Status" type="string" (changes each update period) Column name="On Schedule" type="boolean" (changes each update period)
system_table	tabular	Table containing 15 rows of 5 columns: Column name="System" type="string" (does not change) Column name="Status" type="string" (changes each update period) Column name="%Free Space" type="double" (changes each update period) Column name="CPU Usage" type="int" (changes each update period) Column name="On Site" type="boolean" (does not change)
trade_table	tabular	Table containing 100 rows of 7 columns: Column name="Customer" type="string" (does not change) Column name="Symbol" type="string" (does not change) Column name="Shares" type="int" (does not change) Column name="Purchase Price" type="double" (does not change) Column name="Current" type="double" (changes each update period) Column name="High" type="double" (changes only if current values gets higher than this value) Column name="Low" type="double" (changes only if current values gets lower than this value)

Additionally, the simulator will output the following data keys when using the default data file:

Data Key	Data Type	Data Description
step50_150by10	int	Value ranges between 50 and 150, updating by 10.
step-100_100by10	int	Value ranges between -100 and 100, updating by 10.
step50_75	int	Value ranges between 50 and 75, updating by 1.
step0_100	int	Value ranges between 0 and 100, updating by 1.
step0_50by2.5	double	Value ranges between 0 and 50, updating by 2.5.
step0_100by5.0	double	Value ranges between 0 and 100, updating by 5.
step-10_150.0	double	Value ranges between -10 and 150, updating by 1.
random0_100	int	Value is a randomly generated number between 0 and 100.
random0_100.0	double	Value is a randomly generated number between 0 and 100.
step_100	int	Value ranges between 0 and 100, updating by 1.
step_circ_50_200.0	double	Value ranges between 50 and 150, updating by 1.

You may create your own data file for use with the simulator; however, tabular data cannot be generated from a data file.

Use the following syntax to create custom data files:

Command	Description
key step min max start increment	The key value initially equals start and then increases by increment until max is reached. Then it decreases by increment until min is reached and starts over.
key step_circ min max start increment	The key value initially equals start and then increases by increment until max is reached. Then it starts over at min.
key random min max	The value of the key is randomly assigned to a number between min and max.

From the command line, specify the name of the custom data file for the simulator to use:

run_simdata mydatfile

An example data file:

```
step50_75                step 50 75 50 1
step0_100                step 0 100 0 1
step0_50by2.5          step 0 50. 0 2.5
step0_100by5.0        step 0 100. 0 5
step-10_150.0           step -10 150 -10 1
random0_100            random 0 100
random0_100.0         random 0 100.
step_100                  step 0 100
step_circ_50_200.0   step_circ 50 200.
```

XML Data Source Command Line Options

In addition to general options, the following command line arguments are enabled with the XML data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: "**-sub:$data:my Data"**).

Name	Description
-xmlsource:(filename.xml)	Name (including path) of data update file. If file contains static data, append **:0** at the end of file name. This will signal RTView to read the file only once. Example: **-xmlsource:updatefile.xml**
-xmlredirect	Redirect XML sources that don't start with **http:** or **https:** through the Data Server. Append **:all** to redirect all XML sources through the Data Server. Otherwise, XML data sources will be read directly by RTView. **Note:** This option only applies when reading Data Server data via socket.
	Example: **-xmlredirect** **-xmlredirect:all**

CHAPTER 11 Application Options

This section contains the following:

Overview

The **Application Options** dialog allows you to set general and data source specific options used in RTView. Available options and dialog tabs depend on which data sources are licensed in your RTView application. For information on Application Options for your data source, refer to the "RTView Data Sources" section of this documentation.

Options specified in the **Application Options** dialog are saved into initialization (**.ini**) files when you click the **Save** button. On startup, these initialization files are read by the Display Builder, Display Viewer, Display Server, Data Server, and Historian to set initial values for all application options. It is possible to specify a directory for your initialization files. If no directory is specified and initialization files are not found in the directory where you started the current application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

Note: Options specified using command line arguments will override values saved in initialization files. See "Command Line Options: Display Builder and Display Viewer" for more information.

The following initialization files are saved from the **Application Options** dialog and read by RTView:

- General options are read from **OPTIONS.ini**.
- Additional initialization files are read for alerts and for data sources that are licensed in your RTView application. For information on Application Options for your data source, refer to the "RTView Data Sources" section of this documentation.

The **Application Options** dialog can opened within the Display Builder or run as a stand-alone application on Windows or UNIX.

Setting Options in the Display Builder

In the Display Builder, select **Tools>Options** to open the **Application Options** dialog.

Setting Options via Configuration Utility

To run the dialog stand-alone, you will need to use the Configuration Utility.

In an initialized command window (see "Initializing a Command Prompt or Terminal Window") type: **run_configutil**

Note: When running stand-alone, Java options specified in "RTV_JAVAOPTS" will be used by the **run_configutil** scripts.

By default, the Application Options Utility requires a "Login" to support "Role-based Security". The default user name and password are:

User Name: admin

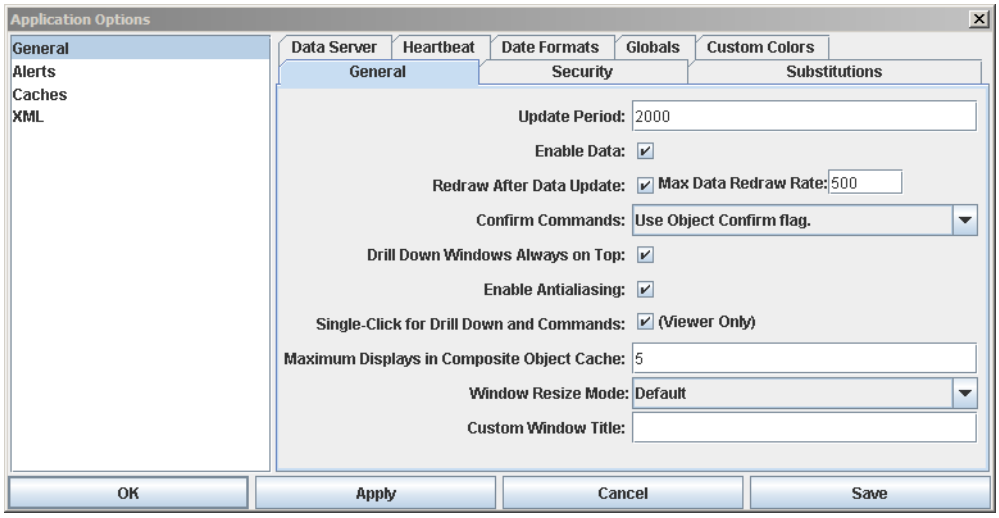
Password: admin

Note: If your system administrator has configured a user name and password for you, use these instead. In this case, you may also need to select a role. See "Role-based Security" for more information.

Application Options Dialog

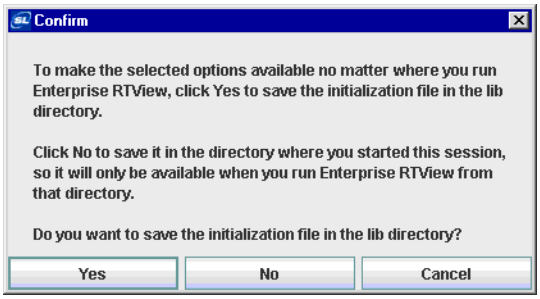
Select a category from the left frame to display corresponding dialog tabs in the main window. Click the **Save** button to record the new application options in an initialization file.

Note: Do not close the dialog without first saving your changes.



A confirmation dialog will appear to verify in which directory you would like to save the initialization file. If you specified a directory for your initialization files, all initialization (**.ini**) files will be saved to, and read from, that directory. If you select the **lib** directory, the initialization file will be available from any directory where you run RTView. If you do not select the **lib** directory, the initialization file will be saved in the directory where you started the current session and will only be available when you run RTView from that particular directory. See "RTV_JAVAOPTS" for more information.

Note: If no initialization file is found, RTView will use the defaults shown above.

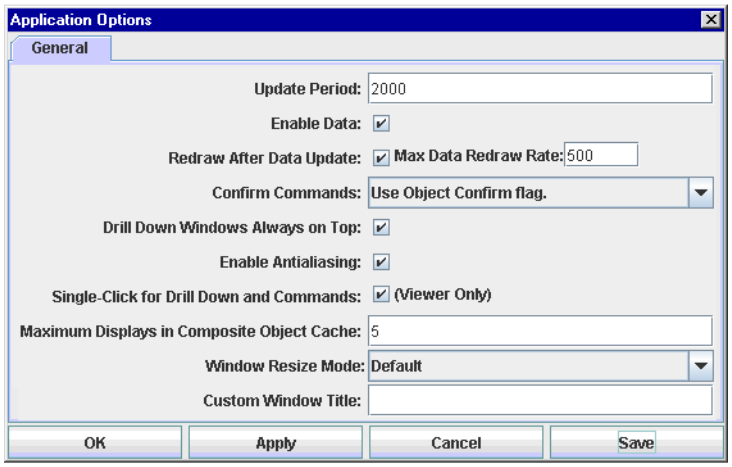


The following describes **Application Options** dialog commands:

Command	Description
OK	Applies values and closes the dialog. **Note:** Do not close the dialog until you have saved your changes.
Apply	Applies values without closing the dialog.
Cancel	Closes the dialog with last values applied.
Save	Applies values, saves them to an initialization file, and closes the dialog.

The **Application Options > General** option contains the following:

General Tab



Field Name	Description
Update Period	Enter an integer value indicating the number of milliseconds at which RTView will update data. On each update, all synchronous data sources are updated after which all displays are redrawn. Default is **2000** milliseconds.
Enable Data	Check to enable data updates. When not enabled, incoming data is ignored. The default is enabled.
Redraw After Data Update	Check to specify data-driven redraws. Data from an asynchronous data source can arrive at any time between update periods. This means there could be a delay between the time an asynchronous data source receives a data update and when the display showing this data is updated. If selected, displays containing data from asynchronous data sources that have changed since the last update will be redrawn at the rate specified in the **Max Data Redraw Rate**. Displays where no data has changed will only be redrawn on the update. If not selected, displays are only redrawn based on the update period.

Max Data Redraw Rate
Enter the maximum data redraw rate when data is updated. The default is **500** milliseconds.

Confirm Commands
Set the confirm policy for all commands. Overrides confirm policies set on individual objects. See the "Define/Execute Command" section for more information on commands,

Policy	Description
Do not confirm	Indicates that no commands require confirmation (regardless of each object's confirmation status).
Confirm all	Indicates that all commands require confirmation (regardless of each object's confirmation status).
Use Object Confirm flag	Indicates that the confirmation status of each object will determine whether confirmation is required. This is the default policy.

Drill Down Windows Always on Top
Drill down displays will permanently remain in front of the main display until drill down windows are closed. When this setting is changed, the new settings won't apply to windows opened before the change was made.

Enable Antialiasing
Smooth graphics in the display. Default is enabled

Single-Click for Drill Down and Commands
Open drill down windows or execute commands in the Display Viewer with a single click.

Note: This option does not apply while working in the Display Builder.

Maximum Displays in Composite Object Cache
Sets the maximum number of display (.rtv) files with composite objects to cache. Default is **5**. If value is set to 0, no displays are cached. This is a performance optimization option for the composite object. Of course the higher you set this value, the more memory the cache will require. The impact of increasing the value of this option depends on your application. For example if you have an object grid with 100 objects showing the same display (.rtv) file, then that file can be cached and cloned for each composite object in the grid. However in a situation where you have 100 composite objects showing different display (.rtv) files, caching the files would not significantly optimize performance.

Window Resize Mode
Globally controls object layout when a display window is resized. It is also possible to set a specific Resize Mode for each particular display (**.rtv**) file using the "Background Properties" dialog.

In the Display Builder, the selected Resize Mode is only applied to drill down windows. The main window of the Display Builder is always in Crop mode.

All three resize modes support zooming the display (right-click -> zoom). In both Scale and Layout modes if the window is resized while the display is zoomed, then the resize will further zoom the display.

Note: If you already have windows open and change the resize mode, the new setting will only apply to new windows that are opened (i.e. windows already open will not change.)

Select from the following options:

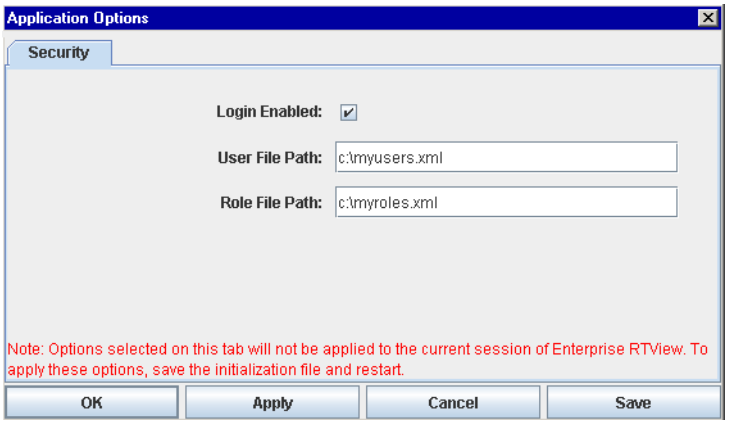
Default	Use application level default mode. Defaults are: **Scale** for the Display Builder and Display Viewer Application and **Crop** for the Thin Client.

	Crop	When the window is resized, the display stays the same size. If the window is bigger than the display, empty space will show around the display. If the window is smaller than the display, scrollbars will be added. The window is not forced to maintain its aspect ratio. This is the default for the Thin Client.
	Scale	When the window is resized, the display and all of the objects in it are scaled to fit the available space. The window is forced to maintain its aspect ratio. This is the default for the Display Builder and Display Viewer Application.
	Layout	When the window is resized, the display is resized to fit the available space. The objects in the display are positioned according to their anchor and dock properties. The window is not forced to maintain its aspect ratio. Objects that are not docked or anchored will move relative to their offset from the top left corner of the display. For example, if the object is centered on the display, the object will move 50% of the resize amount. If the object is centered at 3/4 of the display, it will move 75% of the resize amount. **Note:** To prevent objects from overlapping, set **resizeWidthMin** and **resizeHeightMin**. If a panel containing a display is resized, the display will not reduce smaller than the specified minimum size. See "Background Properties" for more information.
Custom Window Title	Specify a custom window title. To specify an empty window title, enter a single space. By default, window titles contain the name of the application followed by the name of the display (.rtv) file (e.g. RTView mydisplay.rtv). A Custom Window Title: • Takes precedence over the title specified in your panel configuration file for "Multiple Display Panels". • Is superseded by the Window Title option in the "Drill Down Properties" dialog. **Note:** If you already have windows open and enter a Custom Window Title, the setting will only apply to new windows that are opened (i.e. titles of windows already open will not change).	

Security Tab

The **Security** tab allows you to configure security options for RTView. This tab is only available if you are logged in with the admin role. If login is disabled, you can force the login to come up with the **-login** command line argument.

Options specified on the **Security** tab cannot be applied to the current session of RTView. Click the **Save** button to record all **Application Options** in an initialization file that will be used to apply settings when RTView is restarted.

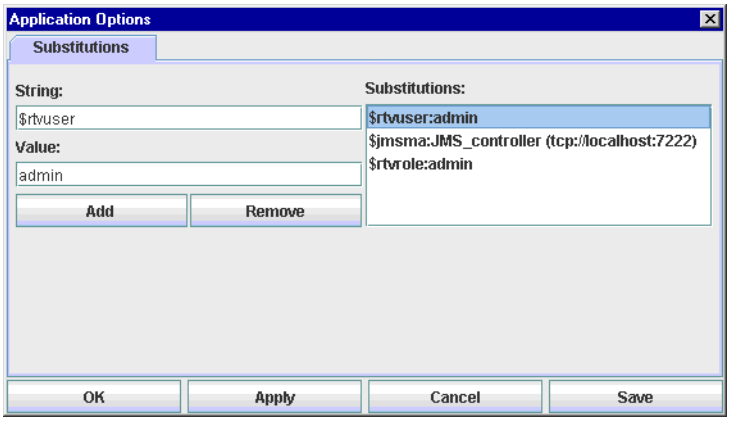


Field Name	Description
Login Enabled	Select to require a login for your application.
User File Path	Enter the name (including path) of the user definition file. This will not be used if you have implemented the Custom User Manager. See "User Definitions" for more information.
Role File Path	Enter the name (including path) of the role definition file. This will not be used if you have implemented the Custom Role Manager. See "Role Definitions" for more information.

Substitutions Tab

Add, edit or remove substitutions.

Note: The substitution string **$value** is reserved for internal use.



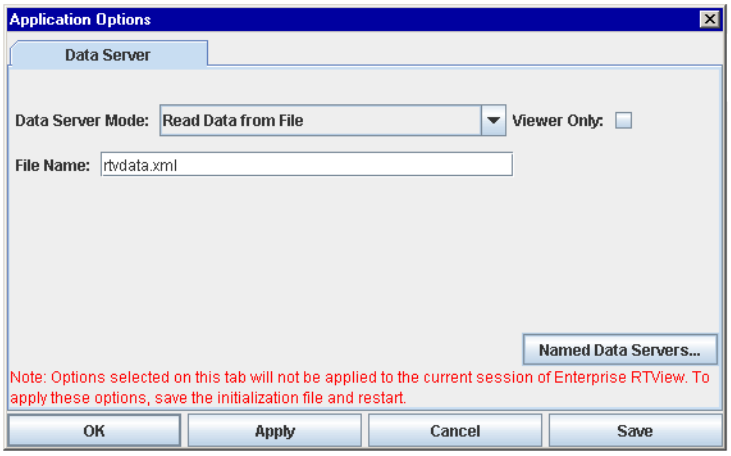
Field Name	Description
Add	Enter a String and a Value. Click **Add** to insert the substitution into the listing. **Note:** Substitution strings cannot contain the following:: | . tab space , ; = < > ' " & / \ { } [ ] ( )
Edit	Select a substitution from the list and edit the String or Value. Click **Add** to replace the substitution currently listed.
Remove	Select a substitution from the list and click **Remove**.

Data Server Tab

This tab allows you to redirect data requests to the "Data Server". Once you select to read Data Server data, direct access to data sources will be disabled. RTView clients (Display Builder and Display Viewer Application) can read Data Server data from a file, directly via socket or through the Data Servlet using HTTP/HTTPS requests.

Note: Settings made in this **Application Options** dialog must correspond with information entered in the **Data Server** "Configuration Tab".

Options selected on the **Data Server** tab cannot be applied to the current session of RTView. Click the **Save** button to record all Application Options in an initialization (**DATASERVER.ini**) file that will be used to apply settings when RTView is restarted. It is possible to specify a directory for your initialization files. If no directory is specified and **DATASERVER.ini** is not found in the directory where you started the Data Server, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.



Field Name	Description
Data Server Modes	By default RTView reads directly from data sources and not from the Data Server. To configure your default Data Server, select one of the following three options on this tab.

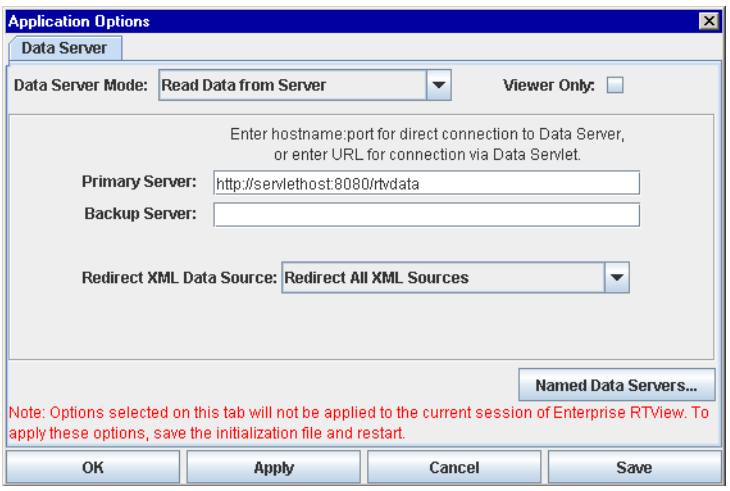
Do Not Read Data from Data Server -- Reads directly from data sources without redirecting data requests to the Data Server.

Read Data from File -- Redirect data requests to the Data Server and read XML file output by the Data Server. Once you select Read Data from File, direct access to data sources, except the XML data source, will be disabled when you save your options and restart RTView.

Note: The XML data source is never redirected through the Data Server when you read data from a file.

Read Data from Server -- When you choose to Read Data from Server, a primary and a backup server can be specified for the default Data Server and also for each **Named Data Server**. At startup, RTView clients will connect to the primary server if available, otherwise to the backup server. If neither server is available, clients will periodically retry connecting to both. If a connection is made to either server and later that connection is lost, then an attempt will be made to connect to the other server. If a connection is made to the backup server and later the primary server becomes available, clients will not switch to the primary server unless the connection to the backup server is lost.

All servers specified should use the same initialization files (*OPTIONS.ini) and should have access to the same external data sources, so that displays will appear the same regardless of which server is connected.



Primary Server - Enter host:port for direct connection to Data Server or enter URL for connection via Data Servlet.

Backup Server - Enter host:port for direct connection to Data Server or enter URL for connection via Data Servlet. If Primary Server is unavailable the client will transfer over the to the specified Backup Server.

Redirect XML Data Source -- This option only applies once you select Read Data from Socket. Select from the following options:

Do Not Redirect XML Data Source	XML data is not redirected through the Data Server. RTView will connect to the XML data source directly.
Redirect XML Sources Without HTTP Prefix	Redirect XML sources that don't start with http: or https:.
Redirect All XML Sources	Redirect all XML sources through the Data Server.

Note: Use the **Get Data Server Connection Status** function to access information about the status of connections from the Display Builder and Display Viewer Application to the default Data Server and any Named Data Servers. See "Tabular Functions" for more information.

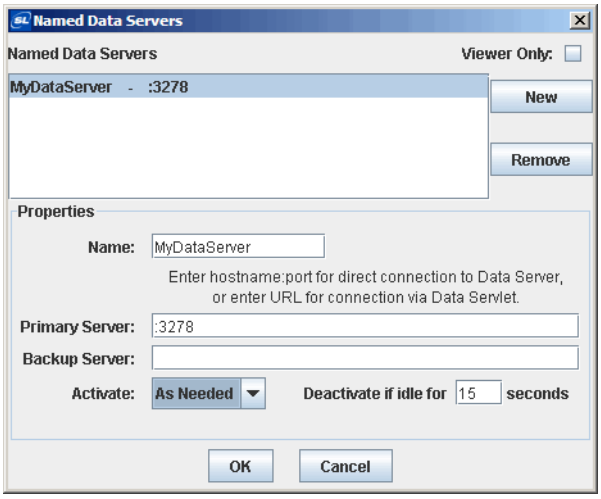
Viewer Only Select this check box if the Display Viewer should read data from the Data Server, but the Display Builder should read directly from data sources.

File Name Enter name (including path or URL base) of the Data Server output file. Default file name is rtvdata.xml.

Note: This information must correspond with the Data Server "Configuration Tab", including name and location of Data Server output file.

Named Data Servers Select **Named Data Servers** to open the Named Data Servers dialog, which allows you to configure connections to additional data servers. Data Servers are configured by the **DATASERVER.ini** file, which is generated from the Data Server, in addition to all ***OPTIONS.ini** files, which are generated when you click **Save** in the **Application Options** dialog. When additional Data Servers are configured you should specify a directory in which you can store their respective initialization files so they will not be overwritten. See "RTV_JAVAOPTS" for more information.

Note: All servers specified should use the same initialization files (***OPTIONS.ini**) and should have access to the same external data sources, so that displays will appear the same regardless of which server is connected.



Named Data Servers **New** - Select **New** to enter the Name of a Data Server to add to the list.

Remove - Select a Data Server from the list and click **Remove** to delete.

Viewer Only Select this check box if the Display Viewer should read data from the specified Data Server, but the Display Builder should read directly from data sources.

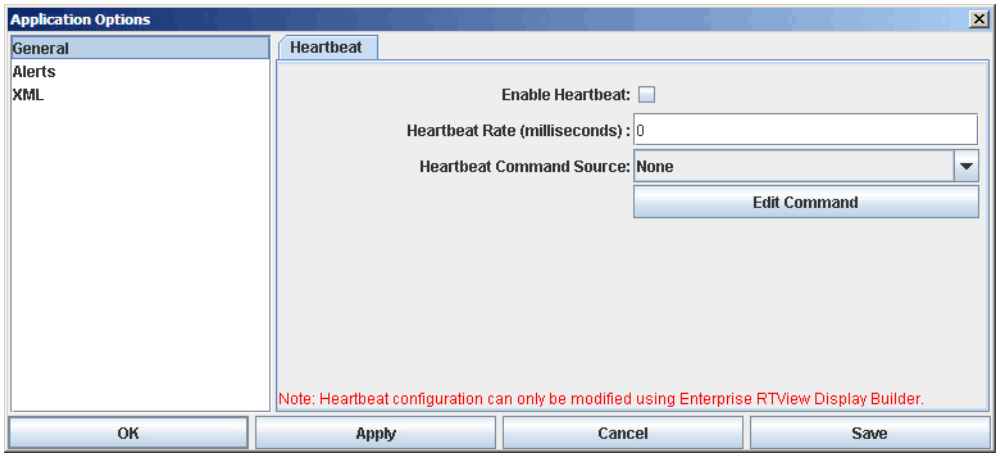
Name Click the **New** button and enter the name of the Data Server. When you attach an object to data or define a command, this name will appear in **Data Server** drop down menu.

Primary Server	Enter host:port for direct connection to Data Server or enter URL for connection via Data Servlet.
Backup Server	Enter host:port for direct connection to Data Server or enter URL for connection via Data Servlet. If Primary Server is unavailable the client will transfer over the to the specified Backup Server.
Activate	**At Startup** - All connections to Data Servers are opened when the selected RTView client application starts. Those connections are closed when that client application exits. **As Needed** - A Data Server connection is not opened until the RTView client application loads a display that contains data attachments directed to a particular Data Server. The connection is closed if all such data attachments are removed and/or the specified idle period has elapsed.

Heartbeat Tab

Configure system or data source commands to execute periodically in the Display Builder, Display Viewer, Historian, Display Server, Data Server and Transaction Monitor. See "Define/Execute Command" for more information on where your command will execute in the Display Server or Data Server.

In addition to commands, it is possible to utilize application level substitutions (in conjunction with built-in substitutions **$appName** and **$appVersion--**see below) to allow the Display Builder, Display Viewer, Historian, Display Server, Data Server, and Transaction Monitor to all use the same heartbeat configuration.



Field Name	Description
Enable Heartbeat	Check to enable the heartbeat command. The default is disabled.
Heartbeat Rate (milliseconds)	Enter an integer value indicating the number of milliseconds at which the heartbeat command will be executed.

Heartbeat Command Source	Select the source of the heartbeat command. This list will include None, System, Alert and all licensed data sources that support commands. If you select **None**, the configured command will be cleared.
Edit Command	Click to open the **Define Command** dialog for the selected source.

Note: Data source commands are not supported by the Transaction Monitor. Additionally, the following System commands are not supported by the Historian, Display Server, Data Server and Transaction Monitor:

- Drill Down or Set Substitution
- Open Browser
- Close Window
- Execute Custom Command (when UI function is invoked)

Substitutions

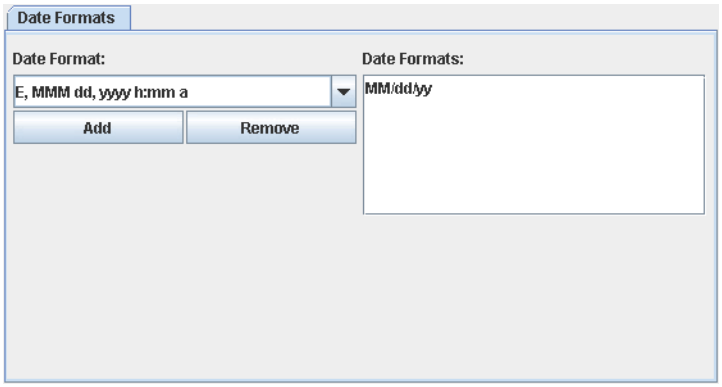
Application level substitutions, along with the following built-in substitutions, are supported in any field of the **Define Command** dialog. See "Substitutions Tab" and "Define/Execute Command" for more information.

$appName	Displays name of the application executing the command.
$appVersion	Displays version of the application (e.g. V: 4.6a0 Date: 01 May 2007) executing the command.

Date Formats Tab

This tab allows you to enter custom date/time formats for use in parsing dates in RTView. The formats entered in this tab will be used when parsing any date entered into RTView, whether the date was typed into a dialog or came in as a string from a data source.

Note: The formats entered in this tab must follow the syntax of the date and time pattern strings documented in the **java.text.SimpleDateFormat** class.



Field Name	Description
Add	Click to add a Date Format from the drop down list and then click **Apply** to execute.
Date Format	Select a Date Format. **Note:** Date Formats listed follow the syntax of the date and time pattern strings documented in the **java.text.SimpleDateFormat** class.
Date Formats	Formats selected from the **Date Format** drop down list.
Remove	Click to remove selected item from the **Date Formats** list and then click **Apply** to execute.

Globals Tab

Specify a Global Definition file to be read by the Display Builder, Display Viewer, Display Server, Data Server and Historian. In the Display Builder, select **Tools>Options>General>Globals** to add, remove or refresh Global Definition files.

Note: Global files are updated even if currently open displays do not use those results.

Field Name	Description	
Add	Click to add a (**.rtv**) file to the Global Definition file list, and optionally a corresponding substitution, then click Apply to execute.	
	Global Definition File	Select from this list to edit a file name or substitution. See "Global Functions and Variables" for information on creating Global Definition files for function data.

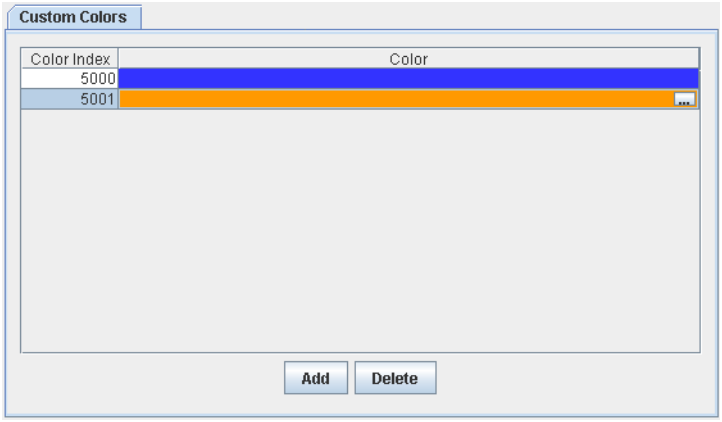
Substitutions	Specify substitutions for this Global Definition File. Substitutions are optional and must use the following syntax: **$subname:subvalue $subname2:subvalue2** If a substitution value contains a single quote, it must be escaped using a / : **$filter:Plant=/'Dallas/'** If a substitution value contains a space, it must be enclosed in single quotes. Do not escape these single quotes: **$subname:subvalue $subname2:'sub value 2'** A substitution string cannot contain the following: : | . tab space , ; = < > ' " & / \ { } [ ] ( )
Remove Selection	Click to remove the selected Global Definition file, then click **Apply** to execute. The Global Definition file is removed from the data source and results for this data will no longer be available.
Refresh Selection	Click to reload the selected Global Definition file. Allows you to edit a Global Definition file and apply changes without restarting the Display Builder. The **Refresh Selection** option is enabled when you select an Global Definition file that has already been added and applied.

Custom Colors Tab

Specify custom colors to use in your displays. Select **Tools>Options>General>Custom Colors** to add, remove or edit custom color definitions.

Custom colors are saved to an initialization file (**COLORS.ini**) when you click **Save** in the **Application Options** dialog. This initialization file must be deployed with your RTView application.

A **Custom Colors** tab has been added to the Color Chooser in the Display Builder, so you can select a custom color in the same way you would select a standard color. You must click **Apply** or **Save** on this options tab in order to make your custom colors available in the Color Chooser.



Field Name	Description
Add	Click to add a Custom Color to the list. Choose from the following options: **Swatches** - Standard Java color palette **HSB** - Color selection by hue, saturation and brightness **RGB** - Color selection by red, green and blue intensity
Color Index	RTView stores custom colors according to Color Index numbers, not RGB values. Therefore if an object property is defined by a custom color and you change the **Color Index** number, the color setting for that object property will revert to white. It is possible to set your own Color Index numbers, but the value must be greater than 5000
Color	Click on the ... button of a selected color to edit that color definition. Once you have clicked **Apply** or **Save**, mouse over any Color row to view RGB values for that color.
Delete	Click to delete the selected Color. **Note:** If an object property is defined by a custom color and you delete that color, the color setting for that object property will revert to white.

Limitations

Object limitations

Some objects (e.g.: the bar graph legend, pie wedges and legend, and some control object properties) cache their colors and therefore do not update when a custom color definition changes. To see the color change for these objects, you will either need to restart RTView or reload the display.

Style Sheet Tab

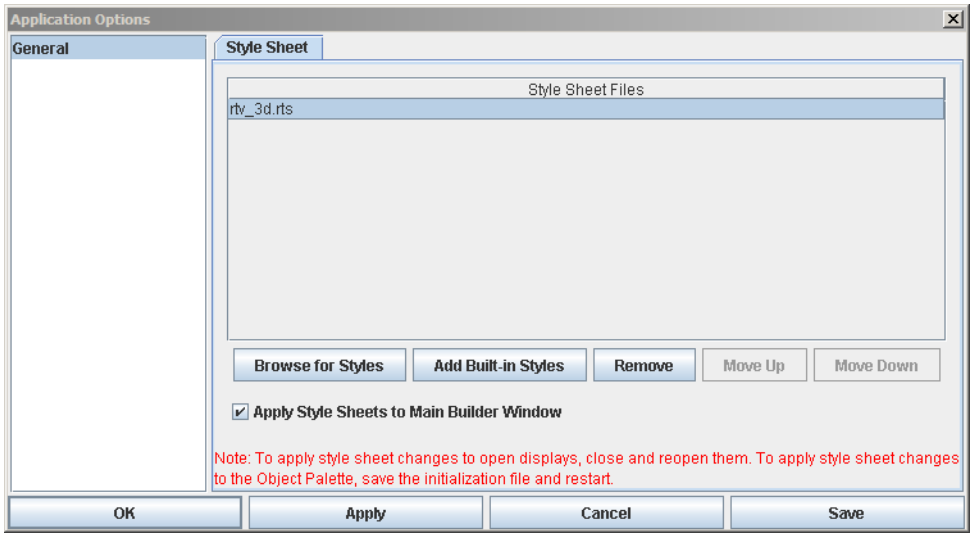
Select **Tools>Options>General>Style Sheet** to add, remove or reorder style sheets applied to your applications. It is also possible to apply specific style sheets to a single display. See "Display-Specific Style Sheets" for more information.

Note: In the Display Builder, style values are also applied to the Object Palette as well as objects added from the **Display Builder** toolbar.

Application level style sheets are read once, at startup, and applied when display (.rtv) files are opened. If you edit a style sheet, then you need to restart to see those changes. If you add, remove or re-order style sheets, changes will not be applied to open displays. To see these changes, close and reopen your displays. To see changes in the **Object Palette** you must always restart.

Note: In the following cases, style changes are applied immediately:

- Objects added from the Display Builder toolbar
- Objects added to Object Grids
- Objects added to Composites



Field Name	Description
Add	**Browse for Styles** - Locate a specific style sheet (.rts) file. **Built-in Style** - Choose from available "Built-In Style Sheets".
Remove	Select a style sheet (**.rts**) file from the **Style Sheet Files** list and click to remove.
Move Up **Move Down**	Re-order the **Style Sheet Files** list. **Note:** When multiple style sheet (**.rts**) files are applied, they are processed in the order specified. Therefore if the same property is specified in multiple style sheets, the value in the last style sheet applied will take precedence.
Apply Style Sheets to Main Builder Window	If selected, style sheets are applied to the main **Display Builder** window (a.k.a Working Area) and in the **Object Palette**. To prevent display (**.rtv**) files from being saved with unwanted style values applied you can choose to deselect this option.

CHAPTER 12 Deployment

This section contains the following:

Overview

One of the biggest challenges of providing data visualization to a variety of consumers is choosing the ideal deployment technology for your organization. There are many factors to consider, including optimizing the end user experience, hardware costs, maintenance costs, performance, scalability and security.

This section is designed to help you understand the various options, consider the pros and cons of each, and choose the optimal deployment.

This section contains the following:

Deployment Options

One major advantage of RTView is that it is a portable delivery platform. This means that all displays, including graphical elements, data attachments, drill downs, functions, substitutions and security settings, are portable to any deployment option - without reengineering. This facilitates implementation and rollout since development, testing, and production systems can use different deployment technologies without significant porting costs. If situations change in your organization and you would like to choose a different deployment option, you can readily do so.

SL keeps current on the latest information delivery technologies, such as browser and portal options, scalable data distribution, application servers and security. This ensures that the most suitable option for your enterprise remains deployable with minimal costs. Currently there are five options for RTView deployment.

If you have already chosen your deployment option, you can go directly to the deployment process from the following links.

Two Application Deployment Options:

"Application with Served Data - Manual Deployment Process" on page 811

"Application with Direct Data Connection - Manual Deployment Process" on page 818

Two Browser Deployment Options:

"Thin Client Browser with Served Data" on page 823

"Thin Client Browser with Direct Data Connection" on page 824

Thin Client Browser with Direct Data Connection is comprised of data sources, a client and possibly a server. Figure 1 is a high level illustration of the five deployment scenarios.

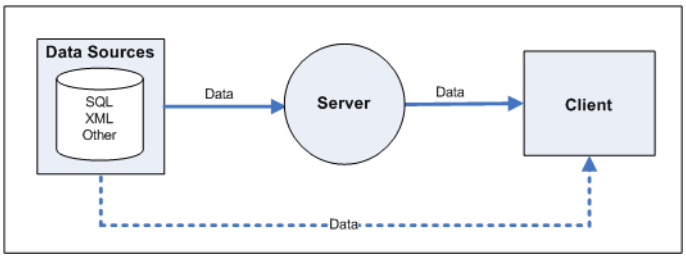


Figure 1: Deployment Overview

Application Deployment

In cases where the information delivered is of a critical, real-time, actionable nature, it may be desirable to deploy information via an application. For instance, infrastructure monitoring applications and critical operational dashboards can have a dedicated platform which only runs this application. End users can focus on monitoring and responding to the delivered information. In these cases it may not be desirable to deliver the information in a browser which might encourage an end user to browse away from critical information.

There are two application deployment options, Application with Served Data and Application with Direct Data Connection, as illustrated in Figure 2.

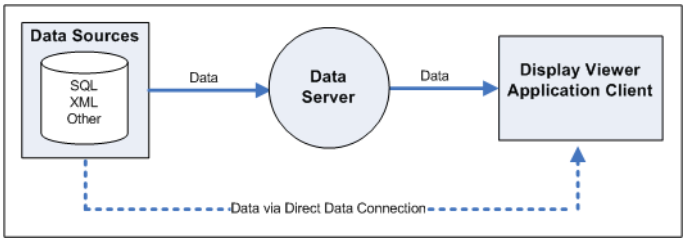


Figure 2: Application Deployment Overview

Browser Deployment

The most common deployment option chosen is a browser based deployment. Browsers are chosen for several reasons: they are a familiar tool for navigating to multiple sources of information, they are a standard application already resident on most enterprise platforms, and new information sources can be provided to end users without the additional IT expense of installing specialized software at each client.

Thin Client Browser deployments are popular with IT departments that prefer the ease of deployment and maintenance achieved by this technology. With this option, the client need only have a standard browser installed to access the RTView dashboards. This deployment requires the installation of the Display Server which uses AJAX technology to provide interactive dashboards to any standard browser. Recent advances in this technology have enabled the Display Server to perform comparably to Rich Client Browser deployments in most use cases.

As illustrated in Figure 4, there are two Thin Client Browser deployment options: Thin Client Browser with Served Data and Thin Client Browser with Direct Data Connection.

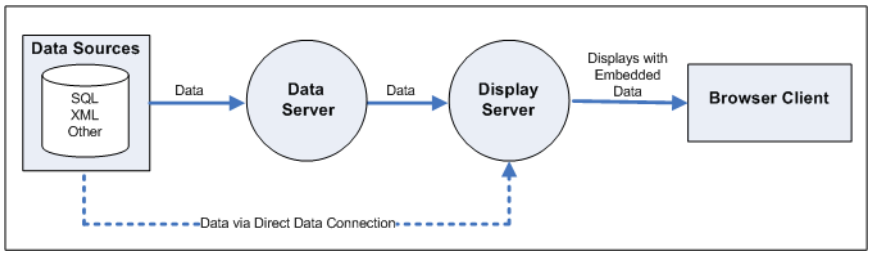


Figure 4: Thin Client Browser Deployment Overview

Choosing The Right Deployment

There are advantages and disadvantages to each type of RTView deployment. Depending on your environment, there are at most three decisions to make. To make it easier, we provide a step-by-step guide to help you consider the pros and cons of each scenario, and readily determine which deployment is the right one for your organization.

Figure 5 illustrates your choices, starting with the first decision; whether to have an application or browser deployment.

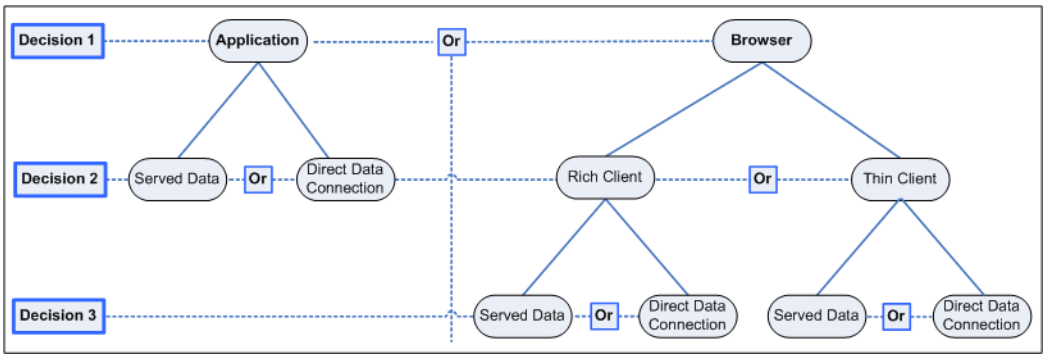
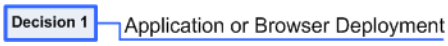


Figure 5: Deployment Decision Process

Let's get started!



Application Versus Browser Deployment

There are a variety of differences between application and browser deployments detailed in the table below. However, the major drivers in making a choice are usually based on two things: optimal user experience and cost factors.

In the majority of situations the browser is the most popular deployment option. The browser is seen as the easiest way to maintain information access to a wide variety of users, and it is a well known end user application. These factors can reduce costs in deployment, maintenance and training.

Yet in small, localized deployments, the application choice may still be the most cost effective. This is because in its simplest form it does not require an application server. The application option is also the obvious choice when a browser environment might lend to end users browsing away from critical information, such as in critical 24/7 monitoring solutions.

Pros and Cons

Issue	Application Deployment	Browser Deployment
Setup	Server: • No application server required Client: • No browser required on client but more labor intensive deployment for wide geographic areas	Server: • Requires application server Client: • Browser required on client but easy deployment for wide geographic areas
Usability	• Users must learn a new application but they cannot browse away from critical information being monitored	• Users do not need to learn a new application but they may browse away from critical information
Performance	• Load balancing and fault tolerance must be implemented	• Load balancing and fault tolerance are included with application server
Security	• Most applications run on LAN so no security issues	• Security issues must be considered
Scalability	• Fully scalable	• Fully scalable
Server Management	• File and version maintenance manually created	• Centralized file and version maintenance handled via application server
Cost	• Lower for small deployment and higher for large deployment	• Lower for large deployment and higher for small deployment

Choose "Application Deployment"

Choose "Browser Deployment"

Deployment Wizard

The Deployment Wizard will take your RTView application and package it up for deployment. It allows you to define a project, and one or more deployments for that project. The generated deployment files can be copied to the system where you want to deploy RTView. The Deployment Wizard supports Application, Thin Browser Client and Rich Browser Client deployments. See "Deployment Options" for help selecting the deployment type that will work best for you.

This section includes:

Setting up your Project for the Deployment Wizard

The Deployment Wizard will help you package up your RTView project for deployment. It has some assumptions on how you have set your project up. The Deployment Wizard assumes that you have already created all of your **.rtv** files and configuration files. In order to use the Deployment Wizard, you will need to make sure that all of your project files are under a single directory. Some of your project files may be in subdirectories of that single directory, but none can be outside of the project directory. Your project files include the following:

- Display (**.rtv**) files
- Style sheet (**.rts**) files (optional)
- Panel configuration file (optional). See "Multiple Display Panels" for more information.
- jar file containing custom classes (optional)
- Security configuration files (optional). See "Configuration" for more information.
- OPTIONS.ini. See "Application Options" for more information.
- COLORS.ini (only required if Custom Colors have been defined). See "Custom Colors Tab" for more information.
- DISPLAYSERVER.ini (only required for Display Server deployments). See "Display Server Configuration" for more information.
- DATASERVER.ini (only required if using the Data Server). See "Running the Data Server" for more information.
- Custom Servlet Files for the Display Servlet (only needed for Display Server deployments)
- Custom Servlet Files for the Data Servlet (only needed if using the Data Server and accessing it using HTTP/HTTPS)
- Refer to Deployment in the Data Sources section of this documentation for information on configuration (.ini) files specific to your data sources.

If you will be deploying the Display Server and have any custom servlet files, these will also need to be in your project directory. The most common custom servlet file is **rtvdisplay.properties** configuration file. Follow instructions in the "Display Servlet" section for setting up your custom servlet files. You do not need to make the Display Servlet war file as described there. The war file will be generated by the Deployment Wizard.

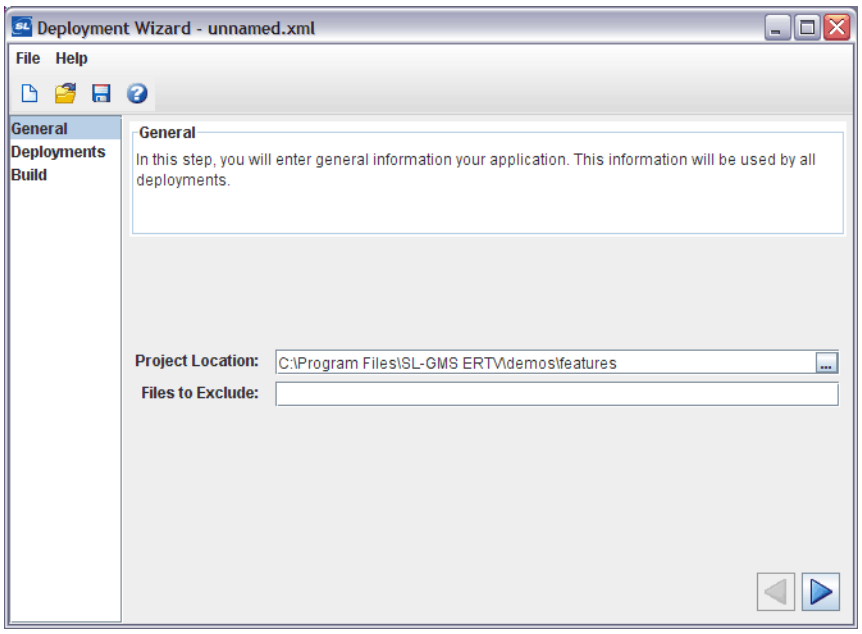
If your deployment will be accessing the Data Sever via HTTP/HTTPS and you have custom servlet files for the Data Servlet, these must also be in your project directory. Typically, the only custom servlet file for the Data Servlet is the **servlet.properties** configuration file. You do not need to make the Data Servlet war file. The war file will be generated by the Deployment Wizard. See "Data Servlet" for more information.

If your deployment will be using more than one Data Server, you will need to include multiple **DATASERVER.ini** files, since each Data Server will run on a different host and/or port. It is recommended that you place the **DATASERVER.ini** file and any other initialization files for each Data Server instance in a separate subdirectory of the project directory. When you run the Data Server, specify the directory for your initialization files. The Deployment Wizard does not have an option for generating multiple Data Servers in a single deployment, so you should select the **Create Separate Archive for Data Server** option and install this archive onto each host where you want to run the Data Server. If multiple Data Servlets are needed, create multiple deployments, one for each Data Servlet. See "RTV_JAVAOPTS" for more information.

Running the Deployment Wizard

In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window") type:

run_depwizard



See the Help in the Deployment Wizard for detailed information on the dialog fields. The **Project Location** field should point to the directory described in the Setting up your "Setting up your Project for the Deployment Wizard" section above. Once you have filled in the Project Location and created one or more deployments, you can build your deployment(s).

Command Line Options

The Deployment Wizard supports the following command line options:

Name	Description
-nogui	Run the Deployment Wizard without the gui. A project file must also be specified on the command line to use this options. It will open the project, build all of the deployments and then exit.
-project:	Open the specified project file. Example: **run_depwizard -project:myproject.xml**
-verbose	Enable verbose build status output.

Building Deployments

Click on the Build All Deployment button in the Build screen to build your deployment(s). The deployment wizard create a directory named deploy under the Project Location directory. The directory will contain a subdirectory for each deployment, named after the deployment name. For example, if your Project Location is **c:\myproject**, and your deployment is called **mydeployment**, the files for that deployment will be saved in **c:\myproject\deploy\mydeployment**. A **readme** is generated for each deployment that will describe what to do with each generated file. The **readme** describes where to run the different components of your deployment, but does not describe how to run them. Here are some convenient links for configuring and running the different components:

- "Display Server"
- "Display Viewer Application"
- "Data Server"

Upgrading Deployments

When a new version of RTView is released, you may want to upgrade your deployment to use the new release. To do this, run the Deployment Wizard from your new installation, and open your project. If the Include RTView option is selected for any of your deployments, the new version of RTView will be included when you build those deployments. Redeploy them as described in the readme for each deployment. If you do not have the Include RTView option selected for your deployments, you will need to re-install RTView on the systems where you deployed RTView manually.

Data Server

The Data Server is an option when it is not desirable, or possible, to directly connect the Display Viewer Application or Display Server to data sources. Select the "Data Server Tab" in the **Application Options** dialog to configure your Data Server(s).

The Data Server uses EII and XML technologies to gather, federate and distribute information from disparate data sources based on information currently in demand. It also caches the data so that multiple demands are delivered to any number of clients - without need of subsequent data queries. These important factors greatly enhance processing speed. Because the Data Server can exist behind firewalls, it also greatly simplifies and strengthens the secured delivery of information to clients beyond the firewall.

The Data Server can be used to serve data to the Display Viewer Application and Display Server. See the "Overview" section for more information.

The following scenarios, or a combination of them, would indicate that one or multiple instances of the Data Server would be useful.

This section includes:

Data Access

In some cases the data source resides in a sub-network where it is not possible or desirable to connect directly. For example, you may have a data source that communicates via a TCP/IP port that is not exposed across a security firewall. In this case, the Data Server would exist behind the firewall and connect to the chosen display technology via HTTP or an available exposed port.

Data Centralization

While a direct TCP/IP connection to the data sources may be possible, it may not be desirable for each data source and each client. For example, with a Display Viewer Application deployment you might not want a direct TCP connection from each client to every data source. In this case, the Data Server could act as a proxy by directly connecting to all data sources and providing data to each Java application client.

Data Reduction / Aggregation

In some cases, there may be a lot of raw data that does not need to be sent except upon demand. Using the Data Server in this situation can reduce network traffic by performing data calculations from within a sub-network and providing either only the aggregated data or raw data on demand to the chosen display technology. For example the Data Server may publish an average metric over all servers in a subnet, but provide raw data on drill down to a selected server.

Scalability

The Data Server has several data processing features including: an in memory data cache, a persistent alert rules engine, and the ability to perform data calculations. This processing load can be scaled out by dividing the labor across multiple Data Servers that will connect to the chosen display technology.

High Availability

To ensure client access to data in the event of a server failure, a pair of redundant Data Servers can be deployed on two separate hosts. In such a deployment each server has the same configuration and access to the same data sources. Typically, one server is considered the primary and the other is the backup. Each client is configured with two Data Server connections, one to the primary and another to the backup.

At startup, the client connects to the primary server if available, otherwise it connects to the backup server. If neither is available, the client periodically retries connecting to both. Client connections are specified on the "Data Server Tab" of the Option dialog in the Builder, as or via the **-dataserver** command line options when launching the Display Builder, Display Viewer Application, or Display Server. See "Command Line Options: Display Builder and Display Viewer" and "Command Line Options: Display Server" for more information.

A running Data Server is either "active" or "standby". An active server loads all global, cache, and alert definitions and activates them. In a standby server, none of these actions are performed, thereby avoiding the overhead of maintaining alert and cache data sources until the server is active.

Note: All servers specified should use the same initialization files (***OPTIONS.ini**) and should have access to the same external data sources, so that displays appear the same regardless of which server is connected.

A primary/backup redundant server pair can be configured in the following three modes:

1. **Active / Active -** In this mode, both servers are always active. This configuration is the simplest and provides the quickest response to clients in the event of a server failure. However, this mode increases data processing as both servers are always storing cache data, as well as processing and generating duplicate alerts.

2. **Active / Standby with manual reset** - In this mode, the backup server is started with the **-standby:warm** command line option. The backup does not become active until a client connects to it (typically after the primary fails). However, after the backup server becomes active it remains active and clients stay connected to it, even if the primary server becomes active. This mode reduces data processing in normal conditions but after a primary server fails and recovers both servers might become active, possibly with clients connected to both servers. To restore the system to its initial active / standby state, the backup server must be manually stopped and restarted (with the **-standby:warm** command line option). See "Command Line Options: Data Server"for more information.

3. **Active / Standby with auto-reset** - In this mode, both servers belong to a server group, where one server is designated as the preferred primary server and the other as the preferred backup server. Each server monitors the status of the other. The backup runs in standby mode until it detects the failure of the primary and then becomes active. If the

primary server then becomes active the backup server automatically returns to standby mode. Clients then automatically disconnect from the backup and (re)connect to the primary.

This mode ensures that only one server in the group has its alert data source activated at any time, thereby avoiding duplicate alert generation. Also by default, the cache data source is only active in the primary server, thereby avoiding duplicate cache storage. However in some cases it may be preferable to also activate the cache data source in the backup Data Server, Otherwise, caches may miss data during the failover of the primary server to the passive backup server, while the backup server is activating its cache data source. This can be accomplished by the **group_standby_mode** option. See "Command Line Options: Data Server"for more information.

Note: The cache and alert persistence features can also be used to ensure that the backup server gets the correct state of those data sources when it becomes active.

See the "Server Group Tab" section (below) for details about configuring this high availability mode using the Builder GUI. This mode can also be configured on the command line. For details, see the **-group_member** argument.

Security

You can use client access lists to specify which clients can connect to a Data Server. Any RTView application (the Builder, Viewer, Display Server, another Data Server and so forth) can be a Data Server client. By default, the client access lists are empty which means that any client can connect to the Data Server. When a client connection request is denied, that client makes no further attempts to connect to that Data Server unless the client application is restarted. This is true even if the client has a failover connection specified for that Data Server.

There are three access lists:

- **Blacklist**: Clients on this list are denied access to the Data Server.
- **Graylist**: Clients on this list are permitted access to the Data Server if they provide a trusted SSL certificate. **Note:** Graylisting should be used only when necessary since it involves certificate management, delays from SSL handshaking, and overhead from data encryption.
- **Whitelist**: Clients on this list are permitted access to the Data Server, no SSL certificate is required.

Access List Process Flow

When access lists are enabled, the Data Server checks the access lists for entries upon receiving a connection request: first the Blacklist, then (if not blacklisted) the Graylist, and lastly (if not graylisted) the **Whitelist**. To illustrate this process in greater detail, let us say that a Data Server receives a request from a client with IP address **X** and hostname **H**:

1. If the **Blacklist** is not empty (it has one or more clients specified) the Data Server checks the **Blacklist** and if **X** or **H** matches an entry in the **Blacklist**, the server rejects the client connection.

2. Otherwise, if the **Graylist** is not empty and **X** or **H** matches an entry in the **Graylist**, the server performs an SSL handshake with the client. If the client presents a certificate that

the server trusts and vice-versa, the handshake completes successfully and the connection is accepted. Otherwise, the server rejects the connection.

3. Otherwise, if the **Whitelist** is not empty and **X** or **H** matches an entry in the **Whitelist**, the server accepts the client connection.

4. Otherwise, neither **X** or **H** is matched on any list. If the **Whitelist** is empty, the connection is accepted. If the **Whitelist** is not empty, the connection is rejected.

The Blacklist has the highest precedence and the **Whitelist** has the lowest precedence. If a client address or hostname has matches on multiple access lists, the most restrictive list is used. For example, if a client address matches an entry on the **Blacklist** its connection request is denied even though the client address also matches an entry on the **Graylist** or **Whitelist**.

If the **Whitelist** has at least one entry all clients are rejected that have no match in the **Whitelist** or the **Graylist**. However, if the **Whitelist** is empty, then all clients that are not blacklisted are accepted (with an SSL handshake if its graylisted, unconditionally otherwise).

Using the hostname **H** in a list is effective only if the Data Server's host system is able to lookup that hostname via its address **X**, using a naming service. For example, if hostname **H** is specified in a list then "**ping H**" should complete successfully if run on the Data Server's host system.

If any **client_whitelist** values are specified and the Data Server belongs to a redundant server group for failover, specify a **client_whitelist** entry for the address or hostname of the other group member. Otherwise, the server cannot connect to its partner.

If the **client_whitelist** values are specified and the rtvdata servlet is used, you must also specify a **client_whitelist** for the address or hostname of the system on which the servlet is deployed. Otherwise, the servlet cannot connect to the Data Server.

Specifying Client Lists

To enable this feature you add client hostnames or IP addresses to the access lists, as appropriate, using the **client_blacklist**, **client_graylist** and **client_whitelist** options. Each option can be specified multiple times. The order in which you specify the client list does not affect the connection request process. The client access list options can be specified as Data Server command line arguments, as options in the **DATASERVER.ini** file, or as entries in a properties file. The value of each option is either a client hostname or IP address which may contain * characters as wildcards, or a range of IPv4 addresses in the format of **n.n.n.n-n.n.n.n**, where each n is a number between **0** and **255**. The ***** character cannot be used in an address range.

Note: For best performance use IPv4 addresses, rather than hostnames, in the access lists whenever possible. This mitigates the possible performance penalty involved in using hostnames, as the Data Server uses a name service to perform hostname lookups.

When this feature is enabled, the Data Server checks both incoming socket connections and HTTP connections (via the rtvdata servlet) against those lists using the process described previously. Case is ignored when a client hostname is compared to a list entry. For example, a list entry of host1 matches HOST1, Host1, and host1.

Examples:

```
run_dataserver -client_whitelist:192.168.1.* -client_whitelist:localhost
```

In this command line argument example, the Data Server accepts connections from all clients on localhost or with IPv4 addresses that begin with **192.168.1**. Connections from any other clients are denied (because the **Whitelist** is not empty).

run_dataserver -client_blacklist:DMZ

In this command line argument example, the Data Server denies connections from a client with hostname **DMZ** and accepts connections from all other clients.

client_whitelist localhost

client_blacklist 192.168.1.44

client_blacklist DEV*

client_whitelist 192.168.1.1-192.168.1.255

In this **DATASERVER.ini** file example, the Data Server accepts connections from the localhost and any clients with IPv4 addresses in the range of **192.168.1.1** through **192.168.1.255** except for address **192.168.1.44** or any client whose hostname begins with **DEV**, **dev**, **Dev**, etc. Connections from any other clients are denied (because the **whitelist** is not empty).

```
sl.rtview.dataserver.client_whitelist=localhost
sl.rtview.dataserver.client_blacklist=192.168.1.44
sl.rtview.dataserver.client_blacklist=DEV*
sl.rtview.dataserver.client_whitelist=192.168.1.1-192.168.1.255
```

This rtview properties file example specifies the same configuration as the previous **DATASERVER.ini** file example.

Graylist and SSL Certificates

If a client matches an entry on the graylist, an SSL handshake is performed between the client and the server in which the client must provide a certificate the server trusts, and the server must provide a certificate the client trusts. If the SSL handshake is successful the client connection is accepted. All traffic between the client and server on that connection is encrypted, typically using a 128-bit key. (This might impact CPU and network performance on connections that transmit high volumes of data).

Example:

The following example describes the self-signed certificate process using the keytool utility provided with the Java JDK. For illustrative purposes, some of the keytool commands appear on multiple lines, but must be entered as a single line. The Data Server runs on host1 and the client runs on host2. The filenames, passwords, aliases and so forth used here are only for illustration and are not required values. For example, there is nothing significant about the use of rtview in the example strings. In an actual configuration all of these strings could be replaced with strings of your choice (if they are replaced consistently).

Note: For details about SSL and certificates, see vender documentation. For details about **keytool**, execute "**keytool -help**" for usage information.

---

1. Create a **keystore** filed named **server_keystore.jks** to store the server's private key and self-signed certificate, with a 10-year expiration. For example:

   keytool -genkey -keystore server_keystore.jks -alias rtview_server -validity 3650 -keyalg RSA -storepass mypswd -keypass mypswd -dname cn=host1

2. View the certificate just created. For example:

   keytool -list -v -keystore server_keystore.jks -storepass mypswd

   Output is similar to the following:

```
Keystore type: JKS
Keystore provider: SUN
Your keystore contains 1 entry
Alias name: rtview_server
Creation date: Nov 15, 2012
Entry type: PrivateKeyEntry
Owner: CN=host1
Issuer: CN=host1
Serial number: 50a567d2
Valid from: Thu Nov 15 17:08:18 EST 2012 until: Sun Nov 13 17:08:18 EST 2022
Certificate fingerprints:
MD5: 96:89:2D:37:71:8E:7F:89:7A:08:71:9F:7F:C2:C1:50
SHA1: 63:1B:1B:ED:75:58:11:34:F0:80:56:45:9B:C9:74:02:CD:D4:C6:89
Signature algorithm name: SHA1withRSA
Version: 3
```

3. Export the server's certificate to a file named **server.cer**:

   keytool -export -alias rtview_server -keystore server_keystore.jks -storepass mypswd -file server.cer

4. Import the server certificate into a **truststore** file for the client:

   keytool -importcert -alias rtview_server -file server.cer -v -noprompt -keystore client_truststore.jks -storepass mypswd

5. Create a **keystore** filed named **client_keystore.jks** to store the client's private key and self-signed certificate:

   keytool -genkey -keystore client_keystore.jks -alias rtview_client -validity 3650 -keyalg RSA -storepass mypswd -keypass mypswd -dname cn=host2

6. Export the client's certificate to a file named **client.cer**:

   keytool -export -alias rtview_client -keystore client_keystore.jks

   -storepass mypswd -file client.cer

7. Import the client certificate into a **truststore** file for the server:

   keytool -importcert -alias rtview_client -file client.cer -v -noprompt -keystore server_truststore.jks -storepass mypswd

8. Verify that you now have four files with a **.jks** extension:

   client_keystore.jks

   client_truststore.jks

   server_keystore.jks

   server_truststore.jks

9. Copy the files as follows:

   - Copy **server_keystore.jks** and **server_truststore.jks** to the directory on host1 where the Data Server will run.

- Copy **client_keystore.jks** and **client_truststore.jks** to the directory on host2 where the client will run.
- If you copy the **keystore** (**.jks**) files between Windows an Linux/UNIX hosts, copy them in binary mode.
- The **client.cer** and **server.cer** files are not needed and can be deleted.

10. On host1, run the Data Server as follows (Note that the "set RTV_JAVAOPTS" command is shown in Windows command syntax and should be entered on a single line):

set RTV_JAVAOPTS=-Djavax.net.ssl.keyStore=server_keystore.jks -Djavax.net.ssl.keyStorePassword=mypswd

-Djavax.net.ssl.trustStore=server_truststore.jks -Djavax.net.ssl.trustStorePassword=mypswd

run_dataserver -daemon -verbose -client_graylist:host2

11. On host2, run a Viewer client as follows

set RTV_JAVAOPTS=-Djavax.net.ssl.keyStore=client_keystore.jks -Djavax.net.ssl.keyStorePassword=mypswd

-Djavax.net.ssl.trustStore=client_truststore.jks -Djavax.net.ssl.trustStorePassword=mypswd

run_viewer -dataserver:remote:host1

12. Verify that you now see output similar to the following on the Data Server console, which indicates that the Viewer's connection to the Data Server is accepted:

```
GmsRtViewDataServer connect client 1, 192.168.1.200:55401
created SSL socket 192.168.1.200:55401
cipher=SSL_RSA_WITH_RC4_128_MD5
add Client: 1 (127.0.0.1/127.0.0.1)
accept graylisted client 1 (127.0.0.1)
```

Note the lines indicating creation of an SSL socket and the cipher used. If errors occur, see "Troubleshooting Client Connection Requests" (below).

SSL Encryption Without Certificates

The **-ssl** option can be specified when the Data Server is started. With this option, the Data Server uses an SSL socket for each socket connection to clients but without performing an SSL authentication handshake. All data transmitted on the socket is encrypted using an anonymous cipher.

The **-ssl** option can be used in combination with client access lists. If a client is graylisted, an SSL handshake is performed when it connects to the Data Server (as described in the "Access List Process Flow"). If a client is whitelisted, its connection is accepted without an SSL handshake and an SSL socket with an anonymous cipher is used. The Data Server's verbose output indicates this with messages similar to the following:

```
comparing client 192.168.1.200 to whitelist entry 192.168.1.*: MATCH
created SSL socket 192.168.1.200:55812
cipher=SSL_DH_anon_WITH_RC4_128_MD5
add Client: 4 (host2/192.168.1.200)
```

Note that encryption may impact CPU and network performance on connections that transmit high volumes of data.

Troubleshooting Client Connection Requests

The following are available for troubleshooting Data Server client connection requests.

To Debug Requests

Use the **client_access_debug** option to debug Data Server client connection requests. This option prints messages to the Data Server console as each client connection request is compared to the client lists. This can be useful for determining which list entries match and do not match a client connection.

To specify on the Data Server command line:

-client_access_debug

To specify in **DATASERVER.ini**:

client_access_debug true

To specify in a properties file:

sl.rtview.dataserver.client_access_debug=true

To Debug SSL Authentication and Certificate Issues

For troubleshooting SSL handshaking and certificate issues, set the following before running the Data Server or the client.

To specify in RTV_JAVAOPTS:

set RTV_JAVAOPTS=-Djavax.net.debug=ssl,keymanager

To specify in an RTView properties file:

sl.rtview.jvm=-Djavax.net.debug=ssl,keymanager

Note that this option may produce a great deal of output.

Running the Data Server

You can run the Data Server as an application, a daemon process, or in the background as a Windows Service. See "Running as a Windows Service" for more information.

Note: The first time you run the Data Server you must run it as an application to configure the initial settings.

Running the Data Server as a daemon process will allow you to run without a display. To do so, run the **-daemon** command line parameter from a Windows Command Prompt or UNIX terminal window.

Several command line options are supported for the Data Server. Java options specified in "RTV_JAVAOPTS" will be used by the **run_dataserver** scripts. See "Command Line Options: Data Server" for more information.

Data source specific options are read in from initialization files created in the Display Builder. For information on creating initialization files or command line options for your data source, refer to Application Options or Command Line Options under the "RTView Data Sources" section of this documentation.

Note: To stop the Data Server, you can use the **kill_dataserver** Command Line Option.

Run the Data Server: Windows or UNIX

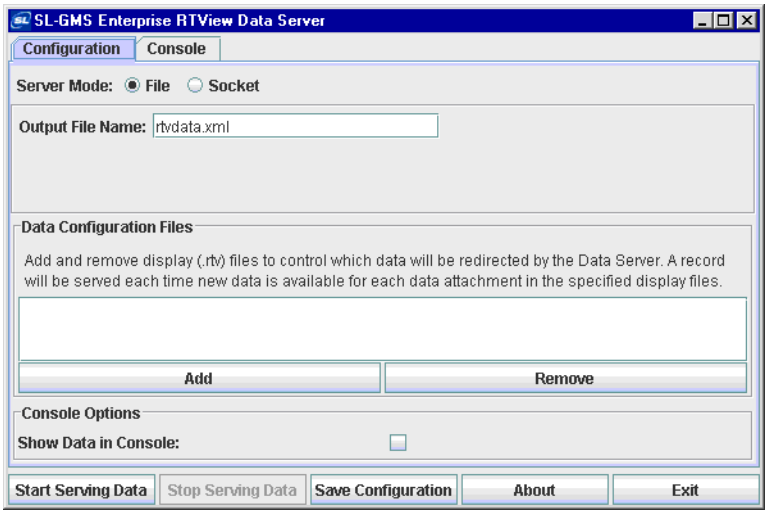
In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window") type:

run_dataserver

Configuration Tab

When running as an application, the **Configuration** tab allows you to specify settings for the Data Server. Click **Save Configuration** to save these settings. If you select the **Show Data in Console** check box, the Data Server Console will output a line for each piece of data that is served. When running as a daemon process, this information will be output to the command window in which the daemon process was started.

To begin serving data, click **Start Serving Data**. Click **Stop Serving Data** to end this process. If you configure the Data Server to output to an XML file, when you start serving data information will be added to the output XML file each time new data is received for each attachment in all of the specified data configuration files. If you configure the Data Server to output data via socket, when you start serving data the Data Server will receive data requests from RTView clients (Display Builder, Display Viewer Application) indicating what data is needed for currently running displays.



Field Name	Description
Server Modes	This information must correspond with information entered in the **Application Options>**"Data Server Tab". **Note:** By default, the Data Server starts in File Mode. If you change the Server Mode, then you must save the configuration and restart the Data Server.
File	Data redirected from configuration files will be output to an XML file. **Note:** The XML data source is never redirected through the Data Server when you output data to a file. When you start serving data, information will be added to the output XML file each time new data is received for each data attachment in all of the specified data configuration files.
Output File Name	Name of file output by the Data Server containing data from specified configuration files. Default output file name is rtvdata.xml.
Data Configuration Files	Specify which data configuration (.rtv) file(s) will be used to generate the output XML file. Data configuration files are display (.rtv) files created in the Display Builder that contain attachments to data you want to redirect through the Data Server. **Add** - Add a data configuration file to the list. If a data configuration file is added while the Data Server is serving data, it will start serving data for attachments in that file as soon as new information becomes available. **Remove** - Remove the selected data configuration (**.rtv**) file from the list. If the Data Server is serving data when a file is removed, it will stop serving data for that file immediately.
Substitutions	To add or edit a substitution on a specified data configuration (**.rtv**) file, double-click in the corresponding field of the Substitutions column. See "Substitutions" for more information.
Socket	Data requested by RTView clients will be output via socket. When you start serving data, the Data Server will receive data requests from RTView clients indicating what data is needed for currently running displays. **Note:** By default the RTView client will connect directly to the XML data source, while all other data sources are redirected through the Data Server. For information on redirecting the XML data source, see **Application Options>**"Data Server Tab".
Port	Specify the port over which the Data Server will communicate with RTView. You do not need to specify a port if you are using the default: **3278**.
Use Secure Sockets	If selected, a secure socket layer (SSL) will be used. RTView clients may connect directly via socket or through the intermediary Data Servlet using HTTP or HTTPS requests to receive data. **Note:** Using the servlet requires a separate installation process, see the "Data Servlet" section for details.

Use Client Credentials for Database Login	Pass RTView login information into all data sources that have the **Use Client Credentials** option enabled. **Note: Use Client Credentials for Database Login** only works in Socket mode. Some data sources do not support this feature. For information on Application Options for your data source, refer to the "RTView Data Sources" section of this documentation.
Send Changed Data Only	If selected, data will be sent only when it has changed. If you want to continuously plot data that has not changed in the trend graph, deselect the **Send Changed Data Only** check box to send all the data regardless of whether or not it has changed. **Note:** Sending all data can lead to performance issues given the amount of information that will be sent over the socket.
Preload Data for Configuration Files	You may specify configuration files for data that you want the Data Server to preload so it will be immediately available to any clients that request it. **Data Configuration Files** - Specify which data configuration (**.rtv**) file(s) will be preloaded by the Data Server. This data will be immediately available to any clients that request it later, and will be updated during Data Server updates. Data configuration files are display (**.rtv**) files created in the Display Builder that contain attachments to data you want to redirect through the Data Server. Enabled if the **Preload Data for Configuration Files** flag is selected. **Add** - Add a data configuration file to the list. **Remove** - Remove the selected data configuration (.rtv) file from the list.

Console Options	**Show Data in Console** - If selected, the Data Server Console will print out a line for each piece of data that is being served.
Start/Stop Serving Data	Start or stop serving data.
Save Configuration	Save settings to an initialization file (**DATASERVER.ini**), which will be used next time you run the Data Server. **Note:** Unless you specify a directory for your initialization files, you must run the Data Server from the same directory in which the initialization (**.ini**) file was saved. See "RTV_JAVAOPTS" for more information.
About	Click on to read about RTView.
Exit	Exit the Data Server, stop serving data and close the **Data Server** window.

Console Tab

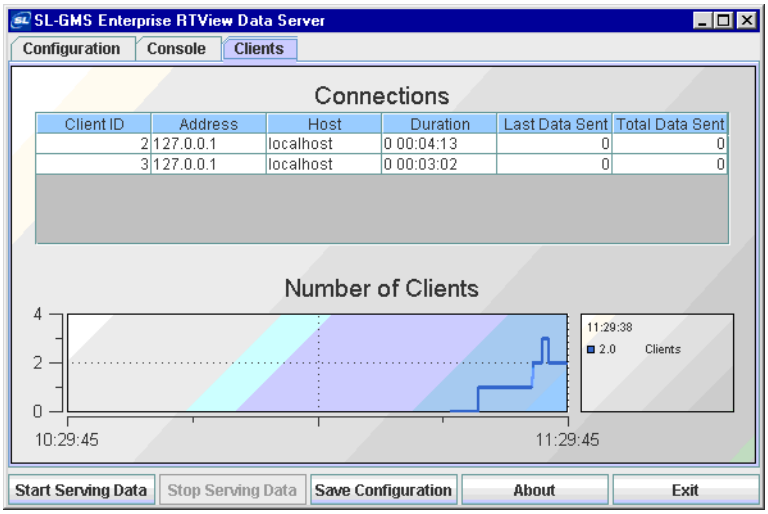
The **Console** tab records errors and information. If you select the **Show Data in Console** check box on the **Configuration** tab, the console will also display a line for each piece of data that is being served. Click the **Clear** button to purge all data from the console.

Note: By default, timestamps are displayed in the **Console** window. Refer to "RTV_JAVAOPTS" for details on how to disable timestamps.

Clients Tab

The **Clients** tab appears when you start the Data Server in **Socket** mode.

Note: By default, the Data Server starts in **File Mode**. If you change the **Server Mode**, then you must save the configuration and restart the Data Server.



Field Name	Description
Connections	This table displays one row for each client that is connected via socket to the Data Server. When clients are connected through the Data Servlet, one connection is shown for the servlet and separate connections are displayed for each HTTP/HTTPS client. If an HTTP/HTTPS client exits under abnormal circumstances, it may not be removed from the table for several minutes. **Note:** Use the **Get Data Server Connection Status** function to access information about the status of connections from the Display Builder and Display Viewer Application to the default Data Server and any Named Data Servers. See "Tabular Functions" for more information. For each client connection, the table contains the following columns.

	Client ID	A unique number assigned by the Data Server to this client connection. The first client connection is assigned an ID of 1, the second is assigned an ID of 2, etc. The ID for an HTTP/HTTPS client will contain extra digits, starting with the ID of the Data Servlet's connection. For example, if the servlet is assigned a client ID of 1, then the first HTTP/HTTPS client will be assigned an ID of 10001, the next will be 10002, etc. ID numbers will not be reused during a Data Server session.
	Address	IP address of the client. Clients running on the same host as the Data Server will display the IP address 127.0.0.1.
	Host	Name of the host on which the client is running. Clients running on the same host as the Data Server will display the host name localhost. If a client's host cannot be determined, the IP address will be shown instead.
	Duration	Time elapsed in this client's session with the Data Server.
	Last Data Sent	Number of characters of XML data contained in this client's most recent update.
	Total Data Sent	Number of characters of XML data contained in all updates during this client's session with the Data Server.
Number of Clients		This trend graph (above) displays the number of clients (direct socket, servlet, HTTP/HTTPS) connected to the Data Server over the past hour.

Server Group Tab

Use the **Server Group** tab to configure the Active / Standby with auto-reset high availability mode. This mode can also be configured on the command line. For details, see the **-group_member** argument in "Command Line Options: Data Server".

Note: For best performance, configure clients to connect to the primary server first (before attempting to connect to the backup server). For example, if the default primary Data Server is **A:5555** and the default backup server is **B:5555**, set each client's connection string to the server pair as "**A:5555,B:5555**". Or, if connections are made via the rtvdata servlet, set each client's connection string to "**http://A/rtvdata,http://B/rtvdata**".

Field Name	Description	
Group Options for this Server	Configure high availability settings for the Data Server.	
	Include in Group	Select to include the server in the server group.
	Priority	Enter a value of 1 or more, with 1 being the lowest priority. A larger value assigns a higher priority. The running server in the pair with the highest priority becomes the primary server. The priority defaults to a value of 1 (the lowest priority) if **Include in Group** is selected and the priority is not specified.
	Timeout	Enter the amount of time (in seconds) that the backup server waits for a connection or response from the primary server, after which it assumes the primary has failed.The default is 5 seconds. (Note that in most cases, if the primary server terminates or the host on which it is running is shut down, the backup server detects this immediately). **Note:** You can use the **group_initial_wait** option to lengthen the wait time for the backup server to connect to its pair at startup. This prevents a failure when the primary and backup server are both started and the backup server attempts to connect to the primary server before it is connection ready. See "Command Line Options: Data Server" for more information.
Connection to other Group Member	Specify the members of the fault tolerant pair.	
	Member	Enter the hostname and port number connection of the other Data Server in this redundant pair using the following format: **hostname:port** If a hostname is specified without a port number, port **3278** is assumed. Do not specify a URL for the rtvdata servlet. **Note:** Selecting Include in Group mitigates the need to specify the **-standby:warm** option. The server remains in standby mode until its determined which server is the primary.

Managing the Data Server Using JMX

The Data Server is instrumented with JMX to allow you to manage and monitor the clients and application settings, and troubleshoot an unresponsive server. To enable JMX, you must run the Data Server in socket mode and using the **-jmxport** command line option:

```
-jmxport:(port number)
```

The JMX Monitor demo, located in **demos\jmxmonitor** in your RTView installation, shows how to use RTView to monitor the Data Server using JMX. This section describes the following MBeans on the Data Server:

- "RTViewDataServer:name=Manager"
- "RTView:name=Troubleshooting"

RTViewDataServer:name=Manager

Use the RTViewDataServer:name=Manager MBean to manage and monitor the clients and application settings. The RTViewDataServer:name=Manager MBean has methods described in the following table.

Method / Attribute	Type	Description
ClientData	TabularData	For each client connection, the table contains the following columns. **Client ID** - A unique number assigned by the Data Server to this client connection. The first client connection is assigned an ID of 1, the second is assigned an ID of 2, etc. The ID for an HTTP/HTTPS client will contain extra digits, starting with the ID of the Data Servlet's connection. For example, if the servlet is assigned a client ID of 1, then the first HTTP/HTTPS client will be assigned an ID of 10001, the next will be 10002, etc. ID numbers will not be reused during a Data Server session. **Address** - IP address of the client. Clients running on the same host as the Data Server will display the IP address **127.0.0.1**. **Host** - Name of the host on which the client is running. Clients running on the same host as the Data Server will display the host name **localhost**. If a client's host cannot be determined, the IP address will be shown instead. **Duration** - Time elapsed in this client's session with the Data Server. **Last Data Sent** - Number of characters of XML data contained in this client's most recent update. **Total Data Sent** - Number of characters of XML data contained in all updates during this client's session with the Data Server. **PID** - The process ID of the client process. Typically, the PID column value appears as **nnnn@hostname**, where **nnnn** is the PID from the client's host system (as reported by jps, top, or tasklist) and **hostname** is the name of the client's host. If the client is the rtvdata or rtvquery servlet, the PID corresponds to the application server (e.g. tomcat) process on the client host. The **PID** column might be blank if the client is unable to obtain its PID from the local operating system. The **@hostname** portion might be omitted if it cannot be obtained from the local operating system. This column is blank if the client process is running an older version of RTView that does not have this enhancement.

		Process Name - This value depends on the client type. If the client is an RTView application, it is the value of the **PROCESS_NAME** property when the application was started. If the client process was started with a standard RTView script (**run_viewer**, **run_displayserver**, etc.) the value is viewer, builder, dataserver, displayserver, or historian. For a server, a "d" at the end of the process name indicates it was started with the **-daemon** option. If the **PROCESS_NAME** property was undefined when the client process was started, then the value is **RTView** (the Viewer), **RTView Display Builder**, **Display Server**, **Data Server**, or **Historian**, or the OEM names assigned to those applications. If the client is the rtvdata or rtvquery servlet the value is **RTVDataServlet** and **rtvquery**, respectively, or the custom servlet name that has been configured for the deployed servlet instance. This column is blank if the client process is running an older version of RTView that does not have this enhancement.
NumberOfClients	int	Number of currently connected clients.
ServingData	boolean	True if the Data Server is currently serving data, false otherwise.
Port	int	The Data Server port number. The default is port 3278. This property is read-only.
PrimaryServer	boolean	True if the Data Server is currently the primary server in a redundant pair, **false** if this server is currently the backup server in a redundant pair. If the server is not in a redundant pair, the value is always **true**. This property is read-only.
ProxyFor	string	If the server is currently a backup server (that is, if **PrimaryServer** is false) this contains the **host:port** of the server that is currently the primary (to which this server forwards all client requests as a proxy). This is empty if the server is currently the primary server. This property is read-only.

The **RTViewDataServer:name=Manager MBean** also supports the following commands:

Method / Attribute	Description
startServingData	Start serving data.
stopServingData	Stop serving data.

RTView:name=Troubleshooting

Use the **RTView:name=Troubleshooting** MBean to detect and troubleshoot an unresponsive server. The Data Server, Display Server, and Historian support the **RTView:name=Troubleshooting** MBean. To attach data to the MBean on an RTView display, enter **RTView:name=Troubleshooting** in the **MBean Name** field of the **JMX Attach To Data** dialog. To access the MBean in jconsole, select the **MBeans** tab, then click **RTView / Troubleshooting** in the tree (jconsole must be connected to the RTView server process).

The **RTView:name=Troubleshooting** MBean has methods described in the following table.

Method / Attribute	Type	Description
ConnectionRequestCount	int	The total number of socket connection requests received from clients. **Note:** The Data Server recycles unused client connection numbers if it reaches its internal limit of 10,000. This can be an issue if client applications from RTView versions prior to 5.9 attempt repeatedly to connect to the Data Server.
ConnectionRequestFailedCount	int	The total number of socket connection requests that failed due to a client/server software version mismatch or because the client is not an RTView client. **Note:** The Data Server recycles unused client connection numbers if it reaches its internal limit of 10,000. This can be an issue if client applications from RTView versions prior to 5.9 attempt repeatedly to connect to the Data Server.
DeadlockCount	int	The count of deadlocked threads. In a healthy server the count is always zero (0). This attribute is only updated every 30 seconds to reduce overhead. You can use the jstack utility to obtain more information about the deadlocked threads.
LastUpdateTime	int	The time at which the server's main update timer most recently changed. In a healthy server, this time changes on each update interval (by default, **2** seconds). If the time stops changing for an extended period, it indicates that the timer thread has stalled or terminated.
UncaughtExceptionCount	int	The count of uncaught exceptions since server started. In a healthy server the count is always zero (0). An uncaught exception terminates the executing thread.
UncaughtExceptions	int	A table of the server's uncaught exceptions listing the type, time, executing thread, and code location for each exception. (For each exception, a stack trace is also written in the server's log file).

Running as a Windows Service

This section describes how to run the Data Server, Display Server, and Historian in the background as Windows Services (on 32- or 64-bit Windows systems). These services run using AlwaysUp CLT, which is included in your RTView deliverable.

Windows Service is run as a local system service. It is not possible to install an application as a Windows Service. If RTView is running off a networked drive, it must be installed on your local system.

Windows Service supports a single log file for both **stdout** and **stderr**. By default, both **stdout** and **stderr** are written to a log file named **appName_out.log**. For example, **displayserver_out.log**. Use the **-serviceout** command line option to override this option.

Logging is appended to the log file each time RTView restarts. When the log file size exceeds 5MB, the oldest 25% of the log file is discarded. For better control over your logging, use either the **-logfile** or the **-log4j** command line options. With both of those options, the service outputs any information written to the console by third party libraries to the log file specified by **-serviceout**. See "Command Line Options: Display Builder and Display Viewer" for more information.

If UAT is enabled on your system, you must run **install_service** and **uninstall_service** as an Administrator.

Some RTView applications use very long command line strings. If the command line string is too long, when you try to view the service properties the Window Services dialog shows the following error: **The stub received bad data**. If this occurs, view look the service properties of your in the Windows Registry under HKEY_LOCAL_MACHINE\SYSTEM**CurrentControlSet**\services\Key, where **Key** is the name of your service.

Command lines with more than 10,240 characters cannot be installed as a service.

This section includes:

- "Install and Start Application as a Windows Service"
- "Stop and Uninstall Application as a Windows Service"
- "Log4j and Windows Service"

Install and Start Application as a Windows Service

After an RTView Data Server, Display Server or Historian is installed as a Windows Service, you can start, stop and configure it in the Windows Services dialog.

For information about using Log4j logging with Windows Service, see "Log4j and Windows Service".

To install and start the application as a Windows Service:

1. Open the command line window:

   Select **Start-->Programs-->Accessories-->Command Prompt-->Right-click Run as administrator**

2. Start the application and Windows Service using the following syntax:

   install_service appName -service:service_name -dir:startup_directory

   Where **appName** is **dataserver**, **displayserver**, or **historian**. **service_name** is the text of your choice which appears in the Windows Event Viewer.

   For example:

   Data Server: **install_service dataserver -service:SLDataserver -dir:c:\myproject**

   Display Server: **install_service displayserver -service:SLDisplayserver -dir:c:\myproject**

   Historian: **install_service historian -service:SLHistorian -dir:c:\myproject**

The following arguments are supported for install_service.

Argument	Description
appName	Required. Must be **dataserver**, **displayserver**, or **historian**.

-dir	Required. Specify the full path to the directory where the application runs. This is the project directory that contains all files necessary (for example, **.rtv** files, configuration files) to run the application. Example: **-dir:c:\myproject** **Note:** If the directory name contains spaces, enclose the entire argument in quotes: **"-dir:c:\my dir"**.
-manualStart	Optional. Set the service startup type to **manual**. The service starts when it is installed but does not automatically start on system startup. If not specified, the service startup type is wet to **auto**.
-service	Required. Specify the name to use for this service. This name is displayed in the Windows Event Viewer. Example: **-service:MyService** **Note:** If the service name contains spaces, enclose the entire argument in quotes: **"-service:my service"**.
-serviceout	Optional. Specify the full path and file name for the **out** log file. If not specified, the **out** log is written to the **-dir** specified and named **appNamed_out.log**. Example: **-serviceout:c:\logs\ds_out.log**

Stop and Uninstall Application as a Windows Service

This section describes how to stop and uninstall the Data Server, Display Server, or Historian as a Windows Service.

To stop and uninstall the application as a Windows Service:

1. Open the command line window:

   Select **Start-->Programs-->Accessories-->Command Prompt-->Right-click Run as administrator**

2. Stop the application and Windows Service using the following syntax:

   uninstall_service appName -service:service_name

   Where **appName** is **dataserver**, **displayserver**, or **historian**, and **service_name** is the name used for this service (displayed in the **Windows Services** dialog).

   For example:

   Data Server: **uninstall_service dataserver -service:SLDataserver**

   Display Server: **uninstall_service displayserver -service:SLDisplayserver**

   Historian: **uninstall_service historian -service:SLHistorian**

Note: If the service name contains spaces, enclose the entire argument in quotes: **"-service:my service"**.

Log4j and Windows Service

This section describes how to install Log4j and start the Data Server, Display Server or Historian as a Windows Service using Log4j. There are three **.properties** files that are used with Log4:

Data Server: **sl-dataserver-service.log4j.properties**

Display Server: **sl-displayserver-service.log4j.properties**

Historian: **sl-historian-service.log4j.properties**

To install Log4j and start your application as a Windows Service using Log4j:

1. Download the file **logging-log4j-1.2.16.zip** from the Apache site *http://archive.apache.org/dist/logging/log4j/1.2.16/*.

2. Extract the Windows **.dll** assembly from the **.zip** file.

3. Select the **NTEventLogAppender.dll** file and place it in a directory that is on the PATH of the Windows system. We recommend the following location for 32-bit systems: *C:\Windows\System32*.

Note: The absence of the **NTEventLogAppender.dll** file causes the **java.lang.UnsatisfiedLinkError** error.

---

4. Open the command line window:

   Select **Start-->Programs-->Accessories-->Command Prompt-->Right-click Run as administrator**

5. Start Log4j and Windows Service using the following syntax:

   install_service appName -service:service_name -dir:startup_directory -log4j -log4jprops:<log4j configuration file> -serviceout:<log directory>\<out log file name> -serviceerr:<log directory>\<err log file name>

   Where **appName** is **dataserver**, **displayserver**, or **historian**. **service_name** is the text of your choice which appears in the Windows Event Viewer.

   For example:

   Data Server: **install_service dataserver -service:SLDataserver -dir:startup_directory -log4j -log4jprops:sl-dataserver-service.log4j.properties -serviceout:c:\logs\dataserver_out.log -serviceerr:c:\logs\dataserver_err.log**

   Display Server: **install_service displayserver -service:SLDisplayserver -dir:startup_directory -log4j -log4jprops:sl-displayserver -service.log4j.properties -serviceout:c:\logs\displayserver _out.log -serviceerr:c:\logs\displayserver _err.log**

   Historian: **install_service historian -service:SLHistorian -dir:startup_directory -log4j -log4jprops:sl-historian-service.log4j.properties -serviceout:c:\logs\historian_out.log -serviceerr:c:\logs\historian_err.log**

Note: Upon startup, Log4j writes a few lines to the log files used by the previous "redirect" logging method (the logging method in RTView v.5.9 and earlier). For this reason the redirect method log file names are used (specified by **–serviceout** and **–serviceerr.** Log4j then switches to the NT Event Viewer.

---

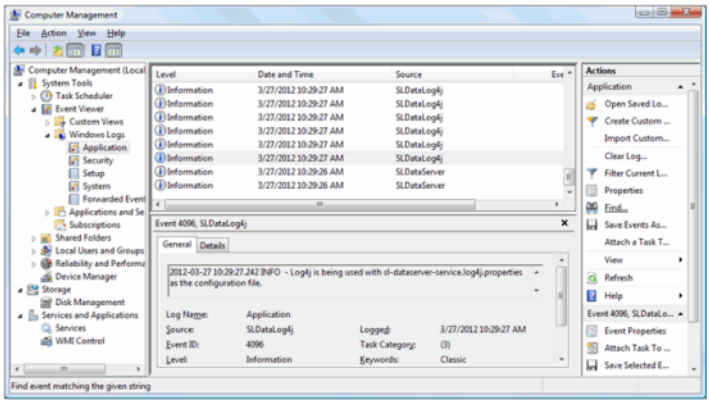
6. View your Log4j log files by opening the Windows Computer Management application:

   Start-->Programs-->Administrative Tools-->Computer Management

   Or, if Administrative Tools is not available:

Start-->Programs-->Right-click Computer-->Manage

The Computer Management application opens.



7. Go to **System Tools-->Event Viewer-->Windows Logs-->Application**.

8. Verify that the center upper panel lists each line in the log file, and the text for the line appears in the **General** tab panel. The **Source** field, in the lower panel, identifies the service name (specified in the **-service** argument).

   The following arguments are supported for **install_service** and Log4j.

Argument	Description
-log4j	Optional. Used for Log4j logging. Specify to use Log4j logging. Example: **install_service dataserver -service:my_data_server -dir:C:\newyork -log4j**
-log4jprops	Optional. Used for Log4j logging. Specify the **.properties** file to use to format the Log4j log file. Use the following files for the: Data Server: **sl-dataserver-service.log4j.properties** Display Server: **sl-displayserver-service.log4j.properties** Historian: **sl-historian-service.log4j.properties** Example: **-log4j -log4jprops:sl-dataserver-service.log4j.properties**
-log4jlevel	Optional. Used for Log4j logging. Specify the Log4j Level. **INFO** is the default. Valid values are: **FATAL**: Indicates a severe error that likely causes the application to abort. **ERROR**: Indicates an event that might not cause the application to abort. **WARN**: Indicates a potentially harmful event. **DEBUG**: Indicates detailed informational about events for debugging the application. **INFO**: Indicates informational messages about the progress of the application at coarse-grained level. Example: **-log4j -log4jlevel:DEBUG**
-showlogcat	Optional. Used for Log4j logging. Specify to turns on the Category column in the log file output. Example: **-log4j -showlogcat**

Data Servlet

This section is intended for users with a web server and a standard working knowledge of servlet deployment.

The Data Server includes a Data Servlet that will run on your web server. The servlet communicates with the Data Server via socket. If you are using multiple Data Servers, you must configure and install a Data Servlet for each Data Server. If you want to install multiple Data Servlets on the same application server, each must have a unique name.

Install Data Servlet

The **servlets\rtvdata** directory contains files necessary to run the Data Servlet. For your convenience, an installation script is provided to install these files on your web server.

In an initialized command window, go to the **servlets\rtvdata** directory and run the following scripts.

Note: These scripts require that you set the environment variable **CATALINA_HOME** to the location of your Tomcat installation directory, see the "Setup" section for details.

---

make_war	This script creates a web archive (.war) named rtvdata.war.
install_to_tomcat rtvdata	This script installs the web archive **rtvdata.war** to your Tomcat server. **Note:** This script will shutdown and restart Tomcat and requires administrative permissions.

Note: If you are using a JSP servlet container other than Tomcat, install the files contained in **servlets\rtvdata** and classes from **lib\gmsjcs_client.jar** on your web server according to instructions given with that product.

Data Servlet Options

The Data Servlet reads the file **servlet.properties**.

Note: If you have already installed the servlet on your web server, you can edit this properties file on your web server. If you edit this file in **servlets\rtvdata** you must reinstall the servlet. In either case, you may need to restart your web server after making changes to this file.

You can set the following options in **servlet.properties**:

Option	Description
ServiceHost	The name of the host on which the Data Server is running. Default is **localhost**. **Note:** Default **localhost** assumes the servlet and Data Server are running on the same machine.

ServicePort	Port the Data Server will use when communicating with the servlet. Default is **3278**.
ServiceTimeout	Amount of time (in seconds) the servlet will wait for replies from the Data Server. Default is **15 seconds**.

The following is an example of the **servlet.properties** file:

```
ServiceHost=localhost
ServicePort=3278
ServiceTimeout=15
```

High Availability Configurations

High Availability configurations are possible for each of the following critical RTView components: Historian, Data Server, Display Server, and Servlets. Redundant components can be set up to provide backup failover capability for each component or for any individual component considered at risk.

Backup components may be run in hot or warm standby mode. In hot standby mode all global variable definitions, as well as cache and alert definitions, are loaded and activated at start up. In warm standby mode none of these actions are performed, thereby avoiding the overhead of maintaining Alert and Cache data sources until the backup component has become the primary.

Note: User interactions with Alerts such as Alert Acknowledgment are not maintained when a backup component becomes the primary -- this limitation will be resolved in an upcoming release.

This section includes:

High Availability Historian

The Historian is an application you can configure to store time-stamped data, derived from either raw real-time data or aggregated/transformed data, in a relational database of choice. To provide high availability, it is possible to designate backup Historians to support a failover event. See "Configuring Failover on the Historian" for details. This high availability solution is intended to be used with a database system that also supports redundancy (through mirroring, clustering, or other techniques) so that any Historian in the redundant group can update the same virtual database.

High Availability Deployments

Details for other high availability configurations depend on the type of deployment selected. The following pages explore five deployment options detailing basic configurations versus high availability configurations. Where applicable, examples are provided on how to configure high availability deployments involving web servers and application servers. The examples below use Apache and Tomcat, although other web servers and application servers can be used.

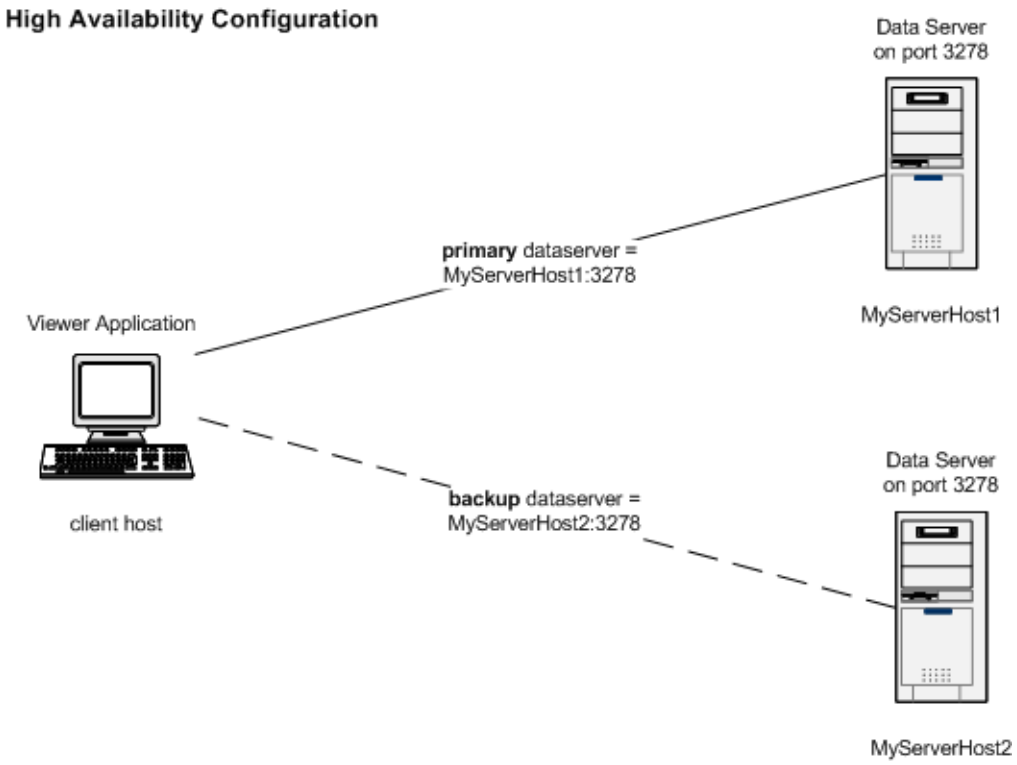
Display Viewer Application

Data Server via Direct Socket

This deployment is chosen when it is more desirable to have a Java application to view RTView dashboards as opposed to a browser-based interface. Direct socket connections are used when there are no firewall issues between clients and the Data Server component.

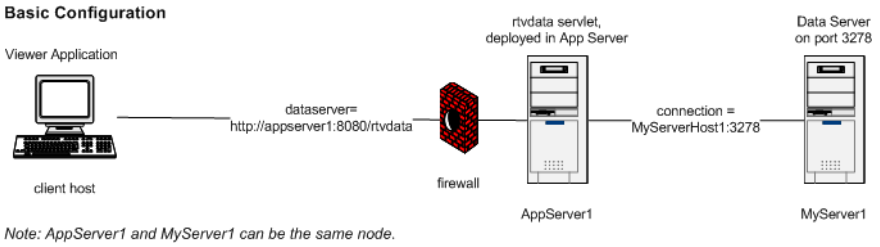
In this high availability configuration, as shown below, if the Data Server connection is lost or unavailable, the Display Viewer Application will switch to a backup server. All backup servers should have access to the same data sources and configuration files.

Primary and backup Data Servers can be configured in the Display Builder, by selecting the Data Server tab in the Application Options dialog, or with the Configuration Utility. See the "High Availability" Data Servers section for details.



Data Server via HTTP

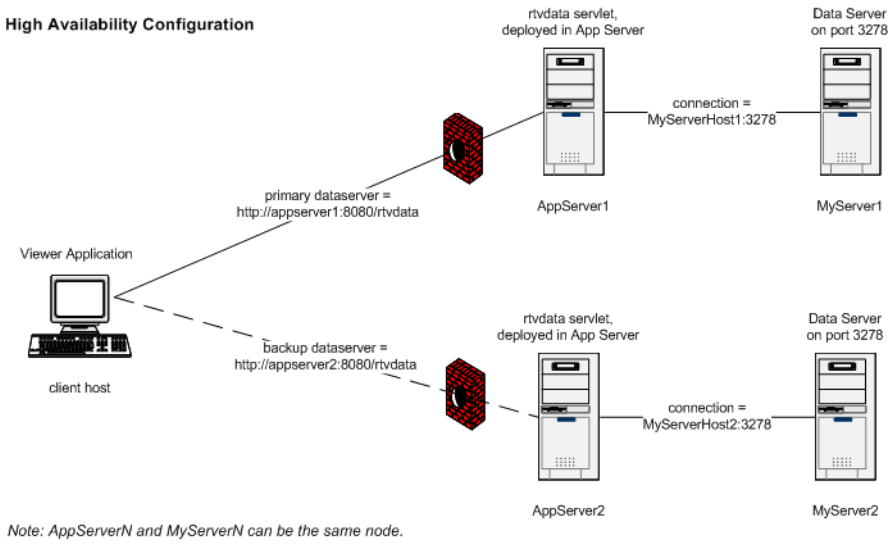
This deployment is chosen when it is more desirable to have a Java application to view RTView dashboards as opposed to a browser-based interface. HTTP connections are used when firewall issues exist between clients and the Data Server component.



In this high availability configuration, as shown below, if the primary Data Server connection (**http://appserver1:8080/rtvdata**) is lost or unavailable, the Display Viewer Application will switch to the backup Data Server connection. Failover to the backup connection will also occur if the primary rtvdata servlet loses its connection to the Data Server (**MyServerHost1:3278**).

Primary and backup Data Servers can be configured in the Display Builder, by selecting the **Data Server** tab in the **Application Options** dialog, or with the Configuration Utility. See the "High Availability" Data Servers section for details.

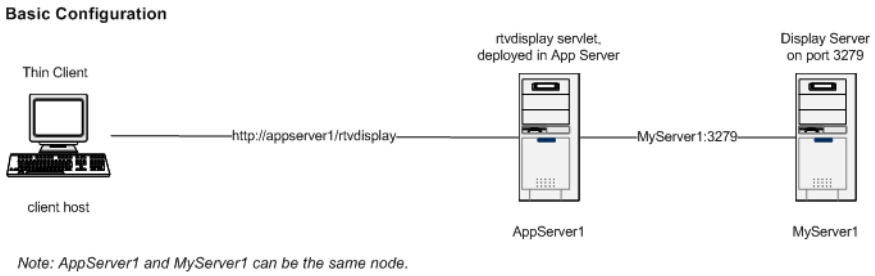
The nodes labeled **AppServer 1** and **MyServer 1** on the diagram could be the same physical node in an actual deployment. Likewise for **AppServer2** and **MyServer2**.



Thin Client

Display Server

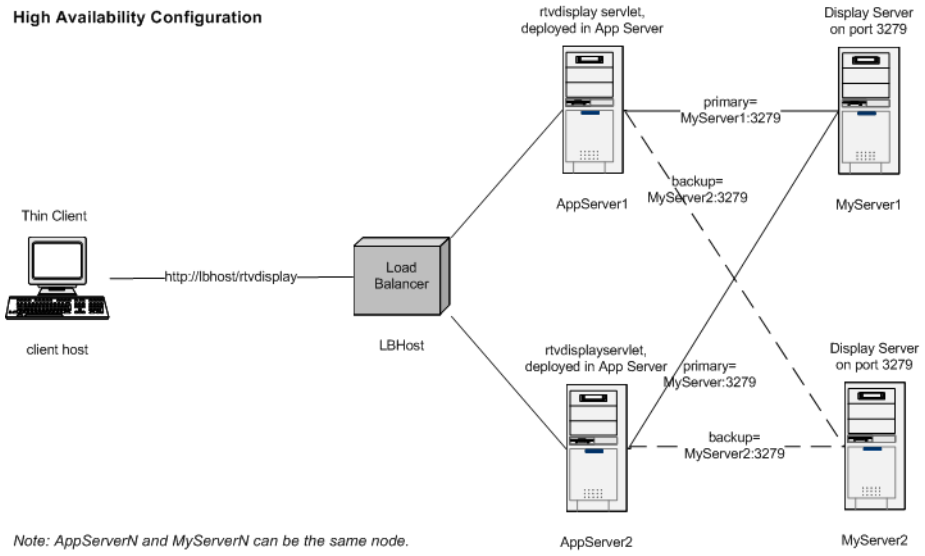
This deployment is used when a browser interface is desired and nothing needs be installed on the client except a browser.



This high availability configuration, as shown below, is perfect for smaller deployments where the Display Server acts to serve client display requests and also acts as the Data Server. See the "High Availability Display Servers" section for configuration details.

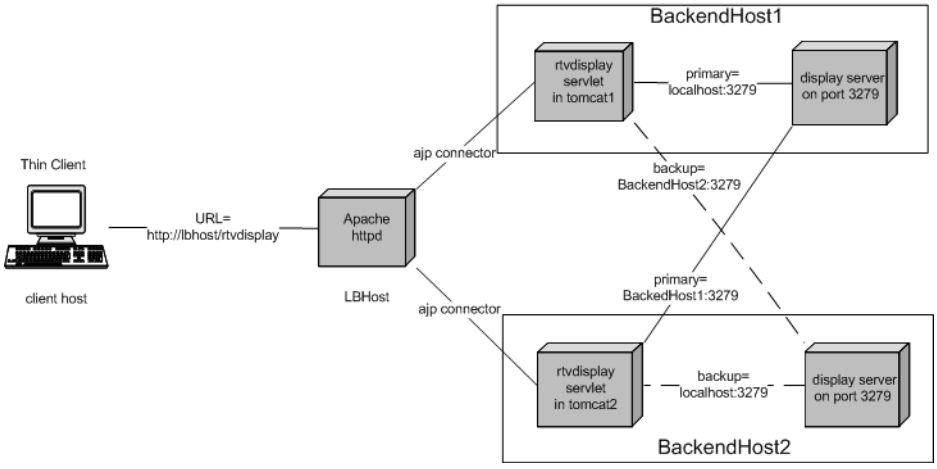
Note: If there are fifty or more potential concurrent users, a large usage of cache or alert definitions, or large amounts of data are being aggregated before they are pushed across a network to be displayed, it is better to opt for a high availability deployment where the Display Server is used in conjunction with the Data Server. See "Display Server and Data Server" for more information.

Note in the diagram below, that the load is distributed to multiple application servers running rtvservlet. Regardless of which servlet is receiving the request, each servlet points to the same primary Display Server. This is to preserve the state of the data, which may include Cache and Alert states



High Availability Configuration with Apache and Tomcat

The diagram and example that follow detail how to configure this high-availability deployment with Tomcat 6.0.18 and Apache 2.2.9.



1. On BackendHost1, in Tomcat's **server.xml** file, modify the Engine entry by adding a jvmRoute attribute:

```
<Engine name="Catalina" defaultHost="localhost" jvmRoute="tomcat1">
```

2. On BackendHost2:

```
<Engine name="Catalina" defaultHost="localhost" jvmRoute="tomcat2">
```

3. On both BackendHost1 and BackendHost2, un-comment the following line in Tomcat's server.xml file:

```
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
```

This enables tomcat clustering. Client session information is shared between the redundant rtvdisplay servlets. So, if the **rtvdisplay** login feature is enabled, clients won't need to login again when a servlet failover occurs. See *http://tomcat.apache.org/tomcat-6.0-doc/cluster-howto.html* for more information.

4. In rtvdisplay/web.xml, un-comment the following line so that the rtvdisplay servlet will use clustering:

```
<distributable/>
```

5. Use **make_war.bat** to rebuild rtvdisplay.war, then deploy it on BackendHost1 and BackendHost2.

6. On the host running the Apache webserver (load-balancer), make the following changes to the httpd.conf file to enable load balancing:

- Un-comment the LoadModule line for all mod_proxy modules: proxy_module, proxy_ajp_module, proxy_balancer_module, proxy_connect_module, proxy_ftp_module, proxy_http_module.
- Define a proxy for the balancer group of rtvdata servlets. The name of this proxy (rtvdisplay) is the name by which clients will access the servlets.

```
# proxy for rtvdisplay load balancer
 ProxyPass /rtvdisplay balancer://rtvdisplay_cluster stickysession=JSESSIONID
```

Note: use same name (rtvdisplay)# as actual workers, otherwise JSESSIONID path may not be found by browser.

- Define the workers (servlets) for the rtvdisplay balancer group. The default tomcat ajp connector on port 8009 is used. Note that the value of "route" for each BalancerMember must match the entry in the corresponding server.xml entry for tomcat, as described above.

```
# list of actual workers for rtvdisplay_cluster
 # (route must match jvmRoute in Engine entry in server.xml
 # for the correpsonding tomcat instance!)
 <Proxy balancer://rtvdisplay_cluster>
         BalancerMember ajp://BackendHost1:8009/rtvdisplay route=tomcat1
         BalancerMember ajp://BackendHost2:8009/rtvdisplay route=tomcat2
 </Proxy>
```

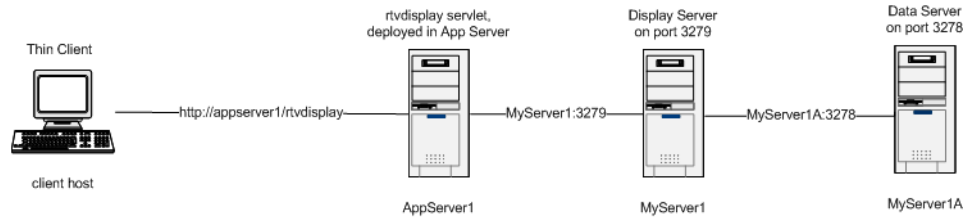
7. For monitoring, enable the apache balancer-manager webapp. This webapp can be viewed in a browser via *http://localhost/balance-manager*.

```
# enable balance-manager
 <Location /balancer-manager>
         SetHandler balancer-manager
 </Location>
```

Display Server and Data Server

This deployment is used when a browser interface is desired and nothing need be installed on the client except a browser.

Basic Configuration

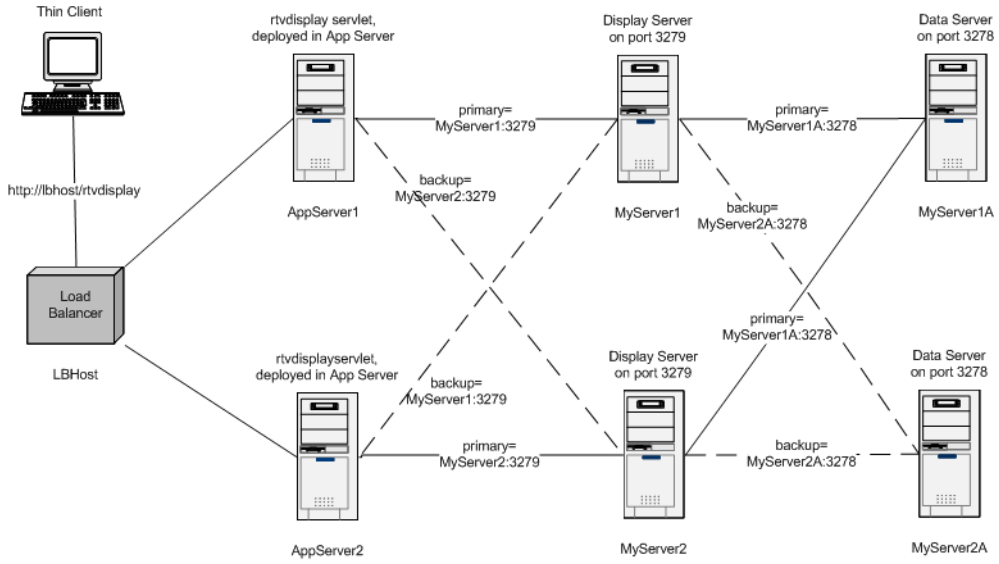


Note: AppServer1, MyServer1, MyServer1A can be the same node.

This high availability configuration, as shown below, is ideal for larger deployments when there are fifty or more potential concurrent users, a large usage of cache or alert definitions, and/or large amounts of data are being aggregated before they are pushed across a network to be displayed. The Display Server acts to serve client display requests, but Data Server(s) provide data access, data aggregation, data caching and alert rule execution. See the "High Availability Display Servers" and "High Availability" Data Servers sections for configuration details.

Note in the diagram below, that the load is distributed to multiple application servers running **rtvservlet**. Display Servers are stateless when utilized strictly for handling client requests and serving up real-time dashboards, therefore it is possible to balance the load between multiple Display Servers in order to handle large numbers of concurrent client users. Notice that each load balanced Display Server points to the same Data Server; this is to preserve the state of the data which may include Cache and Alert states.

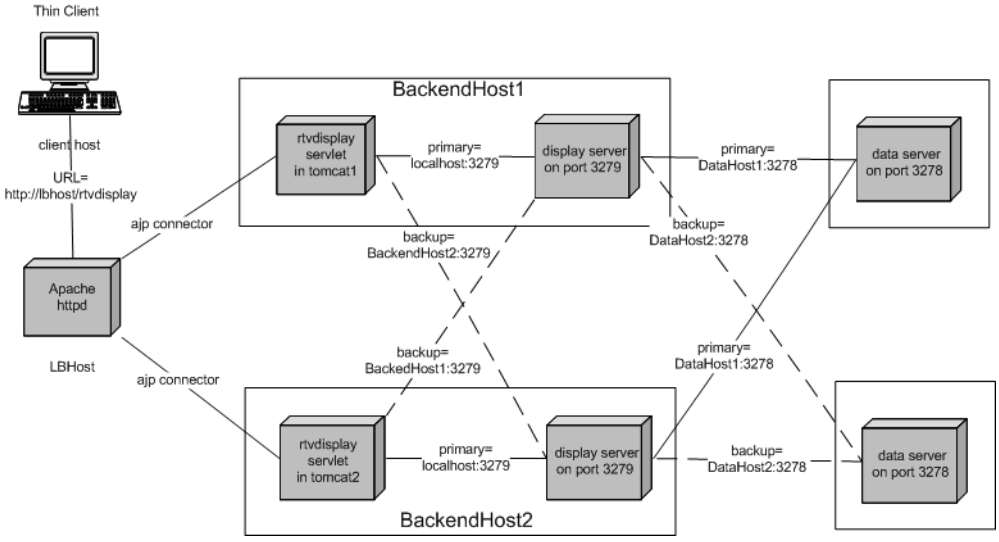
High Availability Configuration



Note: AppServerN and MyServerN can be the same node.

High Availability Configuration with Apache and Tomcat

The diagram and example that follow detail how to configure this high-availability deployment with Tomcat 6.0.18 and Apache 2.2.9.



1. On BackendHost1, in Tomcat's **server.xml** file, modify the Engine entry by adding a **jvmRoute** attribute:

   ```
   <Engine name="Catalina" defaultHost="localhost" jvmRoute="tomcat1">
   ```

2. On BackendHost2:

   ```
   <Engine name="Catalina" defaultHost="localhost" jvmRoute="tomcat2">
   ```

3. On both BackendHost1 and BackendHost2, uncomment the following line in Tomcat's **server.xml** file:

   ```
   <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
   ```

This enables tomcat clustering. Client session information is shared between the redundant rtvdisplay servlets. So, if the rtvdisplay login feature is enabled, clients won't need to login again if a servlet failover occurs. For more info see *http://tomcat.apache.org/tomcat-6.0-doc/cluster-howto.html*

4. In **rtvdisplay/web.xml**, uncomment the following line so that the rtvdisplay servlet will use clustering:

   ```
   <distributable/>
   ```

5. Use make_war.bat to rebuild rtvdisplay.war, then deploy it on BackendHost1 and BackendHost2.

6. On the host running the Apache webserver (load-balancer), make the following changes to the **httpd.conf** file, to enable load balancing:

- Un-comment the LoadModule line for all mod_proxy modules: proxy_module, proxy_ajp_module, proxy_balancer_module, proxy_connect_module, proxy_ftp_module, proxy_http_module.
- Define a proxy for the balancer group of rtvdisplay servlets. The name of this proxy (rtvdisplay) is the name by which clients will access the servlets.

```
# proxy for rtvdisplay load balancer.
 ProxyPass /rtvdisplay balancer://rtvdisplay_cluster stickysession=JSESSIONID
```

Note: use same name (rtvdisplay)# as actual workers, otherwise JSESSIONID path may not be found by browser.

- Define the workers (servlets) for the rtvdisplay balancer group. The default tomcat ajp connector on port 8009 is used. Note that the value of "route" for each BalancerMember must match the entry in the corresponding **server.xml** entry for tomcat, as described above.

```
# list of actual workers for rtvdisplay_cluster
# (route must match jvmRoute in Engine entry in server.xml
# for the correpsonding tomcat instance!)
 <Proxy balancer://rtvdisplay_cluster>
         BalancerMember ajp://BackendHost1:8009/rtvdisplay route=tomcat1
         BalancerMember ajp://BackendHost2:8009/rtvdisplay route=tomcat2
 </Proxy>
```

- For monitoring, enable the apache balancer-manager webapp. This webapp can be viewed in a browser via *http://localhost/balance-manager*.

```
# enable balance-manager
 <Location /balancer-manager>
         SetHandler balancer-manager
 </Location>
```

Application Deployment

This section includes:

Served Data Versus Direct Data Connection

With application deployments, clients either have a direct data connection to each data source or they connect to the Data Server. In most production scenarios it is best to use the Data Server. The Data Server uses EII and XML technologies to gather, federate and distribute information from disparate data sources based on information currently in demand. It also caches the data so that multiple demands are delivered to any number of clients - without need of subsequent data queries. These important factors greatly enhance processing speed.

However, in some small scale implementations the direct data connection may be the best choice. For instance, in order to rapidly implement RTView for prototyping and testing purposes. When testing is complete, it can be ported to a served data deployment for the production environment.

Ultimately, this decision would be determined by weighing the advantages that the Data Server brings; performance, scalability, security, easier setup, and lower impact on backend data sources, against the cost advantage of the direct data connection.

Note: Refer to the Data Sources section of this documentation to see if deployment with a Direct Data Connection is supported by your data source.

The pros and cons of the two scenarios, Served Data and Direct Data Connection, are described below.

Pros and Cons

Issue	Served Data	Direct Data Connection
Setup	**Server**: Requires Data Server setup **Client**: Requires less setup since clients do not need to meet all system requirements for each data source	**Server**: Does not require Data Server **Client**: Requires more setup since each client must meet all system requirements for each data source
Performance	For a very small deployment it is slower, but for all other deployments it is faster	For a very small deployment it is faster, but for all other deployments it is slower
Security	More secure since backend data applications are only accessed by the Data Server	Less secure since backend data applications are directly accessed by clients
Scalability	More scalable due to data requests being federated and caching by the Data Server	Less scalable since each client makes individual data request
Cost	**Large Deployment**: Client maintenance more costly, hardware less costly **Small Deployment**: Hardware cost equal to Direct Data Connection but client maintenance more costly	**Large Deployment**: Client maintenance less costly, hardware more costly **Small Deployment**: Hardware cost equal to Served Data but client maintenance less costly

Application with Served Data

The Application with Served Data deployment involves providing access to the Display Viewer Application for all clients which need to view displays. This could be done by either doing individual installs on each client, sharing an install with a central platform, or via Java Web Start. The Data Server is installed on a server to provide access to all defined data sources and deliver the federated and cached data via XML to the appropriate Display Viewer Applications (see Figure 6). When you deploy RTView as an Application with Served Data, depending on the command, commands are executed either on the server or the client. See command descriptions for information on where commands are executed.

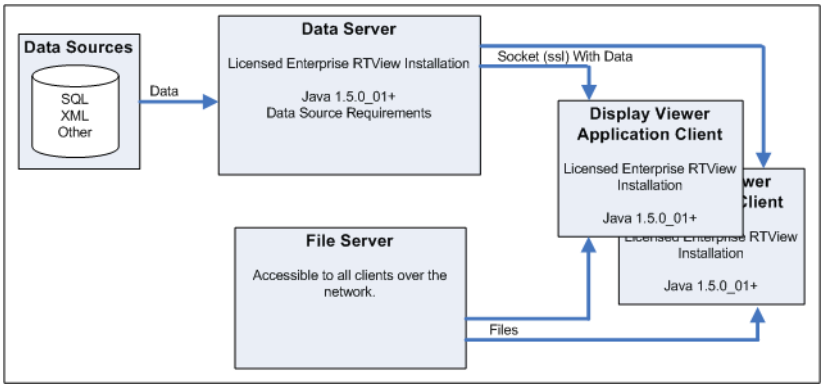


Figure 6: Application with Served Data Deployment Overview

"Application with Served Data - Manual Deployment Process" on page 811

Application with Direct Data Connection

The Application with Direct Data Connection deployment involves providing access to the Display Viewer Application for all clients which need to view displays. This could be done by either doing individual installs on each client, sharing an install with a central platform, or via Java Web Start. Each individual client must be configured so that it can have direct access to data sources. This may involve the installation of additional software such as database drivers or middleware components depending on the data source types used (see Figure 7).

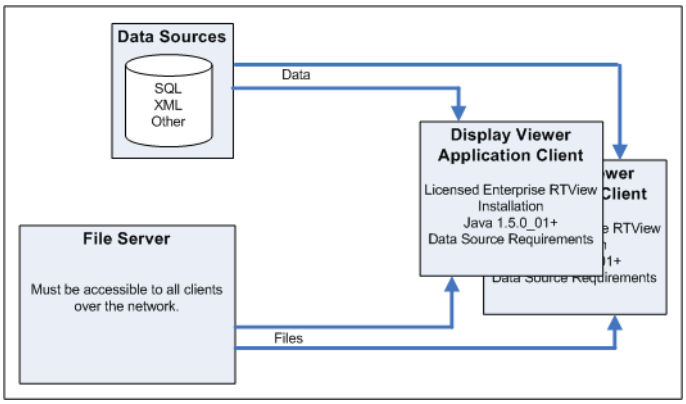


Figure 7: Application with Direct Data Connection Deployment Overview

"Application with Direct Data Connection - Manual Deployment Process" on page 818

Application with Served Data - Manual Deployment Process

The following is an overview of how to manually deploy the Application with Served Data option of RTView. Alternatively, you can use the Deployment Wizard to guide you through this process.

Process Summary

Application with Served Data - Manual Setup

This section provides step-by-step instructions on how to manually deploy RTView. Alternatively, you can use the "Deployment Wizard" to guide you through this process.

Step 1: Install and Configure Data Server

Install RTView on the system where you will run the Data Server. If you are using multiple or high availability Data Servers, you will need to repeat the following steps on each system where you will run Data Server(s). See "Data Server Tab" and "Data Server" for more information.

A: Verify System Requirements

- Basic "System Requirements"
- In addition to basic system requirements, refer to the Data Sources section of this documentation for system requirements and setup specific to your data source.

- Data Server must be accessible via network to all clients.

Note: All clients must have permission to access the Data Server on the port specified for the socket in the **DATASERVER.ini** file (see Step D).

B: Install and Setup RTView

At this point you have verified your system requirements.

1. Install RTView. See "Installation" for more information.
2. Setup RTView. See "Setup" for more information.

C: Register

At this point you have installed and setup RTView.

Register for a license for the Data Server. See "Registration" for more information.

D: Configure Data Server

At this point you have installed, setup, and registered your RTView installation.

1. Create a project directory to store Data Server configuration files.
2. Copy the following files into the directory you just created from the project directory where you developed your RTView application:
   - OPTIONS.ini
   - Refer to Deployment in the Data Sources section of this documentation for information on configuration (**.ini**) files specific to your data sources.
   - All display (**.rtv**) files you want to preload (optional)
   - Style sheet (**.rts**) files (optional)

Note: If you are using multiple or high availability Data Servers, each Data Server needs to run on a different host and/or port. Be sure that the **DATASERVER.ini** file, the **OPTIONS.ini** file, and any related data source initialization files are all correct for this instance of the Data Server. See "Data Server Tab" and "Data Server" for more information.

---

3. If you already created a **DATASERVER.ini** that is configured to run the Data Server in socket mode on the correct port, copy the **DATASERVER.ini** to the project directory you just created. Then go to Step 2: Setup File Server.

   If you have not created a DATASERVER.ini, go to next step.

4. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the project directory you created and type:

   run_dataserver -socket

5. Setup the Data Server configuration. See "Configuration Tab" for more information.

   The Data Server must be configured to run on a socket. The port specified for the socket must match the port specified for the Data Server in Step 2B-5.

6. Click the **Save Configuration** button to save **DATASERVER.ini** and exit the Data Server.

E: Set up RTVAgent Servlet on Application Server (Optional)

Note: This step is only required if you are setting up the Data Server to accept RTVAgent data via HTTP or HTTPS.

At this point you have installed and setup RTView and configured the Data Server.

1. Verify System Requirements

- Application server with a JSP servlet container, such as Apache Tomcat.

2. Install and Setup RTVAgent Servlet

The Data Server uses the RTVAgent Servlet that runs on your application server to access data sent from an RTVAgent application that is sending data via HTTP. The **servlets\rtvagent** directory contains all of the files necessary to configure and install the RTVAgent Servlet.

If you are using multiple Data Servers that are gathering RTVAgent data, you must configure and install a RTVAgent Servlet for each Data Server.

a. Configure RTVAgent Servlet Options

The RTVAgent Servlet reads the **servlet.properties** properties file to get configuration information. If you have not installed the RTVAgent Servlet, modify servlet.properties (in the **servlets\rtvagent** directory) and install (see "b. Install the RTVAgent Servlet"). If you have already installed the RTVAgent Servlet on your application server, you can edit your properties file in the application server. **Note:** You may need to restart your application server after making changes to your properties file.

You can set the following options in **servlet.properties**:

Option	Description
ServiceHost	The name of the host on which the Data Server is running. Default is **localhost**. **Note:** Default **localhost** assumes the servlet and Data Server are running on the same machine.
ServicePort	Port for communicating with the Data Server. Default is **5665**.
ServiceTimeout	Amount of time (in seconds) the servlet will wait for replies from the RTVAgent. Default is **15** seconds.

The following is an example of the **servlet.properties** file:

```
ServiceHost=localhost
ServicePort=5665
ServiceTimeout=15
```

b. Install the RTVAgent Servlet

- In an initialized command window, go to the **servlets\rtvagent** directory and type:

make_war

This script creates a web archive (.war) named rtvagent.war that includes all of the files necessary to run the RTVAgent Servlet.

- Install the files in the **rtvagent.war** file to your application server according to the documentation for that product.

F: Set up RTVPost Servlet on Application Server (Optional)

Note: This step is only required if the application(s) posting data to the HTTP Data Source cannot post directly to your RTView Application due to security or post access limitations.

At this point, you should have installed and setup RTView, configured the Data Server, and (optionally) set up the RTVAgent Servlet.

1. Verify System Requirements
   - Application server with a JSP servlet container, such as Apache Tomcat.
2. Install and Setup RTVPost Servlet

   The Data Server uses the RTVPost Servlet that runs on your application server as a proxy servlet for use with the HTTP Data Source, which is useful for cases where an application cannot post directly to the application running the HTTP Data Source due to security or post access limitations. The **servlets\rtvpost** directory contains all of the files necessary to configure and install the RTVPost Servlet.

   If you are using multiple Data Servers that are gathering HTTP data, you must configure and install a RTVPost Servlet for each Data Server.

   a. Configure RTVPost Servlet Options

   The RTVPost Servlet reads the **rtvpost.properties** properties file to get configuration information. If you have not installed the RTVPost Servlet, modify **rtvpost.properties** (in the **servlets\rtvpost** directory) and install (see "b. Install the RTVPost Servlet"). If you have already installed the RTVPost Servlet on your application server, you can edit your properties file in the application server. **Note:** You may need to restart your application server after making changes to your properties file.

   You can set the following options in **rtvpost.properties**:

Option	Description
proxyHost	The name of the host on which the Data Server is running. Default is **localhost**. **Note:** Default **localhost** assumes the servlet and Data Server are running on the same machine.
proxyPort	Port for communicating with the Data Server. Default is **3275**.
proxyPath	The path to which to post. This is commented out by default.

   The following is an example of the **rtvpost.properties** file:

```
# Properties for the RTView rtvpost servlet.
# proxyHost: the http host to which to post
proxyHost=localhost
# proxyHost: the http port to which to post
proxyPort=3275
# the path to which to post
#proxyPath=
```

   b. Install the RTVPost Servlet

- In an initialized command window, go to the **servlets\rtvpost** directory and type:

  make_war

  This script creates a web archive (.war) named **rtvpost.war** that includes all of the files necessary to run the RTVPost Servlet.

- Install the files in the **rtvpost.war** file to your application server according to the documentation for that product.

Step 1 is completed. Go to Step 2: Setup File Server.

Step 2: Setup File Server

At this point you have installed, setup, and registered RTView, configured the Data Server, (optionally) set up the RTVAgent Servlet, and (optionally) set up the RTVPost Servlet.

Note: Instead of using a file server, you may install your project files onto each client machine. Using a file server allows you to update project files in a single location but is not required.

A: Verify System Requirements

- All clients must be able to map a drive to this system.

B: Install Project Files

At this point you have verified your system requirements.

1. Create a shared project directory to store files for the application.
2. Copy the following files into this directory from the project directory where you developed your RTView application:
   - Display (.rtv) files
   - **COLORS.ini** (only required if Custom Colors have been defined). See "Custom Colors Tab" for more information.
   - Panel configuration file (optional). See "Multiple Display Panels" for more information.
   - jar file containing custom classes (optional)
   - Security configuration files (optional). See "Configuration" for more information.
3. If you already created an **OPTIONS.ini** in which the Display Viewer is configured to connect to the Data Server via socket, copy the **OPTIONS.ini** to the project directory created in Step 1. Then go to Step 3: Install and Setup Client.

   If you have not created an **OPTIONS.ini**, go to next step.
4. On the system where you developed your RTView application, open an initialized command window (see "Initializing a Command Prompt or Terminal Window") and go to the project directory for the application you will be installing and type;

   run_builder -xmlonly -dataserver

Note: The -**xmlonly** option prevents several data sources from loading. This allows us to save only **OPTIONS.ini** and not the other data source configuration files, which are not needed for this step.

---

5. Open the Application Options dialog, select the "Data Server Tab" and do the following:
   - Select **Read Data from Socket (Direct or via HTTP**) from the **Data Server Mode** menu and confirm that the **Connect Directly to Socket** option is selected.
   - Set **Data Server Host** to be the name of the system where you installed the Data Server in Step 1, and the **Data Server Port** to the port specified in Step 1D-5.
   - Configure the **Redirect XML Data Source** option as best suits your application (Redirect XML Data Source is recommended). The **Viewer Only** check box should be selected if you do not want these settings to be used by the Display Builder.

6. Click **Save** to apply and do the following:
   - Save your Data Server options.
   - In the confirm dialog, click **No** to save the **OPTIONS.ini** file to the current directory.

7. Copy the **OPTIONS.ini** file to the directory you created in Step 1.

You have finished Step 2. Go to Step 3: Install and Setup Client.

Step 3: Install and Setup Client

At this point you have installed, setup and registered RTView, configured the Data Server, (optionally) set up the RTVAgent Servlet, (optionally) set up the RTVPost Servlet, and set up the file server.

A: Verify System Requirements

See "System Requirements" Checklist to verify:

- Hardware requirements
- RTView application requirements

Note: You do not need to verify data source requirements.

- Must have permission to access the Data Server on the port specified for the socket in the DATASERVER.inifile (see Step 1D)

B: Install and Setup RTView

Note: Instead of installing RTView on each client system, you can install it only on the file server and have all clients share the installation by running it over a mapped drive. Using a single installation allows you to upgrade your RTView version on a single system. Each client must still register.

---

1. Install RTView. See "Installation" for more information.

2. Setup RTView. See "Setup" for more information.

C: Register

At this point you have installed and setup RTView.

Register for a license for the Display Viewer. See "Registration" for more information.

D: Map Drive to File Server

You have finished Step 3. Go to Step 4: Start / Run Data Server.

Step 4: Start / Run Data Server

At this point you have installed and setup RTView, configured the Data Server, (optionally) set up the RTVAgent Servlet, (optionally) set up the RTVPost Servlet, and set up the file server and a client. If you are using multiple Data Servers, you will need to repeat the following steps on each system where you will run Data Server(s).

A: Run Data Server

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the directory you created in Step 1 and type:

   run_dataserver

2. Click **Start Serving Data**.

Note: You may also run the Data Server as a daemon process. See "Running the Data Server" for more information and additional options.

Note: The Data Server is instrumented with JMX to allow you to manage and monitor the clients and application settings. See "Managing the Data Server Using JMX" for more information.

You have finished Step 4. Go to Step 5: Test Client.

Step 5: Test Client

At this point you have setup RTView, configured the Data Server, (optionally) set up the RTVAgent Servlet, (optionally) set up the RTVPost Servlet, and set up the file server, the client, and are running the Data Server.

A: Testing

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the project directory on the file server and:

   type **run_viewer**

Note: If no file is specified on the command line (on Windows or UNIX), the Display Viewer Application opens **init.rtv**. If the file **init.rtv** is not found, a blank display opens. Once the Display Viewer Application is open, you can access display files by selecting **File>Open**.

Note: Application options set on the command line will override values in initialization files. See "Command Line" for more information on using command line parameters.

---

2. "Login". By default, the Display Viewer requires a login to support "Role-based Security". If your system administrator has configured a user name and password for you, use those. You may also need to select a role. Otherwise, the default user name and password are:

   User Name: admin

Password: admin

Congratulations! RTView deployment is completed.

Application with Direct Data Connection - Manual Deployment Process

The following is an overview of how to manually deploy the Application with Direct Data Connection option of RTView. Alternatively, you can use the Deployment Wizard to guide you through this process.

Note: This documentation is intended for users with a working knowledge of HTML code. If you do not have a full understanding of these topics, you will need assistance from your system administrator.

Process Summary

Application with Direct Data Connection - Manual Setup

This section provides step-by-step instructions on how to manually deploy RTView. Refer to "Application with Direct Data Connection - Manual Deployment Process" for a summary of these instructions. Alternatively, you can use the "Deployment Wizard" to guide you through this process.

Step 1: Setup File Server

Note: Instead of using a file server, you may install your project files onto each client machine. Using a file server allows you to update project files in a single location but is not required.

A: Verify System Requirements

- All clients must be able to map a drive to this system.

B: Install Project Files

At this point you have verified your system requirements.

1. Create a shared project directory to store files for the application.
2. Copy the following files into this directory from the project directory where you developed your RTView application:
   - Display (**.rtv**) files
   - Style sheet (**.rts**) files (optional)
   - Panel configuration file (optional). See "Multiple Display Panels" for more information.
   - **.jar** file containing custom classes (optional)
   - Security configuration files (optional). See "Configuration" for more information.
   - OPTIONS.ini. See "Application Options" for more information.
   - **COLORS.ini** (only required if Custom Colors have been defined). See "Custom Colors Tab" for more information.
   - Refer to "Deployment" in the Data Sources section of this documentation for information on configuration (**.ini**) files specific to your data sources.

You have finished Step 1. Go to Step 2: Install and Setup Client.

Step 2: Install and Setup Client

At this point you have setup the file server.

A: Verify System Requirements

- Basic "System Requirements"
- In addition to basic system requirements, refer to the "RTView Data Sources" section of this documentation for system requirements and setup specific to your data source.

B: Install and Setup RTView

Note: Instead of installing RTView on each client system, you can install it only on the file server and have all clients share the installation by running it over a mapped drive. Using a single installation allows you to upgrade your RTView version on a single system. Each client must still register.

---

1. Install RTView. See "Installation" for more information.
2. Setup RTView. See "Setup" for more information.

C: Register

At this point you have installed and setup RTView.

Register for a license for the Display Viewer. See "Registration" for more information.

D: Map Drive to File Server

You have finished Step 2. Go to Step 3: Test Client.

Step 3: Test Client

At this point you have setup the file server and the client.

A: Testing

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the project directory on the file server and:

type **run_viewer**

Note: If no file is specified on the command line (on Windows or UNIX), the Display Viewer Application opens init.rtv. If the file init.rtv is not found, a blank display opens. Once the Display Viewer Application is open, you can access display files by selecting **File>Open**.

Note: Application options set on the command line will override values in initialization files. See "Command Line" for more information on using command line parameters.

---

2. "Login". By default, the Display Viewer requires a login to support "Role-based Security". If your system administrator has configured a user name and password for you, use those. You may also need to select a role. Otherwise, the default user name and password are:

   User Name: admin

   Password: admin

Congratulations! RTView deployment is completed.

Display Viewer Application

The Display Viewer is a stand-alone Java application. It can either access your data source(s) directly or it can connect to the "Data Server". For step-by-step instructions on how to deploy the Display Viewer Application, see "Application with Served Data - Manual Setup" or "Application with Direct Data Connection - Manual Setup".

This document describes system requirements and how to configure and run the Display Viewer Application.

System Requirements

- Basic "System Requirements"
- In addition to basic system requirements, if you are not using the Data Server, refer to the "RTView Data Sources" section of this documentation for system requirements and setup specific to your data source.

Configuration

If initialization files are present in a specified directory, the current directory or in the lib directory (found in your installation directory) they will be used to set application options. See "Application Options" for more information on creating initialization files. See "RTV_JAVAOPTS" for more information.

Note: Application options set on the command line will override values in initialization files. See "Command Line" for more information on using command line parameters.

RTV_JAVAOPTS

Java options specified in "RTV_JAVAOPTS" will be used by the **run_viewer** scripts.

Start the Display Viewer Application

In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window") type:

type **run_viewer**

Note: If no file is specified on the command line, the Display Viewer Application will open **init.rtv**. If the file **init.rtv** is not found, a blank display will open. Once the Display Viewer Application is open, you can access display files by selecting **File>Open**.

By default, the Display Viewer does not require a login. Login can be enabled at setup to support role based security. The default user name and password are:

User Name: admin

Password: admin

It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role.

View Options

Since editing is not allowed there are no editing dialogs, no toolbar, and menus are limited to **File**, **View**, **Tools**, and **Help**. To include editing dialogs for viewing only, run the Display Builder with the editing disabled instead of running the Display Viewer. Several options are available to control how you view a display, including multiple display panels. See "View Options", "Viewing Multiple Display Panels", and "Command Line Options: Display Builder and Display Viewer" for more information.

Browser Deployment

Thin Client Browser Deployment

The Thin Client Browser deployment involves no installation on the clients. Only a standard browser is necessary on each client. The Display Server must be installed on a server and an application server with a JSP Servlet Container is required to provide visualization to the clients. This solution can be used either from a standard browser or from a portal environment (see Figure 9).

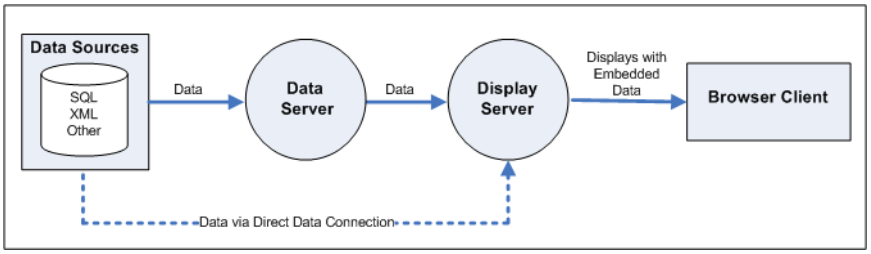


Figure 9: Thin Client Browser Deployment Overview

The Thin Client Browser deployment option requires the installation of the Display Server which is typically installed on a dedicated platform. The Display Server then communicates via a socket to an application server where the Display Servlet is installed. No configuration of the client is necessary other than access to a standard browser.

This section includes:

Served Data Versus Direct Data Connection

In some cases, the Display Server can act as your Data Server to support connections to necessary data sources, handle all defined data calculations, provide cache storage, and maintain the alert rules engine. However, there are situations where this is not possible or desirable.

Data Access/Centralization

Depending on where the Data Server will reside in the network, it may not be desirable to have direct TCP connections to each necessary data source. For example, data sources may reside in a sub-network where it is not possible to open multiple connections to the Display Server. In this case, the Data Server could act as a proxy so that one TCP or HTTP connection to the Data Server could be used to deliver data across the network.

Data Reduction/Aggregation

Sometimes the data exists in a network configuration where bandwidth is very sensitive. Data Servers could be used to maintain data aggregations, data caches and alert information that would only be accessed on demand by the Display Server. This will allow for significant optimization of network bandwidth.

Scalability

If large amounts of data are cached, many data calculations are performed, or large alert rule bases are activated, it may be beneficial to distribute this processing load across one or multiple Data Servers.

The pros and cons of the two scenarios, Served Data and Direct Data Connection, are described below.

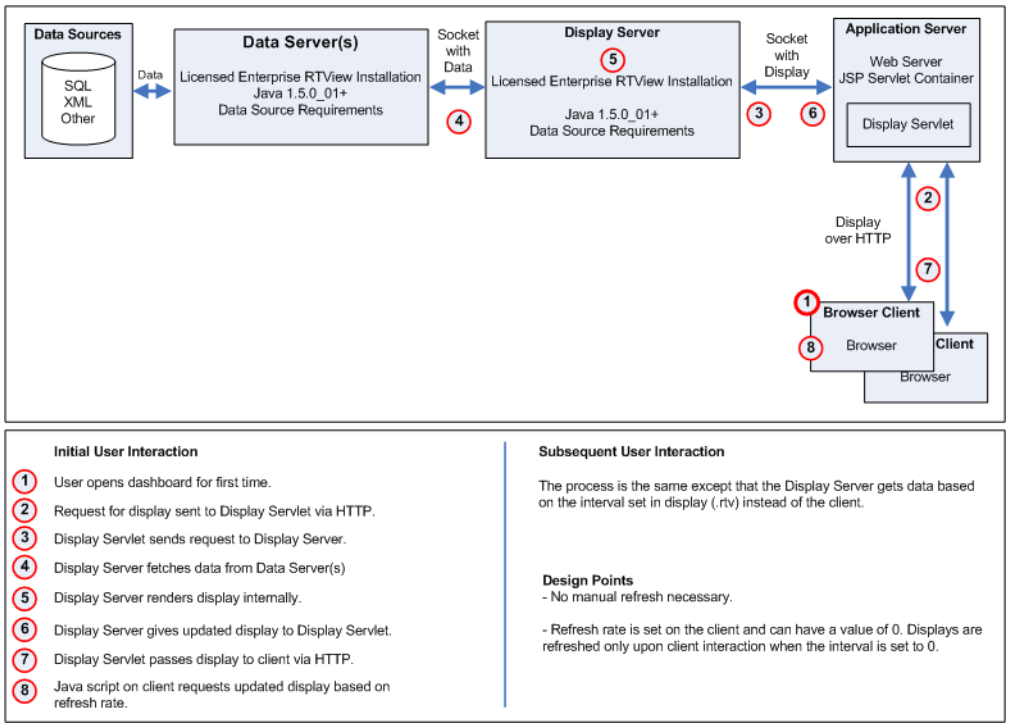
Pros and Cons

Issue	Served Data	Direct Data Connection
Setup	Requires Data Server setup	Does not require Data Server setup
Performance	High performance with additional possibilities for scaling	High performance unless the application needs to scale
Security	A bit more flexible as far as security options since direct data access can be separated from the platform hosting the Display Server	Flexible unless there is an issue with directly connecting to data sources from the platform hosting the Display Server
Scalability	More scalable since Data Servers can be used to divide the processing load	Less scalable since the Display Server must perform all processing
Cost	Hardware and Software costs per additional instance of Data Servers	Less expensive if the Display Server can satisfy all scalability issues and can efficiently connect to all data sources

Thin Client Browser with Served Data

The Thin Client Browser with Served Data deployment involves deploying one Display Server and one or more Data Servers. Each Data Server will have its own configuration files which describe data connections, cache definitions, function definitions, and alert definitions. The Display Server will have one connection to each deployed Data Server and gather the necessary data on demand when users are viewing applicable dashboards.

How It Works

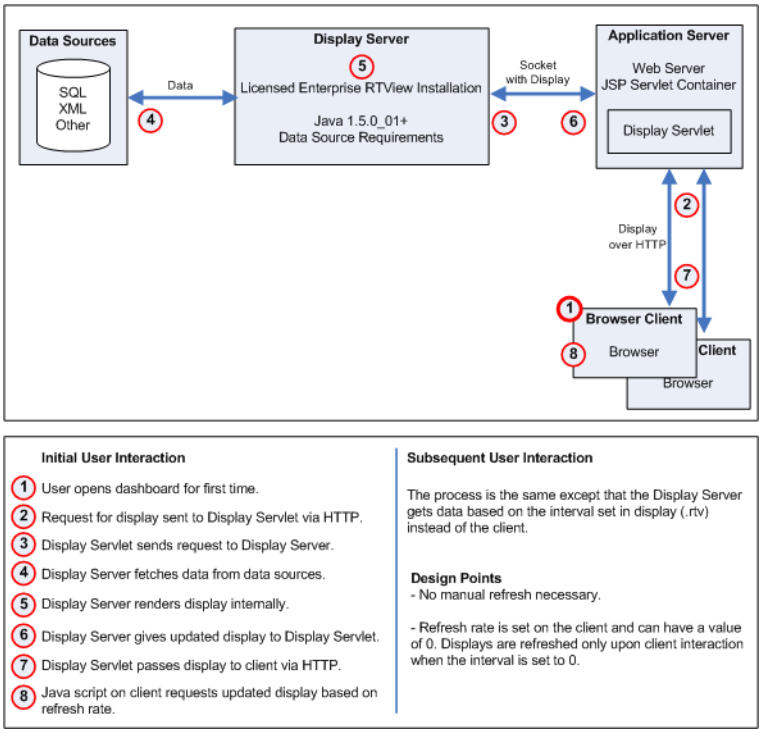


Choose "Thin Client Browser with Served Data"

Thin Client Browser with Direct Data Connection

The Thin Client Browser with Direct Data Connection deployment involves deploying only one Display Server. The Display Server will have a direct connection to all required data sources and will be responsible for serving dashboards, performing data calculations, caching data and processing alerts.

How It Works



Choose "Thin Client Browser with Direct Data Connection"

Thin Client Browser with Served Data - Manual Deployment Process

The following is an overview of how to manually deploy the Thin Client Browser with Served Data option of RTView. The steps need to be followed in the order given. Alternatively, you can use the"Deployment Wizard" to help you deploy your project.

See "How It Works" for a Thin Client Browser with Served Data system overview and sequence of usage.

Note: This documentation is intended for users with a working knowledge of HTML code and application server administration. If you do not have a full understanding of these topics, you will need assistance from your system administrator.

Process Summary

"Step 1: Install and Configure Display Server"

- "A: Verify System Requirements"
- "B: Install Display Server"
- "C: Register"
- "D: Configure Display Server"

"Step 2: Configure and Install Display Servlet"

- "A: Create Display Servlet HTML or JSP Files"

Thin Client Browser with Served Data - Manual Setup

This section provides step-by-step instructions on how to manually deploy RTView. The steps must be done in the order given. Refer to "Thin Client Browser with Served Data - Manual Deployment Process" for a summary of these instructions. Alternatively, you can use the "Deployment Wizard" to help you deploy your project.

This section is intended for users with standard working knowledge of HTML, JSP, and servlet deployment on an application server.

An Apache Tomcat application server is included with your RTView installation for prototyping and testing your deployment before going into the production environment. The following instructions will work for your application server or the one that comes with RTView.

Step 1: Install and Configure Display Server

Install RTView on the system where you will run the Display Server. If you are using "High Availability Display Servers", you will need to repeat the following steps on each system where you will run a Display Server.

A: Verify System Requirements

- Basic "System Requirements"

B: Install Display Server

At this point you have verified your system requirements.

1. Install RTView. See "Installation" for more information.
2. Setup RTView. See "Setup" for more information.

C: Register

At this point you have verified your system requirements and installed the Display Server.

Register for a license to run the Display Server. See "Registration" for more information.

D: Configure Display Server

At this point you have verified your system requirements, and installed and registered RTView.

1. Create a project directory to store Display Server configuration files.
2. Copy the following files into this directory from the project directory where you developed your RTView application:
   - All display (***.rtv**) files
   - Style sheet (**.rts**) files (optional)
   - OPTIONS.ini. See "Application Options" for more information.
   - **COLORS.ini** (only required if Custom Colors have been defined). See "Custom Colors Tab" for more information.
   - Refer to "Deployment" in the Data Sources section of this documentation for information on configuration (**.ini**) files specific to your data sources.
   - Panel configuration file (optional). See "Multiple Display Panels" for more information.
   - Security Configuration files (optional). See "Configuration" for more information.
   - DISPLAYSERVER.ini

Note: If you have no DISPLAYSERVER.ini file, create one now by configuring your Display Server application options. Otherwise, go to Step 2: Configure and Install Display Servlet.

---

3. Configure your Display Server application options. See "Display Server Configuration" for information on how to specify your options. All configuration files should be saved in the directory you just created.

Step 1 is completed. Go to Step 2: Configure and Install Display Servlet.

Step 2: Configure and Install Display Servlet

At this point you have completed the RTView installation and setup.

About Display Servlet

The Display Server uses the Display Servlet, a JSP servlet that runs on your application server. Clients communicate with the Display Servlet using HTTP. The Display Servlet communicates with the Display Server via socket to request HTML for display in the browser. The **servlets\rtvdisplay** directory contains files (JSP, HTML, classes, properties) necessary to install the Display Servlet.

A: Create Display Servlet HTML or JSP Files

The Display Servlet comes with a few HTML files for testing. You can use these to deploy, or you can create your own HTML or JSP files which make calls to the Display Servlet to show your displays.

Skip this step and go to "B: Configure Display Servlet Options" if both of the following are true:

- You will only be deploying one Display Server application on your application server

- You want to deploy using only the test HTML files that come with the Display Servlet

---

Note: If you skip this step, use servlets\rtvdisplay as your project directory referenced in Steps B and C, and use rtvdisplay for the appname argument for all of the scripts.

Otherwise, proceed with the following steps.

1. Create a project directory to store your Display Servlet files.
2. Copy rtvdisplay.properties into this directory from **servlets\rtvdisplay**.
3. Create the HTML files or JSP files for your Display Server application. See "Creating Display Servlet HTML Files" and "Creating Display Servlet JSP Files" for more information.
4. Copy these HTML or JSP files and any referenced files (e.g., images referenced in the HTML files) to this directory.
5. Copy **sample_make_war.bat** (for Windows) or **sample_make_war.sh** (for UNIX) into this directory from **servlets\rtvdisplay** and rename it to **make_war** (with the appropriate extension).
6. This script takes a name for the web archive, without the **.war** extension, and builds a web archive file that includes all of the necessary RTView Display Servlet files along with all .html, .js, .gif, and .jpg files from the current directory. If there are other files you would like to include in your web archive, add them to the following line in your **make_war** script:

```
jar uf %1.war *.html *.jpg *.gif *.js *.jsp WEB-INF
```

Note: You may receive an error message if the script does not find at least one of each of the file types specified (.html, .jpg, etc). Disregard this error message.

B: Configure Display Servlet Options

The Display Servlet reads rtvdisplay.properties to get configuration information. If you have not installed the Display Servlet, modify the rtvdisplay.properties files in your project directory and it will be installed as part of the next step. If you have already installed the Display Servlet on your application server, you can edit this properties file in your application server. You may need to restart your application server after making changes to this file.

Use **servlets\rtvdisplay** as your project directory if you skipped "A: Create Display Servlet HTML or JSP Files".

Set your options in **rtvdisplay.properties** as specified in "Display Servlet Configuration".

C: Install Display Servlet

At this point you have completed Display Servlet setup.

If you skipped "A: Create Display Servlet HTML or JSP Files", use **servlets\rtvdisplay** as your project directory and **rtvdisplay** as your **appname** in this step.

Otherwise, use the project directory that you setup in Step A and your web archive file name for **appname**.

Note: If you will be running multiple Display Server applications on the same application server, each application must have a unique name.

---

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to your project directory and type:

   make_war appname - This script creates a web archive (**.war**) that includes all of the files necessary to run the Display Servlet.

2. Your RTView installation includes an Apache Tomcat application server, so that you can prototype and test your deployment before moving it to your production server. If you will be using this application server, run the following script to install the Display Servlet:

   install_to_demoserver appname - This script installs the web archive to the Apache Tomcat server included in your RTView installation.

Note: This script will shutdown and restart Apache Tomcat and requires administrative permissions.

---

3. If you will be using your own Apache Tomcat application server, run the following script to install the Display Servlet:

   install_to_tomcat appname - This script installs the web archive to your Apache Tomcat server.

Note: This script will shutdown and restart Apache Tomcat and requires administrative permissions.

---

4. If you will be using an application server other than Apache Tomcat, install the files in the web archive to your application server according to the documentation for that product.

Step 2 is completed. Go to Step 3: Configure and Install Display Server Portlet (Optional).

Step 3: Configure and Install Display Server Portlet (Optional)

At this point you have setup the Display Server and the Display Servlet.

The Display Server can optionally be deployed as a portlet. The portlet is tested in Liferay and contains configuration files specific to Liferay, but should work with any JSR-168 compliant portal. If you want to deploy the Display Server as a servlet, skip this step and go to "Step 5: Run Display Server".

Configure install and instance the Display Server Portlet as described in "Configure and Install Display Server Portlet (Optional)".

Step 3 is completed. Go to Step 4: Install and Configure Data Server.

Step 4: Install and Configure Data Server

At this point you have setup the Display Server, Display Servlet and (optionally) the Display Server Portlet.

Install RTView on the system where you will run the Data Server(s). If you are using multiple or high availability Data Servers, you will need to repeat the following steps on each system where you will run Data Server(s). See "Data Server Tab" and "High Availability" for more information. If you want to run the Data Server from the RTView installation in Step 1, skip steps B and C.

A: Verify System Requirements

- Basic "System Requirements"
- In addition to basic system requirements, refer to the "RTView Data Sources" section of this documentation for system requirements and setup specific to your data source.
- Data Server must be accessible via network to all clients.

Note: All clients must have permission to access the Data Server on the port specified for the socket in the DATASERVER.ini file (see Step D).

B: Install and Setup RTView

At this point you have verified your system requirements.

1. Install RTView (see "Installation")
2. Setup RTView (see "Setup")

C: Register

At this point you have installed and setup RTView.

Register for a license (see "Registration") for the Data Server.

D: Configure Data Server

At this point you have installed, setup and registered your RTView installation.

1. Create a project directory to store Data Server configuration files.
2. Copy the following files into the directory you just created from the project directory where you developed your RTView application:
   - OPTIONS.ini
   - Refer to Deployment in the Data Sources section of this documentation for information on configuration (**.ini**) files specific to your data sources.
   - All display (**.rtv**) files you want to preload (optional)

Note: If you are using multiple or high availability Data Servers, each Data Server needs to run on a different host and/or port. Be sure that the **DATASERVER.ini** file, the **OPTIONS.ini** file, and any related data source initialization files are all correct for this instance of the Data Server. See "Data Server Tab" and "High Availability" for more information.

---

3. If you already created a **DATASERVER.ini** that is configured to run the Data Server in socket mode on the correct port, copy the **DATASERVER.ini** to the project directory you just created. Then go to Step 2: Setup File Server.

   If you have not created a **DATASERVER.ini**, go to next step.

4. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the project directory you created and type:

   run_dataserver -socket

5. Setup the Data Server configuration. See "Configuration Tab" for more information.

Note: The Data Server must be configured to run on a socket. The port specified for the socket must match the port specified for the Data Server in Step 2B-5.

---

6. Click the **Save Configuration** button to save **DATASERVER.ini** and exit the Data Server.

E: Set up RTVAgent Servlet on Application Server (Optional)

Note: This step is only required if you are setting up the Data Server to accept RTVAgent data via HTTP or HTTPS.

At this point, you have verified your system requirements, installed and set up RTView, registered a license for the data server, and configured the Data Server.

1. Verify System Requirements
   - Application server with a JSP servlet container, such as Apache Tomcat.

2. Install and Setup RTVAgent Servlet

The Data Server uses the RTVAgent Servlet that runs on your application server to access data sent from an RTVAgent application that is sending data via HTTP. The **servlets\rtvagent** directory contains all of the files necessary to configure and install the RTVAgent Servlet.

If you are using multiple Data Servers that are gathering RTVAgent data, you must configure and install a RTVAgent Servlet for each Data Server.

a. Configure RTVAgent Servlet Options

The RTVAgent Servlet reads the servlet.properties properties file to get configuration information. If you have not installed the RTVAgent Servlet, modify servlet.properties (in the servlets\rtvagent directory) and install (see "b. Install the RTVAgent Servlet"). If you have already installed the RTVAgent Servlet on your application server, you can edit your properties file in the application server. **Note:** You may need to restart your application server after making changes to your properties file.

You can set the following options in servlet.properties:

Options	Description
ServiceHost	The name of the host on which the Data Server is running. Default is **localhost**. **Note:** Default **localhost** assumes the servlet and Data Server are running on the same machine.
ServicePort	Port for communicating with the Data Server. Default is **5665**.
ServiceTimeout	Amount of time (in seconds) the servlet will wait for replies from the RTVAgent. Default is 15 seconds.

The following is an example of the **servlet.properties** file:

```
ServiceHost=localhost
ServicePort=5665
ServiceTimeout=15
```

b. Install the RTVAgent Servlet

In an initialized command window, go to the **servlets\rtvagent** directory and type:

make_war

This script creates a web archive (**.war**) named **rtvagent.war** that includes all of the files necessary to run the RTVAgent Servlet.

Install the files in the rtvagent.war file to your application server according to the documentation for that product.

F: Set up RTVPost Servlet on Application Server (Optional)

Note: This step is only required if the application(s) posting data to the HTTP Data Source cannot post directly to your RTView Application due to security or post access limitations.

At this point, you have verified your system requirements, installed and set up RTView, registered a license for the data server, configured the Data Server, and (optionally) set up the RTVAgent Servlet.

1. Verify System Requirements
   - Application server with a JSP servlet container, such as Apache Tomcat.
2. Install and Setup RTVPost Servlet

   The Data Server uses the RTVPost Servlet that runs on your application server as a proxy servlet for use with the HTTP Data Source, which is useful for cases where an application cannot post directly to the application running the HTTP Data Source due to security or post access limitations. The **servlets\rtvpost** directory contains all of the files necessary to configure and install the RTVPost Servlet.

   If you are using multiple Data Servers that are gathering HTTP data, you must configure and install a RTVPost Servlet for each Data Server.

   a. Configure RTVPost Servlet Options

   The RTVPost Servlet reads the **rtvpost.properties** properties file to get configuration information. If you have not installed the RTVPost Servlet, modify **rtvpost.properties** (in the **servlets\rtvpost** directory) and install (see "b. Install the RTVPost Servlet"). If you have already installed the RTVPost Servlet on your application server, you can edit your properties file in the application server. **Note:** You may need to restart your application server after making changes to your properties file.

   You can set the following options in **rtvpost.properties**:

Option	Description
proxyHost	The name of the host on which the Data Server is running. Default is **localhost**. **Note:** Default **localhost** assumes the servlet and Data Server are running on the same machine.

proxyPort	Port for communicating with the Data Server. Default is **3275**.
proxyPath	The path to which to post. This is commented out by default.

The following is an example of the **rtvpost.properties** file:

```
# Properties for the RTView rtvpost servlet.
# proxyHost: the http host to which to post
proxyHost=localhost
# proxyHost: the http port to which to post
proxyPort=3275
# the path to which to post
#proxyPath=
```

b. Install the RTVPost Servlet

- In an initialized command window, go to the **servlets\rtvpost** directory and type:

make_war

This script creates a web archive (.war) named **rtvpost.war** that includes all of the files necessary to run the RTVPost Servlet.

- Install the files in the **rtvpost.war** file to your application server according to the documentation for that product.

Step 4 is completed. Go to Step 5: Run Display Server.

Step 5: Run Display Server

At this point you have set up the Display Server, Display Servlet, (optionally) the Display Server Portlet, configured the Data Server, (optionally) set up the RTVAgent Servlet, and (optionally) set up the RTVPost Servlet.

A: Start Display Server

1. Start the Display Server. Open an initialized command window (see "Initializing a Command Prompt or Terminal Window") and go to the directory created in Step 1 and type:

   run_displayserver -dataserver:remote://ipaddress:3278

Java options specified in "RTV_JAVAOPTS" are used by the **run_displayserver** scripts.

See "Command Line" for available command line options.

Note: The Display Server is instrumented with JMX to allow you to manage and monitor the display cache and application settings. See "Managing the Display Server Using JMX" for more information.

Step 5 is completed. Go to Step 6: Start/Run Data Server.

Step 6: Start / Run Data Server

At this point you have setup the Display Server, Display Servlet, (optionally) the Display Server Portlet, configured the Data Server, (optionally) setup the RTVAgent Servlet, (optionally) set up the RTVPost Servlet, and run the Display Server.

If you are using multiple or high availability Data Servers, you will need to repeat the following steps on each system where you will run Data Server(s). See "Data Server Tab" and "High Availability" for more information.

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the directory you created in Step 1 and type:

   run_dataserver

2. Click **Start Serving Data**.

Note: You may also run the Data Server as a daemon process. See "Running the Data Server" for more information and additional options.

Note: The Data Server is instrumented with JMX to allow you to manage and monitor the clients and application settings. See "Managing the Display Server Using JMX" for more information.

You have finished Step 6. Go to Step 7: Test Client.

Step 7: Test Client

At this point you have setup the Display Server, Display Servlet, (optionally) the Display Server Portlet, configured the Data Server, (optionally) setup the RTVAgent Servlet, (optionally) set up the RTVPost Servlet, and run the Display Server and Data Server.

1. Open a browser and navigate to the URL for the Display Servlet you set up in Step 2. For example:

http://host:8080/rtvdisplay/index.html	Where host and port are correct for the application server hosting the Display Servlet.

2. Login to the Display Server. By default, the Display Server does not require a login. "Login" can be enabled in the Display Servlet to support "Role-based Security", If login is enabled, the default user name and password are:

   User Name: admin

   Password: admin

Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role. See "Role-based Security" for more information.

Problem: An error message appears in the browser.

Solution: Check the command window where you started the Display Server as well as the Display Servlet log file on your application server.

See the "Browser Client" and "Limitations" for more information.

Congratulations! RTView deployment is completed.

Thin Client Browser with Direct Data - Manual Deployment Process

The following is an overview of how to manually deploy the Thin Client Browser with Direct Data option of RTView. The steps need to be followed in the order given. Alternatively, you can use the "Deployment Wizard" to help you deploy your project.

See "How It Works" for a Thin Client Browser with Direct Data system overview and sequence of usage.

Note: This documentation is intended for users with a working knowledge of HTML code and application server administration. If you do not have a full understanding of these topics, you will need assistance from your system administrator.

Process Summary

Thin Client Browser with Direct Data - Manual Setup

This section provides step-by-step instructions on how to manually deploy RTView. The steps must be done in the order given. Refer to "Thin Client Browser with Direct Data - Manual Deployment Process" for a summary of these instructions. Alternatively, you can use the "Deployment Wizard" to help you deploy your project.

This section is intended for users with standard working knowledge of HTML, JSP and servlet deployment on an application server.

An Apache Tomcat application server is included with your RTView installation for prototyping and testing your deployment before going into the production environment. The following instructions will work for your application server or the one that comes with RTView.

Step 1: Install and Configure Display Server

Install RTView on the system where you will run the Display Server.

A: Verify System Requirements

- Basic system requirements
- In addition to basic system requirements, refer to the Data Sources section of this documentation for system requirements and setup specific to your data source.

B: Install Display Server

At this point you have verified your system requirements.

1. Install RTView (see "Installation")
2. Setup RTView (see "Setup")

C: Register

At this point you have verified your system requirements and installed the Display Server.

Register for a license (see "Registration") to run the Display Server.

D: Configure Display Server

At this point you have verified your system requirements, and installed and registered RTView.

1. Create a project directory to store Display Server configuration files.
2. Copy the following files into this directory from the project directory where you developed your RTView application:
   - All display (***.rtv**) files
   - Style sheet (**.rts**) files (optional)
   - **OPTIONS.ini.** See "Application Options" for more information.
   - **COLORS.ini** (only required if Custom Colors have been defined). See "Custom Colors Tab" for more information.
   - Refer to Deployment in the Data Sources section of this documentation for information on configuration (**.ini**) files specific to your data sources.
   - Panel configuration file (optional). See "Multiple Display Panels" for more information.
   - Security Configuration files (optional). See "Configuration" for more information.
   - **DISPLAYSERVER.ini**

Note: If you have no **DISPLAYSERVER.ini** file, create one now by configuring your Display Server application options. Otherwise, go to "Step 2: Configure and Install Display Servlet".

---

3. Configure your Display Server application options. See "Display Server Configuration" for information on how to specify your options. All configuration files should be saved in the directory you just created.

Step 1 is completed. Go to Step 2: Configure and Install Display Servlet.

Step 2: Configure and Install Display Servlet

At this point you have completed the RTView installation and setup.

About Display Servlet

The Display Server uses the Display Servlet, which is a JSP servlet that runs on your application server. Clients communicate with the Display Servlet using HTTP. The Display Servlet communicates with the Display Server via socket to request HTML for display in the browser. The **servlets\rtvdisplay** directory contains files (JSP, HTML, classes, properties) necessary to install the Display Servlet.

A: Create Display Servlet HTML or JSP Files

The Display Servlet comes with a few HTML files for testing. You can use these to deploy, or you can create your own HTML or JSP files which make calls to the Display Servlet to show your displays. See "Creating Display Servlet HTML Files" and "Creating Display Servlet JSP Files" for more information.

Skip this step and go to Step B: Configure Display Servlet Options if both of the following are true:

- You will only be deploying one Display Server application on your application server
- You want to deploy using only the test HTML files that come with the Display Servlet

---

Note: If you skip this step, use **servlets\rtvdisplay** as your project directory referenced in Steps B and C, and use **rtvdisplay** for the **appname** argument for all of the scripts.

Otherwise, proceed with the following steps.

1. 1. Create a project directory to store your Display Servlet files.
2. Copy rtvdisplay.properties into this directory from **servlets\rtvdisplay**.
3. Create the HTML files or JSP files for your Display Server application.
4. Copy these HTML or JSP files and any referenced files (e.g., images referenced in the HTML files) to this directory.
5. Copy **sample_make_war.bat** (for Windows) or **sample_make_war.sh** (for UNIX) into this directory from **servlets\rtvdisplay** and rename it to **make_war** (with the appropriate extension).
6. This script takes a name for the web archive, without the **.war** extension, and builds a web archive file that includes all of the necessary RTView Display Servlet files along with all **.html**, **.js**, **.gif**, and **.jpg** files from the current directory. If there are other files you would like to include in your web archive, add them to the following line in your **make_war** script:

```
jar uf %1.war *.html *.jpg *.gif *.js *.jsp WEB-INF
```

Note: You may receive an error message if the script does not find at least one of each of the file types specified (**.html**, **.jpg**, etc). Disregard this error message.

B: Configure Display Servlet Options

The Display Servlet reads **rtvdisplay.properties** to get configuration information. If you have not installed the Display Servlet, modify the rtvdisplay.properties files in your project directory and it will be installed as part of the next step. If you have already installed the Display Servlet on your application server, you can edit this properties file in your application server. You may need to restart your application server after making changes to this file.

Use **servlets\rtvdisplay** as your project directory if you skipped Step A: Create Display Servlet HTML Files.

Set the your options in **rtvdisplay.properties** as specified in "Display Servlet Configuration".

C: Install Display Servlet

At this point you have completed Display Servlet setup.

If you skipped Step A: Create Display Servlet HTML Files, use **servlets\rtvdisplay** as your project directory and **rtvdisplay** as your **appname** in this step.

Otherwise, use the project directory that you setup in Step A and your web archive file name for **appname**.

Note: If you will be running multiple Display Server applications on the same application server, each application must have a unique name.

---

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to your project directory and type:

make_war appname	This script creates a web archive (**.war**) that includes all of the files necessary to run the Display Servlet.

2. Your RTView installation includes an Apache Tomcat application server so that you can prototype and test your deployment before moving it to your production server. If you will be using this application server, run the following script to install the Display Servlet:

install_to_demoserver appname	This script installs the web archive to the Apache Tomcat server included in your RTView installation. **Note:** This script will shutdown and restart Apache Tomcat and requires administrative permissions.

3. If you will be using your own Apache Tomcat application server, run the following script to install the Display Servlet:

install_to_tomcat appname	This script installs the web archive to your Apache Tomcat server. **Note:** This script will shutdown and restart Apache Tomcat and requires administrative permissions.

4. If you will be using an application server other than Apache Tomcat, install the files in the web archive to your application server according to the documentation for that product.

Step 2 is completed. Go to Step 3: Configure and Install Display Server Portlet (Optional).

Step 3: Configure and Install Display Server Portlet (Optional)

At this point you have setup the Display Server and the Display Servlet.

The Display Server can optionally be deployed as a portlet. The portlet is tested in Liferay and contains configuration files specific to Liferay, but should work with any JSR-168 compliant portal. If you want to deploy the Display Server as a servlet, skip this step and go to Step 4: Run Display Server.

Configure install and instance the Display Server Portlet as described in "Configure and Install Display Server Portlet (Optional)".

Step 3 is completed. Go to Step 4: Run Display Server.

Step 4: Run Display Server

At this point you have setup the Display Server and the Display Servlet, and (optionally) setup the Display Server Portlet.

A: Start Display Server

1. Start the Display Server. Open an initialized command window (see "Initializing a Command Prompt or Terminal Window") and go to the directory created in Step 1 and type:

   run_displayserver

Java options specified in "RTV_JAVAOPTS" are used by the **run_displayserver** scripts.

See "Command Line" for available command line options.

Note: The Display Server is instrumented with JMX to allow you to manage and monitor the display cache and application settings. See "Managing the Display Server Using JMX" for more information.

Step 4 is completed. Go to Step 5: Test Client.

Step 5: Test Client

At this point you have setup and started the Display Server, and setup and installed the Display Servlet.

1. Open a browser and navigate to the URL for the Display Servlet you set up in Step 2. For example:

http://host:8080/rtvdisplay/index.html	Where host and port are correct for the application server hosting the Display Servlet.

2. Login to the Display Server. By default, the Display Server does not require a login. Login can be enabled in the Display Servlet to support "Role-based Security". If login is enabled, the default user name and password are:

   User Name: admin

Password: admin

Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role. See "Role-based Security" for more information.

Problem: An error message appears in the browser.

Solution: Check the command window where you started the Display Server as well as the Display Servlet log file on your application server.

See "Browser Client" and "Limitations" for more information.

Congratulations! RTView deployment is completed.

Display Server

The Thin Client deployment is implemented using the Display Server. The Display Server consists of two parts, the Display Server application and the Display Servlet. The Display Server application is generally installed on a dedicated platform. It runs the RTView engine, which loads displays and gathers data that it passes on to the Display Servlet via a socket. The Display Server can either access data directly or it can connect to the Data Server for data. The Display Servlet is a JSP servlet that runs on an application server. Clients communicate with the Display Servlet using HTTP, so the only system requirement for the client is a standard browser.

This document describes the system requirements, configuration, installation and runtime information for the "Display Server Application", "Display Servlet", and the "Browser Client". Some objects and interactions behave differently in the Display Server than they do in the other RTView deployments. See the "Runtime Functionality" and "Limitations" sections below for more information.

Display Server Application

System Requirements

The system where you will run the Display Server application must meet the following system requirements:

- Basic "System Requirements"
- In addition to basic system requirements, if you will not be using the Data Server, refer to the "RTView Data Sources" section of this documentation for system requirements and setup specific to your data source.

Display Server Configuration

Create **DISPLAYSERVER.ini** in your project directory with the following options:

Option	Description
cellsperexport	Specify to limit the number of table cells included in HTML/Excel exports requested by the Thin Client. This option avoids out-of-memory exceptions and timeouts when exporting tables with many rows. This option is typically used in conjunction with the **cellsperpage** option, and has the following behavior: • If the **cellsperexport** option is not specified, or if a value of less than a 1000 is specified, the Display Server attempts to export all rows for all tables, regardless of the table size. • If a table contains fewer cells than the **cellsperexport** setting, the Display Server exports all rows for that table. • The exported HTM/Excel table starts with the same first row (or near it) that is visible in the Thin Client. That is, if you scroll to row 900 in the Thin Client and perform an HTML/Excel export, the exported table will begin near line 900. • If the rows included in an export to HTML/Excel are limited by the **cellsperexport** option, the first row in the exported file is the same as the first row currently visible in the Thin Client. • An export to HTML/Excel requires less CPU and memory than an export to PDF (see **cellsperreport**), therefore the value of the **cellsperexport** option is typically larger than the value of the **cellsperreport** option. For example, if an export to HTML/Excel was performed on a table with 5 columns and 100,000 rows, and the option **cellsperexport 30000** was specified when the Display Server was launched, 6000 rows (30000/5) would be included in an exported HTML/Excel file for that table. If the **cellsperreport 5000** option was specified, an export to PDF would include 1000 rows from the table. **Note:** This option is not recognized by the Builder or the Viewer. Example: **cellsperpage 10000** **cellsperexport 30000**
cellsperpage	Specify server-side table paging and sorting mode, also referred to as paging mode, for the Thin Client. Paging mode improves the performance of displays containing table objects (obj_table02) with many rows. In paging mode, the Display Server sends a specified maximum number of table data rows at a time to the Thin Client, rather than sending all table data rows at once. This option avoids out-of-memory exceptions, timeouts and sluggish performance that can otherwise occur from processing and transmitting all of the rows at once. This option is typically used in conjunction with the **cellsperexport** and **cellsperreport** options, and can also be specified on the command line. See "Command Line Options: Display Server" for more information. The page of rows sent from the Display Server to the Thin Client includes all of the rows currently visible in the Thin Client plus additional rows above and/or below the visible rows. If the user scrolls beyond the rows contained in the current page or clicks on a column header to change the sorting order, the Display Server sends another page of rows in response. To specify in the **DISPLAYSERVER.ini** file: **cellsperpage NNNN** where **NNNN** specifies the number of table cells the Display Server includes per page. Typical values for **cellsperpage** are 10000 to 30000.

The number of cells in a table is equal to the number of rows multiplied by the number of columns. For example, if **cellsperpage** = 10000, and the display contains a table with 5 columns, the Display Server uses a page size of 2000 rows for the table. This means the Display Server sends a maximum of 2000 table rows to the Thin Client at a time. If the table contains 40,000 rows the Display Server sends rows 1 through 2000 to the Thin Client when a user opens the display. If the user then scrolls to the bottom of the table, the Display Server sends rows 38,001 through 40,000 to the Thin Client. Similarly, if the user clicks on a column header to sort by that column, the Display Server sorts the table and sends the first 2000 sorted rows to the Thin Client. After the user scrolls or sorts a table in paging mode, the Thin Client displays '**...**' in each cell and an hourglass cursor appears over the table while it waits to receive the new page from the Display Server. The **cellsperpage** option also has the following behavior:

- A smaller value for **cellsperpage** reduces the memory requirements for processing large tables in the Display Server, Application Server, and browser. A larger value smooths scrolling in the Thin Client because it increases the number of rows through which the Thin Client can scroll before it needs to request another page from the Display Server.
- If the **cellsperpage** option is not specified, or if a value less than a 1000 is specified, paging mode is disabled and the Display Server sends all data rows to the Thin Client for all tables regardless of the table size.
- If a table contains fewer cells than the **cellsperpage** setting, the Display Server sends all rows for that table.
- Paging mode only affects the behavior of obj_table02 objects, and only in the Thin Client.
- The **maxNumberOfHistoryRows** property on obj_table02 still limits the maximum number of rows that are shown in the table, regardless of the **cellsperpage** setting.
- After scrolling to a new page, if the user immediately drags the scrollbar again, the table may not react. The user may need to nudge the scrollbar knob or click on the scroll up/down arrows to make the table react.
- The Thin Client Export Table to Excel/HTML/PDF feature is not affected by paging mode. Attempts to export a table with many rows may result in timeouts or out-of-memory exceptions in the Display Server or in the Display Servlet. To avoid those errors, see the **cellsperexport** and **cellsperreport** Display Server options.
- When displaying a table with more than 75,000 rows in Internet Explorer version 8, the last row in the table may be partially obscured even if the scrollbar knob is dragged to the bottom position. The last row can be made visible by using the mouse wheel but that may cause unused space to appear at the bottom of the table.
- After the user clicks a column to sort a table, the table vertical scrollbar is reset to the top position.
- The Thin Client ignores the **scrollToSelectionFlag** property on a table that is in paging mode.
- A table row selection is cleared when a different page of rows is received from the Display Server. If a table is in paging mode, only rows that are on the current page can be selected, even if the table **multiSelectFlag** property is checked.

cellsperreport	Specify to limit the number of table cells included in PDF exports requested by the Thin Client. This option avoids out-of-memory exceptions and timeouts when exporting tables with many rows. This option is typically used in conjunction with the **cellsperpage** option, and has the following behavior: • If the **cellsperexport** option is not specified, or if a value of less than a 1000 is specified, the Display Server attempts to export all rows for all tables, regardless of the table size. • If a table contains fewer cells than the **cellsperreport** setting, the Display Server exports all rows for that table. • In an exported PDF file, the scroll position of the Thin Client has no effect on the starting row in the PDF file, nor any effect on the rows that are included in the PDF report. • An export to PDF requires more CPU and memory than an export to HTML/Excel (see **cellsperexport**), therefore the value of the **cellsperreport** option is typically smaller than the value of the **cellsperexport** option. For example, if an export to HTML/Excel was performed on a table with 5 columns and 100,000 rows, and the option c**ellsperexport 30000** was specified when the Display Server was launched, 6000 rows (30000/5) would be included in an exported HTML/Excel file for that table. If the **cellsperreport 5000** option was specified, an export to PDF would include 1000 rows from the table. **Note:** This option is not recognized by the Builder or the Viewer. Example: **cellsperpage 10000** **cellsperreport 5000**
clearsubs	Set to false to preserve the value of local variables, for which the Use as Substitution option is selected, when navigating to different displays in the same panel. This behavior is consistent with that of the Display Viewer. By default (or if set to true), the Display Server will reset all local variables to their initial values when navigating to a new display regardless of the variable's value in the current display. **Note:** Global variables are not affected by this option.
display_timeout	Amount of time, in seconds, that a display can be kept in memory after the Display Servlet has stopped requesting it. Default is 60 seconds (to allow faster load time when switching between displays).
imageformat	Specify image format: jpg or png. If **imageformat** is not specified, the Display Server will automatically select the image that results in the fewer number of bytes for each display. By default the png format will be selected for most displays, but the jpg format is preferred for: • a background image in the main display • background images in graphs or labels • images with color gradients (especially those with lower imagequality values).
imagequality	Specify a value between 0 and 100 to control the quality of .jpg images. If the value is 100, the Display Server will output the highest quality image with the lowest compression. If the value is 0, the Display Server will output the lowest quality image using the highest compression. Default is 75. If the -verbose option is specified at startup, image creation time/size for each display refresh will be shown in the Display Server console output. **Note:** The Display Server chooses an image format (.jpg or .png) for each display, depending on which format produces the smallest image size (in bytes). The size of .png images is controlled by the **pngcompress** option.

history_config	To preload a display, making data immediately available. Preloaded displays are not unloaded unless the Display Server is restarted or the display cache is cleared via JMX (see "Managing the Display Server Using JMX"). This option may be used multiple times to specify multiple displays to preload: **history_config rtvFileName substitution** Substitutions are optional and must use the following syntax: **$subname:subvalue $subname2:subvalue2** If a substitution value contains a single quote, it must be escaped using a **/** : **$filter:Plant=/'Dallas/'** If a substitution value contains a space, it must be enclosed in single quotes. Do not escape these single quotes: **$subname:subvalue $subname2:'sub value 2'** A substitution string cannot contain the following: : | . tab space , ; = < > ' " & / \ { } [ ] ( )
max_displays	Number of displays kept in memory. Default is **20** (to optimize memory used by the Display Server).
passclientlogin	If true, pass RTView login information into all data sources that have the Use Client Credentials option enabled. **Note:** Some data sources do not support this feature. For information on Application Options for your data source, refer to the Data Sources section of this documentation.

permitpanel	Specifies the panel layout files that the server will read.
	A panel layout for the thin client is requested from the display server with a URL parameter as follows:
	`panels.jsp?file=X`
	where **X** is the name of the panel layout file that the server should read. By default, the display server will attempt to read any filename on the server that is specified by the URL parameter.
	The **permitpanel** option allows you to specify the file(s) that the display server will read in response to a **panels.jsp** request. Requests from **panels.jsp** for any other files are rejected with a **Permission denied** error shown in the browser, regardless of whether the file exists or not, and the server will not attempt to read such files. The option may be specified multiple times to allow access to multiple panel files.
	Command-line example:
	`run_displayserver -permitpanel:PANELS.ini -permitpanel:layout.xml`
	DISPLAYSERVER.ini example:
	`permitpanel PANELS.ini`
	`permitpanel layout.xml`

permitfile	This option prevents attempts to load remote files, as follows:
	`-permitfile:LOCAL_ONLY`
	If this option is specified, any **rtv** or image files that are referenced by URL will not be read and the server will log a message similar to the following:
	`non-local file read permission denied: http://host/somefile`

pngcompress	Specify a value between 1 and 9 to control the quality of .png images. If the value is 1, the Display Server will output the highest quality image with the lowest compression. If the value is 9, the Display Server will output the lowest quality image using the highest compression. Default is **9**. A value of **3** is a good compromise between speed and quality. If the **-verbose** option is specified at startup, image creation time/size for each display refresh will be shown in the Display Server console output. **Note:** The Display Server chooses an image format (.jpg or .png) for each display, depending on which format produces the smallest image size (in bytes). The size of .jpg images is controlled by the **imagequality** option.

port	Port that the Display Server uses to communicate with the Display Servlet. Default is **3279**. **Note:** If you will be running multiple Display Server applications on the same application server, you must specify a unique port for each application.
shared_display	Enables Display Sharing for a display. Add the name of the display (**.rtv**) file and the substitutions, if any, to be applied when the display is opened: **shared_display <filename> [substitutions]** Where: **<filename>** is the name of a display (**.rtv**) file to share. **[substitutions]** is an optional list of substitution **string:value** pairs, separated by spaces. If a substitution value contains spaces it must be enclosed in single quotes. Example: **shared_display navtop.rtv** **shared_display summary.rtv $region:East** See "Overview - Display Sharing" for more information.

Example of the above options set in the **DISPLAYSERVER.ini** file:

```
cellsperpage 10000
cellsperexport 30000
cellsperreport 5000
clearsubs false
display_timeout 60
imagequality 75
history_config trend1.rtv
max_displays 20
port 3279
passclientlogin false
shared_display navtop.rtv
shared_display summary.rtv $region:East
```

RTView offers role based security that allows you to limit access to displays based on the user's role. User and role definitions also support substitutions. When role based security is enabled, users are prompted to login. The default user name and password are:

User Name: admin

Password: admin

Note: To use this feature, you must also enable it in the Display Servlet by specifying **LoginEnabled** in **rtvdisplay.properties**.

To use this security feature, do the following:

1. Define the users that can access RTView, assign each user a password and one or more associated roles. If your user definitions are in an XML file, copy this file to your project directory.

2. Set "Role Definitions" to specify which displays users can access. If your role definitions are in an XML file, copy this file to your project directory.

3. Enable the **Use Client Credentials** option if you want to pass RTView login user name and password to database connections. **Note:** Some data sources do not support this feature. For information on Application Options for your data source, refer to the "RTView Data Sources" section of this documentation.

Use the **-passclientlogin** command line option or the **-passclientlogin** option (see Step 1D-3) in the **DISPLAYSERVER.ini** configuration file to pass RTView login into these database connections. See "Command Line Options: Display Server" for more information.

See "Login", "Roles and Displays", and "Configuration" for more information.

Running the Display Server

You can run the Display Server as an application or in the background as a Windows Service. See "Running as a Windows Service" for more information.

To run the Display Server as an application: open an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to your project directory, and type:

run_displayserver

Several command line options are supported for the Display Server. Java options specified in "RTV_JAVAOPTS" are used by the **run_displayserver** scripts. See "Command Line Options: Display Server" for more information.

Note: The Display Server is instrumented with JMX to allow you to manage and monitor the display cache and application settings. See "Managing the Display Server Using JMX" for more information.

High Availability Display Servers

It is possible to designate one or more backup Display Servers to take over during a failover event when the Display Servlet discovers that the current Display Server is no longer accessible. Backup servers may be run in hot standby mode (default) or warm standby mode (designated as a command line option). In hot standby mode all global variable definitions, as well as cache and alert definitions, are loaded and activated at start up. In warm standby mode none of these actions are performed, thereby avoiding the overhead of maintaining Alert and Cache data sources until the backup server has become the primary. See "Command Line Options: Display Server" for more information.

You can specify a primary Display Server, as well as one or more backup Display Servers, in the **rtvdisplay.properties** file.

At startup, the Display Servlet will connect to the primary server if available, otherwise it will connect to the backup server. If neither is available, the Display Servlet will periodically retry connecting to both. If the Display Servlet connects to either the primary or the backup, and later the connection is lost, it will then try to connect to the other server. If a Display Servlet connects to a backup server and later the primary server becomes available, the Display Servlet will not switch to the primary unless the connection to the backup server is lost. See "Display Servlet Configuration" for more information.

If the Display Server is using the Data Server to connect to data, primary and backup Data Servers can also be specified. Refer to "Running the Data Server" for more information.

Display Servlet

The Display Server uses the Display Servlet, a JSP servlet that runs on your application server. Clients communicate with the Display Servlet using HTTP. The Display Servlet communicates with the Display Server via socket to request HTML for display in the browser. The **servlets\rtvdisplay** directory contains files (JSP, HTML, classes, properties) necessary to install the Display Servlet.

This section is intended for users with standard working knowledge of HTML, JSP and servlet deployment on an application server.

System Requirements

Display Servlet requires an application server with a servlet container such as Apache Tomcat. An Apache Tomcat application server is included with your RTView installation for prototyping and testing your deployment before going into the production environment. The following instructions will work for your application server or the one that comes with RTView. See

Display Servlet Configuration

The Display Servlet reads **rtvdisplay.properties** and **rtvdisplay_strings.properties** to get configuration information. See "Localize Display Server" for more information.

Note: The **rtvdisplay_strings.properties** file provides the ability to localize messages that appear in the browser.

The **rtvdisplay.properties** file is located in **servlets\rtvdisplay** with the default settings. If you do not want to use the default settings, copy this file to your project directory and modify.

Set the following options in **rtvdisplay.properties**:

Option	Description
DisplayServerBackup	The name of the host and port for designated backup servers. Designation of backup servers is optional; multiple backup servers should be separated by a comma.
DisplayServerConnection Failback	When set to true, the rtvdisplay servlet attempts to reconnect to its primary Display Server, approximately every 30 seconds, while connected to its backup Display Server. When the connection to the primary server is restored the servlet disconnects from the backup server and resumes using the primary server. By default, this option is false. To enable this behavior add **DisplayServerConnectionFailback=true**, then rebuild and deploy the **rtvdisplay.war** file.
DisplayServerConnection Pool	Specifies the maximum number of simultaneous connections the Display Servlet makes to the Display Server, and, consequently determines the maximum number of display requests the Display Server processes simultaneously. The default value is **10**. This option improves response times on multi-CPU systems by enabling the Display Servlet to manage a pool of threaded requests to the Display Server. By default, the Display Servlet opens to 10 simultaneous connections to the Display Server, allowing it to process up to 10 display requests simultaneously.
DisplayServerHost	The name of the host on which the Display Server is running. Default is localhost. **Note:** Default localhost assumes the Display Server and Display Servlet are running on the same application server.
DisplayServerPort	Port for communicating with the Display Server. Default is **3279**. This must be the same port specified in your **DISPLAYSERVER.ini** file. **Note:** If you will be running multiple Display Server applications on the same application server, you must specify a unique port for each application.
DisplayServerTimeout	Amount of time (in seconds) the Display Servlet waits for replies from the Display Server. Default is **15** seconds.

DefaultRefreshInterval	Rate (in seconds) at which displays are refreshed. Default is **15** seconds. Set this interval to **0** if you do not want displays automatically refreshed. **Note:** The refresh rate specified in the **rtvRefreshInterval** property or the **URL** or **DIV** tag used to open a display will override this value. See "Background Properties", "Creating Display Servlet HTML Files", and "Creating Display Servlet JSP Files" for more information.
LoginEnabled	Set to **true** to enable role management security. If enabled, a login dialog will appear when the client browses to the URL for the Display Servlet, unless single sign-on is configured. The default is **false**.
MaxSharedDisplays	Used to configure Display Sharing, specifies the maximum number of shared displays for which the Display Servlet stores content. The default is **10**. (This property does not need to be included in the **rtvdisplay.properties** file if the default setting is used). See "Overview - Display Sharing" for more information. **Note:** Best practices dictate that the **MaxSharedDisplays** property be set to a value at least as large as the number of **shared_display** entries in **DISPLAYSERVER.ini**.
MenuItemsToHide	Allows you to remove items from the thin client context menu by specifying the names of the items in **rtvdisplay.properties**. Multiple names are separated by commas. For example, the following will remove the "Export Table To HTML", "Export Table to Excel", "Export PDF", and "Status" items from the menu: `MenuItemsToHide=PDF,ExcelTable,HtmlTable,Stat` After adding this property, the corresponding .war file must be rebuilt and redeployed. Here are the names of all of the menu items, with the English label string of the menu item in quotes if it differs from the item name: Refresh Back Next Command "Execute Command" Drill "Drill Down" ExcelTable "Export Table to Excel" (Internet Explorer only) HtmlTable "Export Table to HTML" PDF "Export PDF" CopyCell "Copy Cell Value" Stat "Status" Logoff "Log Off" Use the menu item name when configuring MenuItemsToHide, not the label string.
SharedDisplayRefreshInterval	Used to configure Display Sharing, specifies the time interval at which the Display Servlet updates the content it uses for a shared display. The default is **15** seconds. See "Overview - Display Sharing" for more information.
VirtualThreshold	Set the cell count of a table at which point Virtual Mode is used. Default is **500**. For smooth scrolling, data is preloaded into tables with less than 500 cells. For tables with more than 500 cells, Virtual Mode is used in which data is paged into the tables to speed processing time for displaying, refreshing and sorting. However, the table may blink while scrolling in this mode.

Example of the **rtvdisplay.properties** file:

```
DisplayServerHost=localhost
DisplayServerConnectionPool=10
DisplayServerPort=3279
```

```
DisplayServerTimeout=15
DefaultRefreshInterval  =15
LoginEnabled=true
MaxSharedDisplays=10
SharedDisplayRefreshInterval=15
VirtualThreshold=500
 DisplayServerBackup=host:3238
DisplayServerConnectionFailback=true
```

Localize Display Server

The **rtvdisplay_strings.properties** file contains settings for text displayed by the thin client (i.e. popup menus, login window, status window, error messages, etc.). This file is located in the **WEB-INF/classes/gmsjsp** directory of the **rtvdisplay.war** file. For localization, extract this file from **rtvdisplay.war** and modify. Make a copy of this file with the desired locale suffix (e.g. **rtvdisplay_strings_ja.properties** for Japanese) and pack into **rtvdisplay.war**, in **WEB-INF/classes/gmsjsp**.

The locale setting of the application server (e.g. tomcat) will determine which **.properties** file to load. If the app server does not have the desired locale setting for the thin client, then edit the original file (**rtvdisplay_strings.properties**) instead.

Create Custom Servlet Files

The Display Servlet comes with a few HTML files for testing. You can use these to deploy, or you can create your own HTML files or JSP files that make calls to the Display Servlet to show your displays. All custom HTML or JSP files should be saved in your project directory or a subdirectory. See "Creating Display Servlet HTML Files", "Creating Display Servlet JSP Files", and "Test Displays" for more information.

Create War File

If you will not be using the Deployment Wizard to generate your deployment, you will need to create a **.war** file that contains **rtvdisplay.properties**, **rtvdisplay_strings.properties** and any custom servlet files:

1. Copy **sample_make_war.bat** (for Windows) or **sample_make_war.sh** (for UNIX) into this directory from **servlets\rtvdisplay** and rename it to **make_war** (with the appropriate extension).

2. This script takes a name for the web archive, without the **.war** extension, and builds a web archive file that includes all of the necessary RTView Display Servlet files along with all .html, .js, .gif and .jpg files from the current directory. If there are other files you would like to include in your web archive, add them to the following line in your **make_war** script:

   ```
   jar uf %1.war *.html *.jpg *.gif *.js *.jsp WEB-INF
   ```

Note: You may receive an error message if the script does not find at least one of each of the file types specified (.html, .jpg, etc). Disregard this error message.

---

3. Type **make_war warName** where **warName** is the name of the war file to create.

Deploying War Files

Once you have created a war file for the Display Servlet, you will need to install it to your application server. An Apache Tomcat application server is included with your RTView installation for prototyping and testing your deployment before going into the production environment. You can either use this application server or your own.

If did not create a war file, use **servlets\rtvdisplay** as your project directory and **rtvdisplay** as your **appname** in the steps below.

Otherwise, use your project directory and your web archive file name for **appname**.

Note: If you will be running multiple Display Server applications on the same application server, each Display Servlet war file must have a unique name.

---

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to your project directory and type:

make_war appname	This script creates a web archive (**.war**) that includes all of the files necessary to run the Display Servlet.

2. Your RTView installation includes an Apache Tomcat application server, so that you can prototype and test your deployment before moving it to your production server. If you will be using this application server, run the following script to install the Display Servlet:

install_to_demoserver appname	This script installs the web archive to the Apache Tomcat server included in your RTView installation. **Note:** This script will shutdown and restart Apache Tomcat and requires administrative permissions.

3. If you will be using your own Apache Tomcat application server, run the following script to install the Display Servlet:

install_to_tomcat appname	This script installs the web archive to your Apache Tomcat server. **Note:** This script will shutdown and restart Apache Tomcat and requires administrative permissions.

4. If you will be using an application server other than Apache Tomcat, install the files in the web archive to your application server according to the documentation for that product.

Configure and Install Display Server Portlet (Optional)

The Display Server can optionally be deployed as a portlet. The portlet is tested in Liferay and contains configuration files specific to Liferay, but should work with any JSR-168 compliant portal. If you want to deploy the Display Server as a servlet, skip this step.

Configure Display Server Portlet Options

The **servlets\rtvportlet** directory contains all of the files necessary to install the Display Server portlet. It includes a web.xml and several other configuration files necessary for use in a Liferay portal. Make any necessary changes to these files, or if you are using another JSR-168 compliant portal, copy any required configuration files to this directory. The Display Server portlet requires the Display Servlet to run and will access it at **http://host:port/rtvdisplay**, where host and port are the host and port where the portlet is running. If this is not the case, modify the value for the **rtvServletUrl** parameter in portlet.xml to point to the correct location of the Display Servlet. Rebuild the **rtvportlet.war** file using the **make_war.bat** script.

Install Display Server Portlet

Install **servlets\rtvportlet\rtvporlet.war** to your portal according to the documentation for that product.

Instance the Display Server Portlet in Your Portal

Once the Display Server portlet is installed to your portal, you can instance it according to the documentation for that product. The Display Server portlet supports edit mode and view mode. In edit mode, you can specify the display, refresh rate, window name, width and height for the portlet.

Browser Client

System Requirements

The client requires a standard browser.

Accessing the Project

1. Open a browser and navigate to the URL for the Display Servlet. For example:

http://host:8080/rtvdisplay/index.html	Where host and port are correct for the application server hosting the Display Servlet.

2. Login to the Display Server. By default, the Display Server does not require a login. Login can be enabled in the Display Servlet to support "Role-based Security", If login is enabled, the default user name and password are:

   User Name: admin

   Password: admin

Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role. See "Role-based Security" for more information.

Problem: An error message appears in the browser.

Solution: Check the command window where you started the Display Server as well as the Display Servlet log file on your application server.

Runtime Functionality

Option	Description
Command Execution	Depending on the command, it executes either on the Display Server or the client. See "Define/Execute Command" for further information. Some commands are not supported on the Display Server. See Display Server "Limitations" (below) for information.
Control Objects	HTML control objects are generated for the following: • Combo Box • Check Box • Text Entry • List Box • Push Button • Slider
Display Refresh	Right-click in the page and select Refresh to refresh the display. **Note:** The web browser's Refresh button resets the display to its original state if you have updated variables or substitutions using an object in the display.
Drill Down	Drill down functionality is supported using hyperlinks. Single-click to activate a drill down display.
LoginEnabled	Set to **true** to enable role management security. If enabled, a login dialog will appear when the client browses to the URL for the Display Servlet, unless "Display Server Single Sign-On" is configured. The default is **false**.
Popup Menu Interaction	1. Right-click in the page to open a Display Server popup menu. 2. Hold down the <Shift> key while you right-click to bring up the web browser's popup menu. 3. Logout by right-clicking and selecting **Log Off** before closing the web browser. Some web browsers may not logout the user after the browser window has been closed.
Resize Mode	To globally specify a Window Resize Mode, select **Tools>Options>**"General Tab" or set **-resizemode** on the command line. It is also possible to set a specific Resize Mode for each particular display (.rtv) file using the "Background Properties" dialog. The default **Resize Mode** for the Thin Client is **Crop**. Runtime behavior in the Thin Client differs in the following cases: • When the initial display is opened, the browser frame is not resized to match the display size as it is in the Display Viewer. Instead, in the default **Crop** mode, the display will appear in its full size. Additionally if the browser frame is larger than the display, then unused space will appear below and to the right of the display. If the browser frame is smaller than the display, then scrollbars will appear. In Layout and Scale modes, the display will briefly appear at its default size and will then resize to fit the browser frame size. This may also occur if another tab is opened in the browser, the browser is resized, and then the browser tab that contains the Thin Client is reselected.

	• In Layout and Scale modes, after resizing the browser frame, table objects will revert to their original states. For example if you clicked on a column header in a table to sort the column, after resize the table would revert to its default sort. Likewise if you scrolled in a table or resized the legend, after resize the scrollbars and legend will revert to their initial position and size. • In Scale mode, there will be unused space in the browser frame. This is due to the fact that, to ensure equal scaling in both dimensions, the display will only use the largest 4"x3" rectangular area of the frame. The unused area will have the same color as the display background, but will not have a gradient fill.
Status	From the Display Server popup menu, select **Status** to access the following information: • Name of the current RTView display • Time the display was last refreshed • URL and connection status for the rtvdisplay servlet • Hostname and port used by the servlet to connect to the Display Server
Table Objects	The Display Server generates JavaScript to display table objects (obj_table02). The JavaScript table allows scrolling, sorting and column resizing. Right-click on a table and select **Export to Excel** to open Excel in a web browser, or **Export to HTML** to export to an HTML file with the data from the selected table.
VirtualThreshold	Set the cell count of a table at which point **Virtual Mode** is used. Default is **500**. For smooth scrolling, data is preloaded into tables with less than 500 cells. For tables with more than 500 cells, **Virtual Mode** is used in which data is paged into the tables to speed processing time for displaying, refreshing and sorting. However, the table may blink while scrolling in this mode.

Limitations

- "Threshold Commands" are not supported in the Display Server, as well as the following "Commands":
  - Beep
  - Play Audio File
  - Run DOS Command or UNIX Shell
- The **commandCloseWindowOnSuccess** property is not supported in the Display Server.
- The Display Server does not support command failure notification on the client. Notification only appears as a console message on the server.
- Drill down displays have the following limitations: ◦The Topmost Window Mode has no effect in the Display Server, it is equivalent to the Normal setting.
  - If the user clicks outside of a display from which a drill down in Modal Window Mode was opened, but still within the same browser instance (i.e.: in another frame or iframe), the modal drill down may be obscured. When the mouse is moved across or clicked in the parent display, the modal drill down will return to the top.
  - In Modal Window Mode the user can close the main browser window without first closing the drill down window.
  - If the Window Position specified is completely off-screen, the coordinates will be adjusted so that the window is partially on-screen.

- If the Window Position is set to Center of Screen or Center of Parent the window open slightly lower than expected.
- In Firefox, when drill down window is closed the next mouse click in the parent window may be ignored.

Table Objects

- The Display Server generates JavaScript to display table objects (obj_table02). The JavaScript table has the following limitations: ◦The **scrollbarMode**, **tabIndex**, and **editDataEnabledFlag** properties are ignored. The default values are used.
  - The minimum requirements for viewing in a web browser are Internet Explorer 5.5, Netscape 7.1 and Firefox 1.0.
  - For smooth scrolling, data is preloaded into tables with less than 500 cells. For tables with more than 500 cells, **Virtual Mode** is used in which data is paged into the tables to speed processing time for displaying, refreshing and sorting. However, the table may blink while scrolling in this mode. To modify, use the **VirtualThreshold** property in **rtvdisplay.properties**. See "Display Servlet Configuration" for information.
  - If a sort column is configured in the table object in the Display Builder, the sort icon does not appear in the column header when viewed in the Display Server. The sort icon appears if the sort column is selected at the time the Display Server is started.
- Table object (obj_table02) has the following limitations:
  - When the **multiSelectFlag** property is enabled:

    a. Multiple rows can only be selected using Ctrl+Left click.

    b. A mouse click must be used to select columns.

    c. A right-click does not select a row under the mouse pointer

    d. Once a table is displayed, changes to the value of the **multiSelectFlag** property are ignored.

    e. In Firefox, a double right-click must be used to open popup menus.
  - When the **indexColumns** property is enabled it is ignored. Also, a selection made in the table gets cleared if the number of rows in the table changes after a data update.
  - The **columnResizeEnabledFlag** cannot be toggled.
  - The **rowHeaderEnabledFlag** cannot be toggled.

Graph Objects

- The **scrollbarSize** property is ignored. The system default value is used.

Object Grid

- The **scrollbarSize** property is ignored. The system default value is used.

Control Objects

- Since Firefox does not support multi-line tool tips, any returns (i.e. \n) found in **mouseOverText** entries will be replaced with spaces.
- Invalid numeric characters that are copied and pasted into text entry fields are not rejected.

Slider Control

- The Slider Control (obj_c1scale) is drawn as a scrollbar rather than a slider.
- The **updateWhileAdjustingFlag**, **bgColor**, and **objHeight** properties are not supported.
- The **enabledFlag** property is not supported in Firefox.

Button Control

- In Firefox, the **defaultButtonFlag** is ignored unless a control object in the display has focus.

Combo Box Control

•The **textEditEnabledFlag** property is not supported.

Date Chooser Control

- Uses the **Datejs** date library for parsing and formatting dates and times. The **Datejs** library subclasses the standard javascript Date object. If you embed a display with the Date Chooser control object inside your javascript application, it may pickup the **Datejs** date class instead of the standard date class. The **Datejs** library is distributed under the MIT License and can be downloaded at *http://www.datejs.com*.
- The month and day names can be localized by replacing the **date.js** file in the Display Servlet with a localized version. To do this, go to **servlets\rtvdisplay** in your RTView installation directory. Extract the appropriate date***.js** file for your language from **datejs_loc.zip**. Rename this file to date.js in the **servlets\rtvdisplay** directory and rebuild the Display Servlet **.war** file. **Note:** The default **date.js** is a copy of **date-en-US.js** from the **datejs_loc.zip** file.
- The following **dateFormat** specifiers are not supported:

  G, w, W, D, F, E, k, K, S, z, Z
- No validation against specified **dateFormat** for date entered in text entry field.

Generated HTML Controls

- Control objects are generated with your browser's default font and color settings. The **bgColor** and **valueTextFont** properties are ignored.
- In older browsers that do not support CSS, HTML control objects may not be positioned correctly and appear outside the image.
- Control objects are reset to their initial values when a Drill Down is executed. To display the current value, attach the **selectedValue** property to a variable or Get Substitution function for the substitution set.

iPad Safari

- In the iPad settings for Safari, JavaScript must be ON and Block Pop-ups should be OFF. As of this writing, the Thin Client has been tested only on iOS 4.3.5 in Safari.
- The Thin Client implements scrollbars for table objects and graph objects. However, unlike the scrollbars used on desktop browsers, the scrollbars used on the iPad do not have arrow buttons at each end. This can make it difficult to scroll precisely (for example, row by row) on objects with a large scrolling range.
- At full size, users may find it difficult to touch the intended display object without accidentally touching nearby objects and performing an unwanted drill-down, sort, scroll, and so forth. This is particularly true of table objects that support drill-down and also scrolling, and also in panel layouts that the tree navigation control. In those cases, the user may want to zoom the iPad screen before interacting with the Thin Client.
- If the iPad sleeps or auto-locks while a Thin Client display is open in Safari, or if the Safari application is minimized by clicking on the iPad's home button, the display is not updated until the iPad is awakened and Safari is reopened. In some cases it may be necessary to refresh the page from Safari's navigation bar.

Because the iPad uses a touch interface there are differences in the Thin Client appearance and behavior in iOS Safari as compared to the conventional desktop browsers that use a cursor (mouse) interface, such as Firefox and Internet Explorer. These are described below.

- **Popup browser windows**: An RTView object's drill-down target can be configured to open a display in a new window. In a desktop browser, when the RTView object is clicked the drill-down display is opened in a popup browser window. But in iOS Safari 4.3.5, only one page is visible at a time, so when the RTView object is touched a new page containing the drill-down display opens and fills the screen. The Safari navigation bar can be used to toggle between the currently open pages or close them.
- **Mouseover text**: When mouseover text and drill-down are both enabled on an RTView object (for example, a bar graph), in iOS Safari the first touch on an element in the object (for example, a bar) displays the mouseover text for that element and the second touch on the same element performs the drill-down.
- **Resize Mode and Layout**: By default, the Display Server runs with **resizeMode** set to crop. In crop mode, if a display is larger than the panel that contains it only a portion of the display is visible. In a desktop browser, scrollbars become available to allow the user to scroll to view the entire display. In iOS Safari, scrollbars do not appear but the display can be scrolled by dragging two fingers inside the display. (Dragging one finger scrolls the entire page, not the display).

  If the Display Server is run with **resizeMode** set to scale or layout, the display is resized to fit into the panel that contains it. If a desktop browser is resized after a display is opened, the display is resized accordingly. On the iPad, the Safari browser can only be resized by reorienting the iPad itself, between portrait mode and landscape mode.

  The panel layout feature is supported in the Thin Client. However, unlike a desktop browser which resizes to match the layout size, the size of Safari is fixed. So if the Display Server is run with **resizeMode** set to crop or scale mode, there may be unused space at the edges of the display(s) or, in crop mode, the panels and displays may be cropped.

  This means that layout mode should be used for best results on the iPad. For layout mode to be most effective, displays should use the anchor and dock object properties. Please see RTView documentation for more information.
- **Scrolling**: The Thin Client implements scrollbars for table objects and graph objects. The scrollbars are activated by dragging with one finger.

If an RTView display is viewed in crop mode and is too large to be displayed entirely in Safari, scrollbars do not appear (as they would in a desktop browser) but the display can be scrolled by dragging with two fingers inside the display.

Scrollbars do not ever appear in a text area control. If the text area contains more text than is visible, use the two finger drag in the text area to scroll the text.

Regardless of the size of a listbox control, it can only display a single item (typically, the selected item). When the listbox is touched, the list of items appear in a popup list. In other words, on iOS Safari the listbox control and the combobox control behave identically.

On the default Thin Client page (opened as *http://hostname/rtvdisplay*), the panels cannot be resized by dragging the border between the left and right frame, as they can in a desktop browser. The list of displays in the left hand panel can be scrolled by dragging with two fingers inside the panel.

- **Context menu**: The Thin Client context menu is opened by a right mouse button click in a desktop browser. It is opened in iOS Safari by touching any location on a display and holding that touch for 2 seconds. The menu appears in the top left corner of the display, regardless of where the display is touched. The items Export Table to Excel, Drill Down, and Execute Command are not included on the context menu in Safari. All other items are available. The Export Table to HTML item is enabled if a table object is touched (unless the table object's **drillDownTarget** is configured to open another display). After and Export to PDF/HTML is performed, the exported content opens on another page in Safari. From there, the content can either be opened by another application (for example, the iBooks application opens PDF) and emailed, or it can be copied ands pasted into an email.

Test Displays

Two HTML files are included in the **rtvdisplay.war** for testing. You can also open a display by entering the JSP directly in the URL.

1. Open **index.html** (e.g., **http://host:port/rtvdisplay/index.html**).

In the left frame of index.html is a list of display (**.rtv**) files available in the directory where you started the Display Server. The main frame should show the selected display.

2. Open **panels.html** (e.g., **http://host:port/rtvdisplay/panels.html**).

The **panels.html** file arranges multiple display panels in frames according to your panel configuration file. If the panel configuration file is not found in the project directory where you are running the Display Server, the Display Server searches under **lib** in your installation directory. See the "Multiple Display Panels" section for information on creating panel configuration files.

Note: The **CardPanel** and **GridPanel** are not currently not supported in the Display Server.

---

3. Specify the panel configuration file name and the border width (in pixels) in the URL:

   http://host:port/rtvdisplay/panels.jsp?file=mypanelconfig&border=1

4. To specify a display to open in the URL:

   http://host:port/rtvdisplay/getdisplay.jsp?display=MyDisplay

5. Specify substitutions to apply to your display and a refresh rate (in seconds) which overrides the **DefaultRefreshRate** in **rtvdisplay.properties** (specified in Step 2A):

   http://host:8080/rtvdisplay/getdisplay.jsp?display=MyDisplay&refresh=30&subs=$sub1:Hello+$sub2:'value+2

Note: To specify multiple substitutions, separate them with a **+** (plus). If a substitution value contains a single quote, escape it with a **/** (forward slash). If a substitution value contains a space, enclose it in single quotes (do not escape these quotes) and replace the space with a **+**.

Creating Display Servlet HTML Files

The Display Servlet comes with a few HTML files for testing. You can use these to deploy, or you can create your own HTML files which reference **getdisplay.jsp** to show the specified display (**.rtv**) file. getdisplay.jsp supports three parameters:

getdisplay.jsp Parameters

Parameter	Description
display	The name of the display (.rtv) file to show.
refresh	The refresh rate (in seconds) at which the display inside this URL should be refreshed. Set to **0** if you do not want displays periodically refreshed. **Note:** This refresh rate overrides the **rtvRefreshInterval** property. If no refresh rate is specified, the interval defined in the rtvdisplay.properties file will be used unless you have set a value for **rtvRefreshInterval**. This parameter is optional. See "Background Properties" and "Display Server Configuration" for more information.
rtvpass	If "Login" is enabled, specify the password in plain text to use for the login. This parameter must be used in conjunction with **rtvuser** and will bypass the login dialog. If the **rtvrole** parameter is not specified for a user with multiple roles, the first role will be used. Use the **rtvsign** parameter instead to specify an encoded user name and password. Any non-alphanumeric characters in the parameter value must be URL encoded as **%xx**, where xx is the hexadecimal value of the ASCII character. **Note:** If the user name or password specified is not valid, the login dialog will appear.
rtvrole	If "Login" is enabled, specify the role to use for the login. This parameter must be used with **rtvsign** or **rtvuser** and **rtvpass**. If this parameter is not specified for a user with multiple roles, the first role will be used. Any non-alphanumeric characters in the parameter value must be URL encoded as **%xx**, where xx is the hexadecimal value of the ASCII character. **Note:** It may be necessary to URL-encode the username, password, or role, depending on the characters they contain.
rtvsign	If "Login" is enabled, specify an encoded user name and password to use for the login, and bypass the login dialog. Contact SL Technical Support at support@sl.com to request a copy of the utility to create the encoded strings. If the **rtvrole** parameter is not specified for a user with multiple roles, the first role will be used. **Note:** If the user name or password specified is not valid, the login dialog will appear.

rtvuser	If "Login" is enabled, specify the user name in plain text to use for the login. This parameter must be used in conjunction with **rtvpass** and will bypass the login dialog. If the **rtvrole** parameter is not specified for a user with multiple roles, the first role will be used. Use the **rtvsign** parameter instead to specify an encoded user name and password. Any non-alphanumeric characters in the parameter value must be URL encoded as **%xx**, where **xx** is the hexadecimal value of the ASCII character. **Note:** If the user name or password specified is not valid, the login dialog will appear.
subs	The substitution(s) to apply to the display. To specify multiple substitutions, separate them with a **+** (plus). If a substitution value contains a single quote, escape it with a **/** (forward slash). If a substitution value contains a space, enclose it in single quotes (do not escape these quotes) and replace the space with a **+**. Any non-alphanumeric characters in the parameter value must be URL encoded as **%xx**, where **xx** is the hexadecimal value of the ASCII character. This parameter is optional.

For example:

```
getdisplay.jsp?display=MyDisplay&refresh=30&subs=$sub1:Hello+$sub2:'value+2&rtvsign=8I
559A5NA8A5864J6J924N0B2
```

The HTML returned by **getdisplay.jsp** is only suitable to be displayed as an entire page, in a window, frame or iframe, so it can be used only in the href tag for links, or the src tag for frames and iframes:

```
<A HREF="getdisplay.jsp?display=MyDisplay"> View My Display </A>
<FRAME SRC="getdisplay.jsp?display=MyDisplay">
<IFRAME SRC="getdisplay.jsp?display=MyDisplay">
```

The **demos\esphere** example shows how to setup a navigation frame on the left that uses **getdisplay.jsp** to load displays into the main frame. To show multiple displays on the same page, use frames or iframes to display the HTML from **getdisplay.jsp**. If the name of the frame or iframe corresponds to the window name specified in a drill down target, drill downs on that object update the named frame.

As an alternative, refer to the "Multiple Display Panels" section for information on how you can use RTView to generate a navigation frame for your display. Once you've created a navigation tree definition as described, reference it from an iframe in your .html file as follows:

```
<iframe name="navtree" id="navtree" src="navtree.xml"scrolling="yes"></iframe>
```

See **demos\dstututorial\displayserver\index.html** or **servlets\ocmonitor\index.html** for an example.

Note: If you are referencing **getdisplay.jsp** from multiple HTML files in a frameset, and login is enabled, a login screen will appear for each HTML file. To bypass this, go to **login.jsp** to login and redirect to your page (where **display1.html** is the name of your page):

```
<script> location.href = "login.jsp?destPath=display1.html"; </script>
```

Creating Display Servlet JSP Files

You can create your own JSP files using the RTView tag library to show any number of display (**.rtv**) files. The RTView tag library defines two tags:

setup — This tag must be included once on each JSP page that uses the RTView tag library. It will insert references to Javascript files and CSS files required by the Display Server. The setup tag must be included in the JSP page before the first display tag. This tag has no attributes.

Example:

```
<rtv:setup/>
```

display — This tag inserts an HTML DIV tag containing an RTView display. The display tag has the following attributes:

Attribute	Description
id	Required. The ID for this tag. This must be unique on the JSP page. An id must start with a letter and may contain only letters and digits.
displayname	Required. The string specifying an RTView display (**.rtv**) file name. This should correspond to the name of a display (**.rtv**) file accessible by the Display Server.
refresh	Optional. A positive number defining the rate, in seconds, at which the display inside this DIV should be refreshed. If zero, the DIV is not periodically refreshed. **Note:** This refresh rate overrides the **rtvRefreshInterval** property. If no refresh rate is specified, the interval defined in the **rtvdisplay.properties** file will be used unless you have set a value for **rtvRefreshInterval**. See "Background Properties" and "Display Server Configuration" for more information.
style	Optional. The CSS style to be used for this DIV.

Example:

```
<rtv:display id="d1" displayname="MyDisplay" refresh=0"/>
```

Requirements

There are THREE requirements for JSP pages using the RTView tag library:

1. Tag library definition

The tag library must be defined at the top of the JSP page. For example:

```
<%@ taglib prefix="rtv" uri="rtv.tld" %>
```

Note: The file **rtv.tld** is contained in **rtvdisplay.war**.

---

2. RTView mouse and key events

For proper operation of the RTView menu, the **onmousedown** and **onkeydown** event handler of the page's BODY must be defined. For example:

```
<BODY onmousedown="rtvBodyMouseDown(event)" onkeydown="rtvBodyKeyDown(event)">
```

If other functions are already assigned to these event handlers, then those functions must call **rtvBodyMouseDown** and **rtvBodyKeyDown**.

3. RTView menu

Each JSP page must include the RTView menu. For example:

```
<jsp:include page="rtvmenu.html"/>
```

This is typically added near the end of the page's BODY, and should not be enclosed in any elements other than <BODY>.

Example

The following is a simple JSP page using the RTView tag library.

```
<%@ page contentType="text/html;charset=UTF-8"%>
<%@ taglib prefix="rtv" uri="rtv.tld" %>
<HTML>
<BODY topmargin='0' leftmargin='0'
 onmousedown="rtvBodyMouseDown(event)"
 onkeydown="rtvBodyKeyDown(event)">

 <rtv:setup/>

 <rtv:display id="d1" displayname="MyDisplay" refresh="0"/>
 <rtv:display id="d2" displayname="AnotherDisplay" refresh="30"/>

 <jsp:include page="rtvmenu.html"/>

 </BODY>
 </HTML>
```

Limitations

A separate timer is used for each **<rtv:display/>** tag that has a nonzero refresh rate. If all of the display tags on a page use the same refresh rate, they may still be refreshed at slightly different times. A display's timer is restarted when a drill down is performed in the display, which may offset its refresh timer from other refresh timers on the page.

Managing the Display Server Using JMX

The Display Server is instrumented with JMX to allow you to manage and monitor the display cache and application settings. To enable JMX, you must run the Display Server using the **-jmxport** command line option (see "Command Line Options: Display Server" for more information):

```
-jmxport:(port number)
```

The JMX Monitor demo, located in **demos\jmxmonitor** in your RTView installation, shows how to use RTView to monitor the Display Server using JMX.

Note: Java 1.7+ is required.

The Display Server contains one MBean named **RTViewDisplayServer:name=Manager** with the following methods:

Method / Attribute	Type	Description
DisplayData	TabularData	One row for each cached display containing: **Display Name** - The name of the .rtv file for the display. **Display Number** - The display number. These values always range from 1 to N, where N is the number of displays currently in the cache. This number is not tied to a particular display (i.e. display name + substitutions), and can change as displays are automatically removed from the cache. **Last Reference Time** - The amount of time that has elapsed since the display was last requested by a client. When this time exceeds the **Display Timeout**, the display is eligible for automatic removal from the cache. **Substitutions** - The substitution string for the display.
DisplayTimeout	int	The current display timeout setting.
ImageQuality	float	The current image quality setting.
MaximumDisplayCount	int	The current maximum display count setting.
NumberOfDisplays	int	The number of cached displays.
Sessions	TabularData	One row for each active client session. Typically, there is one client session for each browser instance that is currently viewing a Thin Client display. The Sessions table contains the following columns: **Session ID** - A unique ID for the client session. The Session ID is also a column in the mbean's **DisplayData** table, which contains one row for each display that is currently open in a Thin Client. The Session ID is a unique string assigned to a session by RTView and is not the same value as the session identifier assigned by the webapp manager (e.g. Tomcat). **Client Address** - The IP address of the client. The Client Address will be in IPv4 (e.g. x.x.x.x) or IPv6 (x:x:x:x:x:x:x:x) notation, depending on which protocol the client browser used to connect to the rtvdisplay servlet. For example, if the client is on localhost then the client address could be shown as **127.0.0.1** for IPv4, or **0:0:0:0:0:0:0:1** for IPv6. **Duration** - The elapsed time since this client session was created. The **Duration** column and the **Last Reference Time** column show elapsed time as "**d hh:mm:ss**" where **d** = days, **hh** = hours, **mm** = minutes and **ss** = seconds. By default, a session expires and is removed from the table after about 10 or 11 minutes of inactivity (that is, when the elapsed time shown in the Last Ref Time column is "**0 00:11:00**" or more). If the Display Server's **display_timeout** property has been set, then the session expiration is 10 times the **display_timeout** value. (The default **display_timeout** is **60** seconds). **Last Reference Time** - The elapsed time since this client opened or refreshed a display. **User** - The login user name. The **User** and **Role** columns will be blank if the login feature of the rtvdisplay servlet is disabled. **Role** - The login user role. The **User** and **Role** columns will be blank if the login feature of the rtvdisplay servlet is disabled.

The **RTViewDisplayServer:name=Manager** MBean also supports the following commands:

Method / Attribute	Description
clearDisplayCache	Clears the cached displays as well as preloaded displays.
setDisplayTimeout	Set the display timeout.
setImageQuality	Set the image quality.
setMaximumDisplayCount	Set the maximum display count.

See "Display Server Configuration" for more information.

Display Server Single Sign-On

In addition to supporting the single sign-on method that utilizes the **rtvuser**, **rtvpass**, **rtvsign**, and **rtvrole** parameters in **getdisplay.jsp**, the Display Server also supports two other types of single sign-on:

- "Login from a Servlet Container"
- "Login from Custom JavaScript"

Login from a Servlet Container

User authentication is a general feature of servlet containers (Tomcat, WebLogic, etc). The **web.xml** servlet file can be configured to require user login before the web server allows access to the servlet. A sample web.xml file included with RTView and located in **rtvdisplay\web.xml.auth_example**, shows how user authentication can be configured for the Display Servlet. It requires the user to login to the servlet container in order to open any Display Servlet URL, then grants authenticated users access to all Display Servlet URLs.

The sample **web.xml** specifies DIGEST authentication mode, which tells the browser to prompt for a user name and password, then encrypt the password before sending it to the web server. Most modern browsers and servlet containers support DIGEST mode. Other authentication modes are also defined, but support for those is determined by your servlet container and browser, not by RTView.

To configure the Display Servlet to accept user names authenticated by the servlet container as the login user name for the Display Server:

1. Set **web.xml** to require user login before the web server allows access to the servlet.
2. Modify the **LoginEnabled** property in the rtvdisplay.properties file as follows:

   LoginEnabled=AUTH

To enable single sign-on to the servlet container and the Display Server, the servlet container login name must match the user name defined in the user configuration file. It is not necessary for the passwords to match unless **passclientlogin** is set to true, since it is assumed that the user name has already been authenticated by the servlet container.

For example, the user database for Tomcat is defined in the file **<tomcat_home>/conf/tomcat-users.xml**. By default, a user named admin is defined. The default user configuration file for RTView also defines a user named admin. If the Display Servlet files web.xml and rtvdisplay.properties are configured as described above, a user could login to Tomcat as admin and the Display Servlet would then automatically login the admin user to the Display Server, without prompting again for a user name and password. See "Configuration" for more information.

Login from Custom JavaScript

The Display Servlet also supports login from custom JavaScript by calling the following JavaScript function:

rtvLogin(username, password, role)

This function returns the string "ok" if the login is successful and returns a string with an error message if it fails. The **role** argument is optional. The password is encrypted before it is sent to the server. To call the **rtvLogin** function from a custom web page, the page must be included in the Display Servlet web archive file and must import the **rtv1.js** and **rtvx.js** script files. An **rtvLogout** function is also available:

rtvLogout()

These functions use the **XMLHttpRequest** object that is included with most modern browsers. In Internet Explorer, if scripting of ActiveX objects is disabled, these functions will fail.

For example, the following HTML file presents the user with a combo box containing the user names admin and guest, and a text box for entering a password. When the user clicks **OK**, the **rtvLogin** function is called. If the login information is validated by the Display Server, the **displays.html** file is opened in the browser. Otherwise an error dialog appears.

Example:

```
<html>
<script src="rtv1.js"></script>
<script src="rtvx.js"></script>
<script>
function doLogin ()
{
                   var user = document.getElementById("usercombo").value;
                   var pwd = document.getElementById("password").value;
                   var msg = rtvLogin(user, pwd);
                   if (msg != "ok")
                                      alert(msg);
                   else
                                      document.location = "displays.html";
}
</script>
<body>
<div>Please log in</div>
<table border="0" cellpadding="2" cellspacing="0">
                  <tr><td align="right" nowrap>Username:</td>
                  <td align="right">
                                     <select id="usercombo" style="width:120px">
                                                  <option value="admin">admin</option>
                                                  <option value="guest">guest</option>
                                     </select>
                   </td></tr>
                   <tr><td align="right" nowrap>Password:</td>
                   <td align="right"><input type="password" id="password"
                                      style="width:120px"
                                value="" onkeypress="if (isEnterKey(event)) doLogin()">
                    </td></tr>
                    <tr><td colspan="2" align="right">
                                      <button onclick="doLogin()">OK
                    </td></tr>
</table>
</body>
```

```
</html>
```

CHAPTER 13 Reporting

This section contains the following:

Overview

RTView has two methods for quickly exporting reports of your displays to a printable PDF file: Automatic Report Generation and On Demand Reporting. Both types of reports can be localized.

The **Automatic Report Generation** and **On Demand Reporting** reporting methods provide the following formats:

Display - With this option, an image of a display is exported.

Report - With this option, an image of a display is exported onto the first page followed by at least one page for each table or object grid in the display. As many pages as are necessary to show all the data in each table or object grid are included in the report. This enables you to view all data in a table or object grid that you otherwise must use a scrollbar to see. If there are no tables or object grids in your display, you will only get an image of the display.

On Demand Reporting

Use On Demand Reporting to export reports a single display from within an RTView application.

This section contains the following:

Generating Reports with the Display Builder

When generating a report in the Display Builder, the **Page Setup** dialog opens, then RTView creates a printable PDF file to save to disk.

Display - Right-click on the display to export and select **Export>Display**.

Report - Right-click on the display to export and select **Export>Report**.

Note: Export>Report / Display from the file menu will export the display in the main window of the Display Builder.

Generating Reports with the Display Viewer

When generating a report in the Display Viewer, the **Page Setup** dialog opens, then RTView creates a printable PDF file to save to disk.

Display - Right-click on the display to export and select **Export>Display**.

Report - Right-click on the display to export and select **Export>Report**.

Selecting **Export>Report / Display** from the file menu will export the display in the main window of the Display Viewer. If you are using "Multiple Display Panels", the display that will be exported depends on the layout:

Border Panel - Display in center panel is exported

Tabbed Panel - Display in selected tab is exported

Grid Panel - Display in first panel in grid is exported

Card Panel - Display in visible panel is exported

Navigation Tree - Display in center panel is exported

Generating Reports with the Thin Client Browser

When generating a report in the Thin Client Browser, the Page Setup dialog opens, then the PDF file opens in a new browser window.

Display - Right-click on the display to export, select **Export PDF**, and choose **Display** in the **Page Setup** dialog.

Report - Right-click on the display to export, select **Export PDF**, and choose **Report** in the **Page Setup** dialog.

Localizing Reports

You can localize the headers and footers of exported reports for the Display Builder, Display Viewer, and the Thin Client.

To localize the headers and footers of your exported reports:

1. Navigate to the **custom\rtvreport** directory.
2. Copy the **rtvreport.properties** file into the directory where you start the RTView application. Typically, this is where the display (**.rtv**) files used by your application reside.

3. Open the **rtvreport.properties** file you just copied in a text editor and edit the following properties as needed:

```
#pdf.header.left=
#pdf.header.center=
#pdf.header.right=
pdf.footer.left=
pdf.footer.center=
pdf.footer.right=
```

The following example **rtvreport.properties** file contains the default values and initial values. Where:

- Date / Time formats are delimited by **|||**.
- Date Time formats use standard Java **SimpleDateFormat** patterns, as defined here: **http://download.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html**
- Page numbers are defined as:

  ### = current page number

  @@@ = total number of pages

Example:

```
# PDF Report Headers and footers properties
#
# This properties file (rtvreport.properties) can contain property
# definitions for the values used as header and footer elements in
# exported PDF files
#
# If a value is defined, it will override the default value
# If a value is not defined, the default value will be used.
# default values are defined below
#
# DEFAULT PROPERTY VALUES
#
#pdf.header.left=
#pdf.header.center=RTView
#pdf.header.right=
#pdf.footer.left=|||EEEE, MMMM d, yyyy|||
#pdf.footer.center=Page ### of @@@
#pdf.footer.right=|||h:mm:ss a z|||
#
# ACTUAL PROPERTY VALUES
#
# To define an actual value,
# Copy the appropriate default value,
# Uncomment, by removing the leading '#'
# And set the property value following the '='
#
 pdf.footer.left=|||EEEE, MMMM d, yyyy|||
 pdf.footer.center=Page ### of @@@
 pdf.footer.right=|||h:mm:ss a z|||
```

Note: To use a single rtvreport.properties file for multiple applications in different directories, use the **RTV_USERPATH** environment variable to define a **classpath** that includes a reference to where the

rtvreport.properties file resides, as the **rtvreport.properties** file is a properties file that is read from the **classpath** at runtime.

Note: The **rtvreport.properties** file is treated as a resource bundle, and thus follows the java convention for localizing resource bundles (**rtvreport** is followed by an underscore and a two-letter language code (ISO 639-2 code) that specifies the language that the resource bundle deals with.) For example, for Spanish and French you create **rtvreport_es.properties** and **rtvreport_fr.properties**, respectively.

Automatic Report Generation

Use Automatic Report Generation to create reports for multiple displays from the command line. You customize the report by modifying the report configuration file.

This section contains the following:

Starting Automatic Report Generation

To start Automatic Report Generation, run the **run_reportgen** script in the directory where your RTView displays reside.

In an initialized command window (see "Initializing a Command Prompt or Terminal Window"):

1. Go to the directory where your displays reside.

   type **run_reportgen -report:<reportconfigfilename>**

See **demos/features/report.xml** for a sample configuration file.

Customizing a Report

The report configuration file can contain multiple reports. Each report can contain multiple sections, each with a separate display that is output to a single PDF file.

The report configuration file is in XML, and must start with the following:

```
<?xml version="1.0" ?>
<rtvreport xmlns="www.sl.com" version="1.0">
```

The file must end with the following:

```
</rtvreport>
```

The following tags are supported:

Tag	Description	Attribute Name	Attribute Description
report	Specify a report definition. Your configuration file may contain multiple reports.	name	The name of the report.
		title	The title for the report. This tag is optional. If specified, this title will be used instead of the display name in the header of each page in the report.
		no_header	Suppresses the header from the report. This tag is optional, takes a boolean value, and uses the following syntax: **no_header="true"**
		no_footer	Suppresses the footer from the report. This tag is optional, takes a boolean value, and uses the following syntax: **no_footer="true"**
section	Specify the display and page setup information for this section. You may have multiple sections in a single report.	display	The name of the RTView display to use for this section.
		subs	The substitution values to apply to the RTView display. Specify initial substitutions for this display. "Substitutions" are optional and must use the following syntax: **$subname:subvalue $subname2:subvalue2** If a substitution value contains a single quote, it must be escaped using a **/** : **$filter:Plant=/'Dallas/'** If a substitution value contains a space, it must be enclosed in single quotes. Do not escape these single quotes: **$subname:subvalue $subname2:'sub value 2'** A substitution string cannot contain the following: : | . tab space , ; = < > ' " & / \ { } [ ] ( ) **Note:** Substitutions set in Application Options will apply to all displays. See "Substitutions Tab" for more information.
		timeout	The time to wait (in seconds) for data sources to update before producing a report for this section.
page_setup	Specify the page setup options for this section.	orientation	The values are portrait or landscape.
		margin_left	Sets the left margin. The values are in inches.
		margin_right	Sets the right margin. The values are in inches.
		margin_top	Sets the top margin. The values are in inches.
		margin_bottom	Sets the bottom margin. The values are in inches.

Tag	Description	Attribute Name	Attribute Description
output	Specify the output options for this report.	filename	The base name of the generated report file. A PDF suffix is added automatically.
		append_timestamp	The values are **none**, **date_time**, or **date_only**. If **date_time**, append timestamp to file basename in the form **yearmonthday_hourminsec**, where hour is on a 24 hour clock. For example, a report with a filename of **myreport** output on September 15, 2006 at 8:45am would be named as follows: **myreport_20060915_084500.pdf**. If **date_only**, append the timestamp in the form **yearmonthday**. For example, a report with a filename of **myreport** output on September 15, 2006 at any time would be named as follows: **myreport_20060915.pdf**. **Note**: Alternatively, you can use **true** instead of **date_time**, and you can use **false** instead of **none**. The **date_only** option does not have an alternate value.
		report_or_display	The kind of report to generate. The values are display or report. **display** - With this option a screen capture of a display is exported. **report** - With this option a screen capture of a display is exported onto the first page followed by at least one page for each table or object grid in the display. As many pages as are necessary to show all the data in each table or object grid are included in the report. This enables you to view all data in a table or object grid that you otherwise must use a scrollbar to see. If there are no tables or object grids in your display, you will only get a screen shot of the display.

Example **report.xml** File

```
<?xml version="1.0" ?>
 <rtvreport xmlns="www.sl.com" version="1.0">
<report name = "report1">
    <section display = "object_variety.rtv"
        subs = ""
        timeout = "0">
        <page_setup orientation = "landscape"
            margin_left = "1.0"
            margin_right = "1.0"
            margin_top = "1.0"
            margin_bottom = "1.0">
        </page_setup>
    </section>
    <section display = "object_variety2.rtv"
        subs = ""
        timeout = "0">
        <page_setup orientation = "landscape"
            margin_left = "1.0"
            margin_right = "1.0"
```

```
            margin_top = "1.0"
            margin_bottom = "1.0">
        </page_setup>
    </section>
    <output filename = "report1"
        append_timestamp = "true"
        report_or_display = "report">
    </output>
 </report>
 <report name = "report2">
    <section display = "object_variety3.rtv"
        subs = "value1:value2 value3:value4"
        timeout = "0">
        <page_setup orientation = "landscape"
            margin_left = "1.0"
            margin_right = "1.0"
            margin_top = "1.0"
            margin_bottom = "1.0">
        </page_setup>
    </section>
    <output filename = "report2"
        append_timestamp = "true"
        report_or_display = "report">
    </output>
 </report>
</rtvreport>
```

CHAPTER 14 Alerts

This section contains the following:

Overview

RTView features a real-time alert engine that enables management and operational personnel to monitor the health and status of business operations. The alert engine can monitor conditions and perform automated actions from any available RTView data source. Alert definitions can include thresholds, severity, notification policies and automated actions, such as email, system commands, performing a SQL statement or sending JMS messages. RTView can load any number of alert definitions and any number of customized dashboards can be created to view alert status, filter alerts, use alerts as drill down navigation for analysis and corrective action, or to interactively change alert status such as alert acknowledgment.

The "Self Service Alerts" feature makes it easy to set and persist threshold, duration and enabled settings for your alerts in a database.

The "Alert Persistence" feature stores all fields and current state of all active alerts, as well as all cleared alerts that have not been removed from the system in a database using the Historian.

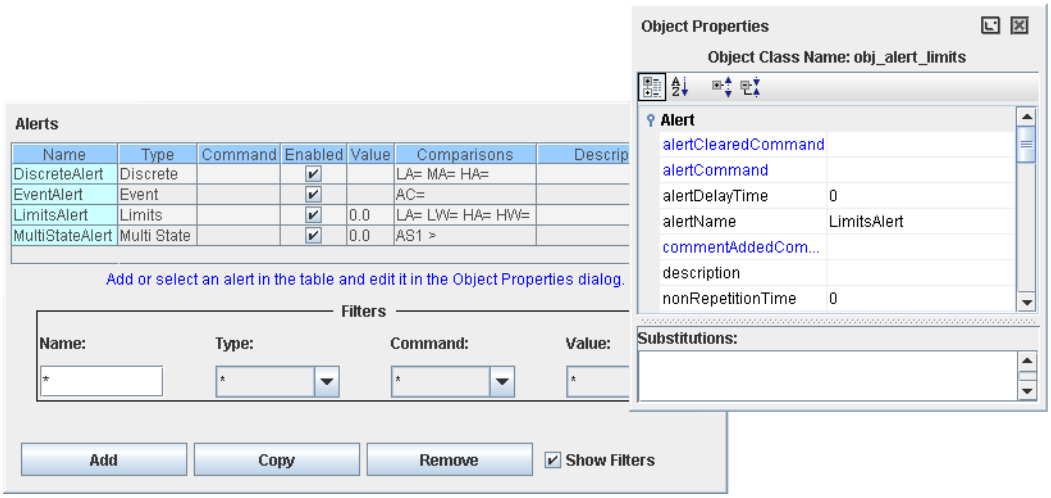
The "Audit Alert Action" feature stores a record in a database each time an alert command is executed.

This section contains the following:

Adding Alerts

In the Display Builder, select **Tools>Alerts** to open the **Alerts** dialog. When you have finished adding all of your alerts and configuring their properties, Save the display (**.rtv**) file and add your "Alert Definition Files" to the Alert data source configuration.

Note: When creating alerts to use with Self Service Alerts, there are a few limitations. See "Alert Construction Limitations" for more information.



Field Name	Description
Add	To add a new alert, click **Add** and enter an alert name and select an alert type. Once you have added an alert, select that alert from the list and edit properties in the **Object Properties** dialog. **Alert Name** - Enter a unique name for each alert listed. Alerts that do not specify a name, as well as alerts with a duplicate names, will not be created and an error will print to the console. **Alert Type** - Choose a type of alert from the drop down menu ("Limits Alerts", "Discrete Alerts", "Multi State Alerts", or "Event Alerts").
Copy	Select an alert from the list and click **Copy** to create a duplicate of that alert. You must enter a unique name for each copy you make. To copy an alert from your current display to another display (**.rtv**) file, select an alert from the list and click the **Copy** button in the toolbar (or Ctrl-C). Then open the other display (**.rtv**) file and click the **Paste** button (or Ctrl-V). If an alert by that name already exists in that display (**.rtv**) file, you will need to rename the alert. An alert pasted into another display (**.rtv**) file will have the same data attachments as the original alert.
Remove	Select an alert from the list and click **Remove** to delete.
Show Filters	Select this check box to filter alerts by **Name**, **Type**, **Command**, or **Value**. When filtering Alerts by **Name**, unnamed alerts will still be included in the filter results.

Alert Definition Files

To create an Alert Definition file, save the display (**.rtv**) file that contains the alert definition properties you set in the **Alerts** dialog. When the Alert data source reads in an Alert Definition file, it adds a line to the "Alert Variables Table" for each alert in the file and creates a variable using the unique name specified for each alert in the file. See the "Alert Definitions Tab" in the **Application Options** dialog for details on how to add an Alert Definition file to the Alert data source configuration.

Creating a Reusable Alert Definition File

You can create a reusable Alert Definition file using the RTView substitution feature. When you enter an Alert Name (**alertName**), include a substitution string as the suffix. Use that same substitution string for the input value in the data attachment. When you subsequently add the Alert Definition file to the Alert data source configuration, you then specify a substitution value.

To give an example, let us say that your sales data is broken down by sales regions, and you need an alert for each. Instead of manually creating an alert for each region, you can create one and reuse it as a template for the others. First create a display containing an alert definition object named **salesAlert.$region**, where **salesAlert** is the alert name and **$region** is the substitution string. The data attachment for the input value also uses the **$region** substitution string. Save the file as **alert_config.rtv**. This is your Alert Definition file. You then add this Alert Definition file to the Alert data source configuration multiple times (in our example, four times), with **$region** set to a different value for each region:

alert_config.rtv	$region:North
alert_config.rtv	$region:South
alert_config.rtv	$region:West
alert_config.rtv	$region:East

You will get four copies of **salesAlert** with the following names:

salesAlert.North

salesAlert.South

salesAlert.West

salesAlert.East

Scheduling Alerts

You can apply schedules to alerts to enable and disable alerts at different times or on different days. For example, if you wanted to turn your alerts off during the time that you were bringing your environment down for maintenance, you could schedule your alerts so that they were disabled during the downtime, thus preventing the firing of alerts during that time.

This section contains:

- "Applying Schedules to Alerts"
- "Monitoring Scheduling Events"

Applying Schedules to Alerts

You can apply a schedule to an alert by setting up a "Rule Table", a "Schedule Table", and an "Alert Schedule Map Table". See "Scheduling Functionality" for more information.

Note: Event alerts that are not configured with an alert index are not currently supported by the scheduler.

Scalar alerts with applied schedules will be enabled or disabled as follows:

- If the **Exception** column for the rule is set to **true** (in the "Rule Table"), then the alert will be disabled from the defined **Start Time** to the defined **End Time**, at which point it will revert back to the enabled property value for the alert. This option will have no effect if the enabledFlag property value for the alert was previously false. When an alert is cleared due to being disabled via a schedule event, the **Cleared Reason** will be **SCHEDULE STARTED** if the alert was disabled because a schedule rule became active. The **Cleared Reason** will be **SCHEDULE ENDED** if the alert was disabled because a schedule rule became inactive.
- If the **Exception** column for the rule is set to **false** (in the "Rule Table"), then the alert will be enabled from the defined **Start Time** to the defined **End Time**, at which point it will revert back to the enabled property value for the alert.This option will have no effect if the **Enable** flag for the alert was previously true.

Tabular alerts with applied schedules will be enabled or disabled as follows:

- If the **Enable** flag for the alert is false, the scheduler will be ignored. In order to use schedules for tabular alerts, the alert's enabledFlag must be turned on.
- If the **Exception** column for the rule is set to **true** (in the "Rule Table"), then the alert index will be disabled from the defined **Start Time** to the defined **End Time**, at which point it will revert back to the rowEnabledTable property value for the alert. This option will have no effect if the alert index's rowEnableTable property value was previously false. When an alert is cleared due to being disabled via a schedule event, the **Cleared Reason** will be **SCHEDULE STARTED** if the alert was disabled because a schedule rule became active. The **Cleared Reason** will be **SCHEDULE ENDED** if the alert was disabled because a schedule rule became inactive.
- If the **Exception** column for the rule is set to **false** (in the "Rule Table"), then the alert index will be enabled from the defined **Start Time** to the defined **End Time**, at which point it will revert back to the alert index's rowEnabledTable property value. This option will have no effect if the alert index's rowEnabledTable property value was previously true for the alert index.
- If an Alert Name and Alert Index combination matches more than one row of the **RtvAlertScheduleMap**, the row with the **Index Type** of **All** takes precedence. For all other Index Type values, the order in which they are specified in the **indexTypes** property on the alert definition controls the order of precedence. If an alert index does not match any of the rows, the schedule for the Default Index Type for the matching alert name is used. If there is no Default Index Type for that alert name, no schedules are applied to the alert.

Monitoring Scheduling Events

The Schedule Events table allows visibility into schedule events, and is updated with a new row each time a schedule enables or disables an alert or alert index. New rows in the table replace the previous table content.

Note: In order to see all schedule events, you will need to store this table in a cache.

This table contains the following columns:

Column Name	Definition
Timestamp	The timestamp of the schedule event.
Alert Name	The name of the alert on which the schedule event occurred.
Alert Index	The alert index on which the schedule event occurred. This will be blank for scalar alerts. For tabular alerts, this will either be ***** or the name of the alert index depending on how the alert is configued in the **RtvAlertScheduleMap**. If the same schedule was applied to all alert indexes by specifying **Default** for both the **Alert Index** and **Index Type**, this value will be *****. Otherwise, it will be the alert index value.
Schedule	The name of the schedule that generated the event.
Rule	The name of the rule in the schedule that generated the event.
Rule State	This will be **STARTED** if a rule became active. It will be **ENDED** if a rule became inactive.
Enabled	The enabled value of the alert or alert index after this schedule event occurred.

Note that this table will be updated any time a schedule event occurs regardless of whether it results in a change to the enabled state of the alert. For example, if you have an alert configured with a schedule rule that sets the enabled state to false and the enabled state for the alert is already false when the schedule rule becomes active, it will not result in a change to the enabled state of the alert.

Running the Alert Engine

The alert engine resides within the Alert data source, so there is no additional process to run the alert engine. How RTView is deployed determines where the alert engine specifically runs:

Thin Client Browser with Direct Data Connection - The alert engine runs on the Display Server. The alert engine is active as long as the Display Server is running, regardless of whether clients are connected.

All Deployments with Served Data - The alert engine runs on the Data Server. The alert engine is active as long as the Data Server is running, regardless of whether clients are connected.

Alert Behavior

Alert Execution

Alerts are evaluated once each update period. By default, this is every two seconds. You may reset the update period on the "General Tab" of the **Application Options** dialog. If the current value of the input data meets an alert condition, the alert executes.

When an alert is executed, the variables in the Alert data source are updated and the **alertCommand** executes.

Note: Multiple asynchronous data updates between updates will be processed on the Max Redraw Rate. By default, this is every 500 milliseconds. If asyncronous data comes in faster than the specified Max Redraw Rate, updates will be missed. For example, if your input data is a JMS message and you receive three messages between Max Redraw Rate updates, only the data from the last message will be used when evaluating the alert condition.

Cleared Alerts

Alerts are evaluated once each update period. By default, this is every two seconds. You may reset the update period on the "General Tab" of the **Application Options** dialog. If the current value of the input data for an alert that has been executed is no longer in an alert state, the alert will clear. Set the time to keep cleared alerts on the "Alerts Tab" of the **Application Options** dialog.

When an alert is cleared, the variables in the Alert data source are updated, the **clearedCommand** executes, and the configured re-notification command (i.e. **alertCommand** or **reNotificationCommand**) no longer executes.

Note: Multiple asynchronous data updates between updates will be processed on the **Max Redraw Rate**. By default, this is every 500 milliseconds. If asyncronous data comes in faster than the specified Max Redraw Rate, updates will be missed. For example, if your input data is a JMS message and you receive three messages between Max Redraw Rate updates, only the data from the last message will be used when evaluating the alert condition. See "General Tab" for more information.

Viewing Alerts

When the Alert data source reads your Alert Definition file, it creates a variable for each alert using the unique **alertName** specified in the Alert Definition file. See "Attach to Alert Data Dialog" for more information on how to create a display to attach to alert variables and view real-time alert data.

For alert definition objects that use scalar input data (i.e. the **useTabularDataFlag** property is not selected), this variable will be scalar. It will list the highest severity for the alert definition. For alerts that use tabular input data, this variable will be a table. See "Per-Tabular Alert Table" for more information.

AlertTable

Lists all active and cleared alerts.

By default, individual index columns will not show up in the **AlertTable**. To display these columns in the **AlertTable**, add a **Custom Alert Object Property** for each column and map them using the **customPropertyMap** property on the alert. See "Custom Alert Fields Tab" for more information.

Note: Different alerts may have different index columns and a different number of index columns, so this allows you to control of whether, and how, index values for different alert definitions are displayed in the **AlertTable**.

Item	Description
Acknowledged	Selected if the alert has been acknowledged. See "Command Types" for more information.
Alert Name	Value of the **alertName** property.
Alert Index	For tabular alerts with multiple index columns this contains the concatenated values for all index columns in the **valueTable** as specified in the **indexColumnNames** property on the alert and delimited by the Multiple Index Delimiter (default is **~**). See "Alert Definitions Tab" for more information. For alerts with a single index column this contains the value of the index column. **Note:** If input data is scalar (i.e. the **useTabularDataFlag** is not selected) this field will be blank.
Alert Index Values	For tabular alerts with multiple index columns this contains the alert index values concatenated by semi-colon (;). **Note:** You can use this value to filter against multiple columns in the Filter By Row function. For tabular alerts with a single index column and for scalar alerts, this will contain the same value as the Alert Index.
Alert Text	Text about the alert. **Note:** Customize this text using the **value*AlertText** properties.
Cleared	Selected if the alert has cleared.

Cleared Reason	DATA UPDATE	Alert received a value that did not meet the alert condition.
	EXPIRED	Alert expired due to the value in the **alertExpireTime** property. This property is only supported for Event alerts.
	DISABLED	Alert definition was disabled. An alert can be disabled by the **enabledFlag** and/or **rowEnabledFlag** properties, or via the **Enable Alert Definition** command. See "Command Types" for more information.
	MANUAL	Cleared by the Clear Alert command. See "Command Types" for more information.
	ALERT HISTORY DEPTH EXCEEDED	Number of rows in the **AlertTable** exceeded the specified Alert History Depth. See "Alerts Tab" for more information.
	SCHEDULE STARTED	Alert was disabled via the scheduler when a rule became active. See "Alert Definition Files" for more information.
	SCHEDULE ENDED	Alert was disabled via the scheduler when a rule became inactive. See "Alert Definition Files" for more information.
Cleared Time	Time that the alert was cleared.	
Comments	This column is initially blank, but is updated by the **Add Comment** and **Clear Comments** commands. See "Command Types" for more information. **Note:** Comments can be added or cleared for any active, cleared or acknowledged alert unless that alert has been purged from the system.	
Count	When an alert is generated, this column contains a value of 1 and increases incrementally each time the alert receives a data update as long as the **Last Update Time** has also changed. If the **Last Update Time** has not increased, the **Count** is not incremented. For tabular alerts, the **Last Update Time** and **Count** are only incremented when a data update comes in for a specified index. However, when a tabular alert is attached to a cache, the cache sends new data for all indexes even when only one index has changed. This results in inaccurate **Last Update Time** and **Count** values. To configure your alert to show correct **Count** and **Last Update Time** values when it is attached to a cache, include the cache's **time_stamp** column in the alert's **valueTable** data attachment. Set the alert's **timeColumnName** property to the name of the cache's **time_stamp** column. This configures the alert to use the cache time stamp for **Last Update Time** and makes the **Count** accurate. You can use the **-ignorelutforcount:true** option to set the **Count** column to increase for all indexes when any row in the table is updated, even if the value in the cache **time_stamp** column only changed for a single row. See "Command Line Options: Data Server" for more information. The **Update Count** on **Acknowledge Alerts** option is selected by default and controls whether or not the **Count** is incremented on acknowledged alerts. • If selected, the count stops incrementing only after an alert is cleared. • If deselected, the count stops incrementing if an alert is either cleared or acknowledged. See "Alerts Tab" for more information.	
ID	Unique ID for the alert.	
Last Update Time	Date/time that the **value** or **valueTable** on the alert last received a data update. This column is updated whenever new data is received for an active (i.e. not cleared) alert.	
Owner	Specify the Owner of an alert using the **Set Owner** command. See "Command Types" for more information.	

Severity	Severity of the alert.
Row Update Time	Date/time that any column in the row for the alert last received an update (as opposed to the **Last Update Time** column which is the last time the **value** or **valueTable** on the alert last received a data update). See "AlertTable" for more information. By default, the **Last Update Time** and **Count** columns are not tracked by the **Row Update** Time. To track the updates of the two columns in the **Row Update Time** column, use the **-lutupdatesnewdata** command line option. When using the **New Data Only Alert Table** as input to a cache, it is recommended that you set the **timestampColumnName** property on the cache to the new **Row Update Time** column. See "Attach to Alert Data" for more information. **Note:** The **RtvAlertTable** cache in the "Self Service Alert Demo" has been enhanced to use the **Row Update Time** for the **timestampColumnName** property, and to limit the columns from the Alert Table that are stored in history.
Time	Time the alert was activated.

Alert Variables Table

Contains all of the active alert variables and their current state.

Item	Description
Alert Name	Value of the **alertName** property. **Note:** Alerts without an **alertName** or with duplicate names will not be added to the Alert Variables Table and an error will print to the console.
Alert State	Highest current severity for the alert. For tabular alerts, it is the highest severity of all alerts in the **valueTable**. **Note:** If an alert is disabled, the Alert State will be -1.
Enabled	Enabled state of the alert: true if enabled, false if disabled.
Description	Description of the alert. This is the value from the description property on the alert.
Alert Type	Type of alert: **Limits**, **Discrete**, **Multi State**, or **Event**.
Tabular Alert	Value is **true** if alert is tabular (i.e. **useTabularDataFlag** property is selected on the alert).
Index Column Names	Index column name(s) for this alert. If you have specified **indexColumnNames** on the alert, this will contain that string. Otherwise, after the alert receives its first data update it will fill in the value with the name of the index column in the **valueTable**. If the alert is scalar (i.e. the **useTabularDataFlag** is not selected) or if it is an unindexed Event alert, this field will be blank.
Index Types	A semi-colon (;) delimited list of available index types for this alert. If a tabular alert does not have any index types, as defined by the **indexTypes** property on the alert, the value will be All. If the alert is scalar (i.e. the **useTabularDataFlag** is not selected), this field will be blank.

Per-Tabular Alert Table

When the Alert data source reads your Alert Definition file, it creates a variable for each alert using the unique **alertName** specified in the Alert Definition file. For alerts that use tabular input data (i.e. the **useTabularData** property is selected), this variable will be a table. It contains one row for each alert index with the following columns:

Item	Description
Alert Index	Name from the first column of the input data for the alert definition. For multiple index alerts, this is the concatenation of all index columns in the **valueTable** as specified in the **indexColumnNames** property on the alert and delimited by the **Multiple Index Delimiter** (default is **~**). The specified **indexColumnNames** will also be included in this table. See "Alert Definitions Tab" for more information.
Alert State	Highest current alert severity for this alert index. If this alert or alert index is disabled, the value of Alert State will become **-1**.
Enabled	Enabled state of the alert index: true if enabled, false if disabled.

Alert Index Types Table

Information about the available index types for your alert. This table only contains rows (one row for each index type) for alerts that have at least one index type specified on the alert's **indexTypes** property.

Item	Description
Alert Name	Name of the alert.
Index Type	Name of the index type.
Column Names	Semi-colon (;) delimited list of column names that define the index type.

Alerting Enabled Table

Indicates whether the alert engine is enabled to the Alert data source.

Item	Description
AlertingEnabled	Enabled state of the alert engine: true if enabled, false if disabled.
AlertTableValid	Validity of **AlertTable**: false at startup, true after one update pass if the following two conditions have been met: **1.** The alert engine has completed initialization. • If self service is configured, alert initialization does not complete until the alert engine has attached to the self service alerts database. • If persistence is configured, alert initialization does not complete until after the alert engine has read the persisted alerts and added them to the alert table or the Persistence Connection Timeout has expired. **2.** The specified Initial Delay time has expired. See "Alert Definitions Tab" for more information. **Note:** If you are running displays that have scalar data attachments to the **AlertingEnabled** table that do not specify a Column Name, then you must modify the data attachment to specify the **AlertingEnabled** column. RTView does not support scalar properties attached to multi-column tables.

Alert Definitions Table

Provides a way to view which file is loading alert definitions. When issues with alerts arise, the information in this table enables users to determine which definition file needs to be modified or re-loaded.

Item	Description
Alert Name	Lists the name of the alert.
File Name	Lists the name of the file containing the alert definition.
Substitutions	Lists the substitutions specified when the alert definition file was loaded.

Managing Alerts

In addition to being able to manage your alerts while developing your display using the **Application Options** dialog, you can also manage them from your deployed display. See "Application Options - Alerts" for more information.

Use the following alert commands to manage your alerts from a deployed display.

- Add Alert Definition File
- Remove Alert Definition File
- Enable Alert Definition
- Enable Alerts
- Acknowledge Alert

See "Define Alert Command" for more information.

Note: Executing these commands from your display in the Display Builder does not cause the **Application Options** dialog to update. For example, if you add an Alert Definition file by using the command, it does not show up in the **Application Options** dialog until you either close and re-open it, or click **Apply**.

Alert Types: Limits, Discrete, Multi State, and Event

This section contains the following:

Limits Alerts

Threshold Values

Limits alerts allow you to compare your input value to a threshold and execute the alert if your input data goes above or below an acceptable range. Attach the value property to your input data if the data is scalar. If the data is tabular, select the **useTabularDataFlag** and attach your input to **valueTable**. The data attached to the **valueTable** property must contain two columns, the first column must be an index column containing a unique value in each row. The second column must contain numeric values to compare to the thresholds. An alert will execute for each row in the table when the value goes above or below the acceptable range.

This alert type supports four thresholds: **valueHighAlert**, **valueHighWarning**, **valueLowAlert**, and **valueLowWarning**. The value and **valueTable** properties are compared against the specified threshold values (**scalar**, **tabular**, or **list**) and an alert is activated when threshold values are exceeded.

When specifying a scalar threshold value, value and each row in the **valueTable** will be compared against the scalar value.

When specifying a tabular threshold value, the input table can contain one, two or four columns:

- If the specified threshold table has one column, the column must contain comparison values and should have the same number of rows as the input table attached to the **valueTable**. Each row in **valueTable** will be compared against values in the corresponding rows of the specified threshold table. If **valueTable** has more rows than the specified threshold table, these extra rows will be compared against the value of the first row of the specified threshold table. The value property will be compared to the first row of the specified threshold table.

Note: Note: This format cannot be used if you will be persisting your alerts. See "Alert Persistence" for more information.

- If the specified threshold table has two columns, the first column must contain indexes and the second column must contain comparison values. The index value for each row in the input table attached to valueTable will be used to lookup the corresponding comparison value from the specified threshold table. If the index is not found in the specified threshold table, no alert is activated. The value property cannot be compared against a threshold table with two columns.
- If the specified threshold table has four columns, include the following:

Alert Index (String)	For single index alerts, this column must contain the alert index. For multiple column alerts, this column must contain concatenated values for the columns used by the specified Index Type. If you have not defined any index types, this column must contain the concatenated values for all index columns. When concatenating index column values, use the Multiple Index Delimiter as specified in **Application Options>**"Alert Definitions Tab" or on the command line. Default is tilde (~). If the specified Index Type is **Default**, then use a value of **Default** for this column as well.
Index Type (String)	Enter one of the index types specified in the **indexTypes** property on this alert or one of the two built-in index types: **Default** - This value will be used for any indexes that are not specified in this table. **All** - The index value will be the concatenated values for all index columns. The **Default** and **All** index types are available for both single index and multiple index alerts even if no custom index types are specified in the **indexTypes** property.
Value (Double)	Enter comparison value to use for rows in the **valueTable** that match the specified Alert Index.
Use Index Value (Boolean)	Indicate whether or not to use this row. If false, this row will be ignored.

When the Alert Index in a row of the **valueTable** matches more than one of the indexes specified in the threshold table, then the row with an Index Type of **All** takes precedence. For all other index types, the order in which they are specified on the **indexTypes** property controls the order of precedence.

In the case where an Alert Index in the **valueTable** doesn't match any of the Alert Indexes in the threshold table, then the row with an Index Type set to **Default** will be used. If there is no row with an Index Type of **Default**, then the value of rows in the **valueTable** with indexes that don't match any of the indexes in the threshold table will be not be evaluated.

When specifying a list of threshold values, the list can contain either index/comparison value pairs or just comparison values:

- If the list only contains comparison values, this should be a semicolon (;) delimited list of comparison values (e.g.: 80;90;100) with the same number of values as there are rows in the input table attached to **valueTable**. Each row in **valueTable** will be compared against corresponding items in the specified threshold list. If **valueTable** has more rows than items in the specified threshold list, these extra rows will be compared against the first item in the specified threshold list. The value property will be compared to the first item the specified threshold list.

Note: This format cannot be used if you will be persisting your alerts. See "Alert Persistence" for more information.

- If the list contains index/comparison value pairs, this should be a semicolon (;) delimited list of comma (,) separated index/comparison value pairs (e.g.: (Chicago,80;Dallas,90;Detroit,100). The index value for each row in the input table attached to valueTable will be used to lookup the corresponding comparison value from the specified threshold list. If the index is not found in the specified threshold list, no alert is activated. The value property cannot be compared against a threshold list of index/comparison value pairs.

Alert Text Values

Specify on the **AlertText** properties (**valueHighAlertText**, **valueHighWarningText**, **valueLowAlertText**, **valueLowWarningText**) the alert text to display in the Alert Text column of the **AlertTable**. If no alert text is specified, the default alert text will be used.

Enter a scalar or tabular value. All alert substitutions available on the **alertCommand**, except **$alertText** and **$alertEmailBody**, can be used in the alert text.

Note: The **$alertText** and **$alertEmailBody** substitutions contain the same string shown in the **Alert Text** column of the "AlertTable".

If a tabular value is specified, the input table requires two or four columns:

- If a two column table is specified, the first must contain indexes and the second must contain alert text values. The index value for each row in the **valueTable** will be used to look up a corresponding index value in the alert text table. If the index value is not found in the specified alert text table and/or the **useTabularDataFlag** is not selected, the default alert text will be used.
- If a four column table is specified, include the following:

Alert Index (String)	For single index alerts, this column must contain the alert index. For multiple column alerts, this column must contain concatenated values for the columns used by the specified Index Type. If you have not defined any index types, this column must contain the concatenated values for all index columns. When concatenating index column values, use the Multiple Index Delimiter as specified in **Application Options>**"Alert Definitions Tab" or on the command line. Default is tilde (**~**). If the specified Index Type is **Default**, then use a value of **Default** for this column as well.
Index Type (String)	One of the index types specified in the **indexTypes** property on this alert or one of the two built-in index types: **Default** - This value will be used for any indexes that are not specified in this table. **All** - The index value will be the concatenated values for all index columns. **Note:** The **Default** and **All** index types are available for both single index and multiple index alerts even if no custom index types are specified in the **indexTypes** property.
Value (String)	Alert text to display in the **Alert Text** column of the "AlertTable" for rows in the **valueTable** that match the specified Alert Index.
Use Index Value (Boolean)	Indicates whether or not to use this row. If false, this row will be ignored.

When the Alert Index in a row of the **valueTable** matches more than one of the indexes specified in the **AlertText** table, then the row with an Index Type of **All** takes precedence. For all other index types, the order in which they are specified on the **indexTypes** property controls the order of precedence.

In the case where an Alert Index in the **valueTable** doesn't match any of the Alert Indexes in the **AlertText** table, then the row with an Index Type set to **Default** will be used. If there is no row with an Index Type of **Default**, then the Alert Text for rows in the **valueTable** with indexes that don't match any of the indexes in the **AlertText** table will use the default alert text.

Alert Command Text Values

Specify on the alert command text properties (**valueHighAlertCommandText**, **valueHighWarningCommandText**, **valueLowAlertCommandText**, **valueLowWarningCommandText**) text to use for the **$alertCommandText** substitution. See "Limits Alert Properties" for information.

Enter a scalar or tabular value. If the value is tabular, the input table requires two or four columns:

- If a two column table is specified, the first must contain indexes and the second must contain alert command text values. The index value for each row in the **valueTable** will be used to look up a corresponding index value in the alert command text table. If the index value is not found in the specified alert command text table and/or the **useTabularDataFlag** is not selected, the **$alertCommandText** substitution will set to an empty string.
- If a four column table is specified, include the following:

Alert Index (String)	For single index alerts, this column must contain the alert index. For multiple column alerts, this column must contain concatenated values for the columns used by the specified Index Type. If you have not defined any index types, this column must contain the concatenated values for all index columns. When concatenating index column values, use the Multiple Index Delimiter as specified in **Application Options>**"Alert Definitions Tab" or on the command line. Default is tilde (**~**). If the specified Index Type is **Default**, then use a value of **Default** for this column as well.
Index Type (String)	One of the index types specified in the **indexTypes** property on this alert or one of the two built-in index types: **Default** - This value will be used for any indexes that are not specified in this table. **All** - The index value will be the concatenated values for all index columns. **Note:** The **Default** and **All** index types are available for both single index and multiple index alerts even if no custom index types are specified in the **indexTypes** property.
Value (String)	Alert command text to use for the **$alertCommandText** substitution for rows in the **valueTable** that match the specified Alert Index.
Use Index Value (Boolean)	Indicates whether or not to use this row. If false, this row will be ignored.

When the Index Value in a row of the **valueTable** matches more than one of the indexes specified, then an Index Type of All takes precedence. For all other index types, the order in which they are specified on the **indexTypes** property controls the order of precedence.

In the case where an Alert Index in the **valueTable** doesn't match any of the Alert Indexes, then the row with an Index Type set to **Default** will be used. If there is no row with an Index Type of **Default**, then rows in the **valueTable** with indexes that don't match any of the indexes in this table will use an empty string for the **$alertCommandText** substitution.

Limits Alert Properties

Property Name	Description
alertClearedCommand	If specified, this command is executed when an alert is cleared. **Note:** This command supports all of the same alert substitutions as the **alertCommand**, as well as the following: **$alertClearedReason** -- Reason the alert was cleared. This is the same value in the Cleared Reason column of the "AlertTable" **$alertClearedTime** -- Time the alert was cleared. It is the same value in the Cleared Time column of the "AlertTable".

alertCommand	The action executed when an alert is activated. Any available RTView commands (see "Define System Command") can be used, as well as the following substitutions: **$alertCommandText** -- Alert command text. **$alertCompValue** -- Value the current input value is being compared against. **$alertCurValue** -- Current input value. **$alertEmailBody** -- Alert email body text. **$alertEmailSubject** -- Alert email subject. **$alertID** -- Unique ID for the alert. **$alertIndex** -- Alert index. This is the same as the value in the Alert Index column of the "AlertTable". **$alertLabel** -- Label indicating the alert type (Discrete alert types: **High Alert**, **Medium Alert**, **Low Alert**; Limits alert types: **High Alert**, **High Warning**, **Low Alert**, **Low Warning**) **$alertName** -- Value from the **alertName** property. **$alertSeverity** -- Severity of the alert. **$alertText** -- Alert text. This is the same text that is displayed in the "AlertTable". **$alertTime** -- Time the alert was generated. This is the same value in the Time column of the "AlertTable".
alertDelayTime	Specifies the amount of time (in seconds) that a value must remain within the alert range before the alert is executed. For example, if **valueHighAlert** is **90** and **alertDelayTime** is **5**, then the input value must stay over 90 for 5 seconds before a high alert is issued. If the **skipDuplicateAlertsFlag** is selected it will apply to both threshold values. So, in the example above, if a high alert has been issued, the **valueHighWarning** is **80** and the input value drops to **85**, it must stay below 90 for 5 seconds before the severity is changed to high warning.
alertName	A unique name for the alert. This property is required and cannot be left blank. To create a reusable Alert Definition file, include the substitution string as the suffix. For example, salesAlert.$region (where **salesAlert** is the **alertName** and $region is the substitution string). See "Creating a Reusable Alert Definition File" for information.
commentAddedCommand	If specified, this command will be executed after a comment is added to the alert by the Add Comment command. See "Command Types" for information. All substitutions supported for **alertCommand** are also supported for this command, as well as the following two additional substitutions: **$alertComment** -- This is set to the value of the **Comments** field for the command after the new comment has been added. **$alertLastComment** -- This is set to the value of the last comment added to the **Comments** field.

customPropertyMap	Use one or more columns from the **valueTable** as the value for one or more Custom Alert Definition Properties that you defined in the "Custom Alert Fields Tab" in the **Application Options** dialog. Use the following syntax: **customPropName:valueTableColumnName;customPropName2:valueTableColumnName2** For example, if you have a defined a Custom Alert Definition Property named **My Custom Property** and you want to use the value from the **My Data Column** column in your **valueTable**, you would specify the following: **My Custom Property:My Data Column** **Note:** In order to use **customPropertyMap**, the **useTabularDataFlag** property must be selected.
description	If specified, a description of the alert will be displayed in the **Alerts** dialog and in the **Description** column of the **Alert Variables Table**.
enabledFlag	Enables or disables the entire alert. If this alert is active when it is disabled, it will be cleared. **Note:** It is not recommended that you attach data to this property if you are using the **Enable Alert Definition** command for this alert. See "Command Types" for information.
indexColumnNames	Enter one or more index column name(s). If left blank, the alert is assumed to have a single index column and the first column of the **valueTable** will be used. For multiple index columns, enter a ; (semicolon) delimited list of column names. The **valueTable** must contain all specified index columns in addition to, and preceding, the required data column. The combination of all index column values in a single row must uniquely identify that Alert Index.
indexTypes	For alerts with multiple indexes, create custom index types for use in the following tables ("Threshold Values", "Alert Text Values", "Alert Command Text Values", and **rowEnabledTable** (see "Limits Alert Properties")) so you can specify a value for a group of alert indexes instead of having to specify a value for each unique alert index combination. Use the following syntax to map one or more index columns to an index type: **typeName:columnName;typeName2;columnName1,columnName2** For example: **PerRegion:Region;PerRegionalService:Region,Service** **Note: Default** and **All** are built-in index types and therefore cannot be entered as a **typeName**.
nonRepetitionTime	Amount of time (in seconds) that must pass after a cleared alert is executed again. If the **skipDuplicateAlertsFlag** is selected, the alert will not be cleared until both thresholds (warning and alert) are cleared.
reNotificationCommand	If specified, this command will be used instead of the value of the **alertCommand** property for renotifications. Otherwise, the specified **alertCommand** will be used. **Note:** This command supports all of the same alert substitutions as the **alertCommand**.

reNotificationMode	Configure how an alert will renotify. Default setting is **Renotify on Timer**. Choose from the following options: **None** -- Do not renotify. The **alertCommand** is executed only once when the alert is activated. **Renotify on Timer** -- Renotify based on **reNotificationTime** property. The **alertCommand** is executed once when the alert is activated and then re-executed every **reNotificationTime** (seconds) until the alert is cleared or acknowledged. If the **reNotificationTime** is set to **0**, then the alert will not renotify. **Renotify on Data Changed** -- Renotify when the input value changes. The **alertCommand** is executed once when the alert is activated and again when a different value is received until the alert is cleared or acknowledged. The new value must be different than the previous value for the alert to renotify. **Renotify on Data Updates** -- Renotify when the input value is updated. The **alertCommand** is executed once when the alert is activated and again whenever a value is received until the alert is cleared or acknowledged. The new data value may be the same or different than the previous value for the alert to renotify.
reNotificationTime	Amount of time (in seconds) that must pass before the **alertCommand** for an unacknowledged or an uncleared alert is re-executed. The **alertCommand** will continue to re-execute on this interval until the alert is acknowledged or cleared. **Note:** This property will be ignored unless the **reNotificationMode** is set to **Renotify on Timer**.
reNotifyOnSevChangeMode	Specifies the conditions for alerts to send renotification when the alert severity increases. There are three modes: **Renotify on First Sev Change**: The alert renotifies when the severity of the alert changes for the first time. **Renotify on All Sev Increases**: The alert renotifies every time the severity increases. **None**: Do not renotify when the severity changes. For the **Renotify on First Sev Change** and **Renotify on All Sev Increases** modes, alert renotification only occurs if the alert is either not acknowledged, or the **Unacknowledge Alerts on First Severity Change** application option is enabled. The renotification executes the **reNotificationCommand** if configured, otherwise it executes the **alertCommand**. By default this option is **None**. For the **Renotify on First Sev Change** and **Renotify on All Sev Increases modes**, alert renotification only occurs if the alert is not acknowledged. The renotification executes the **reNotificationCommand** if configured, otherwise it executes the **alertCommand**. To revert acknowledged alerts to unacknowledged when the severity changes, select the **Unacknowledge on Severity Change Mode** application option. To update the severity of acknowledged alerts you must select the **Update Severity on Acknowledged Alerts** application option. See "Alerts Tab" for more information. **Note:** When run against alerts configured in an older version of RTView, the correct renotification mode is automatically applied. That is, the **reNotifyOnFirstSevChangeFlag** property (from an older RTView version) converts to the equivalent value for **reNotifyOnSevChangeMode** so the alert behavior is the same.

rowEnabledTable	Sets the enabled state for each row in the **valueTable**, so you can enable or disable each index in a tabular alert. This property only applies for tabular alerts, therefore the **useTabularDataFlag** must be selected to activate the rowEnabledTable property. **Note:** If the **enabledFlag** for the alert is disabled, each row index is disabled regardless of the value of **rowEnabledTable**. This property can be set to one of the following: **Scalar data** -- If true, all rows (indexes) in the table will be enabled. If false all rows (indexes) will be disabled. **Tabular data (one or two columns)** -- If your tabular input data has one column, the column must contain boolean values and should have the same number of rows as the **valueTable**. Each row in the **valueTable** will be compared against values in the corresponding rows of the specified **rowEnabledTable**. If the **valueTable** has more rows than the **rowEnabledTable**, these extra rows will be compared against the value of the first row of the **rowEnabledTable**. **Note:** This format cannot be used if you will be persisting your alerts. See "Alert Persistence" for more information. If your tabular input data has two columns, the first column must contain indexes and the second column must contain boolean values. The index value for each row in the **valueTable** will be used to lookup the corresponding enabled value from the **rowEnabledTable**. If the index is not found in the **rowEnabledTable**, the enabled state is set to false and the row (index) is disabled. **List of enabled (boolean) values** -- This should be a semi-colon (;) delimited list of boolean values (e.g.: **true;false;true**) with the same number of values as there are rows in the **valueTable**. Each row in **valueTable** will be set to corresponding enabled values in the specified list. If **valueTable** has more rows than enabled values in the list, then the first value in the list will be used. **Note:** This format cannot be used if you will be persisting your alerts. See "Alert Persistence" for more information. **List of index/enabled value pairs** -- This should be a semi-colon (;) delimited list of comma (,) separated index/enabled value pairs (e.g.: (**Chicago,true;Dallas,false;Detroit,true**). The index value for each row in the **valueTable** will be used to lookup the corresponding enabled value from the specified list. If the index is not found in the enabled value list, the enabled state is set to false and the row (index) is disabled. **Tabular data (four columns)** -- Specify a four column table that includes:

Alert Index (String) -- For single index alerts, this column must contain the alert index.

For multiple column alerts, this column must contain concatenated values for the columns used by the specified Index Type. If you have not defined any index types, this column must contain the concatenated values for all index columns. When concatenating index column values, use the Multiple Index Delimiter as specified in **Application Options>**"Alert Definitions Tab" or on the command line. Default is tilde (**~**).

If the specified Index Type is **Default**, then use a value of Attach your input data to this property if it is tabular. For scalar input data, deselect the **useTabularDataFlag** and attach your input data to the value property.

Data attached to the **valueTable** property must contain at least two columns. The first one or more columns must be index columns. The combination of all index columns must be a unique value for each row and will be used as the Alert Index. If your data contains more than one index column, specify the index column names in the **indexColumnNames** property. Following the index column(s) must be one data column containing numeric values to compare to the thresholds. Additional columns can be mapped to a Custom Alert Definition Property using the **customPropertyMap** property. for this column as well.

Index Type (String) -- Enter one of the index types specified in the **indexTypes** property on this alert or one of the two built-in index types:

Default - This value will be used for any indexes that are not specified in this table.

All - The index value will be the concatenated values for all index columns.

The Default and All index types are available for both single index and multiple index alerts even if no custom index types are specified in the indexTypes property.

Value (Boolean) - Enabled value to use for rows in the valueTable that match the specified Alert Index.

Use Index Value (Boolean) -- Indicates whether or not to use this row. If false, this row will be ignored.

When the Alert Index in a row of the **valueTable** matches more than one of the indexes specified in the **rowEnabledTable**, then the row with an Index Type of All takes precedence. For all other index types, the order in which they are specified on the **indexTypes** property controls the order of precedence.

In the case where an Alert Index in the **valueTable** doesn't match any of the Alert Indexes in the **rowEnabledTable**, then the row with an Index Type set to **Default** will be used. If there is no row with an Index Type of Default, then the value of rows in the **valueTable** with indexes that don't match any of the indexes in the **rowEnabledTable** table will be false.

skipDuplicateAlertsFlag

If selected, the severity of an alert will update (e.g. **valueHighWarning** to **valueHighAlert**) without multiple alerts being activated. That is, only the highest (or lowest) alert will be activated when the input value exceeds both high (or low) thresholds.

Note: By default, once an alert is acknowledged the severity of that alert will no longer update. To enable updating the severity of acknowledged alerts you must select the **Update Severity on Acknowledged Alerts** application option. See "Alerts Tab" for more information.

timeColumnName	Optionally specify a column in the **valueTable** to be used for the Last Update Time in the **AlertTable**. **Note:** The column specified must be of type date or long. If **timeColumnName** is not specified, not of type date or long, or is not found, then the time that the alert last received data will be used. **Note:** This property is only supported if **useTabularDataFlag** is selected and does not apply to scalar alerts. For scalar alerts, the time that the alert last received the data is always used.
useTabularDataFlag	For tabular input data, select **useTabularDataFlag** and attach your input to **valueTable**.
value	Attach your input data to this property if your input data is scalar. For tabular input data, select the **useTabularDataFlag** and attach your input data to the **valueTable** property.
valueCommandFormat	Specify the numeric format (using syntax from the Java **DecimalFormat** class) for the following **alertCommand** substitutions: **$alertCurValue** and **$alertCompValue**. If left blank, values that are eight whole numbers or more will be written in exponential format.
valueDeadband	Specifies a deadband value for the thresholds. If specified, the input value must go below the **valueHighWarning**/**valueHighAlert** minus the deadband value or above the **valueLowWarning**/**valueLowAlert** plus the deadband value for the alert to clear. For example, if the **valueHighAlert** is 90 and the **valueDeadband** is 5, a high alert will be issued when the input value goes above 90, but will not clear until the input value goes below 85. If the **skipDuplicateAlertsFlag** is selected it will apply to both threshold values. So, in the example above, if **valueHighWarning** is set to 80 then the input value must go below 85 before the severity is changed to high warning and below 75 for the alert to be cleared.
valueHighAlert	Specifies the number that the input value must exceed to activate the high alert. Enter either a scalar or tabular value, or specify a list of values. See "Limits Alerts" for more information.
valueHighAlertCommandText	Specify alert command text to use for the $**alertCommandText** substitution. Enter either a scalar or tabular value. See "Limits Alert Properties" and "Alert Command Text Values" for more information.
valueHighAlertEnabledFlag	Enables the high alert threshold. When executed, severity is 2.
valueHighAlertText	Specify the alert text to display in the **Alert Text** column of the **AlertTable**. If no alert text is specified, the default alert text will be used. Enter either a scalar or tabular value. See "Alert Text Values" for more information.
valueHighWarning	Specifies the number that the input value must exceed to activate the high warning. Enter either a scalar or tabular value, or specify a list of values. See "Limits Alerts" for more information.
valueHighWarningCommandText	Specify alert command text to use for the **$alertCommandText** substitution. Enter either a scalar or tabular value. See "Limits Alert Properties" and "Alert Command Text Values" for more information.
valueHighWarningEnabledFlag	Enables the high warning threshold. When executed, severity is 1.

valueHighWarningText	Specify the alert text to display in the **Alert Text** column of the **AlertTable**. If no alert text is specified, the default alert text will be used. Enter either a scalar or tabular value. See "Alert Text Values" for more information.
valueLowAlert	Specifies the number the input value must go below to activate the low alert. Enter either a scalar or tabular value, or specify a list of values. See "Limits Alerts" for more information.
valueLowAlertCommandText	Specify alert command text to use for the **$alertCommandText** substitution. Enter either a scalar or tabular value. See "Limits Alert Properties" and "Alert Command Text Values" for more information.
valueLowAlertEnabledFlag	Enables the low alert threshold. When executed, severity is 2.
valueLowAlertText	Specify the alert text to display in the **Alert Text** column of the **AlertTable**. If no alert text is specified, the default alert text will be used. Enter either a scalar or tabular value. See "Alert Text Values" for more information.
valueLowWarning	Specifies the number the input value must go below to activate the low warning. Enter either a scalar or tabular value, or specify a list of values. See "Limits Alerts" for more information.
valueLowWarningCommandText	Specify alert command text to use for the **$alertCommandText** substitution. Enter either a scalar or tabular value. See "Limits Alert Properties" and "Alert Command Text Values" for more information.
valueLowWarningEnabledFlag	Enables the low warning threshold. When executed, severity is 1.
valueLowWarningText	Specify the alert text to display in the **Alert Text** column of the **AlertTable**. If no alert text is specified, the default alert text will be used. Enter either a scalar or tabular value. See "Alert Text Values" for more information.
valueTable	Attach your input data to this property if it is tabular. For scalar input data, deselect the **useTabularDataFlag** and attach your input data to the value property. Data attached to the **valueTable** property must contain at least two columns. The first one or more columns must be index columns. The combination of all index columns must be a unique value for each row and will be used as the Alert Index. If your data contains more than one index column, specify the index column names in the **indexColumnNames** property. Following the index column(s) must be one data column containing numeric values to compare to the thresholds. Additional columns can be mapped to a Custom Alert Definition Property using the **customPropertyMap** property.

Discrete Alerts

Threshold Values

Discrete alerts allow you to compare the input value to up to three test values (one each for low, medium and high alerts), and execute an alert if your input data equals one of these values. The Discrete alert supports string, number and boolean comparisons. Attach the value property to your input data if the data is scalar. If the data is tabular, select the **useTabularDataFlag** and attach your input data to the **valueTable** property. The data attached to the **valueTable** property must contain two columns, the first column must be an index column containing a unique value in each row. The second column must contain values (string, number, boolean) to compare to the test values. An alert will execute for each row in the table when the value is equal to the specified test values.

This alert type supports three test values: **valueHighAlert**, **valueMediumAlert**, **valueLowAlert**. The value and **valueTable** properties are compared against the specified test values (scalar, tabular or list) and an alert is activated when test values are reached.

When specifying a scalar test value, value and each row in the **valueTable** will be compared against the scalar value.

When specifying a tabular test value, the input table can contain one, two or four columns:

- If the specified test value table has one column, the column must contain comparison values and should have the same number of rows as the input table attached to the **valueTable**. Each row in **valueTable** will be compared against values in the corresponding rows of the specified test value table. If **valueTable** has more rows than the specified test value table, these extra rows will be compared against the value of the first row of the specified test value table. The value property will be compared to the first row of the specified test value table. **Note:** This format cannot be used if you will be persisting your alerts. See "Alert Persistence" for more information.
- If the specified test value table has two columns, the first column must contain indexes and the second column must contain comparison values. The index value for each row in the input table attached to **valueTable** will be used to lookup the corresponding comparison value from the specified test value table. If the index is not found in the specified test value table, no alert is activated. The value property cannot be compared against a test value table with two columns.
- If the specified threshold table has four columns, include the following:

Alert Index (String)	For single index alerts, this column must contain the alert index. For multiple column alerts, this column must contain concatenated values for the columns used by the specified Index Type. If you have not defined any index types, this column must contain the concatenated values for all index columns. When concatenating index column values, use the Multiple Index Delimiter as specified in **Application Options>**"Alert Definitions Tab" or on the command line. Default is tilde (**~**). If the specified Index Type is **Default**, then use a value of **Default** for this column as well.
Index Type (String)	Enter one of the index types specified in the **indexTypes** property on this alert or one of the two built-in index types: **Default** - This value will be used for any indexes that are not specified in this table. **All** - The index value will be the concatenated values for all index columns. **Note:** The **Default** and **All** index types are available for both single index and multiple index alerts even if no custom index types are specified in the **indexTypes** property.
Value (String)	Enter comparison value to use for rows in the **valueTable** that match the specified Alert Index.
Use Index Value (Boolean)	Indicate whether or not to use this row. If false, this row will be ignored.

When the Alert Index in a row of the **valueTable** matches more than one of the indexes specified in the threshold table, then the row with an Index Type of **All** takes precedence. For all other index types, the order in which they are specified on the **indexTypes** property controls the order of precedence.

In the case where an Alert Index in the **valueTable** doesn't match any of the Alert Indexes in the threshold table, then the row with an Index Type set to **Default** will be used. If there is no row with an Index Type of **Default**, then the value of rows in the **valueTable** with indexes that don't match any of the indexes in the threshold table will not be evaluated.

When specifying a list of test values, the list can contain either index/comparison value pairs or just comparison values:

- If the list only contains comparison values, this should be a semicolon (;) delimited list of comparison values (e.g.: 80;90;100) with the same number of values as there are rows in the input table attached to valueTable. Each row in valueTable will be compared against corresponding items in the specified test value list. If valueTable has more rows than items in the specified test value list, these extra rows will be compared against the first item in the specified test value list. The value property will be compared to the first item the specified test value list.

Note: This format cannot be used if you will be persisting your alerts. See "Alert Persistence" for more information.

- If the list contains index/comparison value pairs, this should be a semicolon (;) delimited list of comma (,) separated index/comparison value pairs (e.g.: (Chicago,80;Dallas,90;Detroit,100). The index value for each row in the input table attached to **valueTable** will be used to lookup the corresponding comparison value from the specified test value list. If the index is not found in the specified test value list, no alert is activated. The value property cannot be compared against a test value list of index/comparison value pairs.

Alert Text Values

On the **AlertText** properties (**valueHighAlertText**, **valueMediumAlertText**, **valueLowAlertText**), specify the alert text to display in the **Alert Text** column of the "AlertTable". If no alert text is specified, the default alert text will be used.

Enter a scalar or tabular value. All alert substitutions available on the **alertCommand**, except **$alertText** and **$alertEmailBody**, can be used in the alert text.

Note: The **$alertText** and **$alertEmailBody** substitutions contain the same string shown in the Alert Text column of the "AlertTable".

If a tabular value is specified, the input table requires two or four columns:

- If a two column table is specified, the first must contain indexes and the second must contain alert text values. The index value for each row in the **valueTable** will be used to look up a corresponding index value in the alert text table. If the index value is not found in the specified alert text table and/or the **useTabularDataFlag** is not selected, the default alert text will be used.
- If a four column table is specified, include the following:

Alert Index (String)	For single index alerts, this column must contain the alert index. For multiple column alerts, this column must contain concatenated values for the columns used by the specified Index Type. If you have not defined any index types, this column must contain the concatenated values for all index columns. When concatenating index column values, use the Multiple Index Delimiter as specified in **Application Options>**"Alert Definitions Tab" or on the command line. Default is tilde (**~**). If the specified Index Type is **Default**, then use a value of **Default** for this column as well.
Index Type (String)	One of the index types specified in the **indexTypes** property on this alert or one of the two built-in index types: **Default** - This value will be used for any indexes that are not specified in this table. **All** - The index value will be the concatenated values for all index columns. **Note:** The **Default** and **All** index types are available for both single index and multiple index alerts even if no custom index types are specified in the **indexTypes** property.
Value (String)	Alert text to display in the Alert Text column of the "AlertTable" for rows in the **valueTable** that match the specified Alert Index.
Use Index Value (Boolean)	Indicates whether or not to use this row. If false, this row will be ignored.

When the Alert Index in a row of the **valueTable** matches more than one of the indexes specified in the **AlertText** table, then the row with an Index Type of All takes precedence. For all other index types, the order in which they are specified on the **indexTypes** property controls the order of precedence.

In the case where an Alert Index in the **valueTable** doesn't match any of the Alert Indexes in the **AlertText** table, then the row with an Index Type set to **Default** will be used. If there is no row with an Index Type of **Default**, then the Alert Text for rows in the **valueTable** with indexes that don't match any of the indexes in the **AlertText** table will use the default alert text.

Alert Command Text Values

Specify on the alert command text properties (**valueHighAlertCommandText**, **valueMediumAlertCommandText**, **valueLowAlertCommandText**) text to use for the **$alertCommandText** substitution. See "Limits Alert Properties" for more information.

Enter a scalar or tabular value. If the value is tabular, the input table requires two or four columns:

- If a two column table is specified, the first must contain indexes and the second must contain alert command text values. The index value for each row in the **valueTable** will be used to look up a corresponding index value in the alert command text table. If the index value is not found in the specified alert command text table and/or the **useTabularDataFlag** is not selected, the **$alertCommandText** substitution will set to an empty string.
- If a four column table is specified, include the following:

Alert Index (String)	For single index alerts, this column must contain the alert index. For multiple column alerts, this column must contain concatenated values for the columns used by the specified Index Type. If you have not defined any index types, this column must contain the concatenated values for all index columns. When concatenating index column values, use the Multiple Index Delimiter as specified in **Application Options>**"Alert Definitions Tab" or on the command line. Default is tilde (**~**). If the specified Index Type is **Default**, then use a value of **Default** for this column as well.
Index Type (String)	One of the index types specified in the **indexTypes** property on this alert or one of the two built-in index types: **Default** - This value will be used for any indexes that are not specified in this table. **All** - The index value will be the concatenated values for all index columns. **Note:** The **Default** and **All** index types are available for both single index and multiple index alerts even if no custom index types are specified in the **indexTypes** property.
Value (String)	Alert command text to use for the **$alertCommandText** substitution for rows in the **valueTable** that match the specified Alert Index.
Use Index Value (Boolean)	Indicates whether or not to use this row. If false, this row will be ignored.

When the Index Value in a row of the **valueTable** matches more than one of the indexes specified, then an Index Type of **All** takes precedence. For all other index types, the order in which they are specified on the **indexTypes** property controls the order of precedence.

In the case where an Alert Index in the **valueTable** doesn't match any of the Alert Indexes, then the row with an Index Type set to **Default** will be used. If there is no row with an Index Type of **Default**, then rows in the **valueTable** with indexes that don't match any of the indexes in this table will use an empty string for the **$alertCommandText** substitution.

Discrete Alert Properties

Property Name	Description
alertClearedCommand	If specified, this command is executed when an alert is cleared. **Note:** This command supports all of the same alert substitutions as the **alertCommand**, as well as the following: **Substitution** — **Description** $alertClearedReason — Reason the alert was cleared. This is the same value in the **Cleared Reason** column of the "AlertTable". $alertClearedTime — Time the alert was cleared. It is the same value in the **Cleared Time** column of the "AlertTable".
alertCommand	The action executed when an alert is activated. Any available RTView commands can be used (see "Define System Command"), as well as the following substitutions: **Substitution** — **Description**

$alertCommandText	Alert command text.
$alertCompValue	Value the current input value is being compared against.
$alertCurValue	Current input value.
$alertEmailBody	Alert email body text.
$alertEmailSubject	Alert email subject.
$alertID	Unique ID for the alert.
$alertIndex	Alert index. This is the same as the value in the **Alert Index** column of the "AlertTable".
$alertLabel	Label indicating the alert type (Discrete alert types: High Alert, Medium Alert, Low Alert; Limits alert types: High Alert, High Warning, Low Alert, Low Warning)
$alertName	Value from the **alertName** property.
$alertSeverity	Severity of the alert.
$alertText	Alert text. This is the same text that is displayed in the "AlertTable".
$alertTime	Time the alert was generated. This is the same value that is shown in the Time column of the "AlertTable".
alertDelayTime	Specifies the amount of time (in seconds) that a value must remain equal to the input value before the alert is executed.

alertName

A unique name for the alert. This property is required and cannot be left blank.

To create a reusable Alert Definition file, include the substitution string as the suffix. For example, **salesAlert.$region** (where **salesAlert** is the **alertName** and **$region** is the substitution string). See "Creating a Reusable Alert Definition File" for more information.

commentAddedCommand

If specified, this command will be executed after a comment is added to the alert by the **Add Comment** command (see "Command Types"). All substitutions supported for **alertCommand** are also supported for this command, as well as the following two additional substitutions:

Substitution	Description
$alertComment	This is set to the value of the **Comments** field for the command after the new comment has been added.
$alertLastComment	This is set to the value of the last comment added to the **Comments** field.

customPropertyMap	Use one or more columns from the **valueTable** as the value for one or more Custom Alert Definition Properties that you defined in the "Custom Alert Fields Tab" in the **Application Options** dialog. Use the following syntax: **customPropName:valueTableColumnName;customPropName2:valueTableColumnName2** For example, if you have a defined a Custom Alert Definition Property named **My Custom Property** and you want to use the value from the **My Data Column** column in your **valueTable**, you would specify the following: **My Custom Property:My Data Column** **Note:** In order to use **customPropertyMap**, the **useTabularDataFlag** property must be selected.
description	If specified, a description of the alert will be displayed in the **Alerts** dialog and in the **Description** column of the **Alert Variables Table**.
enabledFlag	Enables or disables the entire alert. If this alert is active when it is disabled, it will be cleared. **Note:** It is not recommended that you attach data to this property if you are using the **Enable Alert Definition** command for this alert. See "Command Types" for more information.
indexColumnNames	Enter one or more index column name(s). If left blank, the alert is assumed to have a single index column and the first column of the **valueTable** will be used. For multiple index columns, enter a **;** (semicolon) delimited list of column names. The **valueTable** must contain all specified index columns in addition to, and preceding, the required data column. The combination of all index column values in a single row must uniquely identify that Alert Index.
indexTypes	For alerts with multiple indexes, create custom index types for use in the following tables ("Threshold Values", "Alert Text Values", "Alert Command Text Values", and **rowEnabledTable** (see "Discrete Alert Properties")) so you can specify a value for a group of alert indexes instead of having to specify a value for each unique alert index combination. Use the following syntax to map one or more index columns to an index type: **typeName:columnName;typeName2;columnName1,columnName2** For example: **PerRegion:Region;PerRegionalService:Region,Service** **Note:** Default and All are built-in index types and therefore cannot be entered as a **typeName**.
nonRepetitionTime	Amount of time (in seconds) that must pass after a cleared alert is again executed.
reNotificationCommand	If specified, this command will be used instead of the value of the **alertCommand** property for renotifications. Otherwise, the specified **alertCommand** will be used. **Note:** This command supports all of the same alert substitutions as the **alertCommand**.

reNotificationMode	Configure how an alert will renotify. Default setting is Renotify on Timer. Choose from the following options: **None** - Do not renotify. The **alertCommand** is executed only once when the alert is activated. **Renotify on Timer** - Renotify based on **reNotificationTime** property. The **alertCommand** is executed once when the alert is activated and then re-executed every **reNotificationTime** (seconds) until the alert is cleared or acknowledged. If the **reNotificationTime** is set to 0, then the alert will not renotify. **Renotify on Data Changed** - Renotify when the input value changes. The **alertCommand** is executed once when the alert is activated and again when a different value is received until the alert is cleared or acknowledged. The new value must be different than the previous value for the alert to renotify. **Renotify on Data Updates** - Renotify when the input value is updated. The **alertCommand** is executed once when the alert is activated and again whenever a value is received until the alert is cleared or acknowledged. The new data value may be the same or different than the previous value for the alert to renotify.
reNotificationTime	Amount of time (in seconds) that must pass before the **alertCommand** for an unacknowledged or an uncleared alert is re-executed. The **alertCommand** will continue to re-execute on this interval until the alert is acknowledged or cleared. **Note:** This property will be ignored unless the **reNotificationMode** is set to Renotify on Timer.
rowEnabledTable	Sets the enabled state for each row in the **valueTable**, so you can enable or disable each index in a tabular alert. This property only applies for tabular alerts, therefore the **useTabularDataFlag** must be selected to activate the **rowEnabledTable** property. **Note:** If the **enabledFlag** for the alert is disabled, each row index is disabled regardless of the value of **rowEnabledTable**. This property can be set to one of the following: **Scalar data** — If true, all rows (indexes) in the table will be enabled. If false all rows (indexes) will be disabled. **Tabular data** (one or two columns) — If your tabular input data has one column, the column must contain boolean values and should have the same number of rows as the **valueTable**. Each row in the **valueTable** will be compared against values in the corresponding rows of the specified **rowEnabledTable**. If the **valueTable** has more rows than the **rowEnabledTable**, these extra rows will be compared against the value of the first row of the **rowEnabledTable**. **Note:** This format cannot be used if you will be persisting your alerts. See "Alert Persistence" for more information. If your tabular input data has two columns, the first column must contain indexes and the second column must contain boolean values. The index value for each row in the **valueTable** will be used to lookup the corresponding enabled value from the **rowEnabledTable**. If the index is not found in the **rowEnabledTable**, the enabled state is set to false and the row (index) is disabled.

List of enabled (boolean) values

This should be a semi-colon (;) delimited list of boolean values (e.g.: **true;false;true**) with the same number of values as there are rows in the **valueTable**. Each row in **valueTable** will be set to corresponding enabled values in the specified list. If **valueTable** has more rows than enabled values in the list, then the first value in the list will be used.

Note: This format cannot be used if you will be persisting your alerts. See "Alert Persistence" for more information.

List of index/ enabled value pairs

This should be a semi-colon (;) delimited list of comma (,) separated index/enabled value pairs (e.g.: (**Chicago,true;Dallas,false;Detroit,true**). The index value for each row in the **valueTable** will be used to lookup the corresponding enabled value from the specified list. If the index is not found in the enabled value list, the enabled state is set to false and the row (index) is disabled.

Tabular data (four columns)

Specify a four column table that includes:

Alert Index (String)

For single index alerts, this column must contain the alert index.

For multiple column alerts, this column must contain concatenated values for the columns used by the specified Index Type. If you have not defined any index types, this column must contain the concatenated values for all index columns. When concatenating index column values, use the Multiple Index Delimiter as specified in **Application Options>**"Alert Definitions Tab"or on the command line. Default is tilde (**~**).

If the specified Index Type is **Default**, then use a value of **Default** for this column as well.

Index Type (String)

Enter one of the index types specified in the indexTypes property on this alert or one of the two built-in index types.

Default - This value will be used for any indexes that are not specified in this table.

All - The index value will be the concatenated values for all index columns.

Note: The **Default** and **All** index types are available for both single index and multiple index alerts even if no custom index types are specified in the **indexTypes** property.

Value (Boolean)

Enabled value to use for rows in the **valueTable** that match the specified Alert Index.

	Use Index Value (Boolean) Indicates whether or not to use this row. If false, this row will be ignored.
	When the Alert Index in a row of the **valueTable** matches more than one of the indexes specified in the **rowEnabledTable**, then the row with an Index Type of All takes precedence. For all other index types, the order in which they are specified on the **indexTypes** property controls the order of precedence. In the case where an Alert Index in the **valueTable** doesn't match any of the Alert Indexes in the **rowEnabledTable**, then the row with an Index Type set to **Default** will be used. If there is no row with an Index Type of **Default**, then the value of rows in the **valueTable** with indexes that don't match any of the indexes in the **rowEnabledTable** table will be false.
timeColumnName	Optionally specify a column in the **valueTable** to be used for the Last Update Time in the **AlertTable**. **Note:** The column specified must be of type date or long. If **timeColumnName** is not specified, not of type date or long, or is not found, then the time that the alert last received data will be used. **Note:** This property is only supported if **useTabularDataFlag** is selected and does not apply to scalar alerts. For scalar alerts, the time that the alert last received the data is always used.
useTabularDataFlag	For tabular input data, select **useTabularDataFlag** and attach your input to **valueTable**.
value	Attach your input data to this property if your input data is scalar. For tabular input data, select the **useTabularDataFlag** property and attach your input data to the **valueTable** property.
valueHighAlert	Specifies the value that the input value must be equal to for the high alert to execute. Enter either a scalar or tabular value, or specify a list of values. See "Discrete Alerts" for more information.
valueHighAlertCommandText	Specify alert command text to use for the **$alertCommandText** substitution. Enter either a scalar or tabular value. See "Discrete Alert Properties" and "Alert Command Text Values" for more information.
valueHighAlertEnabledFlag	Enables the high alert. When executed, severity is 3.
valueHighAlertText	Specify the alert text to display in the **Alert Text** column of the **AlertTable**. If no alert text is specified, the default alert text will be used. Enter a scalar or tabular value. See "Alert Text Values" for more information.
valueLowAlert	Specifies the value that the input value must be equal to for the low alert to execute. Enter either a scalar or tabular value, or specify a list of values. See "Discrete Alerts" for more information.
valueLowAlertCommandText	Specify alert command text to use for the **$alertCommandText** substitution. Enter either a scalar or tabular value. See "Discrete Alert Properties" and "Alert Command Text Values" for more information.
valueLowAlertEnabledFlag	Enables the low alert. When executed, severity is 1.
valueLowAlertText	Specify the alert text to display in the **Alert Text** column of the **AlertTable**. If no alert text is specified, the default alert text will be used. Enter a scalar or tabular value. See "Alert Text Values" for more information.

valueMediumAlert	Specifies the value that the input value must be equal to for the medium alert to execute. Enter either a scalar or tabular value, or specify a list of values. See "Discrete Alert Properties" and "Alert Command Text Values" for more information.
valueMediumAlertCommandText	Specify alert command text to use for the **$alertCommandText** substitution. Enter either a scalar or tabular value.
valueMediumAlertEnabledFlag	Enables the medium alert. When executed, severity is 2.
valueMediumAlertText	Specify the alert text to display in the **Alert Text** column of the **AlertTable**. If no alert text is specified, the default alert text will be used. Enter a scalar or tabular value. See "Alert Text Values" for more information.
valueTable	Attach your input data to this property if it is tabular. For scalar input data, deselect the **useTabularDataFlag** and attach your input data to the value property. Data attached to the **valueTable** property must contain at least two columns. The first one or more columns must be index columns. The combination of all index columns must be a unique value for each row and will be used as the Alert Index. If your data contains more than one index column, specify the index column names in the **indexColumnNames** property. Following the index column(s) must be one data column containing values to compare to the thresholds. Additional columns can be mapped to a Custom Alert Definition Property using the **customPropertyMap** property.

Multi State Alerts

The Multi State alert allows you to define any number of alert states. Each alert state will have a corresponding alert condition. The input value for the alert will be evaluated against all of the alert state conditions and an alert will be issued for highest (1 through N) condition that is met. When an alert is executed, the severity of the alert is set to the alert state number (1 through N), the corresponding alert command is executed and the Alert State is set to the highest active alert state number (1-N). If you would like an alert to be executed for each condition that is met, deselect the **skipDuplicateAlertsFlag** property.

Each alert state evaluation is comprised of two parts: **alertStateNCondition** and **alertStateNComparison**. The **alertStateNCondition** property sets the comparison type (ex. <, >, In Range, Out of Range, etc). The **alertStateNComparison** sets the corresponding comparison value to use in the condition evaluation. However if **alertStateNCondition** is set to In Range or Out of Range, then **alertStateNLowerRangeLimit** and **alertStateNUpperRangeLimit** will be used. The **alertStateNComparison**, **alertStateNLowerRangeLimit** and **alertStateNUpperRangeLimit** properties can contain a scalar, tabular or list of comparison values.

Attach the value property to your input data if the data is scalar. If the data is tabular, select the **useTabularDataFlag** and attach your input to **valueTable**. The data attached to the **valueTable** property must contain two columns, the first column must be an index column containing a unique value in each row. The second column must contain numeric values to evaluate against the conditions. An alert will execute for the highest alert state condition that each row in the table meets, unless the **skipDuplicateAlertsFlag** is turned off, in which case an alert will execute for each alert state condition that each row in the table meets.

When specifying a scalar value for **alertStateNComparison**, **alertStateNLowerRangeLimit** or **alertStateNUpperRangeLimit**, the value and each row in the **valueTable** will be compared against the scalar value.

When specifying a tabular value for **alertStateNComparison**, **alertStateNLowerRangeLimit** or **alertStateNUpperRangeLimit**, the table can contain one, two or four columns:

- If the specified **alertStateNComparison**, **alertStateNLowerRangeLimit** or **alertStateNUpperRangeLimit** table has one column, the column must contain numeric comparison values and should have the same number of rows as the input table attached to the **valueTable**. Each row in **valueTable** will be compared against values in the corresponding rows of the specified comparison table. If **valueTable** has more rows than the specified comparison table, these extra rows will be compared against the value of the first row of the specified comparison table. The value property will be compared to the first row of the specified comparison table. **Note:** This format cannot be used if you will be persisting your alerts. See "Alert Persistence" for more information.
- If the specified **alertStateNComparison**, **alertStateNLowerRangeLimit** or **alertStateNUpperRangeLimit** table has two columns, the first column must contain indexes and the second column must contain numeric comparison values. The index value for each row in the input table attached to **valueTable** will be used to lookup the corresponding comparison value from the specified comparison table. If the index is not found in the specified comparison table, no alert is activated. The value property cannot be compared against a comparison table with two columns (use the **valueTable** property instead).
- If the specified **alertStateNComparison**, **alertStateNLowerRangeLimit** or **alertStateNUpperRangeLimit** table has four columns, include the following:

Alert Index (String)	For single index alerts, this column must contain the alert index. For multiple column alerts, this column must contain concatenated values for the columns used by the specified Index Type. If you have not defined any index types, this column must contain the concatenated values for all index columns. When concatenating index column values, use the Multiple Index Delimiter as specified in **Application Options>**"Alert Definitions Tab" or on the command line. Default is tilde (~). If the specified Index Type is **Default**, then use a value of **Default** for this column as well.
Index Type (String)	Enter one of the index types specified in the **indexTypes** property on this alert or one of the two built-in index types: **Default** - This value will be used for any indexes that are not specified in this table. **All** - The index value will be the concatenated values for all index columns. **Note:** The **Default** and **All** index types are available for both single index and multiple index alerts even if no custom index types are specified in the **indexTypes** property.
Value (Double)	Enter comparison value to use for rows in the **valueTable** that match the specified Alert Index.
Use Index Value (Boolean)	Indicate whether or not to use this row. If false, this row will be ignored.

When the Alert Index in a row of the **valueTable** matches more than one of the indexes specified in the threshold table, then the row with an Index Type of **All** takes precedence. For all other index types, the order in which they are specified on the **indexTypes** property controls the order of precedence.

In the case where an Alert Index in the **valueTable** doesn't match any of the Alert Indexes in the threshold table, then the row with an Index Type set to **Default** will be used. If there is no row with an Index Type of **Default**, then the value of rows in the **valueTable** with indexes that don't match any of the indexes in the threshold table will not be evaluated.

When specifying a list of values for **alertStateNComparison**, **alertStateNLowerRangeLimit** or **alertStateNUpperRangeLimit**, the list can contain either index/comparison value pairs or just comparison values:

If the **alertStateNComparison** list only contains comparison values, this should be a semicolon (;) delimited list of comparison values (e.g.: 80;90;100) with the same number of values as there are rows in the input table attached to **valueTable**. Each row in **valueTable** will be compared against corresponding items in the specified comparison list. If **valueTable** has more rows than items in the specified threshold list, these extra rows will be compared against the first item in the specified threshold list. The value property will be compared to the first item the specified threshold list. **Note:** This format cannot be used if you will be persisting your alerts. See "Alert Persistence" for more information.

If the **alertStateNComparison** list contains index/comparison value pairs, this should be a semicolon (;) delimited list of comma (,) separated index/comparison value pairs (e.g.: **Chicago,80;Dallas,90;Detroit,100**). The index value for each row in the input table attached to **valueTable** will be used to lookup the corresponding comparison value from the specified comparison list. If the index is not found in the specified comparison list, no alert is activated. The value property cannot be compared against a comparison list of index/comparison value pairs (use the **valueTable** property instead).

Property Name	Description
alertClearedCommand	If specified, this command is executed when an alert is cleared. **Note:** This command supports all of the same alert substitutions as the **alertCommand**, as well as the following: **Substitution** — **Description** $alertClearedReason — Reason the alert was cleared. This is the same value in the Cleared Reason column of the "AlertTable". $alertClearedTime — Time the alert was cleared. It is the same value in the **Cleared Time** column of the "AlertTable".
alertCommand	The action executed when an alert is activated. Any available RTView commands (see "Define System Command") can be used, as well as the following substitutions: **Substitution** — **Description** $alertCommandText — Alert command text. $alertID — Unique ID for the alert. $alertCompValue — Comparison value. $alertCurValue — Current input value. $alertName — Value from the **alertName** property. $alertText — Alert text. This is the same text that is displayed in the "AlertTable".

	$alertSeverity	Severity of the alert.
	$alertEmailSubject	Alert email subject.
	$alertEmailBody	Alert email body text.
	$alertLabel	Label indicating the alert level (e.g. Alert State 1, Alert State 2, etc.)
	$alertIndex	Alert index. This is the same as the value that is shown in the Alert Index column of the "AlertTable".
	$alertTime	Time the alert was generated. This is the same value that is shown in the **Time** column of the "AlertTable".

alertDelayTime

Specifies the amount of time (in seconds) that a value must remain in an alert condition before the alert is updated or executed.

If the **skipDuplicateAlertsFlag** is selected it will apply to all alert states. In order for the alert state to change when the condition of one state is no longer met and the condition of another state is met, it must be meet the condition of the new state for more than the **alertDelayTime** specified before the alert severity is changed.

alertName

A unique name for the alert. This property is required and cannot be left blank.

To create a reusable Alert Definition file, include the substitution string as the suffix. For example, **salesAlert.$region** (where **salesAlert** is the **alertName** and **$region** is the substitution string). See "Creating a Reusable Alert Definition File" for more information.

alertStateNAlertCommandText

Specify alert command text to use for the **$alertCommandText** substitution.

Enter a scalar or tabular value. If the value is tabular, the input table requires two or four columns:

- If a two column table is specified, the first must contain indexes and the second must contain alert command text values. The index value for each row in the **valueTable** will be used to look up a corresponding index value in the alert command text table. If the index value is not found in the specified alert command text table and/or the **useTabularDataFlag** is not selected, the **$alertCommandText** substitution will set to an empty string.
- If a four column table is specified, include the following:

Alert Index

(String)

For single index alerts, this column must contain the alert index.

For multiple column alerts, this column must contain concatenated values for the columns used by the specified Index Type. If you have not defined any index types, this column must contain the concatenated values for all index columns. When concatenating index column values, use the Multiple Index Delimiter as specified in **Application Options>**"Alert Definitions Tab" or on the command line. Default is tilde (**~**).

If the specified Index Type is **Default**, then use a value of **Default** for this column as well.

Index Type

(String)

One of the index types specified in the **indexTypes** property on this alert or one of the two built-in index types:

Default - This value will be used for any indexes that are not specified in this table.

All - The index value will be the concatenated values for all index columns.

Note: The **Default** and **All** index types are available for both single index and multiple index alerts even if no custom index types are specified in the **indexTypes** property.

Value

(String)

Alert command text to use for the **$alertCommandText** substitution for rows in the **valueTable** that match the specified Alert Index.

Use Index Value

(Boolean)

Indicates whether or not to use this row. If false, this row will be ignored.

When the Index Value in a row of the **valueTable** matches more than one of the indexes specified, then an Index Type of **All** takes precedence. For all other index types, the order in which they are specified on the **indexTypes** property controls the order of precedence.

In the case where an Alert Index in the **valueTable** doesn't match any of the Alert Indexes, then the row with an Index Type set to **Default** will be used. If there is no row with an Index Type of Default, then rows in the **valueTable** with indexes that don't match any of the indexes in this table will use an empty string for the **$alertCommandText** substitution.

alertStateNAlertText

Specify the alert text to display in the **Alert Text** column of the "AlertTable". If no alert text is specified, the following is used:

Alert State N condition met: current value XX > comparison value YY

If a tabular value is specified, the input table requires two columns or four columns:

- If a two column table is specified, the first must contain indexes and the second must contain alert text values. The index value for each row in the **valueTable** will be used to look up a corresponding index value in the alert text table. If the index value is not found in the specified alert text table and/or the **useTabularDataFlag** is not selected, the default alert text will be used.
- If a four column table is specified, include the following:

Alert Index

(String)

For single index alerts, this column must contain the alert index.

For multiple column alerts, this column must contain concatenated values for the columns used by the specified Index Type. If you have not defined any index types, this column must contain the concatenated values for all index columns. When concatenating index column values, use the Multiple Index Delimiter as specified in **Application Options>**"Alert Definitions Tab" or on the command line. Default is tilde (~).

If the specified Index Type is **Default**, then use a value of **Default** for this column as well.

Index Type

(String)

One of the index types specified in the **indexTypes** property on this alert or one of the two built-in index types:

Default - This value will be used for any indexes that are not specified in this table.

All - The index value will be the concatenated values for all index columns.

Note: The **Default** and **All** index types are available for both single index and multiple index alerts even if no custom index types are specified in the **indexTypes** property.

Value

(String)

Alert text to display in the Alert Text column of the "AlertTable" for rows in the **valueTable** that match the specified Alert Index.

Use Index Value

(Boolean)

Indicates whether or not to use this row. If false, this row will be ignored.

When the Alert Index in a row of the **valueTable** matches more than one of the indexes specified in the **AlertText** table, then the row with an Index Type of All takes precedence. For all other index types, the order in which they are specified on the indexTypes property controls the order of precedence.

In the case where an Alert Index in the **valueTable** doesn't match any of the Alert Indexes in the **AlertText** table, then the row with an Index Type set to Default will be used. If there is no row with an Index Type of **Default**, then the Alert Text for rows in the **valueTable** with indexes that don't match any of the indexes in the **AlertText** table will use the default alert text.

The following substitutions are supported in the alert text.

The **$alertText** and **$alertEmailBody** substitutions contain the same string shown in the Alert Text column of the "AlertTable".

Substitution	Description
$alertID	The unique ID for the alert.
$alertCompValue	The comparison value.
$alertCurValue	The current input value.
$alertName	The value from the **alertName** property.
$alertSeverity	The severity of the alert.
$alertEmailSubject	The alert email subject.
$alertLabel	A label indicating the alert level (ex. Alert State 1, Alert State 2), etc.
$alertIndex	The alert index. This is the same as the value in the Alert Index column of the **AlertTable**.

alertStateNCondition — Condition to evaluate for Alert State N.

Condition	Description

<	An alert is executed when the value is less than the corresponding **alertStateNComparison**. It is cleared when the value is equal to or greater than **alertStateNComparison** if the corresponding **alertStateNValueDeadband** is 0. If the corresponding **alertStateNValueDeadband** is not 0, the alert will be cleared when the value is equal to or greater than **alertStateNComparison** plus **alertStateNValueDeadband**. The value and **alertStateNComparison** must be numbers. The following properties are disabled: **alertStateNUpperRangeLimit** and **alertStateNLowerRangeLimit**.
>	An alert is executed when the value is greater than the corresponding **alertStateNComparison**. It is cleared when the value is equal to or less than **alertStateNComparison** if the corresponding **alertStateNValueDeadband** is 0. If the corresponding **alertStateNValueDeadband** is not 0, the alert will be cleared when the value is equal to or less than alertStateNComparison minus **alertStateNValueDeadband**. The value and **alertStateNComparison** must be numbers. The following properties are disabled: **alertStateNUpperRangeLimit** and **alertStateNLowerRangeLimit**.
<=	An alert is executed when the value is less than or equal to the corresponding **alertStateNComparison**. It is cleared when the value is greater than **alertStateNComparison** if the corresponding **alertStateNValueDeadband** is 0. If the corresponding **alertStateNValueDeadband** is not 0, the alert will be cleared when the value is greater than **alertStateNComparison** plus **alertStateNValueDeadband**. The value and **alertStateNComparison** must be numbers. The following properties are disabled: **alertStateNUpperRangeLimit** and **alertStateNLowerRangeLimit**.
>=	An alert is executed when the value is greater than or equal to the corresponding **alertStateNComparison**. It is cleared when the value is less than **alertStateNComparison** if the corresponding **alertStateNValueDeadband** is 0. If the corresponding **alertStateNValueDeadband** is not 0, the alert will be cleared when the value is less than **alertStateNComparison** minus **alertStateNValueDeadband**. The value and **alertStateNComparison** must be numbers. The following properties are disabled: **alertStateNUpperRangeLimit** and **alertStateNLowerRangeLimit**.

=	An alert is executed when the value is equal to the corresponding **alertStateNComparison**. It is cleared when the value is not equal to **alertStateNComparison**. The value and **alertStateNComparison** must be numbers. The following properties are disabled: **alertStateNValueDeadband**, **alertStateNUpperRangeLimit** and **alertStateNLowerRangeLimit**.
!=	An alert is executed when the value is not equal to the corresponding **alertStateNComparison**. It is cleared when the value is equal to **alertStateNComparison**. The value and **alertStateNComparison** must be numbers. The following properties are disabled: **alertStateNValueDeadband**, **alertStateNUpperRangeLimit** and **alertStateNLowerRangeLimit**.
In Range	An alert is executed when the value is in the range defined by **alertStateNLowerRangeLimit** and **alertStateNUpperRangeLimit** (inclusive of the range limits). It is cleared when the value is outside this range if the corresponding **alertStateNValueDeadband** is 0. If the corresponding **alertStateNValueDeadband** is not 0, the alert will be cleared when the value is not in the range defined by **alertStateNLowerRangeLimit** minus the **alertStateNValueDeadband** and **alertStateNUpperRangeLimit** plus **alertStateNValueDeadband**. The value, **alertStateNUpperRangeLimit** and **alertStateNLowerRangeLimit** must be numbers. The following property is disabled: **alertStateNComparison**.
Out of Range	An alert is executed when the value is not in the range defined by **alertStateNLowerRangeLimit** and **alertStateNUpperRangeLimit** (inclusive of the range limits). It is cleared when the value is in this range if the corresponding **alertStateNValueDeadband** is 0. If the corresponding **alertStateNValueDeadband** is not 0, the alert will be cleared when the value is in the range defined by **alertStateNLowerRangeLimit** minus the **alertStateNValueDeadband** and **alertStateNUpperRangeLimit** plus **alertStateNValueDeadband**. The value, **alertStateNUpperRangeLimit** and **alertStateNLowerRangeLimit** must be numbers. The following property is disabled: **alertStateNComparison**.

Increase	An alert is executed if the difference between the current value and the previous value is an increase greater than or equal to **alertStateNComparison**. It is cleared if the difference between the current value and the previous value is not an increase greater than or equal to **alertStateNComparison**. Since this condition compares the current value to the previous value, this alert will never execute until the alert object has received at least two updates. The value and **alertStateNComparison** must be numbers. The following properties are disabled: **alertStateNValueDeadband**, **alertStateNUpperRangeLimit** and **alertStateNLowerRangeLimit**.
%Increase	An alert is executed if the difference between the current value and the previous value is an increase greater than or equal to **alertStateNComparison** percent. It is cleared if the difference between the current value and the previous value is not an increase greater than or equal to **alertStateNComparison** percent. Since this condition compares the current value to the previous value, this alert will never execute until the alert object has received at least two updates. The value and **alertStateNComparison** must be numbers. The following properties are disabled: **alertStateNValueDeadband** ,**alertStateNUpperRangeLimit** and **alertStateNLowerRangeLimit**.
Decrease	An alert is executed if the difference between the current value and the previous value is a decrease greater than or equal to **alertStateNComparison**. It is cleared if the difference between the current value and the previous value is not a decrease greater than or equal to **alertStateNComparison**. Since this condition compares the current value to the previous value, this alert will never execute until the alert object has received at least two updates. The value and **alertStateNComparison** must be numbers. The following properties are disabled: **alertStateNValueDeadband**, **alertStateNUpperRangeLimit** and **alertStateNLowerRangeLimit**.
%Decrease	An alert is executed if the difference between the current value and the previous value is a decrease greater than or equal to **alertStateNComparison** percent. It is cleared if the difference between the current value and the previous value is not a decrease greater than or equal to **alertStateNComparison** percent. Since this condition compares the current value to the previous value, this alert will never execute until the alert object has received at least two updates. The value and **alertStateNComparison** must be numbers. The following properties are disabled: **alertStateNValueDeadband**, **alertStateNUpperRangeLimit** and **alertStateNLowerRangeLimit**.

	Net Change	An alert is executed if the difference between the current value and the previous value is an increase or decrease greater than or equal to **alertStateNComparison**. It is cleared if the difference between the current value and the previous value is not an increase or decrease greater than or equal to **alertStateNComparison**. Since this condition compares the current value to the previous value, this alert will never execute until the alert object has received at least two updates. The value and **alertStateNComparison** must be numbers. The following properties are disabled: **alertStateNValueDeadband**, **alertStateNUpperRangeLimit** and **alertStateNLowerRangeLimit**.
	% Net Change	An alert is executed if the difference between the current value and the previous value is an increase or decrease greater than or equal to **alertStateNComparison** percent. It is cleared if the difference between the current value and the previous value is not an increase or decrease greater than or equal to **alertStateNComparison** percent. Since this condition compares the current value to the previous value, this alert will never execute until the alert object has received at least two updates. The value and **alertStateNComparison** must be numbers. The following properties are disabled: **alertStateNValueDeadband**, **alertStateNUpperRangeLimit** and **alertStateNLowerRangeLimit**.
alertStateNComparison	Specifies the value used for evaluation in the corresponding **alertStateNCondition**. Not used when **alertStateNCondition** is In Range or Out of Range. Enter either a scalar or tabular value, or specify a list of values.	
alertStateNUpperRangeLimit	Specifies the upper value of the range in the corresponding **alertStateNCondition**. Used only when **alertStateNCondition** is In Range or Out of Range. Enter either a scalar or tabular value, or specify a list of values.	
alertStateNLowerRangeLimit	Specifies the lower value of the range in the corresponding **alertStateNCondition**. Used only when **alertStateNCondition** is In Range or Out of Range. Enter either a scalar or tabular value, or specify a list of values.	
alertStateNEnabledFlag	Enables the evaluation of the corresponding **alertStateNCondition**. If an alert is executed, the severity for that alert is set to N.	
alertStateNValueDeadband	Specifies a deadband value for the corresponding **alertStateNCondition**. If specified, the input value combined with this value must not meet the corresponding **alertStateNCondition** in order to clear. For example, if the **alertStateNCondition** is >, the alert will execute when value is greater than **alertStateNComparison**, and will clear when the value is less than or equal to **alertStateNComparison** minus the deadband. Used only for some **alertStateNCondition** types. See the description for each **alertStateNCondition** type for more information on how this property works with that type of evaluation.	
commentAddedCommand	If specified, this command will be executed after a comment is added to the alert by the **Add Comment** command (see "Command Types"). All substitutions supported for **alertCommand** are also supported for this command, as well as the following two additional substitutions:	

	Substitution	Description
	$alertComment	This is set to the value of the **Comments** field for the command after the new comment has been added.
	$alertLastComment	This is set to the value of the last comment added to the **Comments** field.

customPropertyMap	Use one or more columns from the **valueTable** as the value for one or more Custom Alert Definition Properties that you defined in the "Custom Alert Fields Tab" in the **Application Options** dialog. Use the following syntax: **customPropName:valueTableColumnName;customPropName2:valueTableColumnName2** For example, if you have a defined a Custom Alert Definition Property named My Custom Property and you want to use the value from the **My Data Column** column in your **valueTable**, you would specify the following: **My Custom Property:My Data Column** **Note:** In order to use **customPropertyMap**, the **useTabularDataFlag** property must be selected.
description	If specified, a description of the alert will be displayed in the **Alerts** dialog and in the **Description** column of the Alert Variables Table.
enabledFlag	Enables or disables the entire alert. If this alert is active when it is disabled, it will be cleared. **Note:** It is not recommended that you attach data to this property if you are using the **Enable Alert Definition** command for this alert. See "Command Types" for more information.
indexColumnNames	Enter one or more index column name(s). If left blank, the alert is assumed to have a single index column and the first column of the **valueTable** will be used. For multiple index columns, enter a ; (semicolon) delimited list of column names. The **valueTable** must contain all specified index columns in addition to, and preceding, the required data column. The combination of all index column values in a single row must uniquely identify that Alert Index.
indexTypes	For alerts with multiple indexes, create custom index types for use in the following tables (**alertStateNAlertText**, **alertStateNAlertCommandText** and **rowEnabledTable**) so you can specify a value for a group of alert indexes instead of having to specify a value for each unique alert index combination. Use the following syntax to map one or more index columns to an index type: **typeName:columnName;typeName2;columnName1,columnName2** For example: **PerRegion:Region;PerRegionalService:Region,Service** **Note: Default** and **All** are built-in index types and therefore cannot be entered as a **typeName**.
nonRepetitionTime	Amount of time (in seconds) that must pass after a cleared alert is again executed. If the **skipDuplicateAlertsFlag** is selected, the alert will not be cleared until all alert state conditions are cleared.

numAlertStates	Number of Alert Conditions to Evaluate. This number of **alertStateNCondition**, **alertStateNComparison**, **alertStateNEnabledFlag**, **alertStateNLowerRangeLimit**, **alertStateNUpperRangeLimit** and **alertStateNValueDeadband** properties is generated. (i.e. **alertState1Condition**, **alertState1Comparison**, etc). This property is static and cannot be attached to data.
reNotificationCommand	If specified, this command will be used instead of the value of the **alertCommand** property for renotifications. Otherwise, the specified **alertCommand** will be used. **Note:** This command supports all of the same alert substitutions as the **alertCommand**.
reNotificationMode	Configure how an alert will renotify. Default setting is Renotify on Timer. Choose from the following options: **None** - Do not renotify. The **alertCommand** is executed only once when the alert is activated. **Renotify on Timer** - Renotify based on **reNotificationTime** property. The **alertCommand** is executed once when the alert is activated and then re-executed every **reNotificationTime** (seconds) until the alert is cleared or acknowledged. If the **reNotificationTime** is set to 0, then the alert will not renotify. **Renotify on Data Changed** - Renotify when the input value changes. The **alertCommand** is executed once when the alert is activated and again when a different value is received until the alert is cleared or acknowledged. The new value must be different than the previous value for the alert to renotify. **Renotify on Data Updates** - Renotify when the input value is updated. The **alertCommand** is executed once when the alert is activated and again whenever a value is received until the alert is cleared or acknowledged. The new data value may be the same or different than the previous value for the alert to renotify.
reNotificationTime	Amount of time (in seconds) that must pass before the **alertCommand** for an unacknowledged or an uncleared alert is re-executed. The **alertCommand** will continue to re-execute on this interval until the alert is acknowledged or cleared. The **alertCommand** will continue to re-execute on this interval until the alert is acknowledged or cleared. If set to 0, the **alertCommand** will only execute once. **Note:** This property will be ignored unless the **reNotificationMode** is set to Renotify on Timer.

reNotifyOnSevChangeMode	Specifies the conditions for alerts to send renotification when the alert severity increases. There are three modes: • **Renotify on First Sev Change**: The alert renotifies when the severity of the alert changes for the first time. • **Renotify on All Sev Increases**: The alert renotifies every time the severity increases. • **None**: Do not renotify when the severity changes. For the **Renotify on First Sev Change** and **Renotify on All Sev Increases** modes, alert renotification only occurs if the alert is either not acknowledged, or the **Unacknowledge Alerts on First Severity Change** application option is enabled. The renotification executes the **reNotificationCommand** if configured, otherwise it executes the **alertCommand**. By default this option is **None**. For the **Renotify on First Sev Change** and **Renotify on All Sev Increases** modes, alert renotification only occurs if the alert is not acknowledged. The renotification executes the **reNotificationCommand** if configured, otherwise it executes the **alertCommand**. To revert acknowledged alerts to unacknowledged when the severity changes, select the **Unacknowledge on Severity Change Mode** application option. To update the severity of acknowledged alerts you must select the **Update Severity on Acknowledged Alerts** application option. See "Alerts Tab" for more information. **Note:** When run against alerts configured in an older version of RTView, the correct renotification mode is automatically applied. That is, the **reNotifyOnFirstSevChangeFlag** property (from an older RTView version) converts to the equivalent value for **reNotifyOnSevChangeMode** so the alert behavior is the same.
rowEnabledTable	Sets the enabled state for each row in the **valueTable**, so you can enable or disable each index in a tabular alert. This property only applies for tabular alerts, therefore the **useTabularDataFlag** must be selected to activate the **rowEnabledTable** property. **Note:** If the **enabledFlag** for the alert is disabled, each row index is disabled regardless of the value of **rowEnabledTable**. This property can be set to one of the following: **Scalar data** If true, all rows (indexes) in the table will be enabled. If false all rows (indexes) will be disabled. **Tabular data** **(one or two columns)** If your tabular input data has one column, the column must contain boolean values and should have the same number of rows as the **valueTable**. Each row in the **valueTable** will be compared against values in the corresponding rows of the specified **rowEnabledTable**. If the **valueTable** has more rows than the **rowEnabledTable**, these extra rows will be compared against the value of the first row of the **rowEnabledTable**. **Note:** This format cannot be used if you will be persisting your alerts. See "Alert Persistence" for more information. If your tabular input data has two columns, the first column must contain indexes and the second column must contain boolean values. The index value for each row in the **valueTable** will be used to lookup the corresponding enabled value from the **rowEnabledTable**. If the index is not found in the **rowEnabledTable**, the enabled state is set to false and the row (index) is disabled.

List of enabled (boolean) values

This should be a semi-colon (;) delimited list of boolean values (e.g.: **true**;**false**;**true**) with the same number of values as there are rows in the **valueTable**. Each row in **valueTable** will be set to corresponding enabled values in the specified list. If **valueTable** has more rows than enabled values in the list, then the first value in the list will be used.

Note: This format cannot be used if you will be persisting your alerts. See "Alert Persistence" for more information.

List of index/enabled value pairs

This should be a semi-colon (;) delimited list of comma (,) separated index/enabled value pairs (e.g.: (**Chicago,true**;**Dallas,false**;**Detroit,true**). The index value for each row in the **valueTable** will be used to lookup the corresponding enabled value from the specified list. If the index is not found in the enabled value list, the enabled state is set to false and the row (index) is disabled.

Tabular data

(four columns)

Specify a four column table that includes:

Alert Index (String)

For single index alerts, this column must contain the alert index.

For multiple column alerts, this column must contain concatenated values for the columns used by the specified Index Type. If you have not defined any index types, this column must contain the concatenated values for all index columns. When concatenating index column values, use the Multiple Index Delimiter as specified in **Application Options>**"Alert Definitions Tab" or on the command line. Default is tilde (**~**).

If the specified Index Type is **Default**, then use a value of **Default** for this column as well.

Index Type (String)

Enter one of the index types specified in the **indexTypes** property on this alert or one of the two built-in index types:

Default - This value will be used for any indexes that are not specified in this table.

All - The index value will be the concatenated values for all index columns.

Note: The **Default** and **All** index types are available for both single index and multiple index alerts even if no custom index types are specified in the **indexTypes** property.

Value (Boolean)

Enabled value to use for rows in the **valueTable** that match the specified Alert Index.

Use Index Value (Boolean)

Indicates whether or not to use this row. If false, this row will be ignored.

	When the Alert Index in a row of the **valueTable** matches more than one of the indexes specified in the **rowEnabledTable**, then the row with an Index Type of All takes precedence. For all other index types, the order in which they are specified on the **indexTypes** property controls the order of precedence. In the case where an Alert Index in the **valueTable** doesn't match any of the Alert Indexes in the **rowEnabledTable**, then the row with an Index Type set to Default will be used. If there is no row with an Index Type of Default, then the value of rows in the **valueTable** with indexes that don't match any of the indexes in the **rowEnabledTable** table will be false.
skipDuplicateAlertsFlag	If selected, the severity of an alert will update without multiple alerts being activated. That is, only the highest severity alert will be activated when the input meets multiple conditions. If deselected, an alert will be issued for each **alertStateNCondition** that evaluates to true. **Note:** By default, once an alert is acknowledged the severity of that alert will no longer update. To enable updating the severity of acknowledged alerts you must select the **Update Severity on Acknowledged Alerts** application option. See "Alerts Tab" for more information.
timeColumnName	Optionally specify a column in the **valueTable** to be used for the **Last Update Time** in the **AlertTable**. **Note:** The column specified must be of type date or long. If **timeColumnName** is not specified, not of type date or long, or is not found, then the time that the alert last received data will be used. **Note:** This property is only supported if **useTabularDataFlag** is selected and does not apply to scalar alerts. For scalar alerts, the time that the alert last received the data is always used.
useTabularDataFlag	For tabular input data, select **useTabularDataFlag** and attach your input to **valueTable**.
value	Attach your input data to this property if your input data is scalar. For tabular input data, select the **useTabularDataFlag** property and attach your input data to the **valueTable** property.
valueCommandFormat	Specify the numeric format (using syntax from the Java DecimalFormat class) for the following **alertCommand** substitutions: **$alertCurValue** and **$alertCompValue**. If left blank, values that are eight whole numbers or more will be written in exponential format.
valueTable	Attach your input data to this property if it is tabular. For scalar input data, deselect the **useTabularDataFlag** and attach your input data to the value property. Data attached to the **valueTable** property must contain at least two columns. The first one or more columns must be index columns. The combination of all index columns must be a unique value for each row and will be used as the Alert Index. If your data contains more than one index column, specify the index column names in the **indexColumnNames** property. Following the index column(s) must be one data column containing numeric values to compare to the thresholds. Additional columns can be mapped to a Custom Alert Definition Property using the **customPropertyMap** property.

Event Alerts

The Event alert is meant to be used as a wrapper around an external event source such as IBM® Netcool. For this alert, RTView does not do a threshold comparison to determine when to execute/clear an alert or to determine the severity of the alert. Instead, columns in the valueTable are mapped to the columns in the RTView **AlertTable** using the **valueTableMap** and **customPropertyMap** properties.

Event alerts behave like the other alerts in terms of notifications (i.e. **alertCommand**, **reNotificationCommand**, **commentAddedCommand**, **alertClearedCommand**). They also appear in the **AlertTable** like the other alerts. However, Event alerts do not have a corresponding per-tabular-alert-table like Limits and Discrete alerts. If indexed, the indexes for these alerts are only stored until the alert is cleared and then removed.

The **valueTable** for this alert does not have to be indexed. In order to indicate that you have an index, map the Alert Index property to the index column in your **valueTable** using the **valueTableMap**.

- If no Alert Index is specified in the **valueTableMap**, this indicates that the **valueTable** is not indexed and an alert will be generated for each row of each table. No check for duplicates and no support for status change is supported. This means that the only way for an alert to be cleared is via the **alertExpireTime** property, which will clear the alert after it has been active for the specified amount of time.
- If there is an Alert Index specified in the **valueTableMap**, this indicates that the **valueTable** is indexed. In this case, an alert will be generated for each row where there is not already an active alert for that index. Indexed alerts can be cleared via the **valueTable** if the **Cleared** column is also specified in the **valueTableMap**. **Note:** The value in the **Cleared** column must equal the **alertClearedValue** property. Indexed alerts can also be cleared via **alertExpireTime**. Indexed alerts support updates to the **Severity**, **Cleared**, and **Custom Property** columns.

Property Name	Description
alertClearedCommand	If specified, this command is executed when an alert is cleared. **Note:** This command supports all of the same alert substitutions as the alertCommand, as well as the following:

Substitution	Description
$alertClearedReason	Reason the alert was cleared. This is the same value in the Cleared Reason column of the "AlertTable"
$alertClearedTime	Time the alert was cleared. It is the same value in the Cleared Time column of the "AlertTable".

alertClearedValue	The value specified here will be compared to the value in the **Cleared** column from the **valueTableMap**. If they match, the alert will be cleared. The **alertClearedValue** property is only supported if there is an Alert Index specified in the **valueTableMap** property. If no Alert Index is specified, the only way for an alert to be cleared is via the **alertExpireTime** property.
alertCommand	The action executed when an alert is activated. Any available RTView commands (see "Command Types") can be used, as well as the following substitutions:

Substitution	Description
$alertCommandText	Alert command text.
$alertCompValue	Value the current input value is being compared against.
$alertCurValue	Current input value.
$alertEmailBody	Alert email body text.

	$alertEmailSubject	Alert email subject.
	$alertID	Unique ID for the alert.
	$alertIndex	Alert index. This is the same as the value in the Alert Index column of the "AlertTable".
	$alertLabel	Label indicating the alert type (Discrete alert types: High Alert, Medium Alert, Low Alert; Limits alert types: High Alert, High Warning, Low Alert, Low Warning)
	$alertName	Value from the **alertName** property.
	$alertSeverity	Severity of the alert.
	$alertText	Alert text. This is the same text that is displayed in the "AlertTable".
	$alertTime	Time the alert was generated. This is the same value in the Time column of the "AlertTable".
alertExpireMode	Initial Time	If selected, the alert will clear if the **alertExpireTime** has passed since the alert was generated. The value of **alertExpireTime** must be greater than 0.
	Last Update Time	If selected, the alert will clear if the **alertExpireTime** has passed since the alert was received. The value of **alertExpireTime** must be greater than 0. **Note:** Only indexed Event Alerts can receive data updates, so this option will not work for non-indexed alerts.
alertExpireTime	Specify (in seconds) how long an alert can remain active before it is cleared. **Note:** Alerts will be cleared regardless of whether an alert has been acknowledged or not.	
alertName	A unique name for the alert. This property is required and cannot be left blank. To create a reusable Alert Definition file, include the substitution string as the suffix. For example, **salesAlert.$region** (where **salesAlert** is the **alertName** and **$region** is the substitution string). See "Creating a Reusable Alert Definition File" for more information.	
commentAddedCommand	If specified, this command will be executed after a comment is added to the alert by the **Add Comment** command (see "Command Types"). All substitutions supported for **alertCommand** are also supported for this command, as well as the following two additional substitutions:	
	Substitution	**Description**
	$alertComment	This is set to the value of the **Comments** field for the command after the new comment has been added.

	$alertLastComment	This is set to the value of the last comment added to the **Comments** field.

customPropertyMap	Use one or more columns from the **valueTable** as the value for one or more Custom Alert Definition Properties that you defined in the "Custom Alert Fields Tab" in the **Application Options** dialog.
	Use the following syntax:
	`customPropName:valueTableColumnName;customPropName2:valueTableColumnName2`
	For example, if you have a defined a Custom Alert Definition Property named **My Custom Property** and you want to use the value from the **My Data Column** column in your **valueTable**, you would specify the following:
	`My Custom Property:My Data Column`

description	If specified, a description of the alert will be displayed in the **Alerts** dialog and in the **Description** column of the **Alert Variables Table**.
enabledFlag	Enables or disables the entire alert. If this alert is active when it is disabled, it will be cleared. **Note:** It is not recommended that you attach data to this property if you are using the **Enable Alert Definition** command for this alert. See "Command Types" for more information.
indexColumnNames	Enter one or more index column name(s). If left blank, the Alert Index value in the **valueTableMap** will be used. For multiple index columns, enter a ; (semicolon) delimited list of column names. The combination of all index column values in a single row must uniquely identify that Alert Index. **Note:** If the **valueTableMap** contains an index column and **indexColumnNames** is specified, the **indexColumnNames** property takes precedence.

indexTypes	For alerts with multiple indexes, create custom index types for use in the **rowEnabledTable** so you can specify a value for a group of alert indexes instead of having to specify a value for each unique alert index combination.
	Use the following syntax to map one or more index columns to an index type:
	`typeName:columnName;typeName2;columnName1,columnName2`
	For example:
	`PerRegion:Region;PerRegionalService:Region,Service`
	Note: Default and **All** are built-in index types and therefore cannot be entered as a **typeName**.

reNotificationCommand	If specified, this command will be used instead of the value of the **alertCommand** property for renotifications. Otherwise, the specified **alertCommand** will be used. **Note:** This command supports all of the same alert substitutions as the **alertCommand**.

reNotificationMode	Configure how an alert will renotify. Default setting is Renotify on Timer. Choose from the following options: **None** - Do not renotify. The **alertCommand** is executed only once when the alert is activated. **Renotify on Timer** - Renotify based on **reNotificationTime** property. The **alertCommand** is executed once when the alert is activated and then re-executed every **reNotificationTime** (seconds) until the alert is cleared or acknowledged. If the **reNotificationTime** is set to 0, then the alert will not renotify. **Renotify on Data Changed** - Renotify when the severity of an alert changes. The **alertCommand** is executed once when the alert is activated and again when a different value is received until the alert is cleared or acknowledged. The new value must be different than the previous value for the alert to renotify. **Renotify on Data Updates** - Renotify when the input value is updated. The **alertCommand** is executed once when the alert is activated and again whenever a value is received until the alert is cleared or acknowledged. The new data value may be the same or different than the previous value for the alert to renotify.
reNotificationTime	Amount of time (in seconds) that must pass before the **alertCommand** for an unacknowledged or an uncleared alert is re-executed. The **alertCommand** will continue to re-execute on this interval until the alert is acknowledged or cleared. **Note:** This property will be ignored unless the **reNotificationMode** is set to **Renotify on Timer**.
reNotifyOnSevChangeMode	Specifies the conditions for alerts to send renotification when the alert severity increases. There are three modes: • **Renotify on First Sev Change**: The alert renotifies when the severity of the alert changes for the first time. • **Renotify on All Sev Increases**: The alert renotifies every time the severity increases. • **None**: Do not renotify when the severity changes. For the **Renotify on First Sev Change** and **Renotify on All Sev Increases** modes, alert renotification only occurs if the alert is either not acknowledged, or the **Unacknowledge Alerts on First Severity Change** application option is enabled. The renotification executes the **reNotificationCommand** if configured, otherwise it executes the **alertCommand**. By default this option is **None**. For the **Renotify on First Sev Change** and **Renotify on All Sev Increases** modes, alert renotification only occurs if the alert is not acknowledged. The renotification executes the **reNotificationCommand** if configured, otherwise it executes the **alertCommand**. To revert acknowledged alerts to unacknowledged when the severity changes, select the **Unacknowledge on Severity Change Mode** application option. To update the severity of acknowledged alerts you must select the **Update Severity on Acknowledged Alerts** application option. See "Alerts Tab" for more information. **Note:** When run against alerts configured in an older version of RTView, the correct renotification mode is automatically applied. That is, the **reNotifyOnFirstSevChangeFlag** property (from an older RTView version) converts to the equivalent value for **reNotifyOnSevChangeMode** so the alert behavior is the same.

rowEnabledTable

The **rowEnabledTable** property is only supported if there is an Alert Index specified in the **valueTableMap** property.

Sets the enabled state for each row in the **valueTable**, so you can enable or disable each index in a tabular alert. If the **enabledFlag** for the alert is disabled, each row index is disabled regardless of the value of **rowEnabledTable**.

This property can be set to one of the following:

Scalar data

If true, all rows (indexes) in the table will be enabled. If false all rows (indexes) will be disabled.

Tabular data

(one or two columns)

If your tabular input data has one column, the column must contain boolean values and should have the same number of rows as the **valueTable**. Each row in the **valueTable** will be compared against values in the corresponding rows of the specified **rowEnabledTable**. If the **valueTable** has more rows than the **rowEnabledTable**, these extra rows will be compared against the value of the first row of the **rowEnabledTable**.

Note: This format cannot be used if you will be persisting your alerts. See "Alert Persistence Tab" for more information.

If your tabular input data has two columns, the first column must contain indexes and the second column must contain boolean values. The index value for each row in the **valueTable** will be used to lookup the corresponding enabled value from the **rowEnabledTable**. If the index is not found in the **rowEnabledTable**, the enabled state is set to false and the row (index) is disabled.

List of enabled (boolean) values

This should be a semi-colon (;) delimited list of boolean values (e.g.: true;false;true) with the same number of values as there are rows in the **valueTable**. Each row in **valueTable** will be set to corresponding enabled values in the specified list. If **valueTable** has more rows than enabled values in the list, then the first value in the list will be used.

Note: This format cannot be used if you will be persisting your alerts. See "Alert Persistence" for more information.

List of index/enabled value pairs

This should be a semi-colon (;) delimited list of comma (,) separated index/enabled value pairs (e.g.: (Chicago,true;Dallas,false;Detroit,true). The index value for each row in the valueTable will be used to lookup the corresponding enabled value from the specified list. If the index is not found in the enabled value list, the enabled state is set to false and the row (index) is disabled.

Tabular data

(four columns)

Specify a four column table that includes:

Alert Index (String)

For single index alerts, this column must contain the alert index.

For multiple column alerts, this column must contain concatenated values for the columns used by the specified Index Type. If you have not defined any index types, this column must contain the concatenated values for all index columns. When concatenating index column values, use the Multiple Index Delimiter as specified in **Application Options>**"Alert Definitions Tab" or on the command line. Default is tilde (**~**).

If the specified Index Type is **Default**, then use a value of **Default** for this column as well.

Index Type (String)

Enter one of the index types specified in the **indexTypes** property on this alert or one of the two built-in index types:

Default - This value will be used for any indexes that are not specified in this table.

All - The index value will be the concatenated values for all index columns

Note: The **Default** and **All** index types are available for both single index and multiple index alerts even if no custom index types are specified in the **indexTypes** property.

Value (Boolean)

Enabled value to use for rows in the **valueTable** that match the specified Alert Index.

Use Index Value (Boolean)

Indicates whether or not to use this row. If false, this row will be ignored.

When the Alert Index in a row of the **valueTable** matches more than one of the indexes specified in the **rowEnabledTable**, then the row with an Index Type of All takes precedence. For all other index types, the order in which they are specified on the **indexTypes** property controls the order of precedence.

In the case where an Alert Index in the **valueTable** doesn't match any of the Alert Indexes in the **rowEnabledTable**, then the row with an Index Type set to Default will be used. If there is no row with an Index Type of Default, then the value of rows in the **valueTable** with indexes that don't match any of the indexes in the **rowEnabledTable** table will be false.

valueTable

This table is required, but all columns are optional. Use the **valueTableMap** property to map one or more columns in your data to standard columns in the "AlertTable". If your events are indexed with more than one column, use the **indexColumnNames** property to map your multiple index columns to the Alert Index. Additional columns can be mapped to a Custom Alert Definition Property using the **customPropertyMap** property.

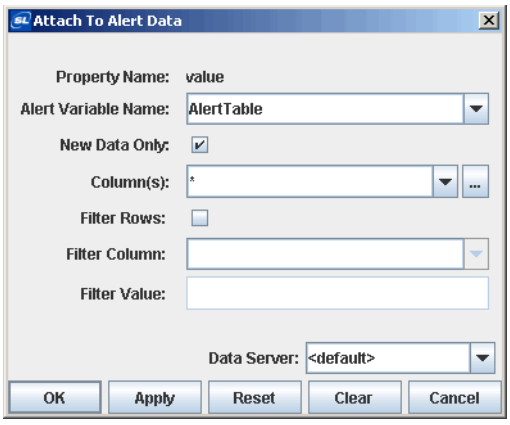
valueTableMap	Map one or more items in the **valueTable** to standard columns in the **AlertTable**. Use the following syntax: `AlertProp:valueTableCol;AlertProp2:valueTableCol2` The following Alert properties are supported, but none are required: Alert Index - If specified, this is used as the index column and will be used in the Alert Index column of the "AlertTable". The **Cleared** column and **alertClearedValue** and **rowEnabledTable** properties will be supported. If not specified, this is an un-indexed table and any new rows coming in to the **valueTable** will just be forwarded to the **AlertTable** with no way to update an existing row. **Time** - If specified, this time will be used in the Last Update Time column of the "AlertTable". The data type of the value of this column must be Long or Date. If not specified, RTView will set the **Last Update Time** to the time the alert received the last update. **Severity** - If specified, this column must contain integers > 0. If a value less than 1 is specified, a value of 1 will be used. This column will be used to set the value in the Severity column of the "AlertTable". The severity of an alert will update whenever the mapped column changes. If not specified, RTView will assign all alerts a Severity value of 1. **Note:** To enable updating the severity of acknowledged alerts you must select the **Update Severity on Acknowledged Alerts** application option. See "Alerts Tab" for more information. **Alert Text** - If specified, the value in this column will be used in the **Alert Text** field of the "AlertTable". If not specified, RTView will assign an Alert Text of Event Received. **Cleared** - If specified, the value in this column will be compared to the **alertClearedValue** property. If they match, the alert will be cleared. This is only supported if an Alert Index was specified. If no Alert Index is specified, the only way for an alert to be cleared is via the **alertExpireTime** property.

Attach to Alert Data

This section contains the following:

Attach to Alert Data Dialog

You can access the **Attach to Alert Data** dialog, which is used to connect an object property to information about your alerts, from the **Object Properties** window. Once a property has been attached to alert data, it receives continuous updates.



Right-click on the Property Name from the **Object Properties** window and select **Attach to Data>Alert** to display the **Attach to Alert Data** dialog. The **Attach to Alert Data** dialog provides four drop down menus that allow you to specify information regarding the alert data you want to display.

The **Alert Variable Name** drop down menu lists all available alert variables. Drop down menus for **Column(s)** and **Filter Column** populate based on the selected Alert Variable Name. The **Column(s)** and **Filter Column** drop down menus will only contain options if the selected **Alert Variable Name** contains tabular information. If the item you require is not listed, type your selection into the field. To add an alert, select **Tools>Alerts** to open the **Alerts** dialog. See "Adding Alerts" for more information.

It is possible to indicate multiple columns for the filter and multiple values to compare against for each column. If the number of specified column names does not correspond to the number of values listed, then extra names and/or values are ignored.

Note: Spaces around separators are not allowed.

Field Name	Description
Alert Variable Name	Name of the alert variable to display. The drop down menu will contain the names of all active alerts. These names correspond to the **alertName** property for each object in your active alert definition files. In addition, the following built-in tables are listed:

"AlertTable" - Contains all active and cleared alerts.

"Alert Variables Table" - Contains all of the active alert variables and their current state. The current state of an alert is the highest severity for the alert. For tabular alerts, it is the highest severity for all alerts in the table.

Alerting Enabled - Indicates whether the alert engine is enabled to the Alert data source. See "Alerting Enabled Table" for more information.

Action Audit Table - Returns the "Alert Action Audit Table" as specified on the **Alerts** tab of the **Application Options** dialog or on the command line.

Action Audit Connection - Indicates the connected status of the Alert Action Audit table.

Alert Settings Table - Contains alert settings.

Note: Only available if Self Service Alerts are enabled and RTView has successfully connected to the Alert Settings Table.

Self Service Audit Table - Contains changes made to the Alert Settings Table if a Self Service Audit Table was specified on the "Self Service Alerts Tab" of the **Application Options** dialog.

Note: Only available if "Self Service Alerts" are enabled and RTView has successfully connected to the Self Service Audit Table.

Self Service Connections - Indicates whether or not RTView is connected to the Alert Settings Table and, if specified, the Self Service Audit Table. The connection is considered good if the SQL connection is good and at least one update has been received.

Note: Only available if "Self Service Alerts" are enabled.

Alert Definitions - Contains the name of the alert, the file containing the alert definition, and the substitutions specified when the alert definition file was loaded. See "Alert Definitions Table" for more information.

Schedule Events - This table allows visibility into schedule events, and is only populated if you have applied schedules to your alerts. See "Monitoring Scheduling Events" for more information.

New Data Only

If selected, only rows that have changed for the selected built-in table (i.e. **AlertTable**, **Alert Variables Table**, or **Alerting Enabled**) are updated. This is useful when a cache or the Historian is attached to this table so that duplicate rows are not stored.

Note: For optimization purposes, if **New Data Only** is selected, the **AlertTable** will not be updated if the only columns that contain changes for that row are **Last Update Time** and **Count**. In many cases these two columns are not used by RTView applications, so selecting **New Data Only** prevents unnecessary data updates. However if your RTView application is using the values in those columns, you can use the **lutupdatesnewdata** command line option to enable this update for the **AlertTable** when **New Data Only** is selected. See "Options Enabled with Alerts" for more information.

Column(s)

If the alert data is tabular, you can select which column(s) to display.

Filter Rows

Check box to indicate whether or not to filter rows. Filters can only be used for tabular data. See "Row Filtering" for more information.

Filter Column

Name of the column to use as a filter. Multiple column names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col 3). If your column name contains a space or a semicolon, then the entire name must be enclosed in single quotes.

Filter Value	Value that the **Filter Column** must equal. Multiple filter values should be entered as a nested list, where values for a given column are separated by commas within a semicolon (;) delimited list (i.e. val1,val2;val3,val4;val5,val6). If your filter value contains a space or a semicolon, then the entire value must be enclosed in single quotes. When * is entered as a filter field value, data for all values in the specified filter column will be used to update the object property. When "*" is entered, only the literal comparative value will be used. These are only allowed for objects which display tabular data.
Data Server	Select to read data through configured Data Server(s) and not directly from alert data. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab" **None** - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source. **Named Data Servers** - Select a **Named Data Server** that you configured in **Application Options>**"Data Server Tab" **Multi-Server Attachment** - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more Named Data Servers (e.g. ds101;ds102). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**). **Note:** The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**. When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received. A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways: **1.** The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows. **2.** The multi-server attachment can be applied as the **Table** argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

When an object property is attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing this value from the **Object Properties** window is no longer possible. To remove the data attachment, and resume editing capability in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Substitutions

Substitutions allow you to build open-ended displays in which data attachments depend on values defined at the time the display is run. Generic names, such as **$alert1** and **$alert2**, are used instead of values for specific alert variable names. Later when the display is running, these generic values are defined by the actual names of specific alert variables, such as **production_alert**. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

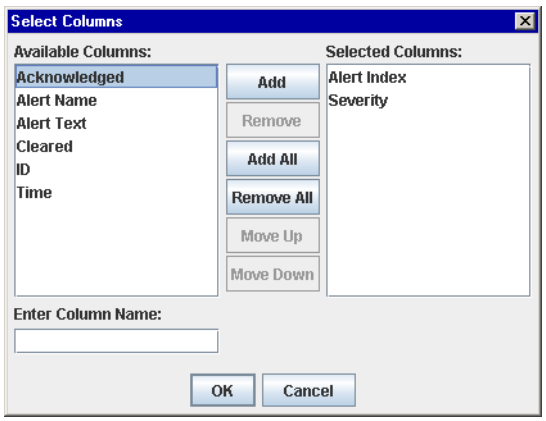
Select Table Columns

You can specify which table columns to display and in what order they will appear from the **Attach to Alert Data** dialog. In order to populate the listing of available columns, you must first select a valid Alert Variable Name.

Click the ellipse button  in the **Column(s)** field (or right-click in the **Column(s)** field and choose **Select Columns**) to display the **Select Columns** dialog. The dialog should contain a list of Available Columns that you can add to your table.

To add a column, select an item from the **Available Columns** list and click on the **Add** button. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of columns in a table by arranging the items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

If no data is available for a table row within a selected column, the table cell will display one the following values: **N/A**, **false**, **0**, or **0.0**.



The following describes the **Attach to Alert Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.

Clear	Clears all fields. Detaches object from data source (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Define Alert Command

This section contains the following:

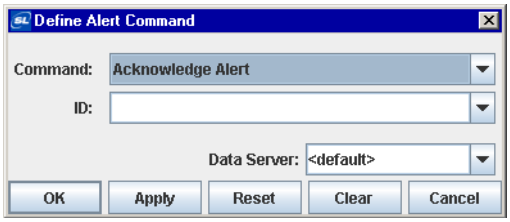
Define Alert Command Dialog

You can access the **Define Alert Command** dialog from the **Object Properties** window. This dialog is used to assign alert commands allowing you to manage your alerts from within an RTView display. If you execute an alert command from a Thin Client with Direct Data Connection or any Served Data deployment, the command will execute on the server.

To enable Alert Action Auditing, select **Tools>Options>**"Alerts Tab". Once Alert auditing is configured and enabled, whenever an alert command is executed the Alert data source will store a record of that event in a database table.

Note: Alert action auditing does not apply to Self Service Alert commands, which are audited separately in the Self Service Audit Table.

Right-click on the appropriate command property in the **Object Properties** window and select **Define Command>ALERT** to display the **Define Alert Command** dialog. The **Define Alert Command** dialog provides a drop down menu with available commands. Argument fields vary based on the selected command type. See the "Define/Execute Command" section for information on how to execute a command.



Field Name	Description
Command	Select a Command Type. See "Command Types" for more information.

Command Argument(s)

Command argument fields depend on which Command Type is selected. To attach a command argument to data, right-click and choose **Attach to Data** or double-click in the field.

Data Server

Select to read data through your configured Data Server(s) and not directly from alert data.

Default - Select the default Data Server you configured in **Application Options>**"Data Server Tab".

None - Bypass data being redirected through the specified data server(s) for this attachment and instead attach directly to the data source.

Named Data Servers - Select a **Named Data Server** that you configured in **Application Options>**"Data Server Tab".

Multi-Server Command - When multiple data servers are specified, the command will be executed on each data server in the list.

To configure multiple data servers, enter a semicolon (;) delimited list containing two or more Named Data Servers (e.g. ds101;ds102). Each name specified must correspond with a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**).

Note: The values **__default** and **__none** begin with two underscore characters.

Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

Command Types

Command Type	Description	Argument	Description
Acknowledge Alert	Specify the ID, or list of IDs, for the alerts to acknowledge. After an alert is acknowledged, the **AlertTable** Acknowledged field updates and if the **reNotificationTime** is greater than zero, it does not re-execute.	ID	The ID of the alert. This can either be a single ID or a semi-colon (;) delimited list of IDs.

Add Alert Definition File	Adds an Alert Definition file to the data source. Alerts without names or with duplicate names are not added.	Alert Definition File	Name of the Alert Definition file.

		Substitution	Specify substitutions for this Alert Definition File. Substitutions are optional and must use the following syntax:
			`$subname:subvalue`
			`$subname2:subvalue2`
			If a substitution value contains a single quote, it must be escaped using a / :
			`$filter:Plant=/'Dallas/'`
			If a substitution value contains a space, it must be enclosed in single quotes. Do not escape these single quotes:
			`$subname:subvalue`
			`$subname2:'sub value 2'`
			A substitution string cannot contain the following:
			: | . tab space , ; = < > ' " & / \ { } [ ] ( )

Add Comment	Adds a time-stamped comment to the specified alert. **Note:** Existing comments on the alert are not replaced, the new comment is added after previous comment(s).	ID	ID of the alert.
		Comment	Enter comment.
Clear Alert	Clears the specified alert.	ID	ID of the alert. This can either be a single ID or a semi-colon (;) delimited list of IDs.
Clear Comments	Clears all comments from the specified alert.	ID	ID of the alert.
Delete Index From Self Service Alerts Table	Remove an alert index setting from the database table. **Note:** This command is only available if Self Service Alerts are enabled and RTView has successfully connected to the Alert Settings table.	Alert Name	Name of the alert containing the index to remove.
		Index	Index to remove.
		Index Type	Index type of the index to remove.
		User	User name to specify in the Self Service Audit Table for this update. **Note:** To specify value of user name from RTView login, if enabled, use **$rtvuser** substitution.

Enable Alert Definition	Enables/disables the specified alert definition object. **Note:** It is not recommended that you use this command on an alert that has the enabledFlag property attached to data.	Alert Name	Name of the alert definition object.
		Enable	Enter **True** to enable and **False** to disable.
Enable Alerts	Enables/disables all alerts in the active Alert Definition files.	Enable	Enter **True** to enable and **False** to disable.
Remove Alert Definition	Removes the Alert Definition file from the data source.	Alert Definition	The name of the Alert Definition file.
		Substitution	Specify substitutions for this Alert Definition file. This must match the substitution used when the Alert Definition file was added.
Set Custom Alert Event Attribute	Set the value of a Custom Alert Event Attribute. **Note:** This command is only available if a Custom Alert Event Attribute has been defined on the "Custom Alert Fields Tab" in the **Application Options** dialog.	ID	ID of the alert. This can either be a single ID or a semi-colon (;) delimited list of IDs.
		Attribute Name	Name of the Custom Alert Event Attribute.
		Attribute Value	Value of the Custom Alert Event Attribute. **Note:** This value must be of the Data Type specified for this Custom Alert Event Attribute.
Set Owner	Specify the Owner of an alert.	ID	ID of the alert. This can either be a single ID or a semi-colon (;) delimited list of IDs.
		Owner	Owner (string value) of the alert.
Unacknowledge Alert	Specify the ID, or list of IDs, for the alerts to unacknowledge. After an alert is unacknowledged, the **AlertTable Acknowledged** field updates, the renotification command resumes, and the **Severity** and **Count** fields in the alert table resume updating. This command is tracked in the Alert Action Audit table as an Event Management alert where the alert id is the target value. For details, see "Audit Alert Action".	ID	The ID of the alert. This can either be a single ID or a semi-colon (;) delimited list of IDs.

Update Self Service Alerts Table	Modify an alert setting. **Note:** This command is only available if "Self Service Alerts" are enabled and RTView has successfully connected to the Alert Settings Table.	Alert Name	Name of the alert.
		Warning Threshold Value	Warning threshold value to store. If this value is not a double, a null value will be stored.
		Alert Threshold Value	Alert threshold value to store. If this value is not a double, a null value will be stored.
		Duration	Duration value to store. If this value is not an integer, a null value will be stored.
		Enabled	Enabled value to store. This can be a boolean (true/false) or 0/1. Other values will be stored as false.
		Index	Index to remove.
		Index Type	Index type of the index to remove.
		Use Index	Set to 1 (or true) to indicate that the alert should use this row in the database, or to 0 (or false) to indicate it should ignore this row in the database.
		User	User name to specify in the Audit Table for this update. **Note:** To specify value of user name from RTView login, if enabled, use **$rtvuser** substitution.

Substitutions

Substitutions allow you to build open-ended displays in which commands depend on values defined at the time the display is run. Generic names are used instead of arguments for specific commands. Later when the display is running, these generic values are defined by the actual arguments. In this way, a single display can be reused to issue a number of different commands. Substitutions are not applied to Drill Down or Set Substitution commands. For more information on creating displays using substitution values, see "Substitutions".

Special Values

$value	When an **actionCommand** is executed $value is replaced with the value from the control. This value may be used in any field in the **Define Alert Command** dialog. **Note:** This value may only be used for Action Commands. See "Commands" for more information.

The following describes **Define Alert Command** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from assigned command (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Application Options - Alerts

This section contains the following:

Overview

Select **Tools>Options** in the Display Builder to access the **Application Options** dialog.

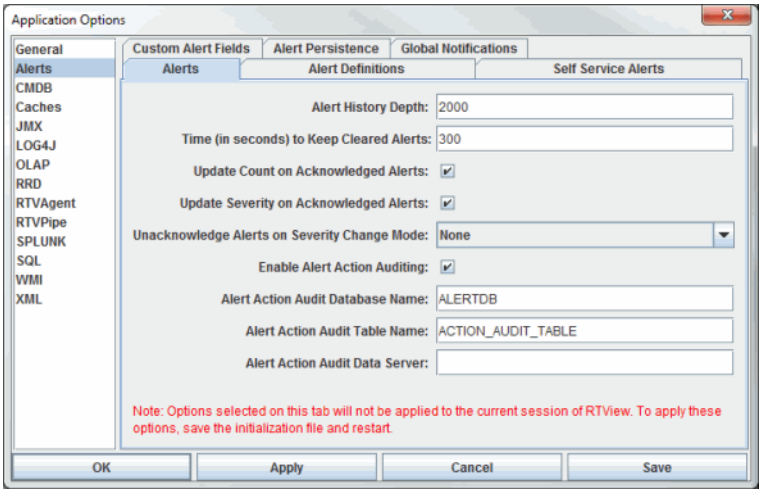
Options specified on **Alerts** tabs can be saved in an initialization file (**ALERTOPTIONS.ini**). On startup, the initialization file is read by the Display Builder, Display Viewer, Display Server, Data Server, and Historian to set initial values. If no directory has been specified for your initialization files and **ALERTOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

Note: Options specified using command line arguments will override values set in initialization files.

There are five Application Options tabs for Alerts: "Alerts Tab", "Alert Definitions Tab", "Self Service Alerts Tab", "Custom Alert Fields Tab", and "Alert Persistence Tab".

Alerts Tab

Select **Tools>Options>Alerts>Alerts** to access the **Alerts** tab, which allows you to set alert options.



Field Name	Description
Alert History Depth	Set the number of rows stored in the "AlertTable". This option limits the size of the **AlertTable** to the number of rows specified. When the **AlertTable** goes over this limit, RTView removes cleared alerts from the table starting with the alerts that have the oldest Cleared Time. If all cleared alerts have been removed and the specified Alert History Depth is still not reached, then RTView will remove active alerts starting with alerts that have the oldest Time value. In this case, these alerts are cleared before being removed and the cleared reason is set to ALERT HISTORY DEPTH EXCEEDED.
Time (in seconds) to Keep Cleared Alerts	Enter the time, in seconds, to control how long RTView retains cleared alerts from the Alert data source. RTView checks for cleared alerts to remove on each update pass. If more than the specified time has elapsed since the time the alert was last cleared, then the alert will be removed. If the value is set to 0, cleared alerts will not be removed until the specified Alert History Depth has been exceeded.
Update Count on Acknowledged Alerts	Controls whether or not acknowledged alerts are included in the value of the **Count** column (in the "AlertTable"). If selected, the count will increase incrementally each time an alert receives an update until that alert is cleared. If deselected, the count will stop if an alert is either cleared or acknowledged.
Update Severity on Acknowledged Alerts	Controls whether or not the severity of acknowledged alerts is updated. By default this option is off, so the severity on acknowledged alerts does not update.

Unacknowledge on Severity Change Mode

Specifies the conditions to automatically unknowledge alerts when the alert severity increases. By default this option is **None**.

When an alert is unacknowledged, the alert resumes renotifying if the **reNotificationMode** property is configured to do so. It also means that if the condition meets the specified **reNofifyOnSevChangeMode** property on the alert, it resumes renotifying regardless of the **reNotificationMode**. This unacknowledge action is not tracked in the Alert Action Audit Table because it is not a user action. There are three modes:

- **First Severity Change**: Reverts acknowledged alerts to unacknowledged when the severity increases for the first time. This means that the alert will start to renotify again, that is if the **reNotificationMode** property is configured to do so. If the **reNofifyOnFirstSevChangeFlag** property is selected, then the alert will renotify once regardless of the selected **reNofication Mode**. **Note:** This option is not tracked in the Alert Action Audit Table since it is not a user initiated action.
- **All Severity Increases**: Reverts acknowledged alerts to unacknowledged every time the severity increases.
- **None**: Do not unknowledge alerts when the alert severity increases.

Note: When run against Application Options set from an older version of RTView where the **Unacknowledge Alerts on First Severity Change** was enabled, that option automatically converts to the **First Severity Change** mode so the behavior is the same.

Enable Alert Auditing

Enable auditing of alert actions. If selected and configured, alert actions will be stored to the specified Alert Action Audit Table Name. See **Alerts>**"Audit Alert Action" for details.

Alert Action Audit Database Name

Enter the name of a database, as defined on the "SQL Tab", that contains the Alert Action Audit table. See **Alerts>**"Audit Alert Action" for details.

Alert Action Audit Table Name

Enter the name of a table, in the specified Alert Action Audit database, in which to store the alert actions audit information. See the **Alerts>**"Audit Alert Action" section for details.

Alert Action Audit Data Server

Optionally specify a **Named Data Server** connection (see "Data Server Tab") to use for reading and writing alerts to the specified Alert Action Audit Database Table.

Maximum Characters Allowed in Comments

Enter a value greater than 0 to limit the number of characters in the **Comments** field. When enabled, new comments are added before older comments in the **Comments** field, and the oldest comments are trimmed if necessary to stay under the limit. Enabling this option prevents the **Comment** field from becoming longer than your corresponding database field can support. By default, this feature is disabled.

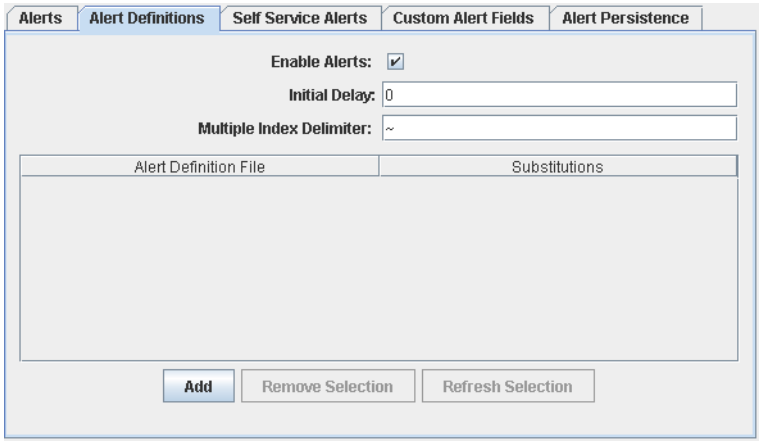
When disabled, the **Comments** field grows unbounded as new comments are appended at the end of existing comments.

This comment limit is available via a data attachment to alert-**commentlimit** to facilitate building a UI that will prevent users from entering comments longer than this limit. Note that the UI text entry limit should be set to 100 characters less than the comment limit in order to account for the time stamp, user name, and hard returns between comments.

Alert Definitions Tab

Select **Tools>Options>Alerts>Alert Definitions**. This tab allows you to enable alerts, set options for your alert definitions, and add, remove or refresh your Alert Definition files. Once you have added an Alert Definition file and, optionally, entered a corresponding substitution, click **Apply** to execute. The Alert Definition file will be added to the data source. The "Alert Variables Table" updates with all active alerts from the Alert Definition file and the alerts begin executing.

Note: Alerts without names or with duplicate names will not be added.



Field Name	Description
Enable Alerts	Enables/disables all alerts in all active Alert Definition files
Initial Delay	Duration (in seconds) to wait after startup before evaluating alerts.
Multiple Index Delimiter	For alerts with multiple index columns, RTView creates a unique Alert Index by concatenating all of the index column values separated by the specified Multiple Index Delimiter. The value can be any string, except the following: • comma (,) • semi-colon (;) or • empty string. Default is tilde (~).
Alert Definition File	File(s) that contain alert definitions. See "Overview" for information on creating Alert Definition files.

Substitutions

Optionally specify substitutions for this Alert Definition file. Substitutions must use the following syntax:

```
$subname:subvalue $subname2:subvalue2
```

If a substitution value contains a single quote, it must be escaped using a / :

```
$filter:Plant=/'Dallas/'
```

If a substitution value contains a space, it must be enclosed in single quotes. Do not escape these single quotes:

```
$subname:subvalue $subname2:'sub value 2'
```

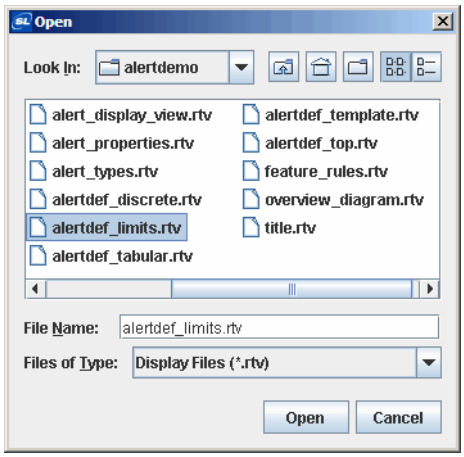
A substitution string cannot contain the following:

: | . tab space , ; = < > ' " & / \ { } [ ] ( )

Add

Click **Add** and select a (**.rtv**) file to include in **Alert Definition File** list.

Note: Alert definitions are only added after you click **OK**, **Apply**, or **Save**.



Remove Selection

Remove the selected Alert Definition file. Alerts in this Alert Definition file will no longer execute and the alert data for these alerts will no longer be available in the Alert data source.

Note: Alert definitions are only removed after you click OK, Apply or Save.

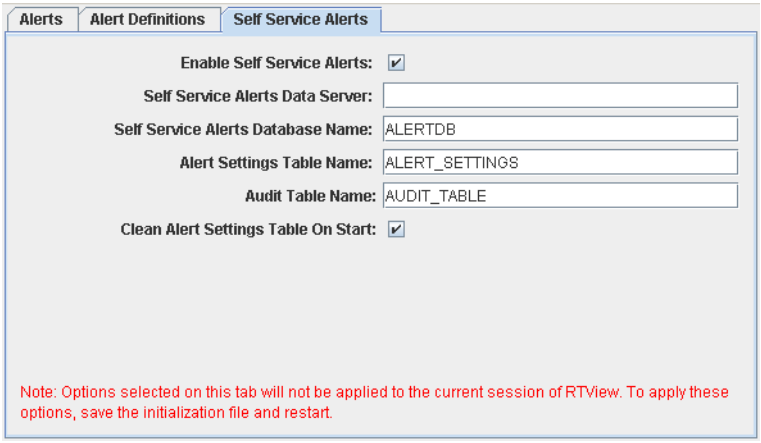
Refresh Selection

Reload the selected Alert Definition file immediately. This option allows the data source to re-read an Alert Definition file without restarting the Display Builder.

Note: Refresh Selection is enabled when you select an Alert Definition file that has already been added and applied.

Self Service Alerts Tab

Select **Tools>Options>Alerts>Self Service Alerts**. Self Service Alerts make it easy to set and persist threshold, duration and enabled settings for your alerts in a database. For details, see "Alert Settings Table".

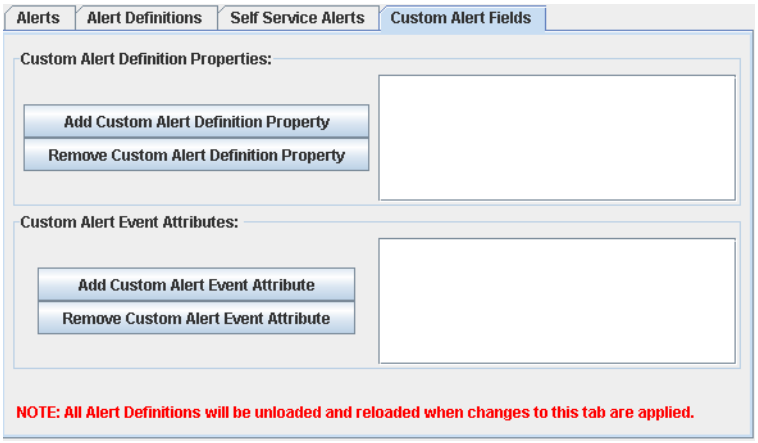


Field Name	Description
Enable Self Service Alerts	Select to enable self service alerts. **Note:** If this option is selected, no alerts will be loaded into the system unless the specified Self Service Alerts Database is configured and connected.
Self Service Alerts Data Server	Enter name of a named data server connection or leave blank to use the default data server connection, if specified. See "Data Server Tab" for more information. This data server will be used when connecting to the Self Service Alerts Database.
Self Service Alerts Database Name	Name of SQL Database connection defined in the "SQL Tab" of the **Application Options** dialog.
Alert Settings Table Name	Name of table where your alert settings will be stored. See below for a description of the schema for this table.
Audit Table Name	Name of the table where a record of all changes to the Alert Settings Table will be stored. This value is optional. If not specified, no record of changes will be stored.
Clean Alert Settings Table on Start	Select to delete entries from the Alert Settings Table for alert names that are not defined in RTView. **Note:** This is done at startup after alert configuration files are processed and all of the alerts are loaded.

Custom Alert Fields Tab

Select **Tools>Options>Alerts>Custom Alert Fields**. This tab allow you to define custom properties on your alert definitions and set custom alert event attributes that will show up as columns in the **AlertTable** when an alert is executed.

Note: All additions, modifications and deletions of Custom Alert Definition Properties and Custom Alert Event Attributes are applied when you click **OK**, **Apply**, or **Save** in the **Application Options** dialog. All of your Alert Definition files will be unloaded and reloaded whenever changes are applied to the **Custom Alert Fields** tab.



Field Name	Description
Custom Alert Definition Properties	These properties are useful when you want to store additional information about an alert definition and have it be accessible in the **AlertTable**. For example, you might want to organize your alerts by Category. In that case, you could add a Custom Alert Definition Property named **Category**. When you define your alerts, you specify a value in the **Category** property for each alert. When the alerts are executed, there will be a **Category** field in the **AlertTable** that you can use for sorting and/or filtering. Once you have applied your Custom Alert Definition Property, a **Custom Properties** section will be added to **Object Properties** window that contains a property for each of your Alert Definition files. These properties can be static or attached to data. You will also see a column in the **AlertTable** for each Custom Alert Definition Property. The value in that column will be the value specified for the property in the corresponding alert definition. The alert will also create a substitution for each custom property by collapsing the name into **$alertXX** (where **XX** is the name of your custom field) in camel case with spaces removed. For example, if your custom property name is my property, the corresponding substitution will be **$alertMyProperty**. You can use this substitution to show the value of your custom property in the **alertCommand**, **reNotificationCommand**, **clearedCommand**, or in the alert text. **Note:** Renotification commands will update with the new value of the property if it changes, but the alert text in the table will not update after the alert is generated.

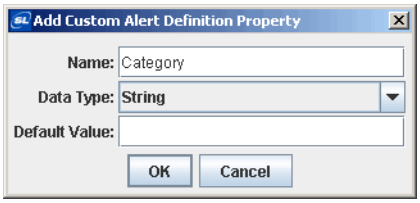
These properties can also be mapped to a column in the **valueTable** input for your alert if the **useTabularDataFlag** is on. Use the **customPropertyMap** property to specify which column from the **valueTable** contains the value to use for the custom property using the following syntax:

```
customPropName:valueTableColumnName;customPropName2:valueTableColumnName2
```

For example, if you have a custom property named **My Custom Property** and you want to use the value from the **My Data Column** column in your value table, you would specify the following:

```
My Custom Property:My Data Column
```

To add a property, click **Add Custom Alert Definition Property**. To edit an existing property, double-click on it in the Custom Alert Definition Properties list.



Add

Name - Name of the Custom Alert Definition Property. This will be used as the property name in the alert definition and as the column name in the "AlertTable". This field is required and the name must be unique within both the Custom Alert Definition Properties and Custom Alert Event Attributes.

Data Type - Select the type of data for this Custom Alert Definition Property: String, Double, Integer or Boolean.

Default Value - Specify a default value for this Custom Alert Definition Property. This field is optional. If specified, this value will be used for new alert definitions.

Note: The **Default Value** must be of the specified Data Type.

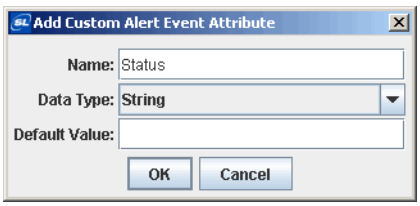
Remove

To delete a Custom Alert Definition Property, select it from the list and click **Remove Custom Alert Definition Property**.

Custom Alert Event Attributes

Alert event attributes show up as columns in the **AlertTable** and are set via the **Set Custom Alert Event Attribute** command. This is useful when you want to add table information on a per-alert instance basis. For example, you might want to add a Status attribute to the alert, so that the user can indicate whether the problem is Under Investigation, Being Tested, or Resolved.

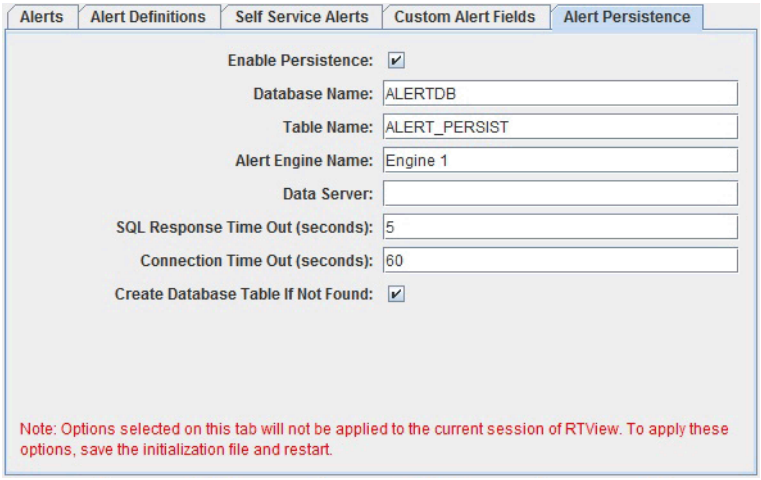
To add an attribute, click **Add Custom Alert Event Attribute**. To edit an existing attribute, double-click on it in the **Custom Alert Event Attributes** list.



Add	**Name** - Name of the Custom Alert Event Attribute. This will be used as the column name for this attribute in the "AlertTable" and will also be the name you will use to reference this attribute in the **Set Custom Alert Event Attribute** command. This field is required and the name must be unique within both the Custom Alert Event Attributes and the Custom Alert Definition Properties. See "Command Types" for more information. **Data Type** - Select the type of data for this Custom Alert Event Attribute: String, Double, Integer or Boolean. **Default Value** - Specify a default value for this Custom Alert Event Attribute. This field is optional. If specified, this value will be used for new alerts. **Note:** The Default Value must be of the specified Data Type.
Remove	To delete a Custom Alert Event Attribute, select it from the list and click **Remove Custom Alert Event Attribute**.

Alert Persistence Tab

Select **Tools>Options>Alerts>Alert Persistence**. Configure the alert engine to persist alerts directly to the database or via data server.

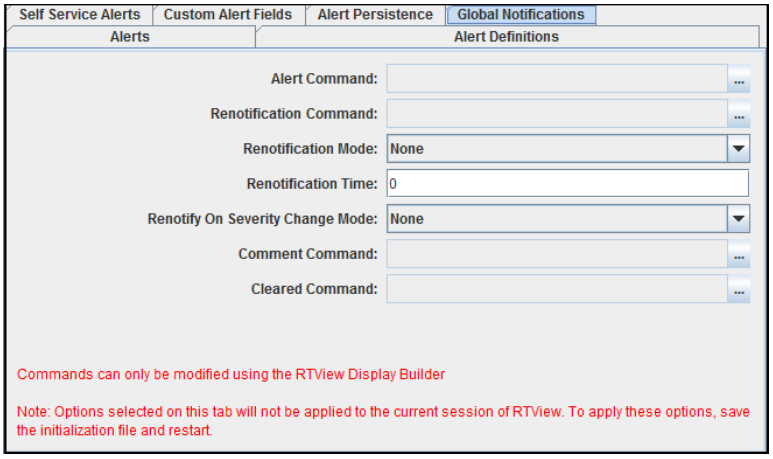


Field Name	Description
Enable Persistence	Select to enable the persistence of alerts during fail over of an alert engine. All fields and current state of all active alerts, as well as all cleared alerts that have not been removed from the system will be stored.

Database Name	Name of SQL Database connection defined in the "SQL Tab" of the **Application Options** dialog. If the Database connection is unavailable at start up, then RTView will wait the amount of time specified by the **Connection Time Out** option before initializing the alerts with persistence disabled. If the Database connection is lost while the alert engine is running, the alert engine will store up alerts to persist until the database becomes available. **Note:** It is possible that alert events that occur between the time the database becomes unavailable and the time the alert engine is notified might not be persisted. Once the Database connection is resolved, all of the stored alerts will be persisted. **Note:** Running the alert engine for an extended period of time with persistence enabled and an unavailable connection will result in memory growth until a connection is made.
Table Name	Name of the table where the alerts will be persisted.
Alert Engine Name	Enter a unique name for the alert engine. Multiple alert engines can persist their alerts to the same database, so the unique identity of each enables the alert engine to restore only alerts for a specific alert engine in the case of fail over.
Data Server	Optionally specify a Named Data Server connection to use for reading and writing alerts to/from the database. See "Data Server Tab" for more information.
SQL Response Time Out (seconds)	Specify (in seconds) the amount of time RTView will wait for a response from the database query to get the persisted alerts. If your database connection is slow, or if you are using a data server to get your persisted alerts and its connection is slow, you should increase this value. **Note:** RTView will block while waiting for this query to return.
Connection Time Out (seconds)	If the persistence database is not available at startup, specify (in seconds) the amount of time RTView will wait for the database to become available before initializing alerts with persistence disabled. Default is 60 seconds. **Note:** RTView will not block while waiting for the connection to resolve.
Create Database Table If Not Found	If selected, RTView will create an alert persistence database table if one is not already found in the specified database.

Global Notifications Tab

Select **Tools>Options>Alerts>Alert Persistence**. This tab allows you to set up global values for alert notifications. For each command field, click the ellipsis button next to the field to define (via a pop-up window--see "Define/Execute Command" for more information), copy, paste, or clear a command. Once the command field contains a command, you can double-click on the field to edit it.



Field Name	Description
Alert Command	Select the command to be used as the **alertCommand** for all alerts whose **alertCommand** property is empty.
Renotification Command	Select the command to be used as the **reNotificationCommand** for all alerts whose **reNotificationCommand** property is empty.
Renotification Mode	Select the mode to be used as the **reNotificationMode** for all alerts whose **reNotificationMode** property is set to **Use Global Setting**. Options include: **None** -- Do not renotify. The **alertCommand** is executed only once when the alert is activated. **Renotify on Timer** -- Renotify based on **reNotificationTime** property. The **alertCommand** is executed once when the alert is activated and then re-executed every **reNotificationTime** (seconds) until the alert is cleared or acknowledged. If the **reNotificationTime** is set to **0**, then the alert will not renotify. **Renotify on Data Changed** -- Renotify when the input value changes. The **alertCommand** is executed once when the alert is activated and again when a different value is received until the alert is cleared or acknowledged. The new value must be different than the previous value for the alert to renotify. **Renotify on Data Updates** -- Renotify when the input value is updated. The **alertCommand** is executed once when the alert is activated and again whenever a value is received until the alert is cleared or acknowledged. The new data value may be the same or different than the previous value for the alert to renotify.
Renotification Time	Select the command to be used as the **reNotificationTime** for all alerts whose **reNotificationTime** property is set to a value less than **0**.

Renotify On Severity Change Mode	Select the mode to be used as the **reNotificationOnSevChangeMode** for all alerts whose **reNotificationOnSevChangeMode** property is set to **Use Global Setting**. Options include: **None** (default) -- Do not renotify when the severity changes. **Renotify on First Sev Change --** The alert renotifies when the severity of the alert changes for the first time. **Renotify on All Sev Increases --** The alert renotifies every time the severity increases. For the **Renotify on First Sev Change** and **Renotify on All Sev Increases** modes, alert renotification only occurs if the alert is either not acknowledged, or the **Unacknowledge Alerts on First Severity Change** application option is enabled. The renotification executes the **reNotificationCommand** if configured, otherwise it executes the **alertCommand**.
Comment Command	Select the command to be used as the **commentAddedCommand** for all alerts whose **commentAddedCommand** property is empty.
Cleared Command	Select the command to be used as the **clearedCommand** for all alerts whose **clearedCommand** property is empty.

Self Service Alerts

This section contains the following:

Overview

The Self Service Alerts functionality allows threshold, duration and enabled settings of alert definitions to be modified at run-time. Alert definitions are created in the Display Builder and are generally loaded at startup by the Display Server and Data Server. However, sometimes end-users of RTView prefer to modify alert settings at run-time rather than reload a modified alert definition. To enable run-time modification, current values of alert settings are maintained in a database. Optionally, this database can also store an audit trail tracking who has modified alert settings and when those changes were made.

To enable and configure Self Service Alerts and the database to persist alert settings, go to the "Self Service Alerts Tab" in the **Application Options** dialog. To provide end-users with an interface to modify alert settings, a "Self Service Alert Demo" is included that can be used stand-alone or integrated into your RTView application.

Two database tables are used for Self Service Alerts: the "Alert Settings Table" is required and the "Self Service Audit Table" is optional. An hsqldb database that contains these tables is provided as part of the Self Service Alerts demo.

Note: For other database types, the **demos\selfservicealerts\dbconfig** directory contains .sql files with the correct table schemas and a README.txt that explains how to use them.

Alert Settings Table

When Self Service Alerts are enabled, RTView will attempt to connect to the Alert Settings Table in the specified Self Service Alerts Database.

Note: No alert configuration files will be processed until it receives the first update of this table. To view your Alert Settings Table table, make a data attachment to the Alert Settings Table from the "Attach to Alert Data Dialog".

Column names, types and order must match exactly.

Column Name	Type
ALERTNAME	String
INDEXTYPE	String
ALERTINDEX	String
WARNINGLEVEL	Double
ALARMLEVEL	Double
DURATION	Integer
ENABLED	Boolean
USEINDEX	Boolean

For all alert types, when the alert is loaded RTView will look for a row in the Alert Settings Table with that alert name. If it isn't found, a row will be added. Once the row has been added, the value in the database will be used instead of the value in the property sheet for the corresponding properties. Note that the specified database and table(s) will apply to all alerts; there is no flag to indicate that for a single alert you want to use the values in the property sheet instead of the database.

The initial values for the **DURATION** and **ENABLED** fields will be set from the **alertDelayTime** and **enabledFlag** properties respectively. For an Event alert, only the **ENABLED** field will be set. **WARNINGLEVEL**, **ALARMLEVEL**, and **DURATION** are not supported for Event alerts. For all other alerts, values for **ALARMLEVEL** and **WARNINGLEVEL** will be set from different properties depending of the alert type:

- For a Limits alert, the **valueHighAlert** and **valueHighWarning** properties map to the **ALARMLEVEL** and **WARNINGLEVEL** fields. If these are both disabled, the **valueLowAlert** and **valueLowWarning** properties are used. If either **valueHighAlert** or **valueHighWarning** are enabled, then **valueLowAlert** and **valueLowWarning** are not mapped to the database even if they are enabled (they'll use the values in the property sheet). **Note:** If only one threshold is enabled only that value will be added to the table and the value for the other field will be inserted into the database as NaN which indicates that the value isn't used.
- For a Discrete alert the **valueHighAlert** property maps to the **ALARMLEVEL** field and the **valueLowAlert** property maps to the **WARNINGLEVEL** field. Again, only fields that are enabled are mapped and written to the database. The **valueMediumAlert** property is never mapped to a field in the database. **Note: ALARMLEVEL** and **WARNINGLEVEL** database columns are of type Double, so string thresholds cannot be entered in the database.
- For a Multi State alert, **AlertState01Comparison** is mapped to **ALARMLEVEL** and **AlertState02Comparison** is mapped to **WARNINGLEVEL**. Again, only fields that are enabled are mapped and written to the database. Also be aware that range comparison types are not mapped to the database, nor are any alert states beyond Alert State 02. **Note: ALARMLEVEL** and **WARNINGLEVEL** database columns are of type Double, so string thresholds cannot be entered in the database.
- For an Event alert, only the **ENABLED** field can be set. The **WARNINGLEVEL**, **ALARMLEVEL**, and **DURATION** are not supported.

Self Service Audit Table

If Self Service Alerts are enabled and a Self Service Audit Table is specified on the **Self Service Alerts** tab of the Application Options dialog, then RTView will insert an entry into the Self Service Audit Table each time it makes a change to the Alert Settings Table. To view your Self Service Audit table, make a data attachment to the Self Service Audit Table from the "Attach to Alert Data Dialog".

Column names, types and order must match exactly.

Column Name	Type
TIME_STAMP	Timestamp
USER	String
ACTION	String
ALERTNAME	String
INDEXTYPE	String
ALERTINDEX	String
WARNINGLEVEL	Double
ALARMLEVEL	Double

DURATION	Integer
ENABLED	Boolean
USEINDEX	Boolean

The Self Service Audit Table contains the TIME_STAMP of the change, the USER that made the change, the ACTION that was done, plus row information.

- If the USER listed is **RTView.GmsRtViewAlertDs**, this indicates that RTView made the change. This happens when a row is added for a new alert, when the threshold enabled flag for an alert that was already in the database changes and when a row is removed due to a selected **Clean Settings Table On Startup** option.
- If the USER value is **No Login**, this indicates that no value was specified for the user in the **Update Self Service Alerts** command. This could be because RTView login is disabled or because the command was not configured to include a user name.

Alert Construction Limitations

The following limitations apply when creating alerts to use with Self Service Alerts:

- Alerts can only use numeric threshold values
- Alerts can only use 1 or 2 thresholds:
  - Limits alerts can use **valueHighAlert** and **valueHighWarning** or **valueLowAlert** and **valueLowWarning**
  - Discrete alerts can use **valueHIghAlert** and **valueLowAlert**
  - Multi-state alerts can use **AlertState01Comparison** and **AlertState02Comparison**.
- Threshold values cannot be attached to data.

Self Service Alert Demo

Self Service Alerts makes it easy to set and persist threshold, duration and enabled settings for your alerts in a database. For details, see "Self Service Alerts".

The Self Service Alerts demo is located in your RTView installation directory under **demos\selfservicealerts**. The demo can be modified and used stand-alone or integrated into your RTView application in order to view and administrate alerts. The demo contains some demo alerts and a pre-configured hsqldb database to store your alert settings. See "Modify the Stand-Alone Demo" and "Integrate the Demo into an RTView Application" for more information.

Note: For other database types, the **demos\selfservicealerts\dbconfig** directory contains .sql files with the correct table schemas and a README.txt that explains how to use them.

The demo contains three main displays (Alert Detail Table, Alert, Administration and Administration Audit). In the top right corner, each display shows the current time, a * button that opens the display in a new window and a ? button that opens Help. These objects are customizable and can be modified, removed or replaced. See the "Customization Options" section for details.

Running the Demo

In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), navigate to the **demos/selfservicealerts** directory:

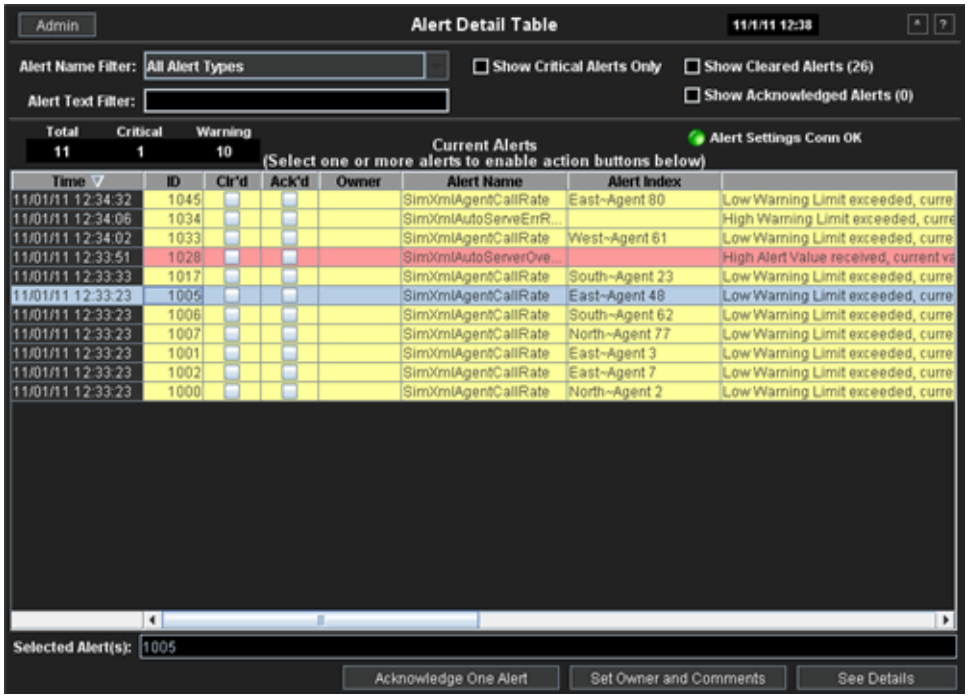
1. Start the XML simulator:
   type **start run_simdata**
2. Start the hsqldb database:
   type **start run_hsqldb**
3. Confirm that the demo server is running, if not start it by typing:
   run_startup_demoserver.

You can view the demo in the Display Viewer or the Thin Client.

4. To view the demo in the Display Viewer:
   type **run_viewer**
   OR
   To view the demo in the Thin Client:
   type **run_displayserver**
   open a browser to **http://localhost:8068/ssa/panels.html**

Alert Detail Table

This display shows all of your current alerts:



Field Name	Description
Admin	Click on this button to open the **Alert Administration** display in a new window.

You can filter the displayed alerts as follows:

Alert Name Filter	Select an alert name from the list to filter the table by the **Alert Name** column.
Alert Text Filter	Enter a value to filter the table by the **Alert Text** column. For example, a value of *High* will filter to rows where the alert text contains the word High. A value of High* will filter to rows where the alert text starts with High.
Show Critical Alerts Only	Only show alerts with a severity greater than 1.
Show Cleared Alerts	Show cleared alerts. The number following the **Show Cleared Alerts** label indicates the number of cleared alerts that match the selected **Alert Name Filter** and **Show Critical Alerts Only** values.
Show Acknowledged Alerts	Show Acknowledged alerts. The number following the **Show Acknowledged Alerts** label indicates the number of active (not-cleared) acknowledged alerts that match the selected **Alert Name Filter** and **Show Critical Alerts Only** values.

The following counts show the number of alerts that match the selected **Alert Name Filter, Show Critical Alerts Only**, **Show Cleared Alerts**, and **Show Acknowledged Alerts** values:

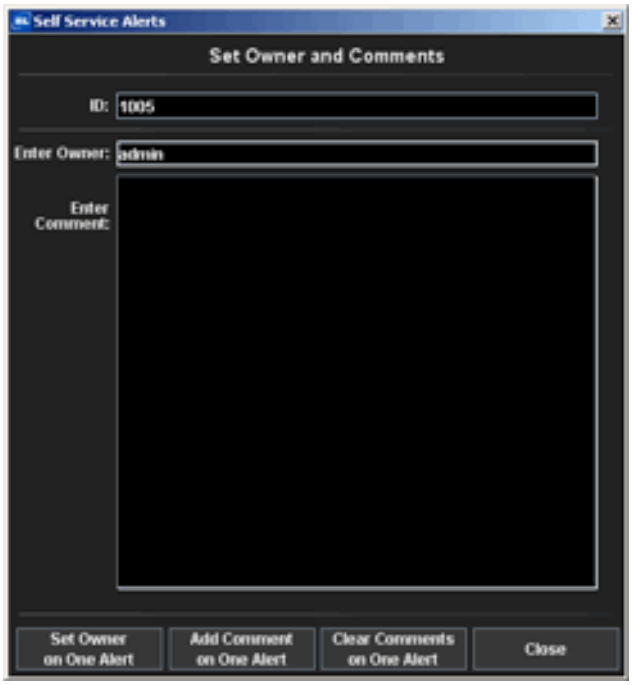
Total	Total number of alerts.
Critical	Number of alerts with a severity greater than 1.
Warning	Number of alerts with a severity of 1.
Alert Settings ConnOK	Indicates the connection status of the Alert Settings table.
Current Alerts Table	Displays all alerts that match the selected filters. • Red rows indicate the alert has a severity greater than 1. • Yellow rows indicate the alert has a severity of 1. Select one or more alerts from the table to enable the action buttons below the table. To select more than one alert: • Click on one alert, then hold the Shift key while clicking on another alert. All alerts in between will be selected. • Click on one alert, then hold the Control key while clicking on one or more other alerts. Only the alerts you click on will be selected. • Click on one alert, then hold the Shift key while pressing the up or down arrow on your keyboard. All alerts in between will be selected. • Click on one alert, then press Control+A to select all alerts in the table.
Selected Alert(s)	Lists ID of currently selected alerts.

The following action buttons are located below the **Current Alerts** table:

Acknowledge One Alert / Acknowledge Multiple Alerts	Acknowledge the selected alert(s). If more than one alert is selected, you will be asked to confirm before the alerts are acknowledged. If only one alert is selected, it will be acknowledged without confirmation.

Set Owner and Comments

Click to open the **Set Owner and Comments** display. The ID field at the top of the display lists the IDs for all selected alerts. Any action will be applied to all alerts in that list.



Set the Owner-The value of the **Enter Owner** field is filled in as the user name used when you logged in. If you are logged in with a role of admin or login is disabled, you can specify a different value for the owner.

Click **Set Owner on One Alert**/**Set Owner on Multiple Alerts** to set the owner field for the selected alert(s) to the value specified in the **Enter Owner** field.

Note: If more than one alert is selected, you will be asked to confirm before the owner is set on the alerts. If only one alert is selected, the owner will be set without confirmation.

Add a Comment - Type a comment in the **Enter Comment** field, then click **Add Comment to One Alert**/**Add Comment to Multiple Alerts** to add the comment to the selected alert(s). The comment will be added to the alert(s) along with the timestamp and the user name from your login.

Note: If more than one alert is selected, you will be asked to confirm before the comment is added to the alerts. If only one alert is selected, the comment will be added without confirmation.

Clear All Comments - Click **Clear Comments on One Alert**/**Clear Comments on Multiple Alerts** to clear all comments from the selected alert(s).

Note: If more than one alert is selected, you will be asked to confirm before the comments are cleared from the alerts. If only one alert is selected, the comments will be cleared without confirmation.

Details

Click to open the **Alert Detail** display to view details about the selected alert.

Note: If more than one alert is selected, this display will show the details for the last alert in the selection list. See the "Customization Options" section for details on customizing this display.

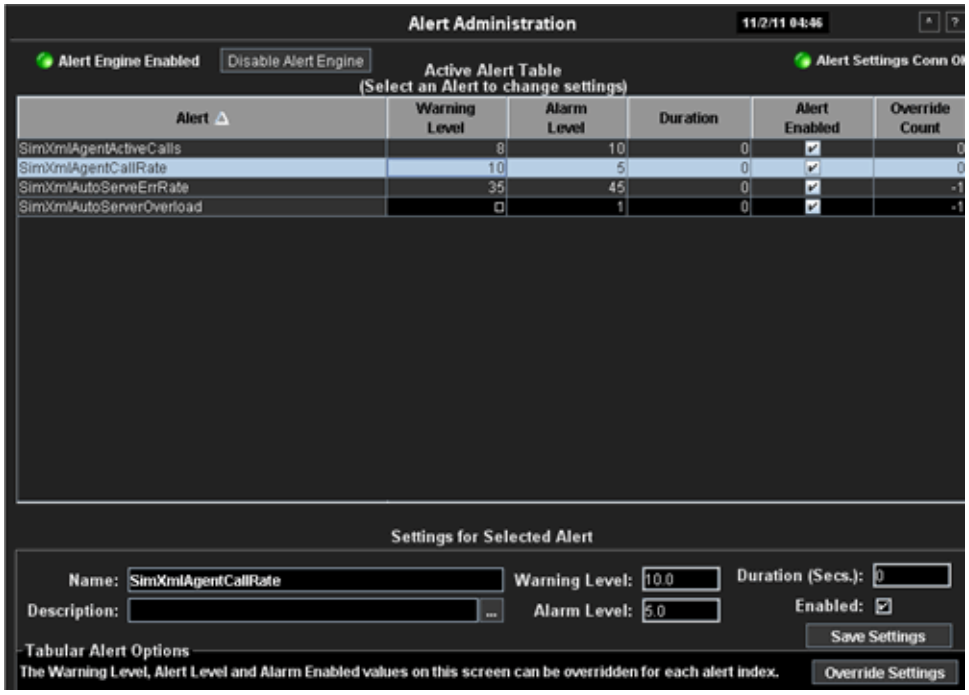
Options

Click to open **Options** display showing options for the selected alert.

Note: By default, the Options button in the **Alert Details** table display is hidden. See the "Customization Options" section for details on showing this button and customizing this display.

Alert Administration

This display shows you the current values in the Alert Settings Table in the Self Service Alerts database.



Field Name	Description
Alert Engine Enabled / Disabled	Indicates the status of the alert engine. If disabled, then click **Enable Alert Engine** to enable. If enabled, click **Disable Alert Engine** to disable. **Note:** The **Enable Alert Engine** and **Disable Alert Engine** buttons are only active if you are logged in with a role of admin or login is disabled.
Alert Settings ConnOK	Indicates the connection status of the Alert Settings Table.

Active Alert Table

Warning Level - Warning level threshold for the alert. If this value is NaN or a square, this means the alert doesn't support this threshold.

Alarm Level - Alarm level threshold for the alert. If this value is NaN or a square, this means the alert doesn't support this threshold.

Duration - Amount of time in seconds the value needs to meet the alert condition before an alert is generated. If the value is -1, this means the alert doesn't support duration.

Alert Enabled - Enabled value for the alert. If an alert is disabled, no alerts of that type will be generated.

Override Count - If this is a tabular Limits, Discrete, or Multi state alert, the settings in this table can be overridden on a per-index basis. This column lists the number of indexes in that alert that override the settings in this table. A value of **-1** indicates the alert doesn't support overriding the settings on a per-index basis.

Active - Indicates whether the alert definition is loaded in the current instance of RTView. If false, then there is a row in the Alert Settings Table in the database for this alert, but the alert is not loaded in the system. This may indicate a problem with your configuration.

By default, the **Active** column in the **Alert Administration** display is hidden. See the "Customization Options" section for details on showing this column.

To configure RTView to remove alerts that are not loaded in the system on startup, select the **Clean Alert Settings Table on Start** option on the **Self Service Alerts** tab of the Application Options dialog or use the -**cleansettingstable:true** command line option.

Settings for Selected Alert

To modify an alert setting, select a row in the Active Alert table.

Make changes to the **Warning Level**, **Alarm Level**, **Duration**, and **Enabled** fields, then click **Save Settings**.

Note: The **Save Settings** button is only enabled if an alert is selected and the connection to the Alert Settings Table is OK and you are logged in with a role of admin or login is disabled.

Name - Reflects value of **alertName** property.

Description - If the description is longer than the field, click on the ellipsis (...) button to see the whole description.

Note: Alert definitions may or may not contain a description.

Enabled - If false, the selected alert will not be evaluated.

Warning Level - Threshold that the value of the selected alert must cross in order to generate a severity 1 alert.

Alarm Level - Threshold that the value of the selected alert must cross in order to generate a severity 2 alert.

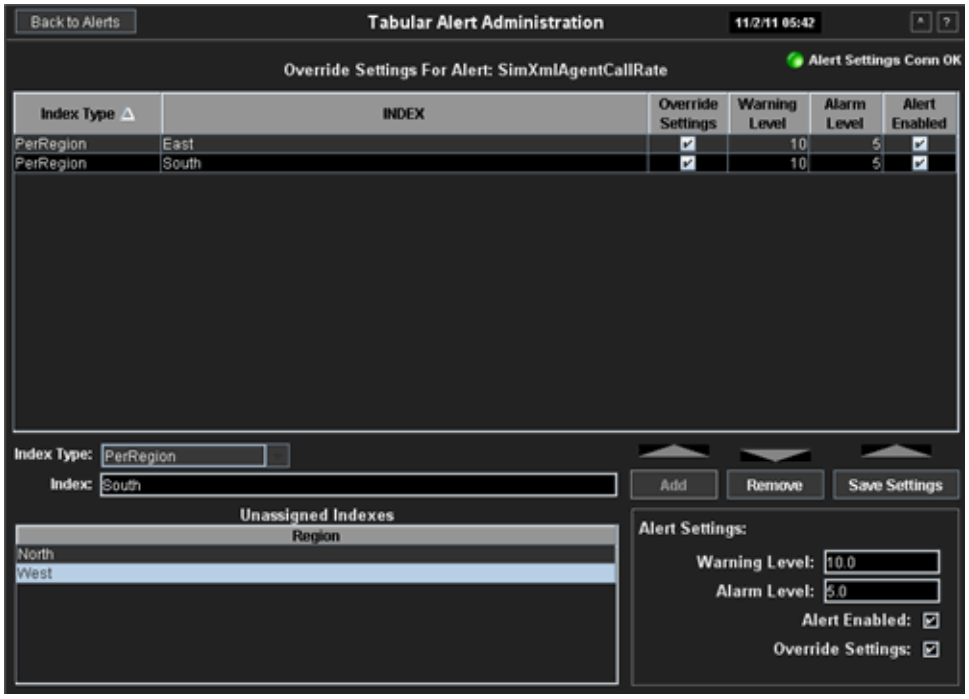
Duration (secs) - Amount of time that the selected alert must be in a Warning or Alarm state before an alert is generated.

Tabular Alert Options

If an alert is a tabular Limits, Discrete or Multi state alert, you can override the Warning Level, Alarm Level and Alert Enabled values specified on this screen. When you select an alert that supports this feature, an Override Settings button will appear at the bottom of the display. Click **Override Settings** to enter index-specific thresholds and enabled values. See "Tabular Alert Administration" section (below) for details.

Tabular Alert Administration

This display allows you to override the default warning, alarm and enabled settings on a per-index bases.



Field Name	Description
Back to Alerts	Click to return to the main **Alert Administration** page.
Alert Settings Conn OK	Indicates the connection status of the Alert Settings Table.
Override Settings Table	Shows all of the per-index settings that you have saved. All indexes that are not overridden here will use the settings specified for this alert on the main **Alert Administration** page. **Index Type** - The index type of this setting. For alerts with no **indexTypes** defined, this will be All. Otherwise this will list the **indexTypes** defined for that alert. In the case of alerts with multiple index columns, the **indexTypes** allow you to set threshold and enabled values on a subset of index columns instead of specifying each index column in the index value. For example, if you have an alert that is indexed on Region and Agent, you can either set a threshold for a specific Region and Agent or you can set a threshold for all Agents in that Region. **Index** - Index value. **Override Settings** - If selected, use these settings for the specified index rather than the default settings specified on the main **Alert Administration** page. If not selected, this row of settings will be ignored. **Warning Level** - The warning level threshold for the alert index. If this value is NaN or a square, this means the alert doesn't support this threshold.

Alarm Level - The alarm level threshold for the alert index. If this value is NaN or a square, this means the alert doesn't support this threshold.

Alert Enabled - The enabled value for the alert index. If an alert index is disabled, no alerts for that index will be generated.

The **Add**, **Remove**, and **Save Settings** buttons are only enabled if you have a valid selection, the connection to the Alert Settings Table is OK, and you are logged in with a role of admin or login is disabled.

In addition, the **Add** button is disabled if the selected index is already defined. In that case **Remove** and **Save Settings** are enabled. If the selected index is not yet defined, the **Add** button is enabled and **Remove** and **Save Settings** are disabled.

Add	To add a new per-index setting, select an Index Type from the list and the table below will list all unassigned indexes available for that Index Type. Select an index from that list and fill in the **Warning Level**, **Alarm Level**, **Alert Enabled**, and **Override Settings** fields. Click **Add** to add to the **Override Settings** table. **Note:** If the **Override Settings** check box is not selected, these settings will be ignored and settings specified in the main **Alert Administration** page for this alert will be used.
Remove	To delete the settings for an index, select it from the table and click **Remove**.
Save Settings	To modify the settings for an index, select it from the table. Make changes and click **Save Settings**.

Administration Audit

This display shows a history of alert administration actions executed in the **Alert Administration** and **Tabular Alert Administration** displays. The **Self Service Audit Table** contains the **TIME_STAMP** of the change, the **USER** that made the change, the **ACTION** that was done, plus row information.

Field Name	Description
Alert Settings ConnOK	Indicates the connection status of the Alert Settings Table.
TIME_STAMP	Time the action was executed.

USER	User name from the RTView login or No Login if login was disabled. If the USER listed is **RTView.GmsRtViewAlertDs**, this indicates that RTView made the change. This happens when: • a row is added for a new alert, • when the threshold enabled flag for an alert that was already in the database changes, and • when a row is removed due to a selected **Clean Settings Table On Startup** option.
ACTION	Action taken.

Alert Action Audit Trail

This display shows a history of alert actions executed in the **Alert Detail** display.

Field Name	Description
Action Audit ConnOK	Indicates the connection status of the Self Service Audit Table.
TIME_STAMP	Time the action was executed.
USER	User name from the RTView login or No Login if login was disabled.
ACTION TYPE	Type of action taken. The **ACTION TYPE** depends on the command.
ACTION	Action taken. This will be the name of the command, with the exception of the **Set Custom Alert Event Attribute** command which will be **Set <Attribute Name>** (e.g. **Set MyCustomAttrField**).
TARGET	Target of the action. The **TARGET** depends on the command.
VALUE	The value of the action. The **VALUE** depends on the command.

Modify the Stand-Alone Demo

Replace Demo Alerts

To use the Self Service Alerts demo to monitor your own alerts, you must first replace the demo alerts with your own. There are two ways to do this:

1. Save your alert definitions in a file named rtv_alertdefs.rtv and replace the **rtv_alertdefs.rtv** in the **selfservicealerts** demo with yours.

   OR

2. Save your alert definitions in one or more **.rtv** files in the **demos/selfservicealerts** directory.
   1. Run the Display Builder from the **selfservicealerts** directory.
   2. In the Display Builder, select **Tools>Options**.
   3. In the Application Options dialog, select **Alerts>**"Alert Definitions Tab" and remove **rtv_alertdefs.rtv** and add your own alert definition files.
   4. Save and exit the Display Builder.

In either case, your alerts must be constructed as documented for self service alerts.

Replace Demo (hsqldb) Database

To use a different database to store your alert settings, modify the ALERTDB SQL Database Definition.

1. Run the Display Builder from the **selfservicealerts** directory.
2. In the Display Builder, select **Tools>Options**.
3. In the **Application Options** dialog, select **SQL**.
4. Edit the ALERTDB database connection to point to your database.
5. If necessary, change the **Settings Table Name** and **Audit Table Name** entries on the **Alerts>**"Self Service Alerts Tab" of the **Application Options** dialog.
6. Save and exit the Display Builder.

The **dbconfig** directory, located in the **selfservicealerts** directory, contains SQL schema files for several databases. You can use these files to create the necessary tables in your database. See the **README.txt** file in that directory for instructions.

Clear Demo (hsqldb) Database

You can also clear the included hsqldb database by running the **DATA\reset_alerts** script while the database is not running. This will remove all entries from the Alert Settings Table and the Self Service Audit Table.

Modify Alert Options

The Self Service Alerts demo is setup to limit the number of active alerts in the Alert Detail table to 2000, which removes cleared alerts every 5 minutes. If you anticipate a larger number of active alerts in the Alert Table at one time, you will need to increase this value. You can also adjust the rate at which cleared alerts are removed. Both of these options are available on the "Alerts Tab" of the **Application Options** dialog.

This demo is setup to use a Custom Alert Definition Property named **DrillDownSuffix**. You may add additional Custom Alert Definition Properties. See "Custom Alert Fields Tab" for more information.

Customization Options

The Self Service Alerts demo is set up to be easily customized in a few ways:

Modify the time label, new window button, and help buttons in the header

- These objects are all contained in **rtv_alerts_header_include.rtv**. Edit this display to add or modify the objects displayed in the header of the displays.

Note: Do not add objects to the center or left top as they will be obscured by objects in the displays.

Use the $rtvAlertDataServer substitution

- If you have deployed your alerts to a data server, specify the name of that data server for the **$rtvAlertDataServer** substitution on the **General>Substitutions tab** in the **Application Options** dialog.

Use the $rtvUserAlertOptionsEnabled substitution

- By default, the **Options** button in the **Alert Details** table display is hidden. To make it visible, specify a value of 1 for the **$rtvUserAlertOptionsEnabled** substitution on the **General>Substitutions tab** in the **Application Options** dialog.

Override the Options display and do a per-alert-type override

- Modify the **user_alert_options.rtv** file to add your own custom alert options. This is useful if you have added Custom Alert Event Attributes that you want your user to set. This file will be displayed when you click on **Options** in the **Alert Detail Table** for an alert that has no value for the **DrillDownSuffix** property in the alert definition. To create a different Options display for different alert definitions, specify a value for the **DrillDownSuffix** on the alert(s) where you don't want to use the default display. When **Options** is selected for an alert where the **DrillDownSuffix** is specified, it will open a file named **user_alert_optionsDrillDownSuffix.rtv** where **DrillDownSuffix** is the value you specified in the **DrillDownSuffix** property for that alert definition.

For example, let's say you have the following alert definitions:

MyAlert1 - This has no value for the **DrillDownSuffix** property. When you select an instance of **MyAlert1** from the Alert Detail Table and click the **Options** button, it will open **user_alert_options.rtv**.

MyAlert2 - This has the value "**_alert2**" for the **DrilldownSuffix** property. When you select an instance of **MyAlert2** from the Alert Detail Table and click the **Options** button, it will open **user_alert_options_alert2.rtv**.

MyAlert3 - This has the value "**_alert3**" for the **DrilldownSuffix** property. When you select an instance of **MyAlert3** from the Alert Detail Table and click the **Options** button, it will open **user_alert_options_alert3.rtv**.

Override the Details display and do a per-alert-type override

- Modify the **user_alert_ details.rtv** file to add your own custom alert details. This is useful if you've added Custom Alert Definition Properties or Custom Alert Event Attributes that you want to display. This file will be displayed when you click on **Details** in the Alert Detail Table for an alert that has no value for the **DrillDownSuffix** property in the alert definition. To create a different details display for different alert definitions, specify a value for the **DrillDownSuffix** on the alert(s) where you don't want to use the default display. When **Details** is selected for an alert where the **DrillDownSuffix** is specified, it will open a file named **user_alert_detailsDrillDownSuffix.rtv** where **DrillDownSuffix** is the value you specified in the **DrillDownSuffix** property for that alert definition.

For example, let's say you have the following alert definitions:

MyAlert1 - This has no value for the **DrillDownSuffix** property. When you select an instance of **MyAlert1** from the Alert Detail Table and click the **Details** button, it will open **user_alert_details.rtv**.

MyAlert2 - This has the value "**_alert2**" for the **DrilldownSuffix** property. When you select an instance of **MyAlert2** from the Alert Detail Table and click the **Details** button, it will open **user_alert_details_alert2.rtv**.

MyAlert3 - This has the value "**_alert3**" for the **DrilldownSuffix** property. When you select an instance of **MyAlert3** from the Alert Detail Table and click the **Details** button, it will open **user_alert_details_alert3.rtv**.

Use the $rtvUserAlertActiveColumnEnabled substitution

- By default, the **Active** column in the **Alert Administration** display is hidden. To make it visible, specify a value of 1 for the **$rtvUserAlertActiveColumnEnabled** substitution on the **General>Substitutions tab** of the **Application Options** dialog.

Help files

- The help button (**?**) in the top left corner of the display is configured to display help files located in the docs directory of the demo. To modify this button to look for help files in another location, add a substitution named **$displayHelpURL** and set it to the url where you are hosting your help files.
- To change the name of the help file for each display, modify **ssa_displays.xml**. This xml file has an entry for each **.rtv** file in the demo with the name of the corresponding html file to load. This file will be loaded from the url specified in **$displayHelpURL**.

Integrate the Demo into an RTView Application

Before integrating this demo into your RTView application, you must configure it as described above to monitor your alerts and store the settings in the database of your choice.

Once you have done this, proceed with the following instructions:

1. Copy all of the **.rtv** files from the **demos/selfservicealerts** directory to your RTView application directory. If you will be using the built-in help files, also copy **ssa_displays.xml** and the **selfservicealerts\docs** directory to your RTView application directory.

2. Copy **ALERTOPTIONS.ini** and **CACHEOPTIONS.ini** files to the directory where you will be running the alerts. This might be your application directory, or it might be the directory where you are running the Data Server if you want the alerts to run in the Data Server.

Note: If this directory is not your application directory, move your alert configuration files (**rtv_alertdefs.rtv** and/or your other alert configuration files) and **rt_alerts_cache.rtv** to the same directory as **ALERTOPTIONS.ini** and **CACHEOPTIONS.ini**. Then, copy values from those files in the **selfservicealerts** directory into the corresponding **OPTIONS.ini** files in your application.

---

3. Add the **ALERTDB** option to the options file in the directory where you put the **ALERTOPTIONS.ini** and **CACHEOPTIONS.ini**:

- If the directory where you put **ALERTOPTIONS.ini** and **CACHEOPTIONS.ini** already contains an **OPTIONS.ini** file, then add the lines that start with sqldb ALERTDB and dbretry from **demos/selfservicealerts/OPTIONS.ini** file to that file.
- If the directory where you put **ALERTOPTIONS.ini** and **CACHEOPTIONS.ini** does not contain an **OPTIONS.ini** file, copy the **OPTIONS.ini** file from **demos/selfservicealerts** to there.

4. In your application directory, add the following display (.rtv) files to your panel configuration file:

   rtv_alerts_table.rtv (Alert Detail View)

   rtv_admin_alerts.rtv (Alert Administration)

   rtv_alerts_audit.rtv (Administration Audit)

5. The Self Service Alerts demo uses the style sheets **rtv_darkstyles.rts** and **rtv_flat.rtv** to set the look and feel of the displays. If your application already uses these style sheets or if you do not want to apply a style sheet, you are finished.

   a. If you want to apply the style sheets to your whole application:

      In the Display Builder select **Tools>Options**. In the **Application Options** dialog, select **General>Style Sheet**. Click **Add Built-in Styles** and select **rtv_darkstyles.rts** and **rtv_flat.rts**. If you do not want these style sheets applied to the main Display Builder window, deselect **Apply Style Sheets to Main Builder Window**. Save your options and exit the Display Builder.

   b. If you only want to apply the style sheets to the Self Service Alerts display files:

      Open each of these files in the Display Builder and in each display, select **Tools>Style Sheets**. Click **Add Built-in Styles** and select **rtv_darkstyles.rts** and **rtv_flat.rts**. Click **OK** and save each display.

6. If you will be deploying the Self Service Alerts in a thin client application and you will be using the built-in help files, include **selfservicealerts\docs** in the **.war** file for your thin client application. See the **make_war.bat** script, located in the **selfservicealerts** directory, as an example of how to include this in your war file.

Alert Persistence

This section contains the following:

Overview

The Alert Persistence feature stores all fields and current state of all active alerts, as well as all cleared alerts that have not been removed from the system in a database. You can configure the alert engine to persist alerts directly to the database or via a data server. More than one alert engine can persist its alerts to the same database table as long as each alert engine specifies a unique Alert Engine Name (specified on the **Alerts>**"Alert Persistence Tab" in the **Application Options** dialog).

With this feature enabled, alerts are persisted as follows: If you are running the alerts in a Data Server that is configured for High Availability, you must run the backup server in warm standby mode. All active and cleared alerts that had not previously been removed from the system will be restored to the standby server when it becomes the active server.

Note: You cannot run the standby server in hot standby mode when using alert persistence since, in that case, both Data Servers will generate alerts and persist them to the database. Also, with High Availability, if your persistence database is slow or you expect a large number of alerts to change each update period, you can use the **persistInitDelayTime** command line option to increase the delay time for when a backup server reads the primary server alert persistence database. Otherwise, there might not be enough time for the failing Data Server to write all the alerts to the database before the backup server reads them.

Even with high availability configurations, there are cases in which some alerts might not be persisted. For example:

- The persistence database fails. In this case, alerts cannot be written to, or read from, the database. If a failover occurs while the database is down, the backup server will not be able to read persisted alerts from the database. This will also happen if persistence is configured to use a Persistence Data Server to access the database, and the Persistence Data Server is down during failover.
- The Alert Server is terminated in a non-orderly shutdown. Alerts are written out once per update period and once during orderly termination. If there is a non-orderly shutdown, some alerts might not be written to the database.

In cases where alerts are not persisted, the new primary Data Server generates new alerts if the data is still in an alert state. The new primary Data Server might also re-use ID's that were used by the failed Data Server.

When the alert engine starts up, it will query the database table containing the persisted alerts and will recreate them at startup in the new alert engine. The persisted alerts will show up in the alert engine after the Initial Delay time (specified on the **Alerts>Alert Definitions** tab in the Application Options dialog) has expired. All alerts that are configured to renotify on a timer will immediately renotify, then continue to renotify on the specified renotification time.

To clear the alerts for an alert engine from the database on startup use the **purgepersistedalerts** command line option. This will clear all alerts for the alert engine from the database table on startup and no persisted alerts will be loaded. If you are persisting alerts for more than one alert engine in the same database table, alerts for other alert engines will not be removed. See "Options Enabled with Alerts" for more information.

For optimization, the alert engine does not store updates to the **Last Update Time** column of the "AlertTable" unless there has also been another change for that alert. In many cases this column is not used by RTView applications. However if you want to restore the alerts with accurate values in the Last Update Time column, you can use the **lutupdatesnewdata** command line option to enable this update. See "Options Enabled with Alerts" for more information.

There are two phases of configuration necessary to enable RTView to support the persistence of alerts:"Database Configuration" and "Alert Engine Configuration".

Database Configuration

You can either let the RTView create the database table or you can create the table. If you want to the RTView to create the table, select the **Create Database Table If Not Found** option (on the **Alerts>Alerts Persistence** tab in the **Application Options** dialog).

If you want to create the table yourself, table schemas for several databases are included in **RTV\dbconfig\alert_persist**. The README.txt file in that directory will assist you in using the schemas to create your table. If you want to create it manually, create a table with the following column names and types in the following order:

Column Name	Type
AlertEngineName	String
Time	Timestamp
Alert Name	String
Alert Index	String
Severity	Integer
Alert	Text String
Cleared	Boolean
Acknowledged	Boolean
ID	Long
Last Update Time	Timestamp
Row Update Time	Timestamp
Comments	String
Owner	String
<Custom Alert Definition Property and Custom Event Attribute Columns>	TBD
current	String
level	Integer

The **AlertEngineName** and **ID** fields are primary keys. If you have defined Custom Alert Definition Properties and/or Custom Event Attributes, you will need to insert columns of the correct type in the order they appear in the **AlertTable** before the current column.

Alert Engine Configuration

In the Display Builder, select **Tools>Options** to open the **Application Options** dialog. To configure the alert engine, select **Alerts>**"Alert Persistence Tab".

Alert Persistence Requirements and Limitations

Requirements

1. Alert definitions and options must be the same before and after failover. For example, if you are running a primary and standby Data Server in two different locations, you must confirm that both are configured with the same alert definitions and option. If your alert options have changed, persisted alerts might be incorrect after failover.

2. If any of the following properties on your alerts are attached to data, those data attachments must be available after failover and you must specify an Initial Delay time. This will allow the alert engine restoring the persisted alerts to resolve those data attachments before finishing the processing of the persisted alerts. The Initial Delay only needs to be long enough for your data attachments to resolve. Any of these properties that use a ; (semicolon) delimited list or a tabular data attachment must include index values, such as:
   - rowEnabledTable
   - value*Alert (i.e. valueHighAlert, valueLowAlert, valueMediumAlert)
   - value*Warning (i.e. valueHighWarning, valueLowWarning)
   - value*Text (i.e. valueHighAlertText, valueHighAlertCommandText, valueMediumAlertText, valueMediumAlertCommandText, valueLowAlertText, valueLowAlertCommandText, valueHighWarningText, valueHighWarningCommandText, valueLowWarningText, valueLowWarningCommandText)
   - alertState*AlertText
   - alertState*AlertCommandText
   - alertState*Comparison
   - alertState*UpperRangeLimit
   - alertState*LowerRangeLimit

3. The database table where the alerts are persisted must contain a column for each column in the **AlertTable**, along with a couple of additional columns. Therefore, if you add, edit or remove Custom Alert Definition Properties or Custom Alert Event Attributes, the database table will need to be modified.

4. If you have multiple alert engines persisting alerts to the same database table, they must have the same Custom Alert Definition Properties and Custom Alert Event Attributes.

Limitations

1. Alerts that attach directly or indirectly to a function that outputs a non-table value (Text or Number) may be incorrectly cleared after failover.

2. The **nonRepetitionTime** for an alert restarts after failover.

3. Depending on the frequency of data that drives the alert, it is possible that an update to an alert might be missed during failover. The likelihood of missing an update increases the longer the time is between the exit of the first engine and the startup of the second.

4. After failover, the per-alert variables will only contain the alert indexes that were persisted in the **AlertTable**. The other indexes will be added after data has been received for them.

Audit Alert Action

This section contains the following:

Overview

In many solutions, Alert Event Management can be the most critical part of an RTView application. Understanding what actions were taken by which user to resolve the situations related to active alerts are important pieces of data to fine tune work flow. Alert audits also help validate that correct actions were performed in a timely fashion when critical events occur.

Once Alert auditing is configured and enabled, the Alert data source will add an entry to the specified "Alert Action Audit Table" whenever an alert command is executed. Select **Tools>Options>**"Alerts Tab" to enable Alert Action Auditing.

The Alert data source provides auditing for the following commands. Each time one of these commands is executed, RTView will write a row to the specified Alert Action Audit Table displaying the Command Name in the **ACTION** field.

Command Name	Action	Type	Target Value
Acknowledge Alert	Event Management	Alert ID	Blank
Add Comment	Event Management	Alert ID	Comment
Clear Alert	Event Management	Alert ID	Blank
Clear Comments	Event Management	Alert ID	Blank
Set Custom Alert Event Attribute	Event Management	Alert ID	Attribute Value
Set Owner	Event Management	Alert ID	Owner
Add Alert Definition File	Configuration Management	Name of the alert definition file	Specified substitution(s) or blank
Enable Alerts	Configuration Management	Blank	Enable value
Enable Alert Definition Alert Name	Configuration Management	Name of the Alert Definition	Enable value
Remove Alert Definition File	Configuration Management	Name of the alert definition file	Specified substitution(s) or blank
Unacknowledge Alert	Event Management	Alert ID	Blank

Alert Action Audit Database

On the "SQL Tab" of the **Application Options** dialog, define connection to the database that contains the **Alert Action Audit** table in which alert actions audit information will be stored. Once defined, you must specify the **Alert Action Audit Database Name** on the "Alerts Tab" of the **Application Options** dialog or on the command line.

Alert Action Audit Table

The **Alert Action Audit** table must be configured with the following columns.

Note: Sample database table schemas are available under **RTV\dbconfig\alert_action_audit**.

Once the table is configured, you must specify the **Alert Action Audit Table Name** on the "Alerts Tab" of the **Application Options** dialog or on the command line.

Name	Type	Description
TIME_STAMP	Timestamp	Time the action was executed.
USER	String	User name from the RTView login or No Login if login was disabled.
ACTION TYPE	String	Type of action taken. The ACTION TYPE depends on the command. See "Audit Alert Action" for more information.
ACTION	String	Action taken. This will be the name of the command, with the exception of the Set Custom Alert Event Attribute command which will be Set <Attribute Name> (e.g. Set MyCustomAttrField).
TARGET	String	Target of the action. The TARGET depends on the command. See "Audit Alert Action" for more information.
VALUE	String	The value of the action. The VALUE depends on the command. See "Audit Alert Action" for more information.
ALERT_NAME	String	Name of the Alert. Valid only for Event Management type alert actions. **Note:** This field will be empty for Configuration Management type actions or if the value of the alert is scalar.
ALERT_INDEX	String	Index number of the Alert. Valid only for Event Management type alert actions. **Note:** This field will be empty for Configuration Management type actions.

CHAPTER 15 Caches

This section contains the following:

Overview

The Cache data source was specifically designed to enable high-speed analytic processing of real-time data and the comparison of current real-time values against history data, either visually or as input to alert rules that activate automated behavior.

The Cache data source allows for easy definition of a memory resident data cache that can store current scalar values, tabular data with optional filters, as well as time-stamped scalar or tabular data (up to a configurable number of entries), which keeps only the most current information and discards older entries. A cache history can be configured to time-stamp incoming data or to recognize a time-stamp already included in incoming data. You can also optionally pre-load a cache with data from another source, for example if you have historical information stored in a database.

In many real-time situations, data enters the system asynchronously. For example, in a financial trading application a series of data might be delivered that contains multi-dimensional data about a particular trading instrument. You might want to store the current value of a particular market data feed based on the instrument name index (IBM) and the activity index ("buy") and require a time-stamped history of this market data to analyze the trend of a particular instrument over a period of time.

The Cache data source allows for much higher performance than the traditional means of either performing analytics on information stored in a database or querying it back out of the database when it is necessary to make calculations.

A cache history table is kept in memory. The size and time range of the data in the history table is determined by the cache **maxNumberOfHistoryRows** property and (optionally) its **historyTimeSpan** property. The cache persistence feature enables the Historian to store cache history data permanently in a database table. The database and table names are specified by the cache object.

When the cache data source is initialized, it preloads the history tables of all caches that use the cache persistence feature by making appropriate SQL queries. However, the number of rows that can be loaded from the database into the cache in-memory history table is limited by the cache **maxNumberOfHistoryRows** property and (if specified) **historyTimeSpan** property values.

When making a data attachment to a cache history table, the **Attach to Data** dialog **Advanced Filter** options can be used to specify the time period of interest (**Time Range**, **Begin Time**, and **End Time** fields).

Extend with SQL

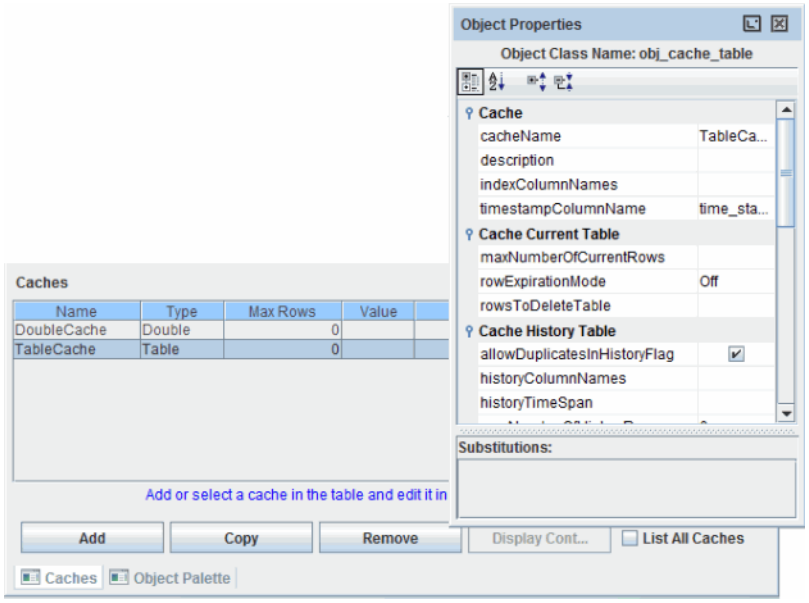
The **Extend with SQL** option allows you to configure a cache to use a SQL query to retrieve data from an external database. This option provides a convenient way to retrieve and display cache data that is stored in an RTView Historian database. When this option is not enabled the cache data source retrieves data from the in-memory cache only. To use this option, make a data attachment to a cache history table, select **Advanced Filter Rows**, specify the time period of interest, and select **Extend with SQL**. For details, see "Filter Rows: Advanced".

Data Compaction

Primary compaction operates on in-memory cache tables, condensing a cache history table by periodically applying a calculation (for example, average, min, max, etc.) to each column. It then stores a single result row in another cache table. Primary compaction is configured using the "Data Compaction: Primary (in-memory) Properties". This feature complements the Historian data compaction, or secondary compaction feature, which operates on database tables. Secondary compaction can be configured to compact rows in database tables as the data "ages". Secondary compaction is configured using the "Data Compaction: Secondary (Historian) Properties".

Adding Caches

In the Display Builder, select **Tools>Caches** to open the **Caches** dialog. When you have finished adding all of your caches and configuring their properties, Save the display (**.rtv**) file and add this cache definition file to the Cache data source configuration. See "Cache Definition Files" for more information.



Field Name	Description
Add	To add a new cache, click **Add** and enter a cache name and select a cache type. Once you have added a cache, select that cache from the list and edit properties in the **Object Properties** dialog. **Cache Name** - Enter a unique name for each cache listed. Caches that do not specify a name, as well as caches with a duplicate names, will not be created and an error will print to the console. **Cache Type** - Choose a type of cache from the drop down menu ("Table Cache" or "Double Cache").
Copy	Select a cache from the list and click **Copy** to create a duplicate of that cache. **Note:** You must enter a unique name for each copy you make. To copy a cache from your current display to another display (**.rtv**) file, select a cache from the list and click the Copy button in the toolbar (or Ctrl-C). Then open the other display (.rtv) file and click the Paste button (or Ctrl-V). If a cache by that name already exists in that display (.rtv) file, you will need to rename the cache. **Note:** A cache pasted into another display (**.rtv**) file will have the same data attachments as the original cache.
Remove	Select a cache from the list and click **Remove** to delete.
Display Content	Select a cache from the list that is attached to data and click **Display Content** to view the contents, as well as the number of columns and rows, in the attached current and/or history table(s).
List All Caches	Select to view caches from all display (**.rtv**) files and included files. See "Include Display Files" for more information. **Note:** Only properties for caches defined within the current display (.rtv) file can be modified.

Cache Definition Files

To create an Cache Definition file, save the display (**.rtv**) file that contains the cache definition properties you set. See "Application Options -- Caches" for details on how to add a Cache Definition file to the Cache data source configuration.

Viewing Caches

See the "Attach to Cache Data" section for more information on attaching to cache data variables.

Table Cache

The Table Cache allows you to cache your tabular data. Attach your input data to the **valueTable** property.

Cache Properties

cacheName	A unique name for the cache. An entry for this property is required. Caches that do not specify a cacheName, as well as caches with duplicate cacheName properties, will not be created and an error will print to the console.
	Substitutions defined in a Cache Definition File allow the same file to be specified multiple times with different sets of substitutions. Note that each cacheName defined in that file should contain a substitution, since cache names must be unique. See "Application Options -- Caches" for more information.
	Substitutions must use the following syntax:
	`$subname:subvalue $subname2:subvalue2`
	If a substitution value contains a single quote, it must be escaped using a / :
	`$filter:Plant=/'Dallas/'`
	If a substitution value contains a space, it must be enclosed in single quotes. Do not escape these single quotes:
	`$subname:subvalue $subname2:'sub value 2'`
	A substitution string cannot contain the following:
	: | . tab space , ; = < > ' " & / \ { } [ ] ( )

description	Enter descriptive text which is displayed in the **Description** column of the **Caches** table.
indexColumnNames	Enter a semicolon (;) separated list of column names from the attached **valueTable** to be used as index columns.
timestampColumnName	Specify the name of a **timestamp** column. If the specified **timestampColumnName** is not found in the incoming data, a column will be inserted and each row will contain the time that row was received.
queryServerName	Specifies the name of the Data Server to use to perform the initial SQL query to preload the cache's tables, and also to perform any Extend-by-SQL queries for data attachments to the cache. By default, the value of the property is blank, which means that the SQL queries are performed by the local application. **Note:** SQL queries are only performed if the cache's **historyTableName** or **currentTableName** properties specify a database table name.
updateListenersImmedFlag	If selected, then each time new data is stored in the cache the updated contents of the cache's tables are immediately applied to any listeners attached to those tables. This is the default setting and is appropriate for most caches. If deselected, the updated contents of the cache's tables are not applied to the listeners immediately after new data is stored in the cache. Instead the listeners are updated on the next RTView update cycle; default is 2 seconds. In very specific conditions, this can improve the performance of the Data Server. For example, in a case where the cache is maintained in a Data Server and raw event-driven data (e.g. high-frequency event data from a JMS, RV, or RTVAgent data attachment) is frequently stored in the cache and the cache's current table or history table can grow to more than 25,000 rows.

Cache Current Table Properties

maxNumberOfCurrentRows

Specifies the maximum number of rows in a cache current table. The oldest rows in the current table are removed if necessary to maintain this limit. The default value is blank, indicating no limit on the number of rows. A value of zero can also be used to specify no limit.

The cache data source checks the **maxNumberOfCurrentRows** limit at approximately 10 second intervals, therefore the limit might be exceeded if new data is applied to the cache between those intervals.

Note: In many cases it is not necessary to place a numerical limit on the rows in the current table, since the rows are limited by the number of unique combinations of index column values, or rows are removed via other properties. In the case of a cache with no index columns, each update replaces the entire current table.

rowExpirationMode

Specifies whether to expire rows in the current table and, if so, the mode for doing so. Possible values are:

Off - Row expiration is disabled.

Mark - An Expired column is added to the current table and its value is set to true in each expired row.

Delete - Expired rows are deleted from the current table.

Mark & Delete - An Expired column is added to the current table, its value is set to true in each expired row. Eventually, expired rows are deleted from the current table.

The default value is **Off**. If this property is set to **Mark**, **Delete** or **Mark & Delete**, the following additional properties are available in the property sheet: **rowExpirationTime**, **rowExpiredIndexColumns** and **rowExpiredColumnName**.

If this property is set to **Mark & Delete**, the **rowExpirationTimeForDelete** property is available in the property sheet.

rowExpirationTime

This property is available when the **rowExpirationMode** property is set to **Mark, Delete**, or **Mark & Delete**. If row expiration is enabled on a cache (**rowExpirationMode** is not **Off**), the current table is periodically checked for expired rows. A row is expired if the difference between the current time and the **timestamp** in the row **timestamp** column is greater than the interval specified by the **rowExpirationTime** property. This check is performed once every 10 seconds, approximately.

Specify the duration in seconds, or specify a number followed by a single character indicating the desired time interval (e.g. 15m for 15 minutes). Valid characters are as follows:

y - years (365 days)

M - months (31 days)

w - weeks (7 days)

d - days

h - hours

m - minutes

s - seconds

The default value is blank, which specifies no expiration time. A row in the current table is considered expired when it is not updated per the time interval specified by the **rowExpirationTime** property.

Note: The cache data source checks the expiration time limit at approximately 10 second intervals, therefore the limit might be exceeded by up to 10 seconds.

rowExpirationTimeForDelete

This property is available only when the **rowExpirationMode** property is set to **Mark & Delete**. A row is deleted if the difference between the current time and the **timestamp** in the row **timestamp** column is greater than the interval specified by this property. The value of this property should specify a longer time interval than that specified for the **rowExpirationTime** property. The default value is blank, which specifies no deletion time.

Specify the duration in seconds, or specify a number followed by a single character indicating the desired time interval (e.g. 15m for 15 minutes). Valid characters are as follows:

y - years (365 days)

M - months (31 days)

w - weeks (7 days)

d - days

h - hours

m - minutes

s - seconds

This property is useful in cases where a row in the cache current table should be marked as expired after the time period indicated by the **rowExpirationTime** has passed and the row has not been updated, and then later the row should be deleted after the time indicated by **rowExpirationTimeForDelete** has passed.

For example, if a cache has the following property settings:

```
rowExpirationMode: Mark & Delete
rowExpirationTime: 30m
rowExpirationTimeForDelete: 4h
```

and a row in the cache current table is not updated for 30 minutes, the value of the **Expired** column for that row will be set to true. If a row is not updated for 4 hours, it will be deleted from the current table.

rowExpiredColumnName

This property is available when the **rowExpirationMode** property is set to **Mark** or **Mark & Delete**.

Specifies the name of the boolean column that is added to the cache current table to indicate if a row is expired. The default value is **Expired**.

rowExpiredIndexColumns	Previously named **indexColumnNamesForDelete**, this property is available when the **rowExpirationMode** property is set to **Mark**, **Delete**, or **Mark & Delete**.
	Enter a semicolon (;) separated list of column names. The list must contain some or all of the column names specified in the cache **indexColumnNames** Property. The default value is blank. See "Table Cache" for more information.
	On each update to the cache, the set of values contained in the update for the columns specified in this property is recorded. A row in the current table is considered expired when it has a matching value for those columns but was not updated.
	This property can be used in conjunction with the **rowExpirationTimeForDelete** property. For example, if a cache is configured as follows:
	`rowExpirationMode = 3 (Mark + Delete)`
	`rowExpirationTime = -1`
	`rowExpirationTimeForDelete = any value > 0`
	`rowExpiredIndexColumns = any subset of the cache's index columns`
	then if a row X exists in the cache's current table and that row's values in the columns specified by the **rowExpiredIndexColumns** property are not matched in the data table applied to the cache's **valueTable** property, the row is marked Expired. Subsequently, if row X is not updated again and the time specified by the **rowExpirationTimeForDelete** property elapses, then row X is deleted from the cache's current table.

rowsToDeleteTable	Previously named **deleteTable**, this property is intended to be attached to a data table that specifies the rows to be deleted from the cache current table. The data table attached to the **rowsToDeleteTable** is expected to contain the columns specified in the **indexColumnNames** property. If a row in the **rowsToDeleteTable** has the same values in each of the index columns as does a row in the cache current table, that row will be deleted from the current table. The history table is not affected

Cache History Table Properties

The properties **historyTimeSpan**, **maxNumberOfHistoryRows**, and **maxNumberOfRowsPerIndex** work together to control the size of the History table.

allowDuplicatesInHistoryFlag	If this property is deselected, then before adding a new row to the History table the cache data source will first check if an existing row has the identical **timestamp** and index column values. If a duplicate does exist, the existing row will be replaced with the new row. **Note:** If duplicate history rows are acceptable or unlikely to occur, then it is recommended that this property remain selected to maintain optimum performance.
historyColumnNames	Enter a semicolon (;) separated list of column names from the cache current table to be stored in the cache history. The order in which the column names are specified does not affect the order in which they appear in the history table, they appear in the same order as they do in the current table. By default, the cache current table and cache history table store the same columns. In many cases this is desirable. Data sources may supply data tables containing columns whose values are static (for example, units, version number, description, etc). It can be useful to store these values in the cache current table, but they can take up memory and database space when storing data in the history tables at every update.

	The columns specified by **historyColumnNames** must be a subset of the columns in the cache current table. If a specified column does not appear in the current table, it is ignored. If a column is not specified in **historyColumnNames** but the column is specified in either **indexColumnNames** or **timestampColumnName**, the column is still included in the history table. This is because the index column and the **timestamp** column are needed for use and maintenance of the history table. Changing the names specified in **historyColumnNames** could lead to incompatibility with data previously stored by the Historian. Note that this is also an issue with changes made to the cache **schemaTable** or the cache **valueTable** attachment. The default value is blank, which specifies that all columns from the current table be stored in the history table.
historyTimeSpan	Previously named timeSpan, this property specifies the amount of time that rows will be kept in the History table according to their **timestamp**. The cache trims its History table by removing rows with timestamps that are older than the limit. Specify the duration in seconds or specify a number followed by a single character indicating the desired time interval (e.g. 15m for 15 minutes). Valid characters are as follows: **y** - years (365 days) **M** - months (31 days) **w** - weeks (7 days) **d** - days **h** - hours **m** - minutes **s** - seconds The **historyTimeSpan** property only determines the duration of rows kept in the History table by the cache data source. It does not affect database storage, if any, associated with the cache. **Note:** The **maxNumberOfHistoryRows** property will limit the total number of rows in the History table regardless of the specified **historyTimeSpan**.
initialTable	Attach this property to a table to pre-populate initial data for a cache. Specify the **initialTable** property to ensure that data is available for viewing when the cache is accessed for the first time. To properly populate the initial data for the cache using the **initialTable** property: • Specify the index columns for queries using the **indexColumnNames** property. See "Table Cache" for more information. • The SQL query that populates the **initialTable** must have the same index column(s) specified as those used to setup the cache. • The table must contain the same column(s) as the input table attached to the **valueTable** property. • The table must contain a **timestamp** column if **timestampColumnName** is specified. If the specified **timestampColumnName** is not found in the incoming data, a column is inserted.

	Data attached to this property works best if it is only updated once at the startup of RTView. **Note:** A table attached the **initialTable** property will be modified if you did not include a column with the **timestampColumnName** and/or if the **rowsToDeleteTable** property is updated. If other objects in your RTView application are attached to this same table, those objects might also be affected.
maxNumberOfHistoryRows	The number of rows in the **History** table. Default is **0**. If set to 0, no **History** table is created and only the Current table is available in the Table drop down menu in the "Attach to Cache Data" dialog. The "Extend with SQL" option requires this property to have a value greater than 0.
maxNumberOfRowsPerIndex	Enter the maximum number of rows the History table will contain for each unique combination of the cache's index column values. By default the value is blank, meaning that no per-index limit is applied to the History table. If the **condenseRowsFlag** is selected, then the **maxNumberOfRowsPerIndex** will be applied to the cache's **history_condensed** table. If **condenseRowsCombineHistoryFlag** is also set, then the **maxNumberOfRowsPerIndex** will also be applied to the cache's **history_combo** table. The **maxNumberOfRowsPerIndex** is checked only once per minute or, in the case of the **history_condensed** table, at the condense interval, which ever is less frequent. This means that the per-index limit may be exceeded until the next time the limit is checked. **Note:** The **maxNumberOfRowsPerIndex** property is not visible if the value of **maxNumberOfHistoryRows** is 0 and therefore no **History** table exists.

Data Properties

resetTrigger	When data values attached to this property receive an update the cache will be cleared.
schemaTable	Optionally attach this property to any tabular data source (e.g. SQL query, XML file, etc.) to provide the structure of a table you would like the Table Cache object to use. Only the table's structure will be used from this data attachment, any row data will be ignored. Once a **schemaTable** is specified, Current and History tables will be initialized accordingly and the **Attach to Cache Data** dialog will show the columns found in that table structure. Once data becomes available, Table Cache objects will contain the columns found in the specified **schemaTable**, in addition to columns found in the data. **Note:** If the data types of columns defined in the **schemaTable** differ from data types found in the actual data attachment, then the schema types will be used and incoming data will be converted. If the specified **timestampColumnName** is not found in the incoming data, a column will be inserted.

schemaTableStrictFlag	If selected, then the **schemaTable** will be applied to all incoming data tables before they are stored in the cache even when the data table and the schema table have the same number of columns. If deselected, the schema is applied only when the data table and schema table have different numbers of columns. **Note:** This option should only be selected if it is possible for a data table to have the expected number of columns, but those columns may be in the wrong order. If incoming data tables have a large number of columns and a high update frequency, then applying the schema in all cases will incur additional CPU overhead when updating a cache.
valueTable	Attach your input data to this property if it is tabular. For scalar input data, select the **Double Cache** object and attach your input data to the value property.
valueTableCount	Allows the **Table Cache** object to support multiple data attachments. Default is **1**, which results in a single **valueTable** property. If **valueTableCount** is increased to 2, then **valueTable02** will be added to the **Object Property** list and so on. Maximum value is 99.

Data Compaction: Primary (in-memory) Properties

The Data Compaction: Primary (in-memory) category of properties is available when a tabular cache object is selected in the Builder.

condenseRowsCombineHistoryFlag	If selected, the **history_combo** table is made available. This property is available when the **condenseRowsFlag** property is selected. The **history_combo** table stores rows of recent raw data followed by rows of older condensed data. The raw data rows extend from the current time back for the number of seconds specified by the **condenseRowsRawDataTimeSpan** property. The condensed rows are limited by the **historyTimeSpan** and/or **maxNumberOfHistoryRows** properties.
condenseRowsFlag	If selected, enables the primary compaction feature for the cache. This property makes the following tables available (in addition to the tables available by default, the Current and History tables.): • **current_condensed**: This table stores the most recent row of condensed data for each unique index column value combination. • **history_condensed**: This table stores condensed data history, limited by the cache **historyTimeSpan** and/or **maxNumberOfHistoryRows** properties. If you also select the **condenseRowsCombineHistoryFlag** property, the **history_combo** table is made available: • **history_combo**: This table stores rows of recent raw data followed by rows of older condensed data. The raw data rows extend from the current time back for the number of seconds specified by the **condenseRowsRawDataTimeSpan** property. The condensed rows are limited by the **historyTimeSpan** and/or **maxNumberOfHistoryRows** properties.

	This property affects RTView behavior when persisting a cache to, or restoring it from, a database. For example, if **condenseRowsFlag** is selected on cache C1: • If the cache definition file is used as an Historian configuration file, the Historian stores only the condensed rows for C1 in the database table specified by C1's **historyTableName** property. • When the cache definition file is loaded by an RTView application (other than the Historian) the database table specified by C1's **historyTableName** is queried to get the initial rows for C1.history_condensed and, if enabled, C1.history_combo. This is useful in cases where the Historian should only store data that has already undergone primary compaction, rather than storing raw data which arrives in high volumes. When **condenseRowsFlag** is selected, the following additional properties are available in the property sheet: **condenseRowsInterval**, **condenseRowsGroupBy**, **condenseRowsRawDataTimeSpan** and **condenseRowsCombineHistoryFlag**.
condenseRowsGroupBy	A string that specifies the calculation to be performed on each data column in the cache table. This property is available when the **condenseRowsFlag** property is selected. Specify the string using the following format: `name1:calc1;name2;calc2...` where **nameN** is the name of data column **N**, and **calcN** is the calculation to be performed on the values in the column for each time interval. The supported calculations are average, min, max, and sum. The value for this property is configured from a dialog which allows the user to pick a column name and the calculation for that column. If a calculation is not specified for a data column, sum is used. Calculations are not applied to the data table index columns, nor to the **timestamp** column.
condenseRowsInterval	The time interval at which the cache history is condensed. This property is available when the **condenseRowsFlag** property is selected. Specify a value using the following format: **NNu** where **NN** is a number and **u** is a single character. Valid characters are as follows: **w** - weeks (7 days) **d** - days **h** - hours **m** - minutes **s** - seconds For example, to specify a ten minute interval: **10m** If only a number is entered, it is assumed to be seconds.
condenseRowsRawDataTimeSpan	The time span of raw data kept in the cache history table and, if enabled, its **history_combo** table. This property is available when the **condenseRowsFlag** property is selected.

Data Compaction: Secondary (Historian) Properties

compactionType	Select from the following options: • **None** - No compaction. • **Aggregate** - Compact via data aggregation. • **Displace** - Compact via table displacement. You must specify at least one index column via the **indexColumnNames** property and, where possible, **quoteColumnNamesFlag** should be selected to preserve column name case. **Note:** Based on your specified **compactionRules**, use the **-smoothcompaction** command line option to perform compaction on old data which currently exists in your database from prior executions of the Historian. See "Command Line Options: Historian" for more information. **compactionGroupBy** - This property applies only if the selected **compactionType** is Aggregate. Click on the button to specify the methods used to perform compaction (i.e. sum, count, average, min or max). **compactionRules** - Specify compaction rules. **Note:** If the selected **compactionType** is Displace, then currently this property only supports the use of one displacement rule. **Note:** These object properties apply to Advanced Historian users only. See "Advanced Historian - Cache Properties" for more information.

Historian Properties

You must specify your Cache definition (**.rtv**) file as an Historian data configuration (**.rtv**) file in order use the following object properties. See "Configuring the Historian" for more information.

Property Name	Description
currentTableName	Enter the name of a table in your Historian database in which to store the cache's current data table. If you specify a **currentTableName** and no table by that name exists in your database, then one will be created for you the first time you run the Historian. On each update of the data attached to the **valueTable** property, the Historian will create or update corresponding rows in the specified **currentTableName** in the Historian database. The **indexColumnNames** property will be used to locate rows to update otherwise, if no **indexColumnNames** are defined, the Historian will replace the entire table on each update. If the **rowsToDeleteTable** property is used to delete rows from the cache's current table, then Historian will also delete those rows from the specified **currentTableName** table in the Historian database. On startup of an RTView application, the specified **currentTableName** will be queried from the Historian database for initial rows to be loaded into the cache's current table. To prevent this initial data from being loaded, you can run the Display Builder or Display Viewer from a command window and use the **-nohistory** command line option. The **currentTableName** property is not supported in combination with the Historian's **-kdbformat** command line option.

databaseName	Name of your Historian database that contains history and/or current tables for this cache. If left blank, the database name RTVHISTORY is assumed. The "Extend with SQL" option requires that the database specified in this property have a SQL database connection. Or, if this property is blank, there must be a database connection to the default RTVHISTORY database.
historyTableName	Enter the name of a table in your Historian database in which to store the cache history data table. If you specify a **historyTableName** and no table by that name exists in your database, one is created for you the first time you run the Historian. When rows are appended to the cache history table, the Historian also appends those rows to the specified **historyTableName** in the Historian database. If you specify a **timestampColumnName**, the Historian assumes that column is to be used for sorting and purging rows in chronological order and does not append its own **timestamp** columns to the specified **historyTableName** table. Before data is stored in the database, the cache **schemaTable**, if any, is applied. If you specify a **historyTableName**, RTView applications automatically query the Historian database table for the initial rows to be loaded into the cache history table, provided that: • The **initialTable** property is not attached to data (indicating an explicit query has been configured to load cache history data), and • A **timestampColumnName** is specified (so that it can be used to sort the SQL query result by time). The number of rows loaded by this initial query is limited by the **maxNumberOfHistoryRows** property. To prevent this initial data from being loaded, you can run the Display Builder or Display Viewer from a command window and use the **-nohistory** command line option. The "Extend with SQL" option requires that this property specify the cache database table name. The table specified in this property is used for queries.
quoteColumnNamesFlag	If selected, column names will be quoted in all SQL queries and commands for this cache.

Object Properties

Property Name	Description
anchor	Specifies where to anchor an object in the display.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objX	Plot x-axis position of object.
objY	Plot y-axis position of object.
styleClass	Enables you to apply distinct styles to different instances of the same object class. The value entered for **styleClass** must not contain spaces and must not start with rtv.

Double Cache

The Double Cache allows you to cache your scalar (double) data. Attach your input data to the **value** property.

Cache Properties

cacheName

A unique name for the cache. An entry for this property is required. Caches that do not specify a **cacheName**, as well as caches with duplicate **cacheName** properties, will not be created and an error will print to the console.

Substitutions defined in a Cache Definition File allow the same file to be specified multiple times with different sets of substitutions. Note that each **cacheName** defined in that file should contain a substitution, since cache names must be unique. See "Application Options -- Caches" for more information.

Substitutions must use the following syntax:

```
$subname:subvalue $subname2:subvalue2
```

If a substitution value contains a single quote, it must be escaped using a / :

```
$filter:Plant=/'Dallas/'
```

If a substitution value contains a space, it must be enclosed in single quotes. Do not escape these single quotes:

```
$subname:subvalue $subname2:'sub value 2'
```

description

Enter descriptive text which is displayed in the **Description** column of the Caches table.

timestampColumnName

Specify the name of a **timestamp** column. If the specified **timestampColumnName** is not found in the incoming data, a column will be inserted and each row will contain the time that row was received

Cache History Table Properties

historyTimeSpan

Previously named timeSpan, this property specifies the amount of time that rows will be kept in the History table according to their **timestamp**. Specify the duration in seconds or specify a number followed by a single character indicating the desired time interval (e.g. 15m for 15 minutes). Valid characters are as follows:

y - years (365 days)

M - months (31 days)

w - weeks (7 days)

d - days

h - hours

m - minutes

s - seconds

The **historyTimeSpan** property only determines the duration of rows kept in the History table by the cache data source. It does not affect database storage, if any, associated with the cache.

The **maxNumberOfHistoryRows** property will limit the total number of rows in the History table regardless of the specified **historyTimeSpan**.

initialTable	Attach this property to a table containing initial values for the cache. This attachment must result in a table with a single column of type double, as well as a **timestamp** column if **timestampColumnName** is specified. **Note:** If the specified **timestampColumnName** is not found in the incoming data, a column will be inserted. Data attached to this property works best if it is only updated once at the startup of RTView. **Note:** A table attached the **initialTable** property will be modified if you did not include a column with the **timestampColumnName** and/or if the **rowsToDeleteTable** property is updated. If other objects in your RTView application are attached to this same table, those objects may also be affected.
maxNumberOfHistoryRows	Previously named **maxNumberOfRows**, this property specifies the number of rows in the History table. Default is **100**. If set to 0, no History table is created and only the Current table will be available in the Table drop down menu in the "Attach to Cache Data" dialog. The "Extend with SQL" option requires this property to have a value greater than 0.

Data Properties

resetTrigger	When data values attached to this property receive an update the cache will be cleared.
value	Attach your input data to this property if it is scalar. For tabular input data, select the Table Cache object and attach your input data to the **valueTable** property.

Historian Properties

Property Name	Description
databaseName	Name of your Historian database that contains history and/or current tables for this cache. If left blank, the database name RTVHISTORY is assumed. The "Extend with SQL" option requires that the database specified in this property have a SQL database connection. Or, if this property is blank, there must be a database connection to the default RTVHISTORY database.
historyTableName	Enter the name of a table in your Historian database in which to store the cache's history data table. If you specify a **historyTableName** and no table by that name exists in your database, then one will be created for you the first time you run the Historian. When rows are appended to the cache's history table, the Historian will also append those rows to the specified **historyTableName** in the Historian database. If you specify a **timestampColumnName**, the Historian assumes that column will be used for sorting and purging rows in chronological order and will not append its own **timestamp** columns to the specified **historyTableName** table. Before data is stored in the database, the cache's **schemaTable**, if any, is applied.

Property Name	Description
	If you specify a **historyTableName**, RTView applications will automatically query the Historian database table for the initial rows to be loaded into the cache's history table, provided that: • The **initialTable** property is not attached to data (indicating an explicit query has been configured to load cache history data), and • A **timestampColumnName** is specified (so that it can be used to sort the SQL query result by time). The number of rows loaded by this initial query is limited by the **maxNumberOfHistoryRows** property. To prevent this initial data from being loaded, you can run the Display Builder or Display Viewer from a command window and use the **-nohistory** command line option. The "Extend with SQL" option requires that this property specify the cache database table name. The table specified in this property is used for queries.
quoteColumnNamesFlag	If selected, column names will be quoted in all SQL queries and commands for this cache.

Object Properties

Property Name	Description
anchor	Specifies where to anchor an object in the display.
objName	Name given to facilitate object management via the **Object List** dialog. Select **Tools>Object List**.
objX	Plot x-axis position of object.
objY	Plot y-axis position of object.
styleClass	Enables you to apply distinct styles to different instances of the same object class. The value entered for **styleClass** must not contain spaces and must not start with rtv.

Attach to Cache Data

The Cache data source stores real-time data from data attachments in memory resident tables, which can themselves be used as data sources. See the "Overview" section for information on adding a cache. From the **Object Properties** window you can access the **Attach to Cache Data** dialog to connect an object to this in-memory data. Once a property has been attached to cache data, it receives continuous updates.

To attach your data, right-click on the Property Name from the **Object Properties** window and select **Attach to Data>Cache**. The **Attach to Cache Data** dialog opens. This dialog provides drop-down menus that allow you to specify information regarding the cache data you want to display. If the item you require is not listed, type your selection into the field.

You can choose to read data directly from the Cache data source, or through a configured Data Server. When you read data directly from the Cache data source, the **Cache** drop-down menu in the **Attach To Cache Data** dialog lists all available caches. See the "Overview" section for information on adding a cache. Drop down menus for **Column(s)** and **Filter Column** populate based on the selected Cache and Table. The **Table**, **Column(s)**, and **Filter Column** drop-down menus will only contain options if the selected Cache contains tabular information.

To read data through a configured Data Server the Builder must be connected to that Data Server. In the **Attach To Cache Data** dialog, select the Data Server from the Data Server drop-down list. The names of caches deployed on the default or named Data Server populate the **Cache**, **Table**, and **Column(s)** drop-down lists.

It is possible to indicate multiple columns for the filter and multiple values to compare against for each column. If the number of specified column names does not correspond to the number of values listed, extra names and/or values are ignored.

Note: Spaces around separators are not allowed.

When an object property is attached to data, the Property Name and Value in the **Object Properties** window is displayed in green. This indicates that editing this value from the **Object Properties** window is no longer possible. To remove the data attachment, and resume editing capability in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. An object property is detached from the data source when the Property Name and Value are no longer green.

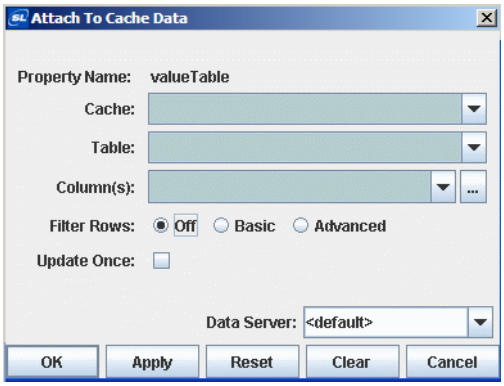
Attach To Cache Data - Filter Modes

There are three filter options, or modes, for attaching to cache data:

- "Filter Rows Off" on page 987
- "Filter Rows: Basic" on page 989
- "Filter Rows: Advanced" on page 992

Filter Rows Off

Select this option if you do not need to filter rows.



Field Name	Description
Cache	Name of the cache to display. The drop-down menu contains the names of all active caches. See the "Overview" section for information on adding a cache. If a selection is made in this dialog from the **Data Server** drop-down list, this field is populated by that selection.
Table	Select the type of table data to display: **Current** or **History**. If a selection is made in this dialog from the **Data Server** drop-down list, this field is populated by that selection. If you selected the **condenseRowsFlag** property for the cache, the **current_condensed** table and the **history_condensed** table are available in the list. If you also selected the **condenseRowsCombineHistoryFlag** property, the **history_combo** table is available in the list. When **history_combo** is selected, the **Update Once** check box will be replaced by three Update buttons: **Once**, **On Condense**, and **Always**. Default is On **Condense**.
Column(s)	Select the column(s) to display from the selected Table. If a selection is made in this dialog from the **Data Server** drop-down list, this field is populated by that selection.
Filter Rows	**Off** - Select not to filter rows. Filters can only be used for tabular data. **Basic** - For details, see "Filter Rows: Basic". **Advanced** - For details, see "Filter Rows: Advanced".
Update	**Once** - Update listeners only once when display is opened. **On Condense** - Update at the interval specified by the cache's **condenseRowsInterval** property, rather than whenever new raw data is applied to the cache. This is useful when, for example, the **traceValueTable** of a **trendgraph** is attached to the **history_combo** table. **Always** - Update whenever new raw data is applied to the cache. This is useful when, for example, the **valueTable** property of a table object is attached to the **history_combo** table. **Note:** The **On Condense** and **Always** options only apply if the selected Table is **history_combo**.
Data Server	Select to read data through your configured Data Server and not directly from the Cache data source. The names of caches deployed on the default or named Data Server populate the **Cache**, **Table**, and **Columns** drop-down lists. **Note:** If the Display Builder requests the cache names from a Data Server and does not get a response within 10 seconds, the drop-down lists will be empty. **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab". Populates the **Cache**, **Table**, and **Columns** drop-down lists with the names of the caches deployed on the default Data Server. If the Builder is not connected to a default Data Server, the names of the caches loaded locally by the cache data source are shown.

None - Select to bypass data being redirected through the specified Data Server(s) for this attachment and instead attach directly to the data source. Populates the **Cache**, **Table**, and **Columns** drop-down lists with the names of the caches loaded locally by the cache data source.

Named Data Servers - Select a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". Populates the **Cache**, **Table**, and **Columns** drop-down lists with the names of the caches deployed on that Data Server.

Multi-Server Attachment - To configure multiple data servers, enter a semicolon (;) delimited list containing two or more Named Data Servers (e.g. ds101;ds102). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**).

The values **__default** and **__none** begin with two underscore characters.

Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

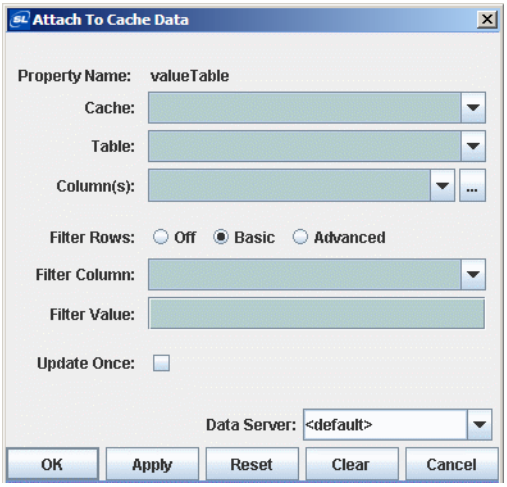
When multiple data servers are specified, the data attachment will be directed to each data server in the list. For tabular data attachments, a column named **DataServerName** will be added as the first column of the table and contain the name of the server from which the data was received.

A multi-server attachment will receive data independently from each of the servers it specifies, so in most cases it will be necessary to combine the tables received into a single table. This can be accomplished in two ways:

1. The multi-server attachment can be applied to a local cache that has the **DataServerName** column specified as an index column. The current table of that cache will contain the combination of the tables received from all servers. **Note:** It may also be necessary to configure cache row expiration settings to remove defunct rows.
2. The multi-server attachment can be applied as the **Table** argument of the RTView function named **Combine Multi-Server Tables**. See "Tabular Functions" for more information.

Filter Rows: Basic

Select this option to filter rows by column(s) or values(s). See "Row Filtering" for more information.

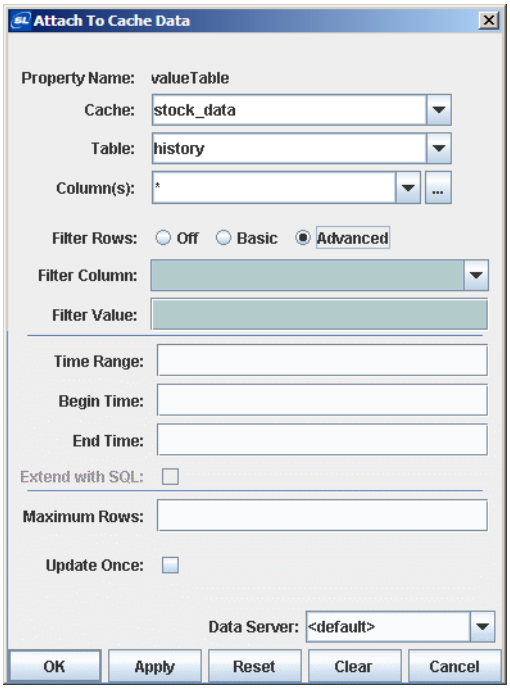


Field Name	Description
Cache	Name of the cache to display. The drop-down menu will contain the names of all active caches. See the "Overview" section for information on adding a cache. If a selection is made in this dialog from the **Data Server** drop-down list, this field is populated by that selection.
Table	Select which type of table data to display: **Current** or **History**. If a selection is made in this dialog from the **Data Server** drop-down list, this field is populated by that selection. If you selected the **condenseRowsFlag** property for the cache, the **current_condensed** table and the **history_condensed** table are available in the list. If you also selected the **condenseRowsCombineHistoryFlag** property, the **history_combo** table is available in the list. When **history_combo** is selected, the **Update Once** check box will be replaced by three Update buttons: **Once**, **On Condense**, and **Always**. Default is **On Condense**.
Column(s)	Select which column(s) to display from the selected Table. If a selection is made in this dialog from the **Data Server** drop-down list, this field is populated by that selection.
Filter Rows	**Off** - For details, see "Filter Rows Off". **Basic** **Filter Column** - Name of the column to use as a filter. Multiple column names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col 3). If your column name contains a space or a semicolon, then the entire name must be enclosed in single quotes. **Filter Value** - Value that the **Filter Column** must equal. Multiple filter values should be entered as a nested list, where values for a given column are separated by commas within a semicolon (;) delimited list (i.e. val1,val2;val3,val4;val5,val6). If your filter value contains a space or a semicolon, then the entire value must be enclosed in single quotes. When ***** is entered as a filter field value, data for all values in the specified filter column will be used to update the object property. When ***** is entered, only the literal comparative value will be used. These are only allowed for objects which display tabular data. **Advanced -** For details, see "Filter Rows: Advanced"

Update	**Once** - Update listeners only once when display is opened. **On Condense** - Update at the interval specified by the cache's **condenseRowsInterval** property, rather than whenever new raw data is applied to the cache. This is useful when, for example, the **traceValueTable** of a trendgraph is attached to the **history_combo** table. **Always** - Update whenever new raw data is applied to the cache. This is useful when, for example, the **valueTable** property of a table object is attached to the **history_combo** table. **Note:** The **On Condense** and **Always** options only apply if the selected Table is **history_combo**.
Data Server	Select to read data through your configured Data Server and not directly from the Cache data source. The names of caches deployed on the default or named Data Server populate the Cache, Table, and Columns drop-down lists: **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab". Populates the **Cache**, **Table**, and **Columns** drop-down lists with the names of the caches deployed on the default Data Server. If the Builder is not connected to a default Data Server, the names of the caches loaded locally by the cache data source are shown. **None** - Select to bypass data being redirected through the specified Data Server(s) for this attachment and instead attach directly to the data source. Populates the **Cache**, **Table**, and **Columns** drop-down lists with the names of the caches loaded locally by the cache data source. **Named Data Servers** - Select a **Named Data Server** that you configured in **Application Options>**"Data Server Tab". Populates the **Cache**, **Table**, and **Columns** drop-down lists with the names of the caches deployed on that Data Server. If the Builder requests the cache names from a Data Server and does not get a response within 10 seconds, the drop-down lists will be empty.

Filter Rows: Advanced

Select this option to filter rows by column(s) or values(s), time range, and to enable the "Extend with SQL" option.



Field Name	Description
Cache	Name of the cache to display. The drop-down menu will contain the names of all active caches. See the "Overview" section for information on adding a cache. If a selection is made in this dialog from the **Data Server** drop-down list, this field is populated by that selection.
Table	Select which type of table data to display: Current or History. If a selection is made in this dialog from the Data Server drop-down list, this field is populated by that selection. If you selected the **condenseRowsFlag** property for the cache, the **current_condensed** table and the **history_condensed** table are available in the list. If you also selected the **condenseRowsCombineHistoryFlag** property, the **history_combo** table is available in the list. When **history_combo** is selected, the **Update Once** check box will be replaced by three Update buttons: **Once**, **On Condense**, and **Always**. Default is **On Condense**.
Column(s)	Select which column(s) to display from the selected Table. If a selection is made in this dialog from the Data Server drop-down list, this field is populated by that selection.

Filter Rows

Off - For details, see "Filter Rows Off".

Basic - For details, see "Filter Rows: Basic"

Advanced

Filter Column - Name of the column to use as a filter. Multiple column names should be entered as a semicolon (;) delimited list (i.e. col1;col2;col 3). If your column name contains a space or a semicolon, then the entire name must be enclosed in single quotes.

Filter Value - Value that the **Filter Column** must equal. Multiple filter values should be entered as a nested list, where values for a given column are separated by commas within a semicolon (;) delimited list (i.e. val1,val2;val3,val4;val5,val6). If your filter value contains a space or a semicolon, then the entire value must be enclosed in single quotes.

When ***** is entered as a filter field value, data for all values in the specified filter column will be used to update the object property. When ***** is entered, only the literal comparative value will be used. These are only allowed for objects which display tabular data.

Time Range- Enter the time range of rows to be returned. Default time unit is seconds, but a suffix (e.g. m, h, d, w, M, q, y) can be specified to indicate minutes, hours, days, weeks, months, etc.

If **Time Range** is specified but neither a **Begin Time** or **End Time** is entered, then the most recent rows of data within the specified time range are returned.

If both **Begin Time** and **End Time** are specified, then **Time Range** is ignored.

Begin Time - Minimum timestamp of rows to be returned. You must specify as a date string or as a long timestamp.

If **Time Range** is specified but **End Time** is not, then data returned will span from *Begin Time to Begin Time + Time Range*.

End Time - Maximum timestamp of rows to be returned. You must specify as a date string or as a long timestamp.

If **Time Range** is specified but **Begin Time** is not, then data returned will span from *End Time - Time Range to End Time*.

Extend with SQL

Select to configure a cache to use a SQL query to retrieve data from an external database. This option is available when either the **Time Range** or the **Begin Time** fields are not blank, and the selected table name in the **Table** drop-down list is not Current. A substitution string, such as **$beginTime**, can be entered for these fields. When the **Extend with SQL** option is not enabled, the cache data source retrieves data from the in-memory cache only.

This option also requires the following:

- The **historyTableName** property for the cache must specify the name of the database table used for queries.
- The **maxNumberOfHistoryRows** property for the cache must have a value greater than 0.
- The the display (**.rtv**) file that defines the cache must be loaded by the RTView Historian as a Data Configuration file (so that the Historian stores the cache data in the database table specified by the cache **historyTableName** property).
- There must be an SQL database connection defined for the database specified in the cache **databaseName** property. Or, if the cache **databaseName** property is blank, there must be a database connection to the default RTVHISTORY database.

Maximum Rows	If the value specified is not zero, it is used to limit the row size of any SQL query that is performed to satisfy the time period in the attachment. When **Extend with SQL** is selected, a default value of **1000** is automatically entered but can be changed by the user. A value of zero or blank means there is no limit.
Update	**Once** - Update listeners only once when display is opened. **On Condense** - Update at the interval specified by the cache's **condenseRowsInterval** property, rather than whenever new raw data is applied to the cache. This is useful when, for example, the **traceValueTable** of a trendgraph is attached to the **history_combo** table. **Always** - Update whenever new raw data is applied to the cache. This is useful when, for example, the **valueTable** property of a table object is attached to the **history_combo** table. **Note:** The **On Condense** and **Always** options only apply if the selected Table is **history_combo**.
Data Server	Select to read data through your configured Data Server and not directly from the Cache data source. The names of caches deployed on the default or named Data Server populate the **Cache**, **Table**, and **Columns** drop-down lists: **Default** - Select the default Data Server you configured in **Application Options>**"Data Server Tab". Populates the **Cache**, **Table**, and **Columns** drop-down lists with the names of the caches deployed on the default Data Server. If the Builder is not connected to a default Data Server, the names of the caches loaded locally by the cache data source are shown. **None** - Select to bypass data being redirected through the specified Data Server(s) for this attachment and instead attach directly to the data source. Populates the **Cache**, **Table**, and **Columns** drop-down lists with the names of the caches loaded locally by the cache data source. **Named Data Servers** - Select a Named Data Server that you configured in **Application Options>**"Data Server Tab". Populates the **Cache**, **Table**, and **Columns** drop-down lists with the names of the caches deployed on that Data Server. If the Builder requests the cache names from a Data Server and does not get a response within 10 seconds, the drop-down lists will be empty.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid. Information entered into the dialog is validated against your active Caches.

The following describes the significance of the **Attach to Cache Data** validation colors:

Blue	Unknown	Entry does not match any known Cache.
White	Valid state	Entry is valid.
Red	Invalid state	Cache is valid, but Column(s) or Filter Column selected are not.

Substitutions

Substitutions allow you to build open-ended displays in which data attachments depend on values defined at the time the display is run. Generic names, such as **$cache1** and **$cache2**, are used instead of values for specific cache names. Later when the display is running, these generic values are defined by the actual names of specific cache names. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

RTViewDs

The RTViewDs cache contains tables with information about each table maintained by the cache data source.

Note: RTViewDs tables do not support SQL queries, row filters, custom query intervals or maximum row settings.

In the **Attach to Cache Data** dialog, enter **RTViewDs** in the **Cache** field and then select one of the following RTViewDs tables from the **Table** drop down menu:

- **CacheObjectProperties** - Contains all of the scalar property values for each tabular cache.
- **Tables** - Contains runtime information about the size of each cache table.
- **cacheDefs** - Contains cache configuration information.

RTViewDs tables contain the following columns, with one row for each table defined by the cache data source. The **CacheObjectProperties** table contains a row for each tabular cache, with the cache name in the first column followed by a column for each **obj_cache_table** scalar property listed in alphabetic order by property name. Tabular properties, like **initialTable** and **valueTable**, are not included. Non-tabular caches defined by **obj_cache_double** instances are not listed in the table.

RTViewDs Cache Table Name	Column Available	Description
CacheObjectProperties	One column for each cache property.	See "Overview" for descriptions of cache properties.

Tables	%Full	The amount of space, in percent, used in the cache table. In history tables, the **%Full** column shows the table row count divided by its **maxNumberOfHistoryRows** limit. Or, if the cache **historyTimeSpan** limit is not zero, the **%Full** column displays the table time span divided by its **historyTimeSpan** limit, whichever is larger. The value is converted to a percentage between 0 and 100. **Note:** If the history table size is limited by the **historyTimeSpan** property, the **%Full** value may never reach 100%. This is because when the old rows are removed from the table, the actual time span of the remaining rows may be less than the **historyTimeSpan** limit. For current tables, if the **maxNumberOfCurrentRows** property has a value greater than 0, the **%Full** column shows the table row count divided by **maxNumberOfCurrentRows**, converted to a percentage between 0 and 100. Otherwise, the value of the **%Full** column is **NaN**.
	Table	The name of the cache table.
	Rows	The number of rows in the table.
	Columns	The number of columns in the table.
	Memory	The estimated current size of the table, in bytes.

cacheDefs	Tag	The cache name and the table name:
		`<cachename>.<tablename>`
		Example:
		`sales.cacheDefs`

	Cache	The name of the cache.
	Table	The name of the table.
	AllColumns	The names of all columns in the table.
	IndexColumns	The names of all index columns in the table.
	DataColumns	The names of all non-index columns in the table.
	TimestampColumn	The name of the **timestamp** column in the table.
	AllColumnTypes	A list of column data types (for example, string, int, double, date, etc.) in the table.
	MaxRows	The value of the **maxNumberOfCurrentRows** property **Note:** This applies only to history tables.
	TimeSpan	The value of the **rowExpirationTime** property. **Note:** This applies only to history tables.
	Description	The value of the description property.
	File	The name of the .rtv file from which the cache definition object was loaded.

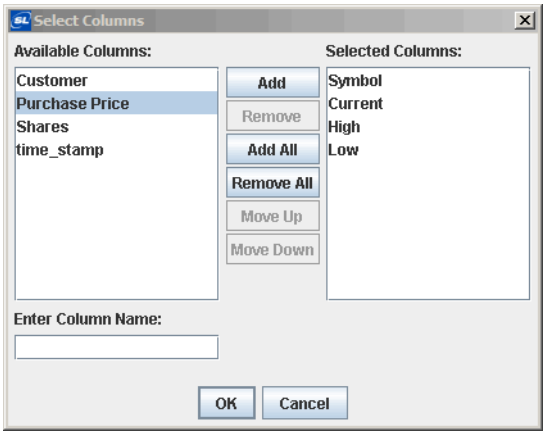
Select Table Columns

From the **Attach to Cache Data** dialog you can specify which table columns to display and in what order they will appear. In order to populate the listing of available columns, you must first select a valid Cache.

Click on the ellipses button in the **Column(s)** field (or right-click in the **Column(s)** field and choose **Select Columns**) to display the **Select Columns** dialog. The dialog should contain a list of Available Columns that you can add to your table.

To add a column, select an item from the **Available Columns** list and click **Add**. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click **Remove** to delete an item previously added to the **Selected Columns** list. You can control the order of columns in a table by arranging the items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

If no data is available for a table row within a selected column, the table cell will display one the following values: N/A, false, 0, or 0.0.



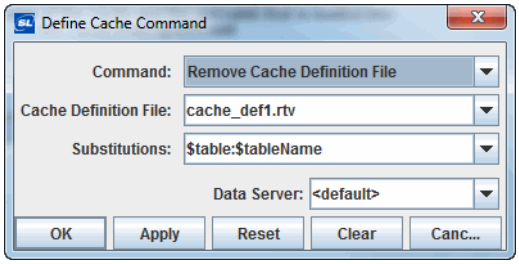
The following describes the **Attach to Cache Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied.

Define Cache Command

You can access the Define Cache Command dialog from the **Object Properties** window. This dialog is used to assign an action to an object's command property, giving you the ability to execute an action from within an RTView display. If you execute a cache command from a Thin Client with Direct Data Connection or any Served Data deployment, the command will execute on the server. If your command fails on a Direct Data deployment, an error dialog will popup with information on why the command failed. In other deployments, the error is only output to the console.

To open the **Define Cache Command** dialog, right-click on the appropriate command property in the **Object Properties** window and select **Define Command>Cache**. The **Define Cache Command** dialog provides a drop down menu with available commands. Argument fields vary based on the selected command type. See the "Define/Execute Command" section for information on how to execute a command.



Field Name	Description	
Command	Select a command from the drop-down menu.	
	Command Types	**Description**
	Add Cache Definition File	Loads the specified cache definition file. If the file was already loaded, all of the caches defined in that file are deleted and the file is loaded again.
	Remove Cache Definition File	Deletes all of the caches defined in the specified file.
Cache Definition File	Select a cache definition file from the drop-down menu. If an asterisk (*) is specified as the cache definition file for an Add Cache Definition command, the command unloads and then reloads all cache definition files. (The substitutions specified for the command, if any, are ignored). If an asterisk (*) is specified as the cache definition file for a **Remove Cache Definition** command, the command unloads all cache definition files. (The substitutions specified for the command, if any, are ignored).	

Substitutions	Select a substitution from the drop-down menu. Substitutions are optional and must use the following syntax:
	`$subname:subvalue $subname2:subvalue2`
	If a substitution value contains a single quote, it must be escaped using a / :
	`$filter:Plant=/'Dallas/'`
	If a substitution value contains a space, it must be enclosed in single quotes. Do not escape these single quotes:
	`$subname:subvalue $subname2:'sub value 2'`
	A substitution string cannot contain the following:
	: | . tab space , ; = < > ' " & / \ { } [ ] ( )
	Note: The name of the cache definition file and the substitutions, if any, to be applied when loading the file are specified when the command is defined.

Data Server	Select the Data Server on which the command is to be executed. **Default** - Execute the command in the Data Server you specified in **Application Options>**"Data Server Tab", or locally if no Data Server was specified. **None** - Always execute the command locally rather than in the default Data Server. **Named Data Servers** - Select a **Named Data Server** that you configured in **Application Options>Data Server**. **Multi-Server Command** - When multiple data servers are specified, the command will be executed on each data server in the list. To configure multiple data servers, enter a semicolon (;) delimited list containing two or more Named Data Servers (e.g. ds101;ds102). Each name specified must correspond with a Named Data Server that you configured in **Application Options>**"Data Server Tab". It is also possible to specify **__default** and **__none** (e.g. **__default**;**ds101**;**ds102**). **Note:** The values **__default** and **__none** begin with two underscore characters. Alternatively, a value of ***** can be entered to specify all data servers, including **__default** and **__none**.

Substitutions

The Substitutions feature allows you to build open-ended displays in which commands depend on values defined at the time the display is run. Generic names are used instead of arguments for specific commands. Later when the display is running, these generic values are defined by the actual arguments. In this way, a single display can be reused to issue a number of different commands. Substitutions are not applied to **Drill Down** or **Set Substitution** commands. For more information on creating displays using substitution values, see "Substitutions".

Special Values

$value	When an **actionCommand** is executed **$value** is replaced with the value from the control. This value may be used in any field in the **Define Cache Command** dialog. **Note:** This value may only be used for Action Commands. See "Commands" for more information.

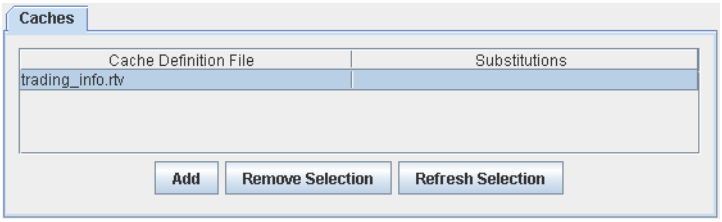
The following describes **Define Cache Command** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.
Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from assigned message subject (once **Apply** or **OK** is selected).
Cancel	Closes the dialog with last values applied.

Application Options -- Caches

Select **Tools>Options** and click **Caches** to add, remove, or refresh your Cache Definition files. Cache Definition files are added or removed after you click **OK**, **Apply**, or **Save**.

See the Caches section for information on creating Cache Definition (**.rtv**) files.



Field Name	Description
Add	Click to add a Cache Definition (**.rtv**) file. Once you have added an Cache Definition (**.rtv**) file and, optionally, entered a corresponding substitution, click Apply to execute. The Cache Definition (**.rtv**) file is added to the data source and data for these caches is collected. See the "Attach to Cache Data" section for more information on attaching to cache data variables. Caches that do not specify a name, as well as caches with a duplicate names, will not be created and an error will print to the console. **Note:** Once a Cache Definition file has been added to the data source, data is collected for that cache regardless of whether any currently open display (**.rtv**) files are showing that data.
Cache Definition File	List of (**.rtv**) files that contain cache definition settings.

Substitutions

Specify substitutions for this Cache Definition File. This allows the same file to be specified multiple times with different sets of substitutions.

Note: Each cache that is defined in the file should contain a substitution in the cacheName property, since cache names must be unique.

Substitutions are optional and must use the following syntax:

```
$subname:subvalue $subname2:subvalue2
```

If a substitution value contains a single quote, it must be escaped using a / :

```
$filter:Plant=/'Dallas/'
```

If a substitution value contains a space, it must be enclosed in single quotes. Do not escape these single quotes:

```
$subname:subvalue $subname2:'sub value 2'
```

A substitution string cannot contain the following:

: | . tab space , ; = < > ' " & / \ { } [ ] ( )

Remove Selection

Select a file from the list and click **Remove** to delete, then click **Apply** to execute. The Cache Definition (**.rtv**) file is removed from the data source and results for this data will no longer be available.

Refresh Selection

Select a file from the list and click Refresh Selection to reload cache definition settings. This allows you to edit a Cache Definition (**.rtv**) file and apply changes without restarting the Display Builder.

Note: The **Refresh Selection** option is enabled when you select a Cache Definition (**.rtv**) file that has been both added and applied.

CHAPTER 16 CMDB

This section contains the following:

Overview

This section contains the following:

What is a CMDB?

A Configuration Management Database (CMDB) is an IT service view database that enables organizations to establish a trusted source for IT service configurations. At its most basic implementation, a CMDB contains information about hardware assets and operating systems, as well as application components that reside on those systems.

A core concept in a CMDB is the notion of a CI (Configuration Item). To manage these different types of IT components, the CMDB contains the definitions of each CI for each uniquely defined entity that must be managed to support an IT service. Each CI in the database will describe what type of item it is and which attributes are necessary to describe the item.

Some typical CIs might be:

- Operating System (OS): Name ID and version
- Application Component: Name ID, version, patch level and component name
- Hardware Asset: Includes device name and ID, memory, disk size, and MAC and/or IP address

CIs also have parent/child information, so that you could, for example, indicate a particular Hardware Asset is hosting a particular Application Component. More advanced CMDB implementations would also have Service CIs, that may depend on various application components, and Business Unit CIs, that would rely on specific Services. This hierarchical map that describes from top-down the enterprise business, the services on which they rely, and the IT infrastructure that supports those services is often called a Service Dependency Model or Service Dependency Map.

A CMDB is often created via manual entry, but there are some vendors that provide a CMDB and discovery capabilities in one tool. When this is the case discovery functionality can not only find components, but also construct a portion of the service model.

How is a CMDB used?

In large organizations, the CMDB is vital to understanding the impact of change. If one server is, for example, brought down, reconfigured, or decommissioned, what services will it impact? A CMDB can also be used for capacity planning, disaster recovery planning or data consolidation projects. One other important usage, which is relevant to performance monitoring and Service Desk applications, is the ability to designate components that require performance and availability monitoring and to provide incident correlation by determining which service(s) or application(s) would be impacted by component performance problems.

Why would I want to use a CMDB with my RTView solution?

There are three reasons why a CMDB might be beneficial in RTView solutions:

- **Centralized connection repository**

  When you use RTView in an agentless manner to collect information from a variety of components, you must define connections to your sources of data. Perhaps this might be host and port information for a JMX connection or host and port of an IBM WebSphere® MQ server. Typically this information is stored in individual initialization (*.ini) files that an RTView Data Server could use to make connections and start gathering performance metrics. This can become a management problem when, for example, an RTView deployment has many Data Servers collecting data and the sources of data might either increase or decrease in number or change connection information. Rather than individually modify many files which contain this information, a centralized database can hold the information for all Data Servers and all sources of data.

- **Service and component centric view of data for display construction**

  It is a common best practice to gather performance metrics from common types of components and use an RTView Data Server to store that information in an RTView cache. For example, you might have a cache that stores metrics for all of your TIBCO EMS Servers. The source for each entry in the cache is then differentiated by one or more index columns that indicate which Server has provided the data.

  When constructing an RTView display with the Display Builder that uses this cache data, you are required to enter data server name and the cache name as well as indexes of interest if you want to perform filtered queries. With a CMDB definition, these Server components can be related in a hierarchical manner to the services or applications that depend on them. Display builders can then use names familiar to the business to navigate the data of interest and input to a particular display or alert rule, ultimately simplifying the construction of displays.

There is also the option of support for creation of composite objects that can be driven from CMDB definitions. Composite objects can be constructed from a number of RTView graphic objects and are often used to convey various pieces of information related to a particular data object. If constructed as described in the "Creating Composites" section, these composite objects can be driven by indexed data in a cache by simply navigating through the hierarchical definition of objects to find the object of interest.

- **Alert propagation (coming soon)**

  Though not currently available, in future releases users will be able to associate alert definitions with CIs described in the CMDB. With this information parent objects, such as a service or business unit, could inherit the alert state of their children facilitating the construction of summary displays that drill down to individual components which indicate they are in a warning or alarm state.

Configuration Requirements

CMDB functionality is dependent on a particular RTView deployment model and the following configuration requirements are necessary to take advantage of the features described above.

- **Cache definitions**

  Data from specific types of objects must be collected using their various data sources and stored with an index into RTView caches that reside in an RTView Data Server.

- **CMDB data definitions**

  A database or XML definition, using the schemas defined in the "CMDB Configuration Tables" section, that describes caches, data source connections and CI dependencies. This might be constructed manually or created automatically from various internal CMDB sources of information in the enterprise.

- **Database**

  If using a database as a repository, a supported database must be instanced, populated with the CMDB information and connection information provided to RTView.

- **Composite Objects**

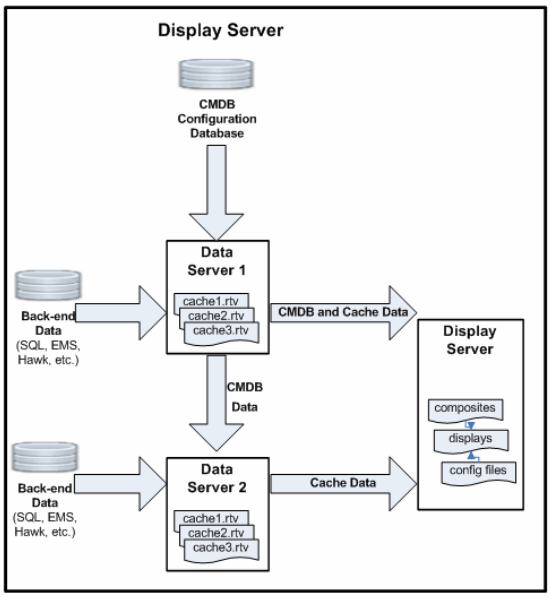
  If composite objects are being used, they must be constructed as described in the "Creating Composites"section and made accessible to the RTView Display Builder.

Deployment

The architecture of a solution using the CMDB is described in the following two diagrams.

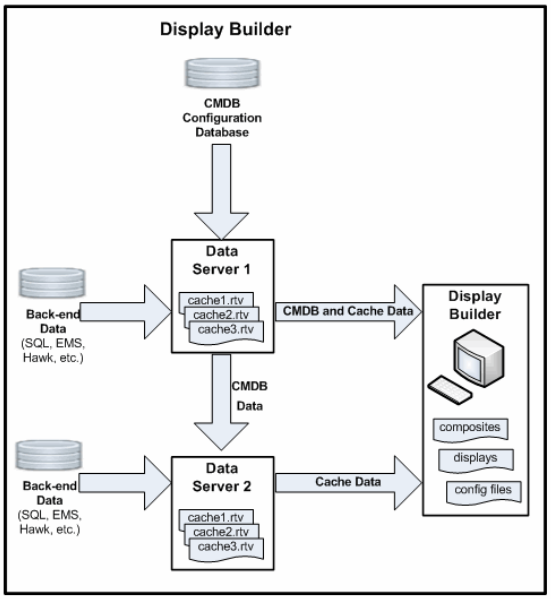
Centralized Connection Repository

Auto-load caches and make the necessary connections in one or more Data Servers based on the information in the CMDB Configuration Tables.



Service and Component Centric View of Data for Display Construction

Use the **Attach to CMDB Data** dialog to navigate through your cache data and attach to groups of cache data using the hierarchy defined in the CMDB Configuration Tables and limit access in the Display Builder to Metric Sources.



CMDB Configuration

This section contains the following:

Overview

The CMDB allows you to do the following:

- Auto-load caches and make the necessary connections in one or more Data Servers based on the information in the CMDB Configuration Tables. This allows you to have a centralized connection repository.
- Use the "Attach to CMDB Data" dialog to navigate through and attach to your cache data using the hierarchy defined in the CMDB Configuration Tables.
- Limit access in the Display Builder to the Metric Sources that are allowed for the user's login.

In order to use the CMDB, you must meet the following prerequisites:

- The CMDB tables can be accessed from a SQL database or an XML file. If you want to use a SQL database to store CMDB tables, you will need to have the database installed where it can be accessed by at least one RTView Data Server.
- All of the data you want to access via the CMDB must be stored in indexed caches in one or more Data Servers.
- If you will be using the "Attach to CMDB Data" dialog in the Display Builder, you must configure the Display Builder to use the RTView login (see "Role-based Security"). This information will be used in a configuration table that describes valid role names and the top level Containers or Metric Sources to which they have access.

The CMDB allows you to define two kinds of Configuration Items (CIs):

Metric Source

A Metric Source maps to the data for a single component in your system, this is done by mapping the Metric Source to a filter value in the index column on the associated caches. For example, this might map to an EMS Server, an application server or application component. Metric Source definitions are used both for the Data Server auto-load and for CMDB data attachments.

Container

A Container is used to add hierarchy and grouping to the Metric Sources. A Container can be the parent of one or more Metric Sources and a Metric Source can have more than one parent. For example, if you have an application named MyApp that uses five servers, you could create a Container named MyApp and set it as the parent for those five servers. If the MyApp application is part of an application group named ProductionApps, then you can create a Container named ProductionApps and set it as the parent for MyApp. As you add other applications in the ProductionApps group, you will set that Container as the parent for those applications as well. **Note:** Containers are not necessary if you select to auto-load caches, they are only used for CMDB data attachments.

"Creating CMDB Caches"

The first step in using the CMDB is to create caches to hold your data. These caches will be auto-loaded into a Data Server using the information in the **Metric Source** and **Metric Source Type** tables described below in the "CMDB Configuration Tables" section. Caches must be constructed as described in Creating Caches.

Data Server Connections

Each cache will be auto-loaded into a Data Server along with the required connections. In order for the client application (i.e. Display Builder, Display Viewer, Historian, Display Server) to access cache data, a named Data Server connection is required for each Data Server you will be using. In a later step we will deploy the caches to those named Data Servers, but first you just need to define them. In the Display Builder or Configuration Utility, go to the **Data Servers** tab and create a named Data Server Connection for each Data Server you will be using. The names you define here will be used in the Metric Source Table described in Configuration Tables.

"CMDB Configuration Tables"

In this step, you will create the CMDB Configuration Tables that define the caches and connections along with the Metric Sources and their relationships. These tables can either be defined in a SQL database or in an XML file.

"Application Options - CMDB"

In order for the Data Server to auto-load the caches and for the client applications (i.e. Display Builder, Display Viewer, Historian, Display Server) to service CMDB data attachments, you must configure the location of the Configuration Tables (either in an SQL database or in an XML file) along with the names of the Configuration Tables and a timeout.

"Deploying CMDB Caches"

In order to deploy the caches to the Data Servers you will need to copy the cache (.rtv) files along with any data source configuration (***OPTIONS.ini**) files to the directory (or directories) where you will be running the Data Server(s).

Persisting Cache Data

If your caches have been configured to persist data, you can run one Historian per Data Server. To do so, you must specify a Data Server for which the Historian will automatically load caches and redirect all data attachments. When you run the Historian, use the –**histDataServer**: command line option:

```
run_historian –histDataServer:ds1
```

The Historian will load all of the caches for Metric Sources in the Metric Source Table with the specified Data Server name. This will not make any data source connections, but it will set the specified Data Server as the default Data Server. This will cause all data attachments that are not configured for a named Data Server to be redirected to the specified Data Server.

Note: It is possible to set and save the name of a Historian Data Server from the **Application Options** dialog on the "RTView Server CMDB Settings Tab".

Building Displays

Once you have deployed your caches to the Data Server(s) and the caches have been auto-loaded by the Data Server(s), there are two ways you can build displays:

1. Use the "Attach to CMDB Data" dialog to attach your Metric Source (cache) data. This will allow you to view the hierarchy and grouping your defined in the Metric Source Table and Container Table. If you use this option, the **CMDBOPTIONS.ini** file you created in the Application Options section and the CMDB Configuration Tables must be available to the Display Builder and all Data Server clients (i.e. Display Viewer, Historian, Display Server).

   When you use the **Attach to CMDB Data** dialog for your data attachments, you can optionally create composite objects (see "Creating Composites") for one or more Metric Source Types to display data from the caches defined for them.

2. Attach to the auto-loaded caches directly using the **Attach to Cache Data** dialog.

Creating CMDB Caches

CMDB caches must be constructed as follows:

- All caches for a single Metric Source Type must use only one parameterized data source. At startup, the Data Server will load all of the caches for a Metric Source Type, make connections, and define substitutions for that cache based on the information in the "CMDB Configuration Tables". One connection information entry is allowed per Metric Source instance, so each cache for the associated "Metric Source Table" will use that connection. Caches can use hard-coded connections for other data sources if desired.
- Each cache must specify at least one column in the **indexColumnNames** property. The Index value listed in the Metric Source Table must exactly match the value in the index column, specified in **indexColumnNames**, of all associated caches to uniquely identify that Metric Source instance. If more than one index column is required to uniquely identify the Metric Source instance, enter a ; (semi-colon) delimited list of values for the Index value in the Metric Source Table. In this case, the order of the values must exactly match the order of **indexColumnNames** in the cache. However, note that more index columns may be used in the cache than are included in the Index value of the Metric Source Table. For example, suppose a cache has three index columns, but only the first column is needed to uniquely identify the Metric Source instance. In this case, only one value needs to be entered for the Index value in the Metric Source Table.
- Caches should be constructed so that they don't reference individual component instances. For example, a cache attached to EMS server data should not use any specific server URLs. Instead, the value ***** should be used for the server URL and the data should be filtered using the **$indexFilter** substitution. When the cache is auto-loaded by the Data Server, the **$indexFilter** substitution will be set to a comma delimited list of index values for all Metric Sources of that type in that Data Server (from the Index column in

the Metric Source Table). For data sources that support connection groups, such as JMX or TIBCO Hawk, the **$connGroup** substitution can be used for the Connection name (or Agent name) if the Index values are JMX Connection names or Hawk Agent names. When the cache is auto-loaded by the Data Server, a connection group is created using the list of index values for all Metric Sources of that type for a single applications, and the **$connGroup** substitution is set to name of that connection group. **Note:** If the **Manual** field in the **Metric Source Type** table is set to true, the caches are not auto-loaded and therefore the **$indexFilter** and **$connGroup** substitutions are not defined. For data sources that do not support the value ***** or groups for the connection in their data attachments, use the **valueTableCount** property to set the number of input tables and then make a separate data attachment for each connection.

- All caches must be Table Caches. Double Caches are not supported.
- The same cache (**.rtv**) file cannot be used for multiple Metric Source Types.
- Each cache name (**cacheName** property) must be unique across all caches in the CMDB. **Note:** The **cacheName** property value cannot contain substitutions.
- The same cache cannot be loaded multiple times each with different sets of substitutions. It will only be instanced once per Metric Source Type per Data Server. Therefore, application level substitutions can be used, but cache specific substitutions cannot.
- If you are going to persist your cache data using the Historian and you are going to load the caches for a single Metric Source Type on multiple Data Servers, you must scope the persisted data. Since the same caches will be loaded by multiple Data Servers, any persisted cache data must also be scoped by a Data Server. When you configure the cache persistence using the **historyTableName** and **currentTableName** properties, you must store the cache data from each Data Server in a different database table. When caches are auto-loaded by the Data Servers, the **$ds** substitution will be set to the name of the Data Server. You can use this substitution in the **historyTableName** and **currentTableName** properties to specify a different table name at runtime for caches loaded into different Data Servers (e.g. **$ds1.myTableName**). This assumes that all tables are in the same database. Alternately, you could use the **$ds** substitution in the **databaseName** property to specify different databases for each Data Server.

CMDB Configuration Tables

To configure the CMDB data source you will need to create the following four tables. These tables describe Metric Sources and their hierarchical relationships, provide associated cache information, and define user accessibility:

- "Metric Source Table"
- "Container Table"
- "Metric Source Type Table"
- "Permission Table"
- "Relationship Table"

These tables should be made available either in a SQL database or an XML source file. To add an SQL database connection, select **Tools>Options>SQL** (see "Application Options - SQL" for more information) in the Display Builder.

Note: If you select to use an XML source, the file must use the format described under **Data Sources>XML>Creating XML Sources** (see "Creating XML Sources" for more information).

In the "CMDB Tables Tab" of the **Applications Options** dialog you will provide the information necessary to allow RTView applications to connect to these tables at startup.

Metric Source Table

A Metric Source represents a single item within the system being monitored. A Metric Source maps to a subset of the rows in the cache(s) defined for the corresponding Metric Source Type. The Metric Source Table describes all of the Metric Sources in the system and allows you to create relationships between Metric Sources by specifying one or more parents for each Metric Source. The Metric Source Table must contain the following columns in the following order:

COLUMN	TYPE	DESCRIPTION
Name	String	Unique name to identify the Metric Source. This name must be unique among both the Containers and the Metric Sources.
Type	String	Name of one of the Metric Source Types defined in the "Metric Source Type Table".
Index	String	Value(s) the first index column(s) in the cache(s) must equal to identify this Metric Source. If the cache has a single index column, enter a single value. If the cache requires multiple index columns to uniquely identify this Metric Source, enter a semi-colon delimited list of values here. **Note:** The values must be entered in the same order as the index columns in the cache(s).
Connection Information	String	Description of connection to make for this Metric Source in the cache file(s). The content will depend on the data source used in the cache files for this Metric Source. See "CMDB - Metric Source Table - Connection Information" for connection string syntax for each data source.
Data Server	String	Name of the Data Server that will host the cache file(s) for this Metric Source. See "Data Server Tab" for more information. **Note:** All clients of the Data Server (i.e. Display Builder, Display Viewer, Historian, and Display Server) must be configured separately with a named Data Server.
Active	Boolean	If set to 0 (FALSE), do not load this Metric Source and ignore any references to it.

Sample Metric Source Table:

NAME	TYPE	INDEX	CONNECTION_INFO	DATA_SERVER	ACTIVE
dev_1_ei	EMS_INFO	tcp://SLXP1:7222	jmsadm_server name=EMS-SERVER-SLXP1 url=tcp://SLXP1:7222	ds2	TRUE
dev_1_si	SYSTEM_INFO	agentxp1;_Total		ds1	TRUE
dev_3_ei	EMS_INFO	tcp://SLXP3:7222	jmsadm_server name=EMS-SERVER-SLXP3 url=tcp://SLXP3:7222	ds2	TRUE

dev_3_si	SYSTEM_INFO	SLXP3;_Total		ds2	TRUE
dev_cb_ei	EMS_INFO	tcp://slpro31:7222	jmsadm_server name=EMS-SERVER-slpro31 url=tcp://slpro31:7222	ds1	TRUE
dev_cb_ji	JVM_INFO	conn_cb	jmxconn conn_cb slpro31 9999 URL:- - - false	ds2	TRUE
dev_cb_si	SYSTEM_INFO	slpro31;_Total		ds1	TRUE
dev_cb_ti	TOMCAT_INFO	conn_cb	jmxconn conn_cb slpro31 9999 URL:- - - false	ds1	TRUE
dev_s3_ei	EMS_INFO	tcp://slserver3:7222	jmsadm_server name=EMS-SERVER-slserver3 url=tcp://slserver3:7222	ds2	TRUE
dev_s3_ji	JVM_INFO	conn_s3	jmxconn conn_s3 slserver3 9999 URL:- - - 'false'	ds1	TRUE
dev_s3_si	SYSTEM_INFO	agentserver3(testConn);_Total	hawkconsole testConn rvd QADomain 7474 ; tcp:7474	ds2	TRUE
dev_s3_ti	TOMCAT_INFO	conn_s3		ds1	TRUE
dev_s5_si	SYSTEM_INFO	slserver5;_Total		ds1	TRUE
doc_1_si	SYSTEM_INFO	agent30p;_Total		ds1	TRUE
qa_rtv_ei	EMS_INFO	tcp://slqa9:7222	jmsadm_server name=EMS-SERVER-slqa9 url=tcp://slqa9:7222	ds1	TRUE
qa_rtv_si	SYSTEM_INFO	slqa9;_Total		ds1	TRUE

CMDB - Metric Source Table - Connection Information

The value of the Connection Information field in the Metric Source table varies according to which RTView data source is used for the associated cache files. To find out the data source being used for a Metric Source look at the value of the **Type** field in the "Metric Source Table", then go find that value in the **Type** column of the Metric Source Type table. The value in the **Dskey** field for that row indicates which data source the cache files will be using.

In some cases, the connection will be only be used by a single item in the Metric Source table. An example of this would be a Metric Source that uses the EMS Administration data source and uses the EMS Server URL for the Index value. In this case, each Metric Source contains information on how to connect to that EMS Server. In other cases, a single connection might be used for multiple items in the Metric Source table. An example of this would be a group of Metric Sources that use the SQL data source and use a value from the database table as the Index value. In this case, the connection need only be defined for one of the Metric Source instances as long as all of the Metric Sources are running on the same Data Server. However, if Metric Sources of this type are being deployed to multiple Data Servers, at least one Metric Source instance for each Data Server must contain the connection information.

Select from the data sources listed below to reference the connection string syntax. If you do not want to create the connection string by hand, you can define the connection in the **Application Options** dialog for that data source (either in the Display Builder or Configuration Utility), then save the configuration file. Open the configuration file in a text editor and copy the connection string from the configuration file into the Connection field in your database.

Note: Data sources that use a different connection token than those listed in the configuration files have been noted below.

DATA SOURCE DSKEY	DATA SOURCE DSKEY
IBMMQ	ibmmq (See "IBMMQ (ds key = ibmmq)")
RTVPipe	pipe (See "RTVPipe (ds key = pipe)")
JMS	jms (See "JMS (ds key = jms)")
SNMP	snmp (See "SNMP (ds key = snmp)")
JMX	jmx (See "JMX (ds key = jmx)")
SQL	sql (See "SQL (ds key = sql)")
Log4j	log4j (See "Log4j (ds key = log4j)")
TIBCO EMS Administration	jmsadm (See "TIBCO EMS Administration (ds key = jmsadm)")
RTVAgent	rtvagent (See "RTVAgent (ds key = rtvagent)")
TIBCO Hawk	hawk (See "TIBCO Hawk (ds key = hawk)")

IBMMQ (ds key = ibmmq)

An example connection string:

```
immed_conn __name=vmrh5-1 host=192.168.200.164 port=1414 channel=SYSTEM.DEF.SVRCONN
modelQueueName=SYSTEM.DEFAULT.MODEL.QUEUE waitInterval=10 maxRetries=5 retryInterval=30
```

The string must start with the connection token **immed_conn**. Note that this token is different than the connection token used in the IBMMQOPTIONS.ini file which uses ibmmqconn instead.

The **immed_conn** token is followed by a sequence of name=value pairs, where name is the name of the field, and value is the value.

The fields are defined as follows:

Field Name	Description
__name	The name of the connection – used to uniquely identify the connection Example: **vmrh5-1**
host	The name or IP address of the computer to connect to. (The host the queue manager of interest is running on) Example: **192.168.200.164**
port	The port number of the connection. This is the listener port address for queue manager. The default port used is 1414 Example: **1414**
channel	Client Channel to use for this connection (use SYSTEM.DEF.SVRCONN) Example: **SYSTEM.DEF.SVRCONN**
modelQueueName	The named model queue of the connection Example: **SYSTEM.DEFAULT.MODEL.QUEUE**
waitInterval	The wait interval of the connection (in seconds). This is the amount of time the connection will wait for a response before timing out (it may be necessary to have a larger values on "slow" machines or where a large amount of data is returned Example: **30**
maxRetries	The Maximum number of subsequent connection retry attempts, when a connection attempt fails. Set **maxRetries** to 0 if you only want one attempt at connection (i.e. 0 retries) Example: **5**
retryInterval	Minimum interval between connection retry attempts (in seconds). The minimum elapsed time between connection attempts, when a connection attempt fails. Example: **30**

JMS (ds key = jms)

An example connection string:

```
jmsconn sampleconn com.tibco.tibjms.TibjmsTopicConnectionFactory tcp://myserver:7222
user1 pass1 -
```

The string must start with the **jmsconn** connection token.

The **jmsconn** token is followed by the following values in the following order. To leave a value blank, use a dash. Enclose any values that contain spaces in single quotes.

Field Name	Description
Connection Name	Name to use when referencing this JMS connection in your data attachments. Example: **sampleconn**
Factory Class Name	The fully qualified name of the topic connection factory class to use when creating this connection. The path to this class must be included in the **RTV_USERPATH** environment variable. Example: **com.tibco.tibjms.TibjmsTopicConnectionFactory**
Server URL	The complete URL for your JMS message server. Example: **tcp://myserver:7222**
User Name	User name to use when creating this connection Example: **user1**
Password	Password to use when creating this connection. This can be entered as plain text or as encrypted text from the **encode_string** option. Example: **pass1**
Client ID	The client identifier to set on this connection. Example: **slclient**
Host Name	For IBM WebSphere MQ only. The name of the host to use for this connection. Example: **myHost**
Port	For IBM WebSphere MQ only. The name of the port to use for this connection. Example: **9999**
Queue Manager	For IBM WebSphere MQ only. The name of the queue manager to use for this connection. Example: **myQueueManager**

JMX (ds key = jmx)

An example connection string:

```
jmxconn MyConnection localhost 9998 user1 pass1
```

The string must start with the **jmxconn** connection token.

The **jmxconn** token is followed by the following values in the following order. To leave a value blank, use a dash except where noted below. Enclose any values that contain spaces in single quotes.

Field Name	Description
Connection Name	Name to use when referencing this JMX connection in your data attachments. Example: **MyConnection**
Host or IP	Host name or IP address of the MBean server. Example: **localhost**
Port	Port that your JMX instrumented application is using to expose MBean methods. Example: **9998**
URL	URL that should be used to create this connection. Enter local to make a local connection to an in-process MBean server. If this field is blank, don't enter a dash for it, just don't enter a value for it. Example: **local**
User Name	User name to use when creating this connection Example: **user1**
Password	Password to use when creating this connection. This can be entered as plain text or as encrypted text from the **encode_string** option. See "Encrypt text using the encode_string option" for more information. Example: **pass1**
Use Client Credentials	If true, the **User Name** and **Password** from the RTView login will be used instead of the **User Name** and **Password** entered in this connection string. Example: **false**
Additional Properties	Add any properties necessary to initiate connections with specific MBean servers. These should be name value pair separated by spaces. If no properties are needed, leave this field blank, do not add a dash. Example: **myPropertyName myPropertyValue**

Log4j (ds key = log4j)

An example connection string:

```
immed_conn __name=sim1_vmrh5-7 host=192.168.200.174 port=4560
```

The string must start with the **immed_conn** connection token. Note that this token is different than the connection token used in the **LOG4JOPTIONS.ini** file which uses log4jconn instead.

The **immed_conn** token is followed by a sequence of name=value pairs, where name is the name of the field, and value is the value.

The fields are defined as follows:

Field Name	Description
__name	The name of the connection – used to uniquely identify the connection Example: **vmrh5-7**
host	The name or IP address of the computer to connect to. (The host the log4j **SocketHubAppender** of interest is running on) Example: **192.168.200.174**
port	The port number of the connection. This is the port address used by the log4j **SocketHubAppender**. The default port is **4560**. Example: **4560**

RTVPipe (ds key = pipe)

This data source does not use connections. It works by executing the command specified in the data attachment and parsing the return. It does not support connections.

RTVAgent (ds key = rtvagent)

This data source does not use connections. It works by configuring the remote Agent applications to send data to a Data Server, not by configuring the Data Server to query the remote Agents.

SNMP (ds key = snmp)

An example connection string:

```
immed_conn __name=local host=127.0.0.1 dataport=161 trapport=162 version=v2c
community=public
```

The string must start with the **immed_conn** connection token. Note that this token is different than the connection token used in the **SNMPOPTIONS.ini** file, which uses **snmpconn** instead.

The **immed_conn** token is followed by a sequence of name=value pairs, where name is the name of the field, and value is the value.

The fields are defined as follows:

Field Name	Description
__name	The name of the connection – used to uniquely identify the connection. Example: **vmrh5-7**
host	The name or IP address of the computer to connect to. Example: **192.168.200.174**
dataport	The connection will poll for data on this port. Example: **161**

trapport	The connection will listen for trap messages on this port. Example: **161**
version	The version of SNMP. This can be v1 or v2c. Example: **v1**
community	The SNMPv2 Community Name string. This is only used if the version is v2c. Example: **public**

SQL (ds key = sql)

An example connection string:

```
sqldb APPDATA user1 pass1 jdbc:hsqldb:hsql://localhost:3380/appdata
org.hsqldb.jdbcDriver - false true
```

The string must start with the **sqldb** connection token. The **sqldb** token is followed by the following values in the following order. To leave a value blank, use a dash. Enclose any values that contain spaces in single quotes.

Field Name	Description
Database Name	The name to use when referencing this database connection in your data attachments Example: **APPDATA**
User Name	The user name to use when connecting to this database. Example: **user1**
Password	The password to use when connecting to this database. The may be entered in plain text or encrypted text from the **encode_string** option. See "Encrypt text using the encode_string option" for more information. Example: **pass1**
URL	The url to use when connecting to this database. Example: **jdbc:hsqldb:hsql://localhost:3380/appdata**
Driver	The fully qualified name of the JDBC driver class to use when connecting to this database. The path to this driver must be included in the **RTV_USERPATH** environment variable. Example: **org.hsqldb.jdbcDriver**
Table Types	Comma delimited list of tables types to retrieve when querying the database for available tables. Refer to your database manual for a list of valid table types. Example: **TABLE,VIEW**

Use Client Credentials	If true, the **User Name** and **Password** from the RTView login will be used instead of the **User Name** and **Password** entered in this connection string. Example: **true**
Run Queries	Concurrently If true, each query on the connection is run on its own execution thread. Example: **false**

TIBCO EMS Administration (ds key = jmsadm)

An example connection string:

```
jmsadm_server=name='my server' url=tcp://myhost:7222 agent=myagent user=myusername
pass=mypassword
```

The string must start with the **jmsadm_server** connection token.

The token **jmsadm_server** token is followed by a sequence of name=value pairs, where name is the name of the field, and value is the value. Any values that contain spaces should be enclosed in single quotes.

The fields are defined as follows:

Field Name	Description
name	The name of the EMS Server. Example: **EMS-SERVER-myserver2**
url	The complete URL for your EMS Server Example: **tcp://myserver2:7222**
agent	Name of the agent containing the microagent method to subscribe. Example: **myagent**
user	The user name to user name to use when connecting to this EMS Server Example: **user1**
pass	The password to use when connecting to this EMS Server. This may be entered in plain text or encrypted text generated from the **encode_string** option. See "Encrypt text using the encode_string option" for more information. Example: password1

TIBCO Hawk (ds key = hawk)

The string must start with the connection token **hawkconsole**.

The token **hawkconsole** token is followed by a name identifying this hawk console.

The name token is followed by the transport type. Valid values are **rvd**, **rva**, and **ems**.

If the transport type is **rvd**, it is followed by 4 values separated by spaces: domainName service network daemon. For example:

```
hawkconsole testConn rvd TestDomain 7474 ; tcp:7474
```

If the transport type is rva, it is followed by 2 values separated by spaces: host port. For example:

```
hawkconsole testConn2 rva MyHost 5555
```

If the transport type is ems, it is followed by 3 values separated by spaces: serverUrl username password. For example

```
hawkconsole testConn3 ems tcp://myserver:7222 user1 pass1
```

If any values in the above strings contain spaces, enclose them in single quotes. If any values are blank, use a -. For the ems transport type, the password may be entered in plain text or encrypted text from the **encode_string** option. See "Encrypt text using the encode_string option" for more information.

Encrypt text using the encode_string option

If you need to provide an encrypted password (rather than expose server password names in a clear text file, use the **encode_string** command line option with the following syntax:

```
encode_string type mypassword
```

where **type** is the key for the data source and **mypassword** is your plain text password.

Note: The type argument is only required when you encrypt a string for a data source.

For example, for the SQL data source you would enter the following in an initialized command window (see "Initializing a Command Prompt or Terminal Window" for more information):

```
encode_string sql mypassword
```

and you will receive an encrypted password:

```
encrypted value: 013430135501346013310134901353013450134801334
```

Copy the encrypted value, paste it into the password field and click **Save** to save this value to the initialization (***.ini**) file. Or, if necessary, manually edit the (***.ini**) file to include the encrypted value.

Note: If you need to manually edit a configuration (***.ini**) file, contact SL Technical Support at support@sl.com for information about supported syntax.

Container Table

Containers allow you to create a hierarchy of Metric Sources. A Container does not map to cache data like the Metric Sources, it just allows you to group and create hierarchies of Metric Sources. A Container can be specified as a parent to any number of Metric Sources or Containers. Circular references are not allowed.

Note: This table can be left empty if you selected to auto-load caches and will not use the Attach to CMDB Data dialog to attach to your cache (Metric Source) data.

The Container Table defines all Containers in the system and describes their relationships; it must contain the following columns in the following order:

COLUMN	TYPE	DESCRIPTION
Name	String	Unique name to identify the Container. This name must be unique among both the Containers and the Metric Sources.
Type	String	A label to describe the type of Container. This value will allow you to group Containers of the same type at the same level of the hierarchy. **Note:** Metric Source Types cannot be used in this field.

Sample Container Table:

NAME	TYPE
App1	Application
App2	Application
dev_3_ems	EMSSERVER
dev_server_3_web	WEBSERVER
production_group1	PRODUCTION
production_group2	PRODUCTION
qa_group1	QA
dev_1_ems	EMSSERVER
dev_cb_web	WEBSERVER
dev_server_3_fs	FILESERVER
dev_server_5_fs	FILESERVER
qa_rtv_ems	EMSSERVER
dev_1_fs	FILESERVER
dev_cb_ems	EMSSERVER
dev_server_3_ems	EMSSERVER
doc_1_fs	FILESERVER
dev_cb_fs	FILESERVER

Metric Source Type Table

A Metric Source Type maps your Metric Sources to their corresponding caches. It allows you to define one or more caches that contain the data for all Metric Sources of that type. The Metric Source Type Table describes all of the Metric Source Types and their associated cache file information; it must contain the following columns in the following order:

COLUMN	TYPE	DESCRIPTION
Type	String	Unique value to identify a Metric Source type. The values in this column must be used as the values for the Metric Source Type in the "Metric Source Table".
CacheFileNames	String	Comma delimited list of one or more cache (.rtv) file names. **Note:** The same cache file cannot be used for multiple Metric Source Types.
DsKey	String	Data source key used in your cache files. Connection information from the Metric Source Table will be send to this data source. See "CMDB - Metric Source Table - Connection Information" for connection string syntax for each data source.
Manual	Boolean	If set to **1** (TRUE), caches of this type and connections for this Metric Source Type will not be auto-loaded into the Data Server or Historian.

Sample Metric Source Type Table:

TYPE	CACHE_FILE_NAMES	DSKEY	MANUAL
EMS_INFO	ems_cache.rtv,ems_queue_cache.rtv	jmsadm	FALSE
JVM_INFO	jvm_cache.rtv	jmx	FALSE
SYSTEM_INFO	fs_cache.rtv	hawk	FALSE
TOMCAT_INFO	tomcat_cache.rtv	jmx	FALSE

Permission Table

The Permission Table describes all valid Roles and the top level Containers or Metric Sources to which they have access. This information is used by the Display Builder to limit access to available Metric Sources. When a user runs the Display Builder, they must login using one of the Roles defined in this table in order to access CMDB data.

Note: This table can be left empty if you selected to auto-load caches and will not use the Attach to CMDB Data dialog to attach to your cache (Metric Source) data.

The Permission Table must contain the following columns in the following order:

COLUMN	TYPE	DESCRIPTION
Role	String	Specify a role value from the RTView login. This value must match exactly. If the role allows access to multiple applications, each should be listed in a separate row.
CI_Name	String	Name of a CI listed in the "Container Table" or "Metric Source Table". This CI will be added as a top level CI in the Attach to CMDB Data dialog for this user, and all data for that CI and all of its children will be available. Enter a value of * to add all CIs from the Container and Metric Source tables that have no Parents as top level CIs.

Sample Permission Table:

ROLE	CI_NAME
admin	*
app1user	App1
app2user	App2
poweruser	App1
poweruser	App2

Relationship Table

The Relationship Table contains all relationship information for Containers and Metric Sources.

The Relationship Table must contain the following columns, with the specified names and types, in the following order:

COLUMN	TYPE	DESCRIPTION
ID1	String	Name of the parent container.
ID2	String	Name of the child container or metric source.
TYPE	String	Relationship type. Currently, only PC (parent-child) is supported.

Sample Relationship Table:

ID1	ID2	TYPE
production_group1	App1	PC
production_group1	App2	PC
App1	dev_3_ems	PC

App1	dev_1_fs	PC
App2	dev_3_ems	PC
App2	dev_cb_web	PC

Application Options - CMDB

Select **Tools>Options** in the Display Builder to access the **Application Options** dialog.

Options specified in CMDB tabs are saved in an initialization (**.ini**) file. This file, **CMDBOPTIONS.ini**, along with general options saved in **OPTIONS.ini**, must be copied to all Display Builder, Display Viewer, Historian, Data Server, and Display Server deployments. Any related data source specific options (**.ini**) files must be copied to all of the Data Server and Power User Display Builder deployments.

On startup, these initialization files are read by the Display Builder, Display Viewer, Display Server, Data Server, and Historian to set initial values. If no directory has been specified for your initialization files and **CMDBOPTIONS.ini** is not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "RTV_JAVAOPTS" for more information.

Note: Options specified using command line arguments (see "Command Line Options - CMDB" for more information) will override values set in initialization files.

There are two **Application Options** tabs for CMDB:

- "CMDB Tables Tab"
- "RTView Server CMDB Settings Tab"

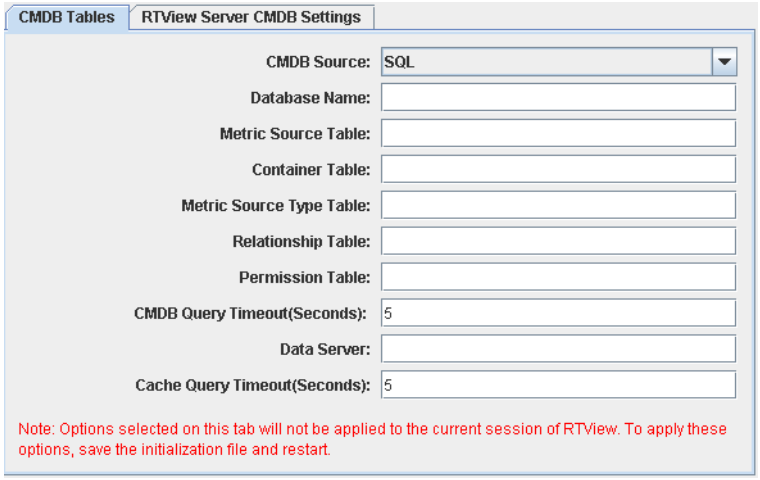
CMDB Tables Tab

In the **CMDB Tables** tab of the **Applications Options** dialog, you will provide the information necessary to allow RTView to connect to the required CMDB configuration tables (see "CMDB Configuration Tables" for more information), which can be made available via a SQL database or an XML source file. To add a SQL database connection, select **Tools>Options>**"SQL Tab" in the Display Builder.

Note: If you select to use an XML source, that file must use the format described under **Data Sources>XML>Creating XML Sources** (see "Creating XML Sources" for more information).

Click **Save** to save these options to the **CMDBOPTIONS.ini** file which will be read at startup.

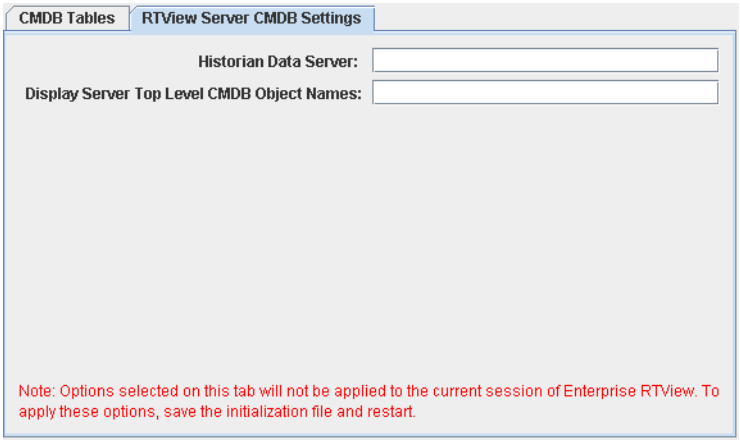
Note: These options, along with any named Data Server definitions and the SQL Database or XML Source definition, must be made available to all Display Builder, Display Viewer, Data Server, Historian, and Display Server instances that want to use this CMDB data.



Field Name	Description
CMDB Source	XML or SQL
Database Name / XML Source Name	Name of a SQL database connection or an XML source that is already defined. See "Creating XML Sources" for more information.
Metric Source Table	Name of the Metric Source Table in the database or XML file.
Container Table	Name of the Container Table in the database or XML file.
Metric Source Type Table	Name of the Metric Source Type Table in the database or XML file.
Relationship Table	Name of the Relationship Table in the database or XML file.
Permission Table	Name of the Permission Table in the database or XML file.
CMDB Query Time Out (Seconds)	Enter the time (in seconds) to wait for the data attachment for each CMDB table to return. Default is 5 seconds. **Note:** The data attachment may return faster than the specified time out. Data attachments to all CMDB tables are made once at startup in the main thread, so this timeout will impact startup time if the CMDB data is not available.
Data Server	Optionally specify the name of a defined Data Server to route data attachments for the CMDB tables to that Data Server. Select the "Data Server Tab" in the **Application Options** dialog to configure your Data Server(s). If left blank and a default Data Server is specified on the **Data Server** tab, the CMDB table data attachments will be routed to that Data Server. If left blank and no default Data Server is specified on the Data Server tab, the data source will connect directly to the CMDB tables which will result in the Display Builder, Display Viewer, Display Server and Data Servers all making separate connections to the CMDB tables.
Cache Query Time Out (Seconds)	Enter the time (in seconds) to wait for the data attachment for each cache table to return. Default is 5 seconds. **Note:** The data attachment may return faster than the specified time out. Data attachments to cache tables are made at startup in the main thread for all applications except the Data Servers, so this timeout may impact startup time if the cache data is not available.

RTView Server CMDB Settings Tab

This **CMDB** tab of the **Applications Options** dialog allows you to limit the number of CMDB objects loaded by the Display Server and to specify a Data Server for which the Historian will automatically load caches and redirect all data attachments.



Field Name	Description
Historian Data Server	Enter the name of a Data Server. The Historian will load all of the caches for Metric Sources in the Metric Source table with that Data Server. This will not make any data source connections, but it will set the specified Historian Data Server as the default Data Server. This will cause all data attachments that are not configured for a named Data Server to be redirected to the Historian Data Server. **Note:** To utilize this option your caches must have been configured for persistence.
Display Server Top Level CMDB Object Names	Specify the name(s) of one or more top level CMDB objects. By default, the Display Server will load in all Containers and Metric Sources without parents along with all of their children.

Deploying CMDB Caches

In order for the Data Server to auto-load the caches, the cache files and their corresponding data source configuration files must be copied to the Data Servers you specified for the corresponding Metric Sources. Each Data Server must also have access to the **CMDBOPTIONS.ini** file you created in the Application Options section above and it must have access to the SQL database or XML file containing your CMDB Configuration Tables either directly or via Data Server.

To deploy the caches to the Data Servers, you must copy the cache (**.rtv**) files along with any data source configuration (***OPTIONS.ini**) files* to the directory (or directories) where you will be running the Data Server(s).

Note: The system(s) where you run the Data Server(s) must meet the System Requirements and Setup for the data sources used in the cache files for that Data Server.

Run each Data Server using the **–processName:** command line option to specify a process. This **processName** must match the name in the Data Server column of the Metric Source Table.

```
run_dataserver –processName:ds1
```

The Data Server will query the Metric Source Table for all rows where the Data Server column value equals the **processName** and load caches and create connections for all of those Metric Sources.

*****It is important to note that while connection information for Metric Sources is contained in the Metric Source Table, other data source specific options (contained in data source specific configuration ***OPTIONS.ini** files) are not. Therefore, directory (or directories) where you run the Data Server must also contain configuration (**.ini**) files with required options for data source(s) that are being used for a Metric Source. For example, in order for TIBCO Hawk caches to work correctly, the **Disable Data Caching** and **Include Index Columns** options should be enabled. For TIBCO EMS Administration, you might want to turn off auto-discover options since all of the EMS-Server connection information is contained in the Metric Source Table. In these cases, the files **HAWKOPTIONS.ini** and **JMSADMOPTIONS.ini** must be included to successfully run caches for those data sources.

Steps for Deploying Caches

This process takes you through running the cache files locally, then on the Data Server without the CMDB to confirm that the caches are properly constructed.

1. Construct your cache file as detailed in the "Creating CMDB Caches" section.

2. Confirm that the cache file running locally in the Display Builder works as expected and contains all of the data you expect to see. This will require that all of your data source specific configuration (**.ini**) files are setup correctly.

3. Deploy your cache file to the named Data Server that will be hosting it. Move your data source specific configuration (***OPTIONS.ini**) files and **CACHEOPTIONS.ini** to the Data Server along with the cache files. Run the Data Server without using the **–processName:** command line option.

4. From the Display Builder, make data attachments to the cache using the Named Data Server connection in your data attachment. Confirm that these data attachments return the expected behavior.

5. Now that you've confirmed that the cache file is properly constructed and that the data source options are all correct, add it to the Configuration Tables in your CMDB Configuration Database. When adding the connection information to your Metric Source definitions, copy the corresponding line from your data source options (**.ini**) file for that connection and paste it into the database. This will avoid problems with typos.

   There are three cases where you will need to slightly modify the connection string before adding it to the database:

   - For the IBMMQ data source: change **ibmmqconn** to **immed_conn**.
   - For the LOG4J data source: change **log4jconn** to **immed_conn**.
   - For the SNMP data source: change **snmpconn** to **immed_conn**.

6. Remove the connections you copied into the database from the data source specific configuration (**.ini**) files.

7. Remove the **CACHEOPTIONS.ini** file.

8. Restart the Data Server where you are hosting the cache with the appropriate –**processName:** command line option.

9. Restart the Display Builder and confirm that CMDB data attachments to that cache work as expected.

10. If CMDB data attachments do not show any data, confirm that cache data attachments using the named Data Server connection work.
    - If cache data attachments also do not work, check that the connection strings in your database are correct and that the named Data Server connection did not fail (check the console for connection errors).
    - If the cache data attachments do work, this probably indicates a problem with the cache index column values. The index is required by the CMDB, but only optional for direct cache data attachments. Confirm that your cache contains at least one **indexColumnName**. Also confirm that the value in the Index column of the Metric Source configuration table is the correct value for that Metric Source. It must be the value in the index column of the cache file(s) in the row that represents that Metric Source.

Creating Composites

According to the following three rules, optionally create composite objects for one or more Metric Source Types to display data from the caches defined for them.

All Composites must expose one scalar variable that the user can attach to a single Metric Source name or a list of Metric Source names.

For example, in the Display Builder select **Tools>Variables** to open the Variables dialog. Enter a variable name (e.g. **msciList**), then click **Add**. This variable must not be mapped to a sub and it must be public in order for it to show up in the list of properties on the composite object.

The Composite must contain a function that sets the value of the above variable into a substitution.

For example, in the Display Builder select **Tools>Functions** to open the **Functions** dialog. Click **Add** and enter the following:

- **Function Name** = **setMsciList**
- **Function Type** = **Set Substitution**
- **Substitution String** = **$MsciList**
- **Substitution Value** = Right-click and select **Attach to Data>Variable** and select **msciList**

All data attachments to CMDB caches must be configured using the **Attach to CMDB Data** dialog. The **Filter Type** must be **By Metric Source(s)** and the Metric Source name should be the substitution set by the function (e.g. **$MsciList**).

Note: Data attachments to CMDB caches should not be made using the **Attach to Cache Data** dialog, as this will prevent the composite from reflecting changes in the CMDB configuration.

Attach to CMDB Data

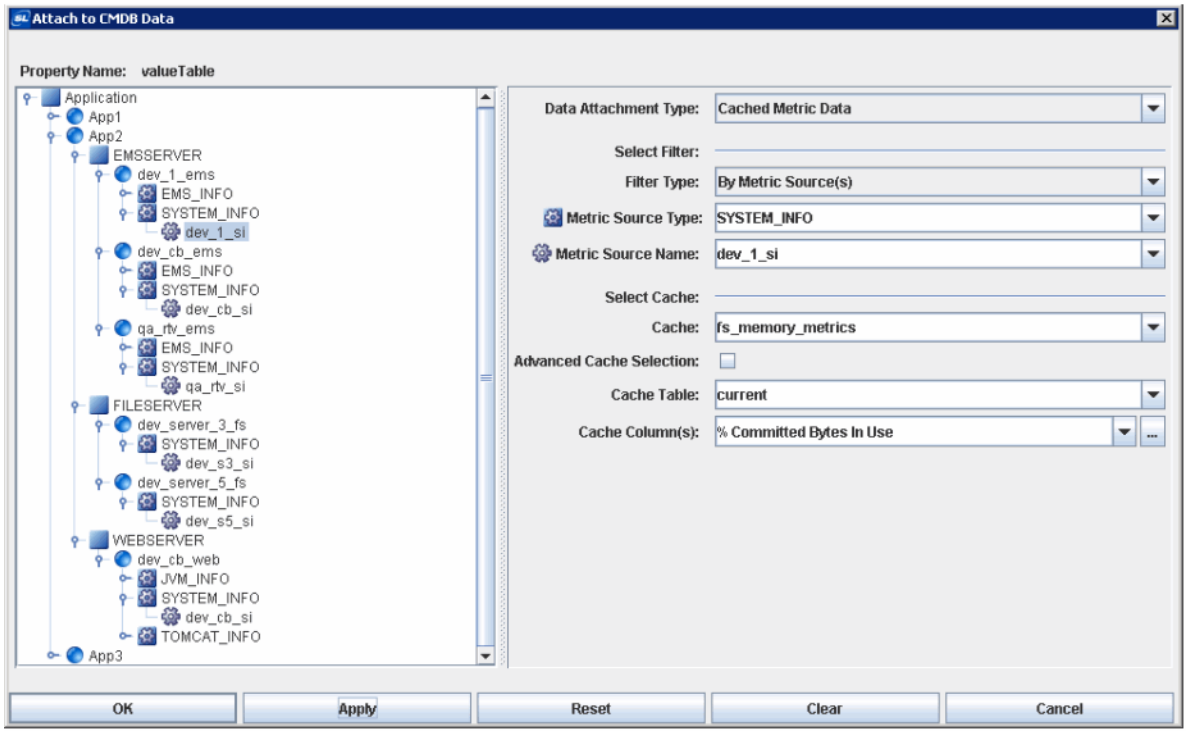
Note: This section applies only if you selected to use the Attach to CMDB Data dialog to attach your cache (Metric Source) data and chose not to auto-load caches.

From the **Object Properties** window you can access the **Attach to CMDB Data** dialog, which allows you to browse through the Container and Metric Sources defined in the "CMDB Configuration Tables" and create data attachments either to a list of Metric Source names or to a Cache filtered by a list of Metric Sources. The tree on the left is a graphic representation of the objects and relationships defined in the Container and Metric Source tables. This list is restricted depending on which CMDB objects the logged in user has permission to access.

Note: CMDB caches must be deployed and named Data Servers must be running in order to auto-load those caches and make connections defined in the Metric Source Table. Additionally, the Display Builder must contain definitions for all of those named Data Servers in order for the **Attach to Data** dialog to list them.

To display the **Attach to CMDB Data** dialog, right-click on the Property Name from the Object Properties window and select **Attach to Data>CMDB**. Options in the **Attach to CMDB Data** dialog vary depending on the selected Data Attachment Type: **Cached Metric Data** or **Metric Source List**. For either Data Attachment Type, you must select to filter by **Metric Source(s)** or by **Container(s)**. The fields in the **Select Filter** section can be set from the tree (on the left) or from the drop down menus. The advantage of filtering by Container is that when new children are added to the Container, data attachments will update automatically to include those children.

For example (in the screenshot below), for the Containers and Metric Sources shown you can enter App2 for the Container, * for the Child Container Type, and SYSTEM_INFO for the Metric Source Type. This will create a filter containing all of the SYSTEM_INFO Metric Sources under the App2 Container (dev_1_si, dev_cb_si, qa_rtv_si, dev_s3_s1,dev_s1_si). If you specify EMSSERVER for the Child Container Type, the filter will contain all SYSTEM_INFO Metric Source children of all EMSSERVER Container Children of App2 (dev_1_si, dev_cb_si, qa_rtv_si). You can enter * for both the Container and Child Container Type to get all Metric Sources of the specified Metric Source type.



Field Name	Description	
Data Attachment Type	Cached Metric Data	Allows you to make a data attachment to a cache in the CMDB system. Returns a table containing the cache data filtered by the index values of the Metric Sources that match the specified filter parameters.
	Metric Source List	Returns a string that is a comma delimited list of Metric Source names matching the filter parameters, which could be used in another CMDB data attachment when passed into the Metric Source Name field. **Note:** This Data Attachment Type only applies if composite objects have been created for one or more Metric Source Types. See "Creating Composites" for more information.
Select Filter	Metric Source(s)	**Metric Source Type** - Enter or select a Metric Source Type. If the Data Attachment Type is Metric Source List, you may enter *, otherwise you must specify a single type. **Metric Source Name** - Enter or select one or more Metric Source names. The list of available names in the drop down menu is filtered by the selected Metric Source Type. **Note:** It is possible to enter more than one name using a semi-colon delimited list (e.g. dev_1_si;dev_cb_si).

	Container(s)	**Container Name** - Enter or select a Container that is the parent of the Metric Sources you want to filter. **Child Container Type** - Enter or select a Child Container Type to further restrict the Metric Sources to filter within the selected Container. **Metric Source Type** - Enter or select a Metric Source Type. If the Data Attachment Type is Metric Source List, you may enter *****, otherwise you must specify a single type.
Select Cache	Cache	Enter or select the name of the cache you would like to attach. The list of available caches is determined by the selected Metric Source Type. **Cache Table** - Select the name of the cache table. **Cache Column(s)** - Select one or more columns. Click on the ... button to open the **Select Columns** dialog. The list of available columns is determined by the selected Cache Table.
	Advanced Cache	If selected, click on the **Select Advanced Cache Data** button to open the **Attach to Cache Data** dialog. Select the name of the cache table and one or more columns. Click on the ... button to open the "Select Columns" dialog. The list of available columns is determined by the selected Cache Table. Select the "Filter Rows: Advanced" option to specify additional time based filtering and/or configure a cache to use an SQL query to retrieve data from an external database. **Note:** The **Cache Name**, **Filter Column**, and **Filter Value** fields are disabled and auto-populate based on information you entered in the **Attach to CMDB Data** dialog.

When an object property is attached to data, the Property Name and Value in the **Object Properties** window will be displayed in green. This indicates that editing this value from the **Object Properties** window is no longer possible. Once a property has been attached to data, it receives continuous updates. To remove the data attachment, and resume editing capability in the **Object Properties** window, right-click on the Property Name and select **Detach from Data**. You will recognize that an object property has been detached from the data source when the Property Name and Value are no longer green.

Validation Colors

Fields in the dialog change colors according to the information entered. These colors indicate whether or not information is valid.

The following describes the significance of the Attach to CMDB Data validation colors:

Red	Invalid state	Entry is not valid.
White	Valid state	Entry is valid.

Substitutions

The Substitutions feature allows you to build open-ended displays in which data attachments depend on values defined at the time the display is run. A generic name such is used instead of a specific name. Later when the display is running, this generic value is defined. In this way, a single display can be reused to show data from a number of different sources. For more information on creating displays using substitution values, see "Substitutions".

Select Columns

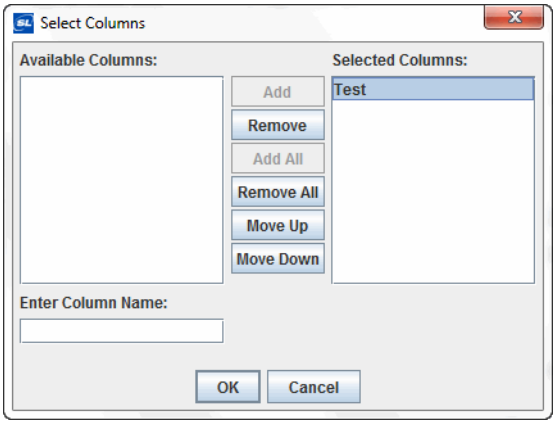
From the **Attach to CMDB Data** dialog you can specify which cache columns to display and in what order they will appear.

To display the **Select Columns** dialog, click on the ellipse button ... in the **Cache Column(s)** field (or right-click in the **Cache Column(s)** field and choose **Select Columns**). The dialog should contain a list of Available Columns that you can add to your table.

To add a column, select an item from the **Available Columns** list and click on the **Add** button. If the item you require is not listed, type your selection into the **Enter Column Name** field. Click the **Remove** button to delete an item previously added to the **Selected Columns** list. You can control the order of columns in a table by arranging the items in the **Selected Columns** list with the **Move Up** and **Move Down** buttons.

Validation colors indicate whether selected columns are valid. However, if even one column selected is invalid, the **Cache Column(s)** field in the **Attach to CMDB Data** dialog will register as an invalid entry.

Note: Invalid columns will not update.



The following describes the **Attach to CMDB Data** dialog commands:

Command	Description
OK	Applies values and closes the dialog.
Apply	Applies values without closing the dialog.

Reset	Resets all fields to last values applied.
Clear	Clears all fields. Detaches object from data source (once Apply or OK is selected).
Cancel	Closes the dialog with last values applied.

RTView Deployment - CMDB

This page contains details about the deployment process specific to the CMDB. Please go to the Deployment section of this documentation for instructions on how to implement your RTView deployment option. Return to this page whenever you are instructed to refer to deployment information that is specific to your data source. The CMDB deployment differs from other data source deployments in two ways:

1. The CMDB references data in cache files. Therefore, System Requirements and Setup for data sources used in your cache files must be met on the system(s) hosting the Data Server for that cache.

2. The CMDB requires access to the CMDB configuration file (CMDBOPTIONS.ini) and the CMDB Configuration Database from both the Data Server and the client applications.

Deploy CMDB Caches

To deploy CMDB caches to the Data Server(s), copy cache (.rtv) files along with any data source specific configuration files to the directory (or directories) where you will be running the Data Server(s). See "Deploying CMDB Caches" for more information.

Configuration Files

RTView saves general application settings as well as CMDB configuration options and any data source specific options in initialization files that are read at startup. If no directory has been specified for your initialization files and files are not found in the directory where you started the application, then RTView will search under **lib** in your installation directory. See "Application Options", "Application Options - CMDB", and "Deploying CMDB Caches" for more information.

The Configuration Utility, Display Builder, Display Viewer, Data Server, Historian, and Display Server require the following configuration files when you deploy:

File Name	Description
CMDBOPTIONS.ini	Contains options for the CMDB.
OPTIONS.ini	Contains general options and SQL Database connection options. If the CMDB is directly connecting to the Configuration Database, this file is needed to define the database connection.
***OPTIONS.ini (e.g. HAWKOPTIONS.ini)**	Contains options for data source(s) being used for a Metric Source. **Note:** The directory (or directories) where you run the Data Server must contain configuration (.ini) files with required options for data source(s).

There may be other common options that you want to deploy to RTView applications, such as role based security configuration.

Note: Options specified using command line parameters override values set in these initialization files.

Troubleshooting - CMDB

This section contains the following:

Display Builder

Problem: The tree in the **Attach to CMDB Data** Dialog is empty.

Solution: There are several reasons this could be happening:

- The Display Builder could not access the CMDB Configuration tables. In this case, you will see an error in the console (the table name might be different):

  ```
  ERROR: Metric Source Table required.
  ```

  Check that your database connection is valid (or if you are accessing the CMDB Configuration Database via a data server that your data server connection is valid). Also check that your CMDB options are correct. If you did not get an error in the console, check that you did see the following in the console:

  ```
  GmsRtViewCmdbDs: loading Metric Source Table
  GmsRtViewCmdbDs: loading Metric Source Type Table
  GmsRtViewCmdbDs: loading Permission Table
  GmsRtViewCmdbDs: loading Container Table
  ```

  All 4 of these lines should print to the console when the Display Builder starts up with the CMDB properly configured. If you only get a subset of these lines, check your database connection and your CMDB options.

- You did not login to the Display Builder or your login doesn't allow you access to the CMDB data. The CMDB requires that you login and that the role is in the Permissions Table in the CMDB Configuration Database. You will get an error in the console describing the permission problem:

  ```
  ERROR: You must login to use the CMDB.
  ERROR: Your role <loggedInRole> is not in the Permissions Table.
  ```

Problem: The tree in the **Attach to CMDB Data** dialog is not showing all of the expected entries.

Solution: There is more than one reason this could be happening:

Your Metric Source definitions contained errors. Check the console for errors about invalid Metric Sources.

- Your login does not allow you access to all of the tree items. Confirm that the entry for your login in the Permissions table is correct.

Problem: The tree in the **Attach to CMDB Data** dialog is ok, but there are no caches listed for one or more of the Metric Source Types.

Solution: This indicates that the Display Builder could not attach to the Data Server(s) hosting the caches. Confirm the following:

- The Display Builder contains a valid named Data Server connection to each Data Server that is hosting CMDB caches. Check the **Application Options** dialog to confirm the connection definitions are correct and check the console to see if there are any connection errors. If there are connection errors, confirm that the Data Server is running and that the connection definition is correct.
- The named Data Server connection names defined in the Display Builder match the Data Server names in the Metric Source table in the CMDB Configuration database.
- The Data Server was run with the **-processName** command line argument and that the process name used matches the Data Server name listed in the Metric Source table in the CMDB Configuration database.
- The Data Server actually loaded the cache file. See the "Data Server" troubleshooting section below.

Display Builder\Display Viewer\Display Server

Problem: CMDB data attachments do not show any data.

Solution: There are a few reasons this could happen.

- The application could not access the CMDB Configuration tables. In this case, you will see an error in the console (the table name might be different):

  ```
  ERROR: Metric Source Table required.
  ```

  Check that your database connection is valid (or if you are accessing the CMDB Configutaion Database via a data server that your data server connection is valid). Also check that your CMDB options are correct. If you did not get an error in the console, check that you did see the following in the console:

  ```
  GmsRtViewCmdbDs: loading Metric Source Table
  GmsRtViewCmdbDs: loading Metric Source Type Table
  GmsRtViewCmdbDs: loading Container Table
  ```

  All 3 of these lines should print to the console when the Display Builder starts up with the CMDB properly configured. If you only get a subset of these lines, check your database connection and your CMDB options.
- See "Display Builder" and read the section titled: *The tree in the Attach to CMDB Data dialog is ok, but there are no caches listed for one or more of the Metric Source Types*.
- Refer to "Steps for Deploying Caches". The CMDB does not contain any of the Metric Source data, it just allows you to access the Metric Source data which is actually stored in caches. Because of this, when there is a problem with missing data, it can be difficult to track down where the problem lies. This process will take you through the steps of running the cache files locally, then on the Data Server without the CMDB to confirm that the caches are properly constructed.

Data Server

Problem: The caches are not auto-loaded into the Data Server.

Solution: Run the Data Server with the **–cmdbdstrace:3** command line argument to show the list of cache files it loaded in the console. If the cache is missing, this indicates an incorrect entry in the CMDB Configuration Database.

Problem: The caches auto-loaded by a Data Server are missing information.

Solution: Run the Data Server with the **–cmdbdstrace:3** command line argument to show the list of cache files and associated data source connections that it loaded in the console. If a connection is missing or incorrect, this indicates an incorrect entry in the CMDB Configuration Database.

If this looks ok, refer to "Steps for Deploying Caches". The CMDB does not contain any of the Metric Source data, it just allows you to access the Metric Source data which is actually stored in caches. Because of this, when there is a problem with missing data, it can be difficult to track down where the problem lies. This process will take you through the steps of running the cache files locally, then on the Data Server without the CMDB to confirm that the caches are properly constructed.

Command Line Options - CMDB

In addition to General Options, the following command line arguments are enabled with the CMDB data source when you run RTView applications from a Windows Command Prompt or UNIX terminal window. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: "**-sub:$data:my Data**").

Name	Description
-dispServerTopCmdbObjs:	Limit the number of CMDB objects loaded by the Display Server at startup. By default, it will load in all Containers and Metric Sources without parents along with all of their children. Example: **-dispServerTopCmdbObjs:App1,App2**
-histDataServer:	Specify a Data Server for which the Historian will automatically load caches and redirect all data attachments. Example: **run_historian –histDataServer:ds1**
-processName:	Specify a process name for your Data Server. **Note:** The **processName** must match the name in the Data Server column of the Metric Source Table. Example: **run_dataserver –processName:ds1**

CHAPTER 17 Historian

This section contains the following:

Overview

The Historian enables you to archive data from any RTView data source to any database using JDBC. The Historian is a separate application process that runs independently from the other RTView Data Servers. The sole purpose of the Historian is to gather data and then, optionally, process and store that data with time-stamped values into your database. This historical information is most often utilized as a source of data (via SQL data attachments) in dashboards and alert definitions in RTView solutions.

The SQL data source is required to view data from the Historian in RTView. If you use the Historian without a license to the SQL data source the data is not visible in RTView. This might be suitable if you are using another SQL client. See "RTView Data Sources" for more information.

Note: You may not be licensed to run all RTView data sources.

Use Cases for Historical Data

The Historian is commonly used for:

- **Business, Operations and Performance Analysis:** Create trend charts to view and compare with current data, and perform calculations on a wide range of metric values over a long period of time.
- **Baselines & Alerting:** Use your historical data as the baseline for the creation of alerts that notify you when your current metrics are outside the normal range. Rather than hard-coding the alert thresholds, you specify the alerts as a percent (%). This makes for a more accurate alerting system.
- **Reporting and Audit Trails:** Create a database that you can leverage for a variety of reporting or auditing tools, and for surveying your historical data.

The Historian is instrumented with JMX which allows you to manage and monitor application settings. See "Managing the Historian Using JMX" for more information.

You configure the Historian by creating Data Configuration (**.rtv**) files, configuring the Historian to create connections to your data sources, and setting final configurations in the Historian application (see "Configuring the Historian"). After the Historian has stored data over a period of time, you can build a display using History data (see "Building a Display Using History Data").

For information about the Advanced Historian, see the "Advanced Historian".

Configuring the Historian

This section describes how to configure the Historian. For information about the Advanced Historian, see the "Advanced Historian" section.

The Historian is used to archive data from any RTView data source to any database using JDBC. The SQL data source is required to view data from the Historian in RTView. It is not required to use the Historian to archive data. You can verify that the SQL data source is licensed in your installation in the Display Builder **About** dialog. See "RTView Data Sources" for more information.

You configure the Historian by creating Data Configuration (**.rtv**) files, configuring the Historian to create connections to your data sources, and setting final configurations in the Historian application. The Data Configuration (**.rtv**) files specify which data the Historian is to archive from each data source. You create Data Configuration (**.rtv**) files in the Display Builder. The Data Configuration (**.rtv**) files also specify how to access the data, and perform required transformations in the data.

When you start the Historian, it immediately begins gathering data as specified and places the time-stamped data into the designated tables. After the Historian has stored data over a period of time, you can build a display using History data. See "Building a Display Using History Data" for more information.

You can install multiple versions of the Historian to segment data collection into different databases, and configure fault-tolerance. The Historian comes with default schemas that are largely configurable. For example, you can use configuration files to specify your own table names and, optionally, let the Historian automatically create the tables in your database if they do not already exist. See "Configuring Failover on the Historian" for more information.

Data Configuration File Options

To configure the Historian you create Data Configuration (**.rtv**) files in the Display Builder. Data Configuration (**.rtv**) files specify which data to archive from each data source, how to access the data, any transformations to perform with the data, and the table in which to then store the data. There are two Data Configuration (**.rtv**) file options, Display Files or Cache Definition Files.

Display Files

Use this option if your RTView deployment does not use caches. With this option you use your existing RTView display (**.rtv**) files (that you created in the Display Builder) to configure the Historian. Your existing display (**.rtv**) files contain objects that are already attached to your data. A record is stored in the history database every time new data is received by the Historian for each data attachment in your configuration display.

To use RTView display (**.rtv**) files to configure the Historian, you must have two history tables configured in your database: one for numeric data, the other for string data (for example, textual data). Numeric values must be attached to numeric object properties, and text values must be attached to string object properties. If you attach a numeric value to a meter label (which is a string object property), the numeric value is stored in the database as a string instead of a number, and the data is not available for trend graphs.

Cache Definition Files

Use this option if your RTView deployment uses caches or, if you need to use the "Advanced Historian". With this option you configure Cache object properties to point to the historical data to be archived. You can also specify which sources of current and/or historical data to keep in memory. To configure the Historian using this option, you specify which of these cached values to archive in a history database and then use these cache definition (**.rtv**) files to configure the Historian. The cache definition (**.rtv**) file specifies the following for the Historian: the data source, the cache and the database in which to store historical data.

The Cache data source allows for much higher performance than the traditional means of either performing analytics on information stored in a database, or querying it from the database to make calculations.

This section contains the following:

Basic Steps To Configure the Historian

Perform the following steps in the order given to configure the Historian:

1. Create or prepare Data Configuration files using the RTView Builder. Choose one:
   - "Display File Configuration"
   - "Cache Definition File Configuration"

   For help choosing the type of Data Configuration file, see Data Configuration File Options.

2. Configure Database Connections (see "Database Connection Configuration"). You can configure the Historian to create connections to your data sources using a direct JDBC connection.

3. Configure the Historian application database connection. See "Historian Application Configuration" for more information.

4. After the Historian has stored data over a period of time, you can build a display using History data. See "Building a Display Using History Data" for more information.

Display File Configuration

Perform the following steps to create a Data Configuration (**.rtv**) Display File for configuring the Historian. These procedures assume you are using your existing RTView display (**.rtv**) files (the displays you currently use to view your data) to configure the Historian. These procedures assume that you have already attached your historical data to an object.

To configure the Historian using a Display File:

1. In the Display Builder, verify that your data is attached to the appropriate type of object property for the corresponding value. That is, numeric values must be attached to numeric object properties, and text values must be attached to string object properties. If you attach a numeric value to a meter label (which is a string object property), the numeric value is stored in the database as a string instead of a number, and the data will not be available for trend graphs).

2. Verify that you have two history tables in your database: one for numeric data, and another for string data (textual data). If you do not, configure the two tables. By default, if the -rebuildtables option is on, the first time you run the Historian two tables are created: a numeric table named HISTORY and a string table named HISTORY_S. See "Command Line Options: Historian" for more information.

If you can verify the default numeric table (HISTORY) and the string table (HISTORY_S) are in your database, and you want to use these tables, proceed to "Database Connection Configuration".

If you do not want the Historian to create the HISTORY and HISTORY_S tables, you can create your own history tables manually. The tables must be created before running the Historian application.

To create your own history tables, use your preferred database client tools or database vendor client tools to do the following:

A. Use the **-tablename** command line option for the Historian to specify a single table name (other than HISTORY and HISTORY_S). The table name you specify is used for the numeric table, and _S is appended to the name to create the string table. See "Command Line Options: Historian" for more information. For example, if you specify:

MY_TABLE

The table name **MY_TABLE** is then established, and another table name is automatically created which has _S appended to the name. You then have the following two tables:

MY_TABLE is used for the numeric table.

MY_TABLE_S is used for the string table.

Note: Table names cannot contain spaces.

B. Configure the numeric table as described in the "Numeric Data Table", below.

C. Configure the string table as described in the "String Data Table", below.

D. Specify the table name on the command line for the Display Builder and Display Viewer Application. See "Command Line Options: Display Builder and Display Viewer" for more information.

Note: The two manually created tables are automatically rebuilt as data is received (you do not need to manually rebuild the tables).

Note: Tables created manually cannot utilize the **valueHistoryFlag** property to load initial history data on graph traces. See "Enable valueHistoryFlag" for more information.

Numeric Data Table

The following fields are supplied in the numeric data table (HISTORY or the table name you specified):

Field Name	Field Type	Field Description	Recommended Field Length (if applicable)
VAR_NAME	Text	The data attachment used to store this record.	80-160
TIME_STAMP	Text	The time this record was stored. This value is rounded to the closest second. Two columns are stored with each record. This field is only necessary if **String Timestamps (2)** is selected in the Historian GUI. This is the default selection. See "The Historian Application" for more information.	30
	Standard SQL TIMESTAMP data type	This field is only necessary if **SQL Timestamp** is selected in the Historian GUI. See "The Historian Application" for more information.	N/A
TIME_STAMP_LS	Text	A string value of the time this record was stored. This value contains the time including milliseconds. This field is only necessary if **String Timestamps (2)** is selected in the Historian GUI. This is the default selection. See "The Historian Application" for more information.	14
VAR_VALUE	double	The value for this field. This value is a number.	N/A

Note: If you select the **historyTableRowNameFlag** check box in Object Properties, then you must include and additional text column placed at the beginning of the table to store row name values. The Historian assumes that columns in the history table you create are in the same order as those in your tabular data element. If not, you can use the **-insertColumnNames** command line option for the

Historian to match data to the correct columns. See "Command Line Options: Historian" for more information.

String Data Table

The following fields are supplied in the string (textual) data table (HISTORY_S or the table name you specified):

Field Name	Field Type	Field Description	Recommended Field Length (if applicable)
VAR_NAME	Text	The data attachment used to store this record.	80-160
TIME_STAMP	Text	The time this record was stored. This value is rounded to the closest second. Two columns are stored with each record. This field is only necessary if **String Timestamps (2)** is selected in the Historian GUI. This is the default selection. See "The Historian Application" for more information.	30
	Standard SQL TIMESTAMP data type	This field is only necessary if **SQL Timestamp** is selected in the Historian GUI. See "The Historian Application" for more information.	N/A
TIME_STAMP_LS	Text	A string value of the time this record was stored. This value contains the time including milliseconds. This field is only necessary if **String Timestamps (2)** is selected in the Historian GUI. This is the default selection. See "The Historian Application" for more information.	14
VAR_SVALUE	Text	The value for this field. This value is a string.	40

Note: When you attach data to a tabular object property, such as **valueTable**, it is recommended that you create a table in the history database to store that data. If you do not create a table, data from your tabular element is stored in standard history tables and, in the case of tabular data with multiple columns, row integrity is lost. See "Tabular Functions" for details about how to store tabular data.

For step-by-step instructions on creating data configuration files for the Historian or to learn how to create a table in the history database to store tabular data, go to **Examples>Historian**.

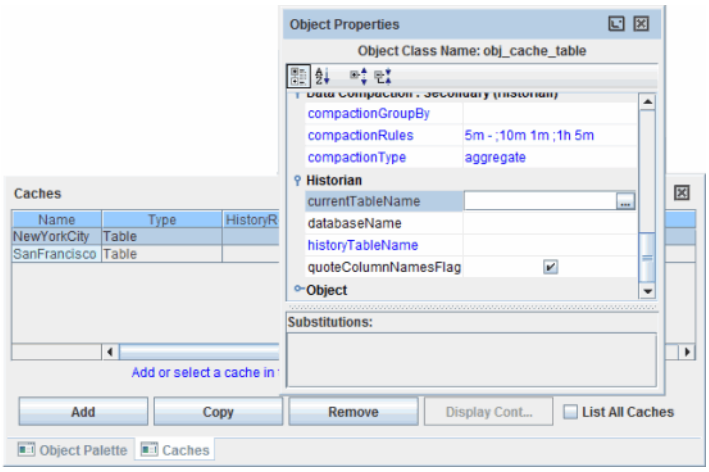
Proceed to "Database Connection Configuration".

Cache Definition File Configuration

Perform the following steps to configure the Historian using a Cache Definition File. The Cache Definition file specifies the following for the Historian: the data source, the cache, and the database in which to store historical data. In these Steps you specify your data source and your cache. See "Cache Definition Files" for more information.

To configure the Historian using a Cache Definition File:

1. Use the RTView Builder to create a Cache Definition file, or select a cache on which you want to enable the Historian.



2. Select the cache from the **Caches** dialog and edit the following Historian properties in the **Object Properties** dialog. (In this Step you specify your cache.)

Field Name	Description
currentTableName	Enter the name of a table in your Historian database in which to store the cache current data table. If you specify a **currentTableName** and no table by that name exists in your database, one will be created for you the first time you run the Historian. On each update of the data attached to the **valueTable** property, the Historian will create or update corresponding rows in the specified **currentTableName** in the Historian database. The **indexColumnNames** property will be used to locate rows to update otherwise, if no **indexColumnNames** are defined, the Historian will replace the entire table on each update. If the **deleteTable** property is used to delete rows from the cache's current table, the Historian will also delete those rows from the specified **currentTableName** table in the Historian database. On startup of an RTView application, the specified **currentTableName** will be queried from the Historian database for initial rows to be loaded into the cache's current table. To prevent this initial data from being loaded, you can run the Display Builder or Display Viewer from a command window and use the **-nohistory** command line option. **Note:** The **currentTableName** property is not supported in combination with the Historian's **-kdbformat** command line option.

databaseName	Enter the name of your Historian database that contains history and/or current tables for this cache. If left blank, the database name **RTVHISTORY** is assumed. The "Extend with SQL" option requires that the database specified in this property have a SQL database connection. Or, if this property is blank, there must be a database connection to the default **RTVHISTORY** database.
historyTableName	Enter the name of a table in your Historian database in which to store the cache history data table. If you specify a **historyTableName** and no table by that name exists in your database, one is created for you the first time you run the Historian. When rows are appended to the cache history table, the Historian also appends those rows to the specified **historyTableName** in the Historian database. If you specify a **timestampColumnName**, the Historian assumes that column is to be used for sorting and purging rows in chronological order and does not append its own **timestamp** columns to the specified **historyTableName** table. Before data is stored in the database, the cache **schemaTable**, if any, is applied. If you specify a **historyTableName**, RTView applications automatically query the Historian database table for the initial rows to be loaded into the cache history table, provided that: • The **initialTable** property is not attached to data (indicating an explicit query has been configured to load cache history data), and • A **timestampColumnName** is specified (so that it can be used to sort the SQL query result by time). The number of rows loaded by this initial query is limited by the **maxNumberOfHistoryRows** property. To prevent this initial data from being loaded, you can run the Display Builder or Display Viewer from a command window and use the **-nohistory** command line option. The "Extend with SQL" option requires that this property specify the cache database table name. The table specified in this property is used for queries. You can use the -verbose command line option for the Historian to see the specific SQL command used to create this table. See "Command Line Options: Historian" for more information.
quoteColumnNamesFlag	If selected, column and table names will be quoted in all SQL queries and commands for this cache.

3. When you have finished configuring the properties, Save the display (**.rtv**) file. You will add this Cache Definition file to the list of Data Configuration files in the Historian application. See "Data Configuration File Options" for more information.

Proceed to "Database Connection Configuration", next.

Database Connection Configuration

Perform the following steps to configure the Historian connections to your data sources. There are two options: "Direct JDBC Connection" or "Historian Application Configuration".

Direct JDBC Connection

For the Historian to use a direct JDBC Connection to communicate with your database, you must have a JDBC driver for your database. JDBC drivers are available from most database vendors.

To configure a direct JDBC Connection:

1. Locate the JDBC driver on your machine.
2. Define an environment variable named RTV_USERPATH that includes a path to the driver class or .jar file containing the driver class.

Note: You can define **RTV_USERPATH** to include paths to multiple driver classes. The **RTV_USERPATH** variable is included in the classpath for RTView and the Historian.

---

3. Note the fully qualified class name for the driver class and the database URL required to connect to your database. You need this to setup the database (when you configure the Historian application--see "Historian Application Configuration"). The database URL typically contains protocol and sub-protocol strings for your database, as well as the path to the database and a list of properties. If you do not know the syntax for your database URL, consult the documentation for your JDBC driver.

Proceed to "Historian Application Configuration".

Historian Application Configuration

This section describes how to add the Data Configuration file you previously created to the Historian application, and configure the database in which your historical data will be stored. See "Data Configuration File Options" for more information.

The Historian Application allows you to configure the database connection. Click **Save Configuration** to save these settings. The **Historian Console** tab records errors and information. If you select the **Show Data in Console** check box, the Historian Application will output a line for each record that is stored in the database.

Data source-specific options are read in from initialization (**.ini**) files created in the Display Builder. For information on creating initialization files or command line options for your data source, refer to Applications Options or Command Line Options in the Data Sources section of this documentation.

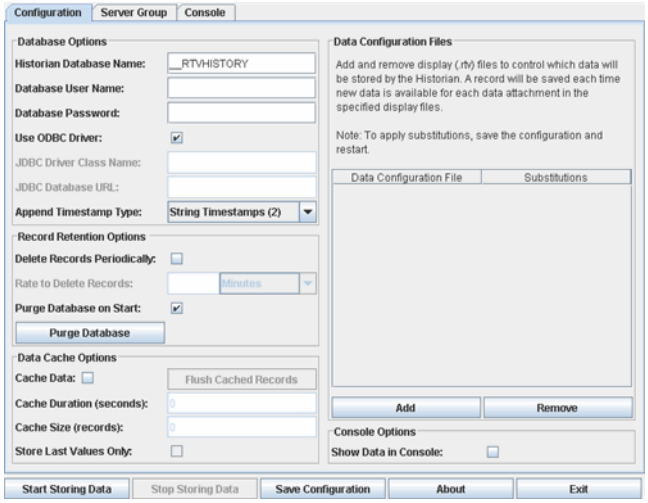
To configure the Historian Application

1. Start the Historian.

   In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window") type: **run_historian**

The Historian application opens and begins gathering and placing your (now time-stamped) data into tables based on your Data Configuration file specifications.

2. Click **Configuration** to edit settings.



3. Make the following **Database Options** entries as appropriate:

Field Name	Description
Historian Database Name	The Data Source Name for the database the Historian will use to store and query information. **Note:** This name must also match the name of the history database configured in the Display Builder **Application Options>**"SQL Tab".
Database User Name	The user name to pass into the history database when making a connection. This parameter is optional. You cannot edit this field while the Historian is storing data.
Database Password	The password to pass into the history database when making a connection. This parameter is optional. You cannot edit this field while the Historian is storing data. If you need to provide an encrypted password (rather than expose server password names in a clear text file, use the encode_string command line option with the following syntax: **encode_string mypassword** where **mypassword** is your plain text password. For example, enter the following in an initialized command window (see "Initializing a Command Prompt or Terminal Window"): **encode_string mypassword** and you will receive an encrypted password: **encrypted value: 013430135501346013310134901353013450134801334** Copy the encrypted value, paste it into the password field and click **Save** to save this value to the initialization (***.ini**) file. Or, if necessary, manually edit the (***.ini**) file to include the encrypted value. **Note:** If you need to manually edit a configuration (***.ini**) file, contact SL Technical Support at support@sl.com for information about supported syntax.
JDBC Driver Class Name	The fully qualified name of the JDBC driver class to use when connecting to the history database. The path to this driver must be included in the **RTV_USERPATH** environment variable. You cannot edit this field while the Historian is storing data.

JDBC Database URL	The full database URL to use when connecting to the history database using the specified JDBC driver. Consult your JDBC driver documentation if you do not know the database URL syntax for your driver. You cannot edit this field while the Historian is storing data.
Append Timestamp Type	Specify the type of **timestamp**: **None** - No **TIMESTAMP** column is stored. **SQL Timestamp** - A single **TIMESTAMP** column is stored using a standard SQL TIMESTAMP data type. **String Timestamps (2)** - Two **TIMESTAMP** columns are stored with each record as strings. This is the default.

4. Make the following **Record Retention Options** entries as appropriate:

Field Name	Description
Delete Records Periodically	If selected, the Historian will periodically delete records from the numeric table (HISTORY or the table name you specified) and the string table (HISTORY_S or the table name you specified) according to the rate set.
Rate to Delete Records	The length of time (in minutes, hours or days) a record can exist before it marked for deletion. For example if you set the rate to 5 minutes, then the next time you start RTView all records older than 5 minutes will be deleted. For the remainder of your work session, RTView will search every 2.5 minutes (half the rate) for records that exceed the rate.
Purge Database on Start	If selected, the Historian will clear out the numeric table (HISTORY or the table name you specified) and the string table (HISTORY_S or the table name you specified) before storing new data.
Purge Database	Clears out the numeric table (HISTORY or the table name you specified) and the string table (HISTORY_S or the table name you specified) immediately.

5. Make the following **Data Cache Options** entries as appropriate:

Field Name	Description
Cache Data	If selected, the Historian will cache data according to the **Cache Duration** and **Cache Size** specifications. If both **Cache Duration** and **Cache Size** are set, then the data records will be committed as soon as the first limit is reached.
Cache Duration	Length of time (in seconds) to cache before committing records to the database. If the value is set to **0**, the Historian will not commit records in the database immediately, but rather will store them in the cache to be committed later.
Cache Size	Number of records to cache before committing them to the database.

Flush Cached Records	Flush all cached records to the database immediately.
Store Last Values Only	If selected, the Historian will store only the last (most recent) values in the cache for each unique data attachment. By default, the Historian stores all records in the cache each time the **Cache Duration** or **Cache Size** limit is reached. This option allows the Historian to store less data than it receives, which can be useful in a configuration where the Historian receives data from the Data Server at a higher rate than necessary for historical storage. For example, suppose the Data Server sends tables from two SQL attachments to the Historian: **Query1: select * from table1** **Query2: select * from table2** If the Data Server executes those queries every 10 seconds and the Historian has **Cache Duration** of 60 seconds, then every 60 seconds the Historian's data cache will contain six result tables from Query1 and six result tables from Query2. By default, the Historian will commit all twelve tables to the database. However, if **Store Last Values** is selected, the Historian will commit only the sixth (most recent) table from Query1 and Query2 and discard all other tables in the cache. **Note:** This option is available only if **Cache Data** is selected and the value of **Cache Duration** is greater than **0**.

6. Make the following **Alert Persistence** entries as appropriate:

Note: Alerts are sent from the application running the alert engine to the Historian via the RTVAgent. Therefore, you must enable connections from remote agent applications on the "RTVAgent Options Tab" and then copy the **RTVAGENTOPTIONS.ini** file to the directory where you will be running the Historian.

Field Name	Description
Alert Persistence Enabled	Select to enable the persistence of alerts during fail over of an alert engine. All fields and current state of all active alerts, as well as all cleared alerts that have not been removed from the system will be stored.
Alert Persistence Table Name	Name of the table where the alerts will be persisted. **Note:** This must be the same table name specified in Alert Persistence tab of the **Application Options>**"Alerts Tab" dialog.

Note: The Historian must run the RTVAgent data source locally (not via a Data Server) to persist the alerts. If you are running the Historian against a Data Server you can run the RTVAgent data source locally using the following command line option: **run_historian -dsenable:RTVAGENT**

---

7. In Data Configuration Files, click **Add**, enter the name of the Data Configuration (**.rtv**) file you just created and, optionally, any corresponding substitutions. "Substitutions" allow the same file to be used multiple times by different sets of substitutions.

Note: Each cache that is defined in the file should contain a substitution in the **cacheName** property, since cache names must be unique.

---

8. Click **Apply** to execute. Data Configuration files are added or removed after you click **OK**, **Apply**, or **Save**.

Note: To edit the Data Configuration (**.rtv**) file and apply changes without restarting the Display Builder, select a file from the list and click Refresh Selection to reload definition settings. The Refresh Selection option is enabled when you select a Data Definition (**.rtv**) file that has been both added and applied.

The Data Configuration (**.rtv**) file is added to the data source and data for these caches is collected. See the "Attach to Cache Data" section for details about attaching to cache data variables.

Note: After a Data Configuration (**.rtv**) file has been added to the data source, data is collected for that cache regardless of whether any currently open display (**.rtv**) files are showing that data.

---

9. Select **Show Data in Console** so that the Historian prints out a line for each record that is being stored in the database.

10. Select **Start Storing Data**.

11. Select Save Configuration.

12. Verify your configuration by clicking **Console** to view the **Show Data in Console** output.

Your setup is complete. After the Historian has stored data for a period of time, proceed to "Building a Display Using History Data".

Optionally, you can configure failover for the Historian. For information, see "Configuring Failover on the Historian".

Configuring Failover on the Historian

This section describes how to configure failover for the Historian. You configure failover in the Historian application. For details about configuring the Historian, see "Configuring the Historian".

High Availability Historians

The High Availability feature is intended to be used with a database system that also supports redundancy (through mirroring, clustering, or other techniques) so that any Historian in the group can update the same virtual database.

To configure failover, create a Server Group and designate backup Historians to support a failover event. By default, no Server Group is defined. You can specify a primary Historian as well as one or more backup Historians. Only the primary Historian is capable of writing to the database.

Each Server Group member is assigned a port number and a priority. Using a socket connection on the assigned ports, each member monitors the status of the others in the group. The member with the highest priority is elected as the primary Historian. If the primary Historian fails, is shut down, or loses its connection to the database, the backup Historian with the highest priority becomes the new primary. A Historian with a priority of 2 is has higher priority than a Historian with a priority of 1.

For example, if we have three Historians with the following priority:

Historian A / Priority 1

Historian B / Priority 2

Historian C / Priority 3

and Historian C is the Primary. If Historian C is shut down, Historian B becomes the Primary. In the event of a tie in priority ranking, the Historian that was started first becomes the primary.

Note: Each Historian in the group should be configured with the same set of data configuration files, retention options, data source options, etc.

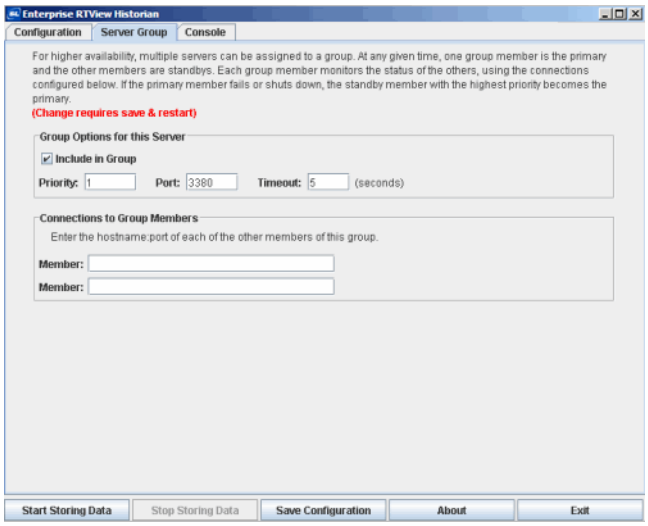
To configure failover for the Historian:

1. Start the Historian application.

   In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window") type: **run_historian**

   The Historian application opens.

2. Click the **Server Group** tab and make the following entries:



Field Name	Description
Include in Group	Check the box to enable.

Priority	Choose a priority for this Historian. The default is **1**. The online Historian with the highest priority is the primary Historian, and updates the database.
Port	Enter a port number on which this Historian accepts connections from other Historians in the group. The default is **3380**.
Timeout	The amount of time, in seconds, at which a connection with a group member is considered down.
Member	For each Historian in the group, enter the **hostname** and **port** in the **Member** text field.

3. Click **Save Configuration**.

Starting the Historian

This section describes how to start the Historian Application and describes the Historian Application interface. For information about configuring the Historian, see "Configuring the Historian".

The first time you start the Historian you must start it as an application to configure the initial settings. You can start the Historian Application on Windows or UNIX. After you configure the Historian, you can run it as a daemon process or as a Windows Service.

There are several ways to start the Historian:

- "Starting the Historian Application"
- "Starting the Historian as a daemon process"
- "Starting the Historian in the background as a Windows Service"

Several command line options are supported for the Historian. Java options specified in **RTV_JAVAOPTS** are used by the **run_historian** scripts.

Rebuilding Historian Tables

To rebuild HISTORY and HISTORY_S tables each time you run the Historian, use the **-rebuildtables** command line option for the Historian. See "Command Line Options: Historian" for more information.

If you manually created your Historian tables, the tables are automatically rebuilt as data is received.

The Historian is instrumented with JMX to allow you to manage and monitor application settings. See "Managing the Historian Using JMX" for more information.

Starting the Historian Application

In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window") type: **run_historian**

The Historian application opens.

Starting the Historian as a daemon process

The first time you start the Historian, you should start it as an application to configure the initial settings.

Running the Historian as a daemon process allows you to run the Historian without a display. To do so, run the **-daemon** command line parameter from a Windows Command Prompt or UNIX terminal window.

Starting the Historian in the background as a Windows Service

The first time you start the Historian, you should start it as an application to configure the initial settings.

See "Running as a Windows Service".

The Historian Application

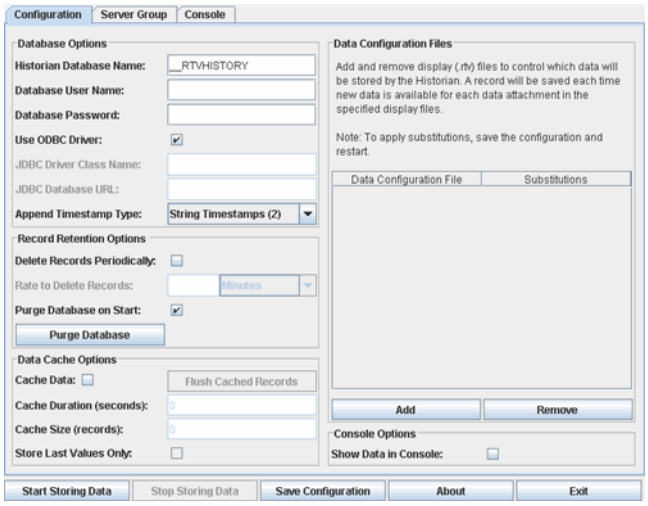
Use the Historian Application to make final configurations for the Historian database connections. For details about configuring the Historian Application, see "Historian Application Configuration".

Note: The Historian Application configuration occurs within a series of Historian configuration steps. For details, see "Configuring the Historian".

Use the Historian Application:

- **Configuration** tab to specify settings for the Historian. Click **Save Configuration** to save these settings. Select **Show Data in Console** to output a line for each record that is stored in the database to the **Console** tab.
- **Console** tab to view errors and information.
- **Server Group** tab to configure failover. For details about configuring failover, see "Configuring Failover on the Historian".

Data source-specific options are read in from initialization (**.ini**) files created in the Display Builder. For information on creating initialization files or command line options for your data source, refer to Application Options or Command Line Options in the "RTView Data Sources" section of this documentation.



Database Options Region

Field Name	Description
Historian Database Name	The Data Source Name for the database the Historian will use to store and query information. **Note:** This name must also match the name of the history database configured in the Display Builder's **Application Options>** "SQL Tab".
Database User Name	The user name to pass into the history database when making a connection. This parameter is optional. You cannot edit this field while the Historian is storing data.
Database Password	The password to pass into the history database when making a connection. This parameter is optional. You cannot edit this field while the Historian is storing data. If you need to provide an encrypted password (rather than expose server password names in a clear text file), use the **encode_string** command line option with the following syntax: **encode_string mypassword** where **mypassword** is your plain text password. For example, enter the following in an initialized command window (see "Initializing a Command Prompt or Terminal Window"): **encode_string mypassword** and you will receive an encrypted password: **encrypted value: 013430135501346013310134901353013450134801334** Copy the encrypted value, paste it into the password field and click **Save** to save this value to the initialization (***.ini**) file. Or, if necessary, manually edit the (***.ini**) file to include the encrypted value. **Note:** If you need to manually edit a configuration (***.ini**) file, contact SL Technical Support at support@sl.com for information about supported syntax.
JDBC Driver Class Name	The fully qualified name of the JDBC driver class to use when connecting to the history database. The path to this driver must be included in the **RTV_USERPATH** environment variable. You cannot edit this field while the Historian is storing data.
JDBC Database URL	The full database URL to use when connecting to the history database using the specified JDBC driver. Consult your JDBC driver documentation if you do not know the database URL syntax for your driver. You cannot edit this field while the Historian is storing data.
Append Timestamp Type	Specify the type of **timestamp**. **None** - No **TIMESTAMP** column is stored. **SQL Timestamp** - A single **TIMESTAMP** column is stored using a standard **SQL TIMESTAMP** data type. **String Timestamps (2)** - Two **TIMESTAMP** columns are stored with each record as strings. This is the default.

Record Retention Region

Field Name	Description
Delete Records Periodically	If selected, the Historian will periodically delete records from the numeric table (HISTORY or the table name you specified) and the string table (HISTORY_S or the table name you specified) according to the rate set.

Rate to Delete Records	The length of time (in minutes, hours or days) a record can exist before it marked for deletion. For example if you set the rate to 5 minutes, then the next time you start RTView all records older than 5 minutes will be deleted. For the remainder of your work session, RTView will search every 2.5 minutes (half the rate) for records that exceed the rate.
Purge Database on Start	If selected, the Historian will clear out the numeric table (HISTORY or the table name you specified) and the string table (HISTORY_S or the table name you specified) before storing new data.
Purge Database	Clears out the numeric table (HISTORY or the table name you specified) and the string table (HISTORY_S or the table name you specified) immediately.

Data Cache Options

Field Name	Description
Cache Data	If selected, the Historian will cache data according to the **Cache Duration** and **Cache Size** specifications. If both **Cache Duration** and **Cache Size** are set, then the data records will be committed as soon as the first limit is reached.
Cache Duration (seconds)	Length of time (in seconds) to cache before committing records to the database. If the value is set to **0**, the Historian will not commit records in the database immediately, but rather will store them in the cache to be committed later.
Cache Size (records)	Number of records to cache before committing them to the database.
Flush Cached Records	Flush all cached records to the database immediately.
Store Last Values	Only If selected, the Historian will store only the last (most recent) values in the cache for each unique data attachment. By default, the Historian stores all records in the cache each time the Cache Duration or Cache Size limit is reached. This option allows the Historian to store less data than it receives, which can be useful in a configuration where the Historian receives data from the Data Server at a higher rate than necessary for historical storage. For example, suppose the Data Server sends tables from two SQL attachments to the Historian: **Query1: select * from table1** **Query2: select * from table2** If the Data Server executes those queries every 10 seconds and the Historian has **Cache Duration** of 60 seconds, then every 60 seconds the Historian's data cache will contain six result tables from Query1 and six result tables from Query2. By default, the Historian will commit all twelve tables to the database. However, if **Store Last Values** is selected, the Historian will commit only the sixth (most recent) table from Query1 and Query2 and discard all other tables in the cache. **Note:** This option is available only if **Cache Data** is selected and the value of **Cache Duration** is greater than 0.

Data Configuration Files Region

Field Name	Description
Data Configuration Files Add	Add a data configuration file to the list. Data configuration files are display (**.rtv**) files that were created in the Display Builder. When you start storing data, a record will be added to the database each time new information is received for each data attachment in all of the specified data configuration files. **Note:** If a data configuration file is added while the Historian is storing data, it will start storing data for attachments in that file as soon as new information becomes available.
Remove	Remove the selected data configuration file from the list. If the Historian is storing data when a file is removed, it will stop storing data for that file immediately.
Substitutions	To add or edit a substitution on a specified data configuration (**.rtv**) file, double-click in the corresponding field of the Substitutions column. See "Substitutions" for more information.

Console Region

Field Name	Description
Show Data in Console	If selected, the Historian Console prints out a line for each record that is being stored in the database.

General Options

Field Name	Description
Start/Stop Storing Data	Start or stop storing data in the database.
Save Configuration	Save settings to an initialization file (**HISTORY.ini**), which will be used next time you run the Historian. **Note:** Unless you specify a directory for your initialization files, you must run the Historian from the same directory in which the initialization (**.ini**) file was saved. See "RTV_JAVAOPTS" for more information.
About	Click on to read about RTView.
Exit	Exit the Historian, stop storing data in the database and close the Historian window.

Building a Display Using History Data

This section explains how to create a display in the Builder that accesses your historical data. This section assumes you have configured the Historian, and that your Historian has had time to gather data. For configuration information, see "Configuring the Historian". For general information about the Historian, see "Overview".

Data saved out of the Historian to the history database can be viewed from an RTView display if you are licensed for the SQL data source. For step-by-step instructions on viewing and configuring historical data, see **Examples>**"The Historian".

Note: The SQL data source may not be licensed in your RTView installation.

Note: The name of the history database must match the Historian Database name specified in the Historian application. See "Historian Application Configuration" for more information.

You build a display using History data by attaching an object to your historical data. You have the following options for attaching your historical data:

- "Enable valueHistoryFlag"
- "Creating a SQL Data Attachment"
- "Extend with SQL"

Enable valueHistoryFlag

The trend graph on the **Graphs** tab of the Object Palette (class name: **obj_trendgraph02**) features a **trace*ValueHistoryFlag** property for each trace. If you select this property, RTView attempts to connect to the history database and load initial data from the Historian for the corresponding trace any time you open a display containing that graph. Only data within the **timeRange** set on the graph is loaded and the graph is updated as live data becomes available. See "Trend Graphs" for more information.

To set up a trace to load from the history database, you must have a corresponding object in a history configuration file that the Historian is using. The data attachment in the configuration file must exactly match the data attachment for your trace. The data must not be stored in a user defined table. Only data stored in the "Numeric Data Table" (HISTORY or the table name you specified) is used to load historical data into your trace.

Creating a SQL Data Attachment

Any object in RTView can display historical data using an SQL data attachment.

For information about configuring history database tables for use in data attachments, see the steps to create your own history table. See "Display File Configuration" for more information.

Extend with SQL

The **Extend with SQL** option is the simplest way to retrieve and display all of your data: current, real-time data and historical data.

As a cache data source receives real-time data it displays the data and stores it in-memory. The cache continues receiving and displaying real-time data, and eventually the in-memory space reaches full capacity. The oldest in-memory data then overflows into a second source, the History database. The data stored in the History database (the "overflow" data) is not included in displays. Only data from the first source, stored in the cache in-memory space, is displayed.

To use **Extend with SQL**, you configure a cache to use a SQL query to retrieve data from an external database. To configure **Extend with SQL**, make a data attachment to a cache history table, select Advanced Filter Rows, specify the time period of interest, and select **Extend with SQL**. For details, see "Extend with SQL" and "Filter Rows: Advanced".

Extend with SQL requires the following Object Properties be specified for a:

Table Cache

- maxNumberOfHistoryRows
- databaseName
- historyTableName

Double Cache

- maxNumberOfHistoryRows
- databaseName
- historyTableName

See "Caches" for more information.

Managing the Historian Using JMX

The Historian is instrumented with JMX to allow you to manage and monitor application settings. To enable JMX, you must run the Historian using the **-jmxport** command line option:

-jmxport:(port number)

The JMX Monitor demo, located in **demos\jmxmonitor** in your RTView installation, shows how to use RTView to monitor the Historian using JMX.

The Historian contains one MBean named **RTViewHistorian:name=Manager** with the following methods:

Method / Attribute	Type	Description
DataConfigurationFileInfo	TabularData	For each Data Configuration File, the table contains the following columns: **File Name** - Name of the Data Configuration File. **Last Access Time** - Last time this file was accessed. **Substitutions** - Substitutions applied to the Data Configuration File.
NumberOfDataConfigurationFiles	int	Number of data configuration files in use.
HistorianSuspended	boolean	True if the Historian is currently suspended (i.e.: not storing data), false otherwise.

The **RTViewHistorian:name=Manager MBean** also supports the following commands:

Method / Attribute	Description
killHistorian	Exit the Historian.
purgeHistoryData	Clear out tables (numeric and string).
resumeHistorian	Start storing data.
suspendHistorian	Stop storing data.

Advanced Historian

The Advanced Historian feature (RTView v.5.4+) provides the following ways to reduce the amount of data stored in the Historian cache table: Table Displacement and Data Aggregation. Both options enable you to automatically reduce the amount of stored data as a background process simply by defining rules for Table Displacement or Data Aggregation. The Advanced Historian feature can be used on new or existing Historian data collections, allowing you to manage existing historical data.

With the standard "Historian", data is reduced by manually backing up or deleting archived data.

The Advanced Historian is an optional licensed feature that (when activated) provides additional Cache properties to describe your data management preferences. You then determine whether and how to reduce the amount of historical data for each table. If Table Displacement is used, archival data is removed and placed in a separate, new table, at a frequency that you specify (for example, once a day). If Data Aggregation is used, the number of data points is reduced. In this latter case, you specify, using rules, both the granularity and the method for reducing the amount of data (for example: the sum, average, count, minimum or maximum).

The Table Displacement option might be suitable if you only need to view recent data, for example, for the last 24 hours. The Data Aggregation option is suitable if you need to view data from further in the past, for example, for the last few months or years.

You configure the Advanced Historian feature using the **obj_cache_table** object and configuring Cache properties. To access the **obj_cache_table** object, in the Display Builder, select **Tools>Caches** to open the **Caches** dialog, add your caches, configure their properties, and Save the display (**.rtv**) file.

You can use either "Table Displacement" or "Data Aggregation" on a single table, not both. If rules are not specified for either Table Displacement or Data Aggregation, neither displacement nor aggregation occurs.

See "Caches" for more information.

Data Aggregation

You can configure Data Aggregation for one or more tables in a display. The Data Aggregation option aggregates, or reduces, the number of data points based on the method you specify using a rule (for example, sum, average, count, min, or max).

The aggregation of data is performed by reading data from the Historian table, aggregating the data based on the rule specifications, and then writing it to the Historian table. Data Aggregation is performed at known intervals, specified by the **compactiontimerinterval** (the default setting is 5 seconds). See "Command Line Options: Historian" for more information.

You setup Data Aggregation by configuring the **obj_cache_table** cache object to supply data to the historian table.

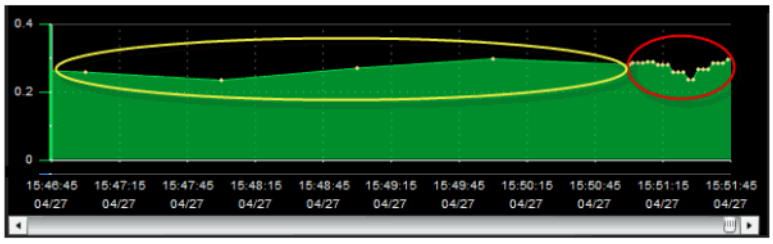
Single Versus Multiple Tables

When aggregated data is stored in a single table, a single RTView object supplies data to a single trend chart. The trend chart shows all the historic data and, as the older data is aggregated, the data points thin out. That is, the length of time between each time-stamp increases. If the raw data is stored at one-second intervals, as it ages, you might not need such granularity. You can configure the Historian to store aged data at longer intervals, for example, one minute intervals. Increasing the time interval for storing aged data makes more table rows available in the Historian.

By default, Data Aggregation uses Single Table Aggregation.

Single Table Aggregation

Use a single table if you want to display all of your data in one table, and if it is suitable for the length of time between data points to increase as your data ages. By default, Data Aggregation uses Single Table Aggregation. The following trend graph illustrates how Single Table Aggregation is displayed: current data, located on the right side of the graph, is gathered at one-second intervals, and, as the data ages in the table, the length of time between data points increases to one-minute intervals.



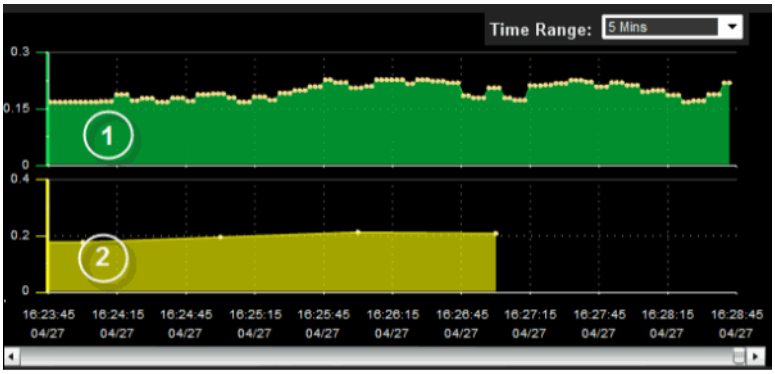
Multiple Table Aggregation

Use multiple tables if you do not need to view all of your data in one table, or you want the option to display your data in different time intervals (for example, **1 hour**, **1 day**, or **1 month** time intervals). Using multiple tables has several advantages. For example, you can use multiple RTView objects to display the data in different ways. For instance, you could have a drop-down menu on a chart that allows the user to select the data view: **raw**, **20 minutes**, **1 hour**, **1 day**, or **1 month** time intervals. Also, the length of time between data points for older data does not increase as they do with a single table.

When multiple tables are used, aggregated data is stored from the most recent point in time for all of the tables. This is different from Table Displacement where, as the data ages, each table storing the data has a different start date and time. This is necessary for a scenario in which multiple RTView data objects supply data to multiple trend charts, which show the historic data separately by interval.

The following trend graphs illustrate how Multiple Table Aggregation might be displayed.

- **Trend Graph 1** data points are per one-second intervals (also referred to as raw data), and the duration is only the previous 5 minutes. The table containing this data is accessed when **2 Mins** or **5 Mins** is selected from the **Time Range** drop-down menu.
- **Trend Graph 2** data points are per one-minute intervals, and the duration is for the previous hour. The table containing this data only shows data points at that one-minute interval, back to the time period chosen (up to an hour). The table containing this data might be accessed when a time period of an hour or less is selected from the Time Range drop-down menu. Until Data Aggregation is performed on the most recent two minutes of this table, Table 1 is accessed to display the most recent two minutes of data.



You configure Multiple Table Aggregation in the **Edit Compaction Rule** dialog. See "Configuring Data Aggregation", Step 6 for information.

By default, rules for all data are stored in a single table.

The algorithm for Data Aggregation does not do compaction of data that was stored before the current historian session. Use **-smoothCompaction** to perform Data Aggregation on your previously existing data.

Configuring Data Aggregation

This section provides step-by-step details for configuring the **Advanced Historian Data Aggregation** option for storing archived data. You configure Data Aggregation using the **obj_cache_table** object and configuring Cache properties. See "Caches" for more information.

These procedures require a Historian Compaction license key.

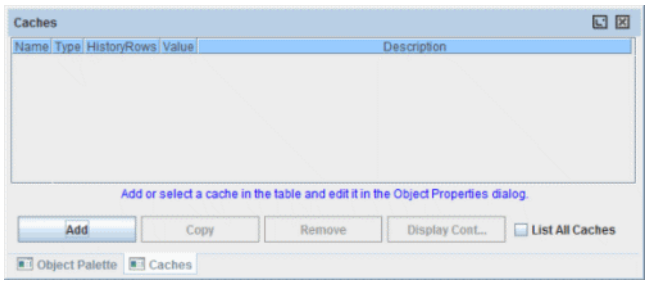
Note: The algorithm for Data Aggregation does not do compaction of data that was stored before the current historian session. The following procedures include instructions to use **-smoothCompaction**, which performs Data Aggregation on your previously existing data.

At this point you have:

- Attached data to the **obj_cache_table** object using the **valueTable** property.
- Obtained a Historian Compaction license key.

To configure Data Aggregation on a cache table:

1. In the Display Builder, select **Tools>Caches** to open the **Caches** dialog. The **obj_cache_table** object appears in the display.



2. Click **Add**. The **Add New Cache** dialog opens.

3. Enter a name for the cache and select **Table** from the **Select the type of the cache** drop-down menu. Add other caches as needed.

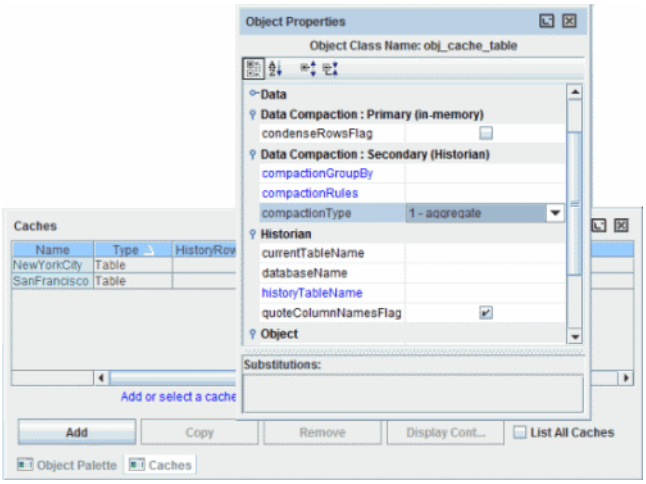
4. Click the **obj_cache_table** object in your display. The **Object Properties** window is populated with properties to configure Data Aggregation.

5. In the **Object Properties** window, under **Data Compaction: Secondary (Historian)**, click the **compactionType** property, then select **Aggregate**.

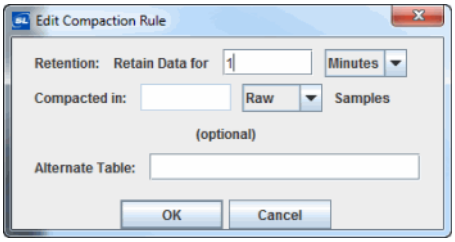
   None – No Data Aggregation nor Table Displacement is performed.

   Aggregate – Performs Data Aggregation.

   Displace – Performs Table Displacement.



6. In the **Object Properties** window double-click the **compactionRules** property to open the **Compaction Rules** dialog, then click **Add** to open the **Edit Compaction Rule** dialog.



7. In the **Edit Compaction Rule** dialog, create a rule by making the following entries.

Note: If you choose to configure Multiple Table Aggregation, this is the Step in which you configure it. For each table, enter its name in the **Alternate Table** field, and configure the Compaction Rule (in the **Retention** and **Compacted** fields), as follows:

Retention: Retain Data for - Specify how long to store data for the current rule.

Compacted in - Specify the time interval at which to store data for the current rule. For Raw data this is the update interval. **Note:** If you select **Raw**, all values are retained and no compaction is performed.

Alternate Table - Use this field to configure Multiple Table Aggregation. For each table, enter a table name and an associated rule. By default, data is aggregated in the table specified by **historyTableName** property. Use the **Alternate Table** field to specify the name of another table.

For example, if we configure the following rules:

Rule 1:

Retain Data for: 5 Minutes

Compacted in: Raw

Rule 2:

Retain Data for: 10 Minutes

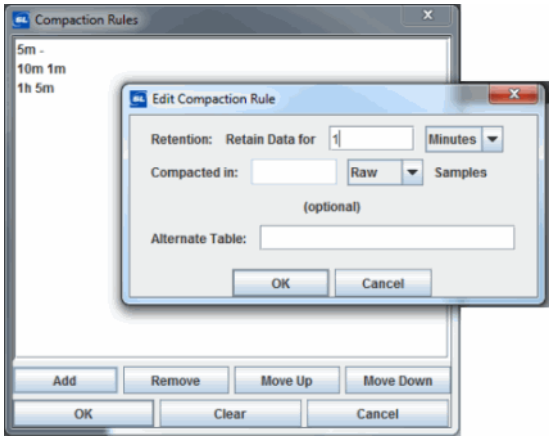
Compacted in: 1 Minute

Rule 3:

Retain Data for: 1 Hour

Compacted in: 5 Minutes

the rules are then displayed in the Compaction Rules dialog as follows:



8. Click **OK** to apply the rule.

9. In the **Object Properties** window, double-click the **compactionGroupBy** property to open the **Compaction GroupBy Columns** dialog, then click **Add** to open the **Edit Compaction GroupBy Column** dialog. The columns in your table are populated in the **Group Column** drop-down menu.

10. In the **Edit Compaction GroupBy Column** dialog, select a column from the **Group Column** drop-down menu, then select the data aggregation method from the by drop-down menu:

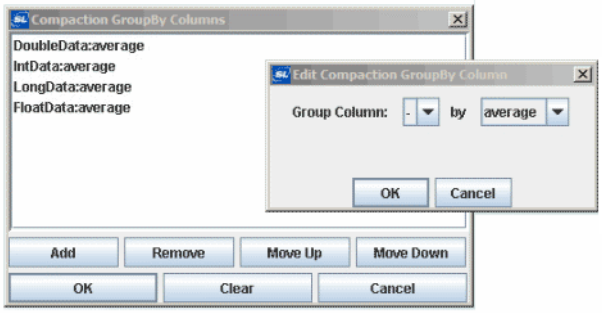
- **sum**: Uses the sum total amount of all data within the specified time period.
- **count**: Uses the total number of data points within the specified time period.

Note: This option is typically not used. count is only accurate for the last aggregation performed and should be used sparingly.

- **average**: Uses the average total of all data within the specified time period.
- **min**: Uses the smallest number of all the data points within the specified time period.
- **max**: Uses the largest number of all the data points within the specified time period.

Note: Typically, the sum or average method is used.

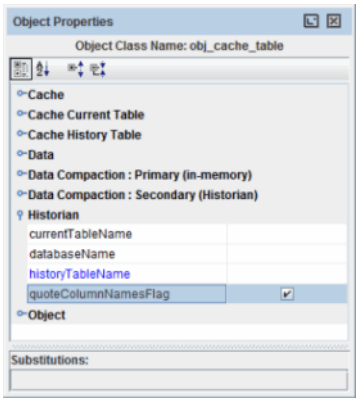
11. Click **OK**. The column and aggregation method is listed in the **Compaction GroupBy Column** dialog.



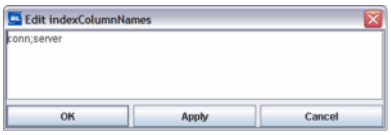
Click **OK** when finished applying the aggregation method to the columns.

12. In the **Object Properties** window, under Historian, click the **quoteColumnNames** property to specify how the column names are processed in the SQL database used for compaction. Choose True (checked) to preserve the column name case. True (checked) is recommended when possible.

Note: This setting is used by Primary Compaction and Secondary Compaction.



13. In the **Object Properties** window, click the **indexColumnNames** property and enter a semicolon (;) separated list of column names to be used as index columns. (See "Table Cache" for details).



14. If you have pre-existing data (table data that was stored before the current historian session) you wish to compact, perform the following. The compaction rules you specified will be applied to the pre-existing data. In a command line, type:

-smoothCompaction

When it is finished compacting the historical data, the Data Aggregation compaction process automatically starts compacting all the data. While smoothing is executing, new (raw) data is also being stored by the Historian.

Note: You may not be licensed to run all RTView features and/or data sources.

---

15. Save the display (**.rtv**) file.

Proceed to "Add a Display (.rtv) File to the Historian Application".

Table Displacement

Use the Table Displacement option if you only need to view recent data, for example, for the last 24 hours. The Table Displacement option removes archival data from a table. Table Displacement supports a single compaction rule. No data compaction occurs, nor do any changes to the data. The removed data is placed into a new, separate table at intervals you define, usually every 24 hours. You can configure Table Displacement to place the removed data into more than one table.

For example, let us say we configure Table Displacement for a table, ProductX, to occur every day at midnight. In our example, we also configure Table Displacement to archive the previous three days of data. This means we will have a total of four tables: the (original or source) ProductX table, plus one table for each of the three additional days (24-hour periods).

At midnight at the end of the first day, the Advanced Historian archives each row in the ProductX table and places the data in a new table. The new table is named ProductX_<date>, where date is the date it is stored. To illustrate this example, we number the tables as they are created in the Advanced Historian process. At the end of the first day we have the following two tables:

#1	**ProductX**	This table is now empty.
#2	**ProductX_2012_1_01_12**	This table contains data from the previous 24 hours.

The ProductX table immediately begins storing data for the next 24 hours. At midnight, the Historian again archives each row in the ProductX table and places the data in a new table, ProductX_<date> (Table #3, below):

#1	**ProductX**	This table is now empty.
#3	**ProductX_2012_1_02_12**	This table contains data from the previous 24 hours.
#2	**ProductX_2012_1_01_12**	This table contains data for the 24 hours prior to Table #3.

The ProductX table, empty again, begins storing data for the next 24 hours. At midnight, the Historian archives each row in the ProductX table and places the data in a new table, ProductX_<date> (Table #4, below):

#1	**ProductX**	This table is now empty.
#4	**ProductX_2012_1_03_12**	This table contains data from the previous 24 hours.

#3	**ProductX_2012_1_02_12**	This table contains data for the 24 hours prior to Table #4.
#2	**ProductX_2012_1_01_12**	This table contains data for the 24 hours prior to Table #3.

The ProductX table, empty again, begins storing data for the next 24 hours. At midnight, the Historian again archives each row in the ProductX table and places the data in a new table, ProductX_<date>. But now three archival tables exist, so the oldest table, Table #2 in this scenario, is deleted.

#1	**ProductX**	This table is now empty.
#5	**ProductX_2012_1_04_12**	This table contains data from the previous 24 hours.
#4	**ProductX_2012_1_03_12**	This table contains data for the 24 hours prior to Table #5.
#3	**ProductX_2012_1_02_12**	This table contains data for the 24 hours prior to Table #4.

The following table describes options for the table name format:

Date part for displacement rule	**Code**	**Table Name**
Year	y	Tablename_2008
Month	M	Tablename_2008_12
Week	W	Tablename_2008_49
Day	D	D Tablename_2008_271
Hour	H	H Tablename_2008_12_10_09
Minute	M	M Tablename_2008_12_10_09_15
Second	S	S Tablename_2008_12_10_09_15_23
"default"		Tablename_2008_12_10_09_15

Configuring Table Displacement

This section provides step-by-step details for configuring the Advanced Historian Table Displacement option for storing archived data. You configure Table Displacement using the **obj_cache_table** object and configuring Cache properties. See "Caches" for more information.

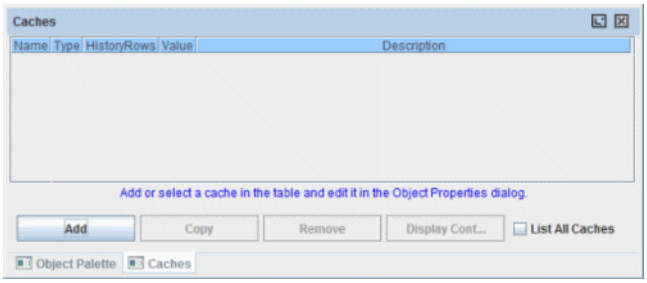
Note: These procedures require a Historian Compaction license key.

At this point you have:

- Attached data to the **obj_cache_table** object using the **valueTable** property.
- Obtained a Historian Compaction license key.

To configure Table Displacement on a cache table:

1. In the Display Builder, select **Tools>Caches** to open the **Caches** dialog. The **obj_cache_table** object appears in the display.



2. Click **Add**. The **Add New Cache** dialog opens.

3. Enter a name for the cache and select **Table** from the **Select the type of the cache** drop-down menu. Add other caches as needed.

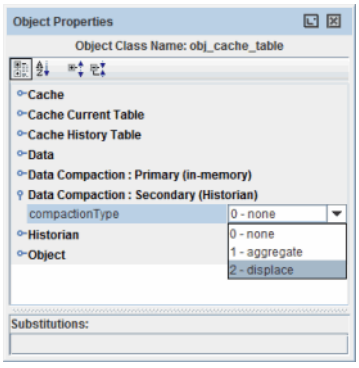
4. Click the **obj_cache_table** object in your display. The **Object Properties** window is populated with properties to configure Table Displacement.

5. In the **Object Properties** window, under Data Compaction: Secondary (Historian), click the **compactionType** property, then select **Displace**.

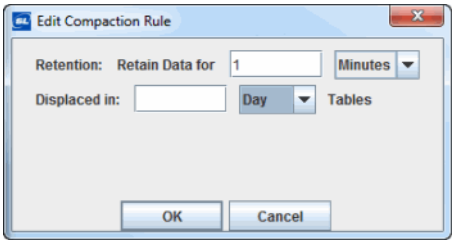
   None – No Data Aggregation nor Table Displacement is performed.

   Aggregate – Performs Data Aggregation.

   Displace – Performs Table Displacement.



6. In the **Object Properties** window double-click the **compactionRules** property to open the **Compaction Rules** dialog, then click **Add** to open the **Edit Compaction Rule** dialog. Table Displacement supports a single compaction rule.



7. In the **Edit Compaction Rule** dialog, create a rule by making the following entries:

   Retention: Retain Data for - Specify how long to store compactions for the current rule.

   Displaced in - Specify the number of tables to retain, and the time period the tables are to store data.

   For example, if you specify:

   Displaced in: 5 Day Tables

   5 tables are created, each storing data for a 24-hour period.

8. Click OK.

Note: Displacement rules are used if the same RTView object is being used to display data that starts again at a predefined interval. Use only a single displacement rule on each table or the rules will conflict.

---

9. Save the display (.rtv) file.

Proceed to "Add a Display (.rtv) File to the Historian Application," next.

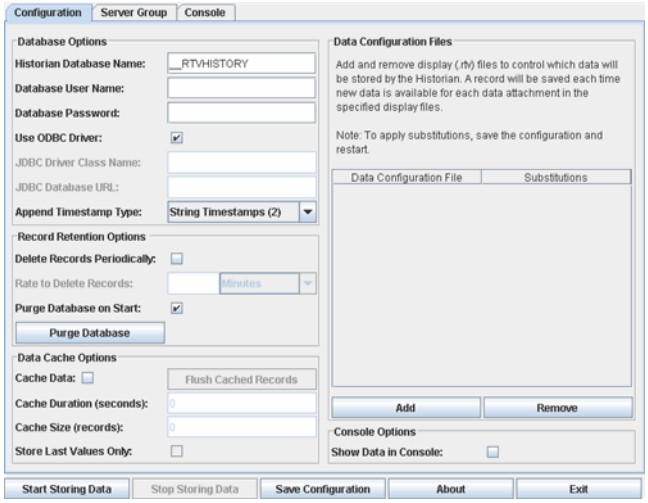
Add a Display (.rtv) File to the Historian Application

1. Start the Historian application to specify the configuration (**.rtv**) file.

   In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window") type: **run_historian**

   The Historian application opens and begins gathering and placing your (now time-stamped) data into tables based on your "Data Configuration File Options".

2. Click **Configuration** to add the configuration (**.rtv**) file.



3. In **Data Configuration Files**, click **Add**, enter the name of the Data Configuration (**.rtv**) file you just created and, optionally, any corresponding substitutions. "Substitutions" allow the same file to be used multiple times by different sets of substitutions.

Note: Each cache that is defined in the file should contain a substitution in the **cacheName** property, since cache names must be unique.

---

4. Click **Apply** to execute. Data Configuration files are added or removed after you click **OK**, **Apply**, or **Save**.

Note: To edit the Data Configuration (**.rtv**) file and apply changes without restarting the Display Builder, select a file from the list and click **Refresh Selection** to reload definition settings. The **Refresh Selection** option is enabled when you select a Data Definition (**.rtv**) file that has been both added and applied.

The Data Configuration (**.rtv**) file is added to the data source and data for these caches is collected. See the "Attach to Cache Data" section for details about attaching to cache data variables.

Note: After a Data Configuration (**.rtv**) file has been added to the data source, data is collected for that cache regardless of whether any currently open display (**.rtv**) files are showing that data.

---

5. Select **Show Data in Console** so that the Historian prints out a line for each record that is being stored in the database.

6. Select Start Storing Data.

7. Select Save Configuration.

8. Verify your configuration by clicking **Console** to view the **Show Data in Console** output.

Your setup is complete. After the Historian has stored data for a period of time, you can build a display using History data. See "Building a Display Using History Data" for more information.

Optionally, you can configure failover for the Historian. For information, see "Configuring Failover on the Historian".

Advanced Historian - Cache Properties

Note: You may not be licensed to run all RTView features and/or data sources.

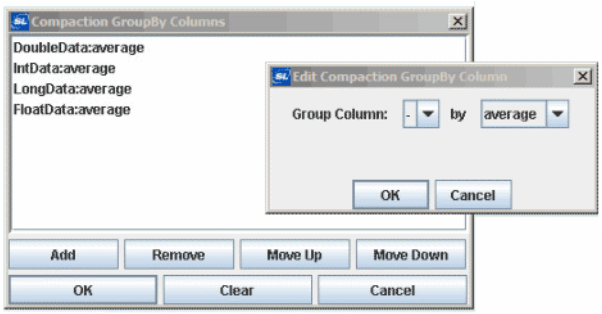
The "Advanced Historian" (when activated) provides the following additional Cache properties (see "Caches") to describe your data management preferences:

compactionType

The Advanced Historian feature allows you to define rules about data compaction via aggregation or displacement.

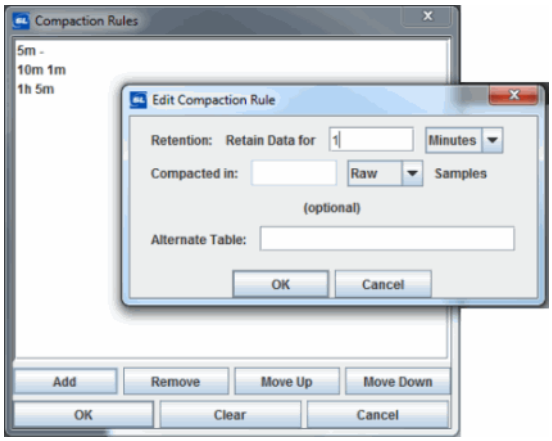
- Aggregate - Specify the granularity of time stamped data and the methods used to perform compaction.
- Displace - Archive data in displaced tables by creating a new table per each defined retention period.

Compaction GroupBy Columns



Property Name	Description
compactionGroupBy	**Note:** This property only applies the selected **compactionType** is **Aggregate**. Specify column and type of compaction to be performed. Double click on the property name and then click **Add** to open the **Edit Compaction** dialog. Once you have added a column, the column name and compaction type will be displayed in the dialog. **Group Column** - Select a column name to compact. **by** - Select a method of compaction from the drop down menu: sum count average min max

Compaction Rules



<table>
<tr><th>Property Name</th><th colspan="2">Description</th></tr>
<tr><td rowspan="3">compactionRules</td><td colspan="2">Specify compaction rules. Double click on the property name and then click Add to open the Edit Compaction Rule dialog. Compaction rules affect data differently depending on the selected compactionType.</td></tr>
<tr><td>Aggregate</td><td>Displace</td></tr>
<tr><td>Once you have added a compaction rule, that rule will be displayed in the dialog as numbers followed by single characters indicating retention period and rate of compacted data samples (e.g. 1d 10m: over a period of 1 day, aggregate data every 10 minutes).

By default, data is aggregated in the table specified by historyTableName. Use the Alternate Table field to enter the name of another table.

Retention - Enter a duration and select the desired time range in which to perform compaction.

Minutes m
Hours h
Days d
Weeks (7 days) w
Months (31 days) M
Years (365 days) y

Compacted in ... - Samples Compaction rate of data samples within the specified retention period. Enter the duration and select the desired time period from the drop down menu.

Note: If you select Raw, all values are retained and no compaction is performed.

Raw -
Second s
Minute m
Hour h
Day d
Week (7 days) w
Month (31 days) M
Year (365 days) y

Alternate Table - Enter an alternate table name in which to compact data. By default, data is compacted in the table specified by historyTableName.</td><td>Once you have added a compaction rule, that rule will be displayed in the dialog as numbers followed by single characters indicating retention period and rate of compacted data samples (e.g. 1d 10m: retain data for 1 day, create a new displaced table every 10 minutes).

Note: Currently, when the selected compactionType is Displace, this property only supports one compaction rule

Displaced tables are created and named according to the specified retention period and rate of compaction. (e.g. historyTableName_2009_06_16_14_35, historyTableName_2009_06_16_14_45, etc.).

Retention - Amount of time displaced tables will be retained before being deleted. Enter the duration and select the desired retention period from the drop down menu (e.g. 1d: retain displaced tables for one day)

Minutes m
Hours h
Days d
Weeks (7 days) w
Months (31 days) M
Years (365 days) y

Compacted in ... - Samples Compaction rate of data samples within the specified retention period. Enter the duration and select the desired time period from the drop down menu. (e.g. 10m: create a new displaced table every 10 minutes).

Note: If you select Raw, all values are retained and no compaction is performed.

Raw -
Second s
Minute m
Hour h
Day d
Week (7 days) w
Month (31 days) M
Year (365 days) y</td></tr>
</table>

CHAPTER 18 Optimizing Performance

This section contains the following:

Overview - Display Sharing

This section describes how to optimize RTView Display Server and Display Servlet performance using Display Sharing.

Display Server and Display Servlet performance can be improved by sharing displays among clients. This is helpful in an environment where many clients typically view the same display simultaneously.

There are "Limitations" on the features that can be used on a shared display, so displays to be shared should be selected carefully.

To configure Display Sharing, you specify the displays to be shared and the substitutions, if any, to be applied when the display is opened, in the **DISPLAYSERVER.ini** file. See "Display Server Configuration" for more information.

This section contains the following:

How It Works

Normally, the Display Server loads a copy of a display for each and every client that requests it. And the Display Servlet normally then generates the Web page content (for example, HTML, images and JavaScript) for each and every client.

When Display Sharing is enabled, the Display Server instead loads a single copy of a display for all of the clients to view. The Display Servlet then generates the Web page content once, stores the content, and reuses that content for all other client requests for the same display. The Display Servlet keeps and uses that content until the time interval specified for the **SharedDisplayRefreshInterval** property in the rtvdisplay.properties file (located in the **servlets\rtvdisplay** directory). By default, the interval is 15 seconds.

Display Sharing is used only for client requests that match both the display name and the substitutions, if any, that are listed in the **DISPLAYSERVER.ini** file. See "Display Server Configuration" for more information.

If the Display Servlet receives a client request for a shared display after the interval has expired, the Display Servlet checks with the Display Server to see if the stored content is "stale". That is, if any data has been applied to the display since the Display Servlet generated the content. If it is not stale the Display Servlet stores the content for another interval. If it is stale, the Display Servlet gets the updated display from the Display Server and regenerates the content. The Display Servlet purges the stored content for shared displays if it has not been accessed within one shared display refresh interval, plus one minute.

The maximum number of shared displays for which the Display Servlet stores content is determined by the **MaxSharedDisplays** property. By default, the value is 10. Best practices dictate that the **MaxSharedDisplays** property be set to a value at least as large as the number of **shared_display** entries in **DISPLAYSERVER.ini**.

The JMX Mbean operation **RTViewDisplayServer.Manager.clearDisplayCache** can be used to clear and reload all shared displays in the Display Server, and to clear all page content for shared displays currently stored in the Display Servlet.

When a client requests a display name that does not match a shared display and substitution entry in the **DISPLAYSERVER.ini** file, the request is processed using the normal process.

Again, the performance benefit of display sharing is obtained only if multiple clients request the same shared display within a single shared display retention interval.

Configuring Display Sharing

To configure Display Sharing, add the name of the display (**.rtv**) file and the substitutions, if any, to be applied when the display is opened, to the **DISPLAYSERVER.ini** file. See "Display Server Configuration" for more information.

To configure a shared display, perform the following steps.

1. Add the **shared_display** property to the **DISPLAYSERVER.ini** file as follows:

   ```
   shared_display <filename> [substitutions]
   ```

   Where:

   <filename> is the name of a display (**.rtv**) file to share.

   [substitutions] is an optional list of substitution **string:value** pairs, separated by spaces. If a substitution value contains spaces it must be enclosed in single quotes.

   For example:

   ```
   shared_display navtop.rtv
   shared_display summary.rtv $region:East
   shared_display summary.rtv $region:West
   shared_display summary.rtv $region:'East and West'
   ```

2. Optionally, add the **SharedDisplayRefreshInterval** property to the **rtvdisplay.properties** file (located in the **servlets\rtvdisplay** directory). See "Display Servlet" for more information. By default, the interval is 15 seconds. (This step is only required if you need to change the default setting.) For example:

   ```
   SharedDisplayRefreshInterval=15
   ```

3. Optionally, specify the maximum number of shared displays for which the Display Servlet stores content by adding the **MaxSharedDisplays** property to the **rtvdisplay.properties** file. See "Display Servlet" for more information. By default, the value is **10**. (This step is only required if you need to change the default setting.) For example:

```
MaxSharedDisplays=10
```

Note: Best practices dictate that the **MaxSharedDisplays** property be set to a value at least as large as the number of **shared_display** entries in the **DISPLAYSERVER.ini** file.

Monitoring Display Sharing

To view performance metrics about requests for shared displays and status information from the Display Servlet, open the **servletstatus.jsp** page in a browser: **http://localhost:8068/rtvdisplay/servletstatus.jsp**.

The **servletstatus.jsp** page contains the following information:

Value	Description
Display Server	The Display Server hostname and port number.
Backup Server	The hostname and port number of the backup Display Server, if one exists.
Connection Pool Size	The value of the **DisplayServerConnectionPool** property in the rtvdisplay.properties file. The **DisplayServerConnectionPool** property specifies the maximum number of simultaneous connections the Display Servlet makes to the Display Server and, consequently, determines the maximum number of display requests the Display Server processes simultaneously.
Timeout	The amount of time, in seconds, for a query to timeout.
Refresh	The value of the **DefaultRefreshInterval** property in the rtvdisplay.properties file. The **DefaultRefreshInterval** property specifies the refresh rate of the Thin Client.
Login	Specifies whether login is enabled.
Request Count	The total number of client display requests received.
Connection	The current connection from the Display Servlet to the Display Server.
Started	The time that the Display Servlet started.
Shared Display Refresh	The value of the **SharedDisplayRefreshInterval** property, in seconds.
Shared Display Hits/ Tries	The amount of client requests for a page that is satisfied by Display Sharing, in percent (%). That is, the number of client requests satisfied by stored content for a shared display divided by the total number of requests for a display. A high amount indicates that display sharing is effective, a low amount indicates that it is not effective for the requests being received.
Shared Display Updates	The number of times the Display Servlet has updated shared display content.
Shared Displays	The current number of shared displays divided by the value of the **MaxSharedDisplays** property. This is followed by a scrolled list of the shared displays currently loaded in the Display Servlet.

The following is an example of data:

```
display server: localhost: 3279
backup server: null
connection pool size: 10
timeout: 15 sec
refresh: 15 sec
login: on

request count: 437
connection: localhost:3279
started: 02/04/11 14:14:33.17
shared display refresh:: 15 sec
shared display hits/tries: 222/437 (50%)
shared display updates: 88
shared displays: 3 / 10
      blank
      nav1
      rbt1+$plants:*
```

See "Display Servlet" for more information.

Thin Client

In the Thin Client, you can determine whether a shared display is being viewed. Right-click in the display and select **Status** from the popup menu, then check the Panel ID string shown in the Thin Client Status window. For a shared display the Panel ID string begins with preload.

Limitations

- The following features should not be used on shared displays, because these features can change the state of a display for each user.
  - A **resizeMode** of Layout or Scale.
  - Scrolling on the following objects: **obj_trendgraph02**, **obj_bargraph**, **obj_objectgrid**.
  - Paging mode on **obj_table02**.
  - User-specific substitutions (for example, **$rtvuser**).
  - Role-specific substitutions (for example, **$rtvrole**).
- The substitutions in a client request for a shared display must exactly match those specified in the entry in **DISPLAYSERVER.ini**, including the order of the substitutions, otherwise the request is processed as a non-shared display.
- All shared displays share the same set of global variables and functions. Therefore, if a user changes the value of a shared display global variable, all users viewing any shared display will see that value change--including users on separate machines. Whereas, for all non-shared displays, the Display Server uses a separate set of global variables that it maintains for each browser session. Therefore, if a user changes the value of a global variable no other non-shared displays are affected.
- A drilldown to a shared display behaves as though the **Remove Existing Substitutions** box was checked in the **Drill Down Properties** dialog. This is necessary so that substitutions in the source display are not included in the drilldown substitutions, which might prevent them from matching the shared display substitutions.

CHAPTER 19 Troubleshooting & Monitoring

This section contains the following:

Overview

This section describes the troubleshooting tools available for RTView. If an RTView application is not performing as expected, investigate the following most critical and commonly encountered issues:

Display Server: Check the Display Server CPU usage. This is the most typical issue for the Display Server as the application processes many user requests.

Data Server and Historian: Check both the amount of available memory and memory used on the Data Server and the Historian. The amount of memory used grows as these applications run, so the amount of memory originally designated might no longer be adequate. If the amount of memory used is close to the amount designated, consider making more memory available.

Note: When troubleshooting, it is helpful to have performance metrics for both before and after a failure.

Troubleshooting Tools

The following tools are available for troubleshooting your RTView system.

"RTView Monitor" - View status, performance metrics, and memory allocation and utilization for all licensed RTView Data Servers, Display Servers, Historian applications and Tomcat applications in your system. Use the RTView Monitor to determine how an RTView application is utilizing its memory.

"Java Tools" - SL recommends that you install JDK v1.6 on systems that run RTView applications. Use tools such as JConsole, Jmap, and JPS to gather more complete information for troubleshooting your RTView system.

"Log Files" - View RTView application output and error streams while installing an RTView application or while operating in the production environment.

RTViewDs - Query the health of your data source data adapter and debug the adapter. Use this debugging tool if you encounter problems accessing your data. RTViewDs provides connectivity and response time information to help you determine whether the problem is with accessing your data source, or if the data source is slow in responding. The following data sources support RTViewDs:

- **JMX:** Provides database connection information. See "RTViewDs" in the "Attach to JMX Data" section for more information.
- **SNMP:** Provides metrics for SNMP requests completed and traps received per connection. See "RTViewDs" in the "Attach to SNMP Data" section for more information.
- **SQL:** Provides two tables containing metrics about SQL database connections and queries. See "RTViewDs" in the "Attach to SQL Data" section for more information.
- **XML:** Provides information about the listeners currently active in the XML data source. Use this information to troubleshoot Data Server clients, as the XML data source handles listeners for ALL RTView data attachments that are redirected to a Data Server. See "RTViewDs" in the "Attach to XML Data" section for more information.
- **TIBCO Rendezvous:** Provides RTViewDs subject fields containing TIBCO Rendezvous monitoring information. See "TIBCO Rendezvous RTViewDs Fields" in the "Attach to TIBCO Rendezvous Data" section for more information.
- **TIBCO Hawk:** Provides agent information, such as agent connection and status information, alert information and IP address. See "RTViewDs" in the "Attach to TIBCO Hawk Data" section for more information.
- **Caches:** Provides cache configuration and cache table size at runtime information for each table maintained by the cache data source. See "RTViewDs" in the "Attach to Cache Data" section for more information.
- **RTVAgent:** Provides an RTViewDs.agents table containing a list of currently connected agents. See "RTViewDs" in the "Attach to RTVAgent Data" section for more information.

JMX MBeans - The Data Server is instrumented with JMX, including the following MBeans:

- "RTView:name=Troubleshooting" MBean: Use this MBean to troubleshoot an unresponsive server.
- "RTViewDataServer:name=Manager" MBean: Use this MBean to manage and monitor clients and application settings.

Troubleshooting Steps

If an RTView Server has become unresponsive or slow, this might be due to heavy CPU load or memory usage on the host system. Perform the following steps:

1. Observe overall system performance on the host machine by checking CPU and memory usage. You can use:
   - Windows Task Manager
   - a Linux utility, such as top
2. Observe RTView Application performance on the host machine. Identify the RTView Application consuming the most resources (the largest CPU load or memory usage), then

determine what action the application is specifically performing while consuming them. You can use:

- "RTView Monitor"
- "Java Tools"

Note the following:

- Overall CPU usage. Is it consistently high? If so, which applications are the biggest CPU users?
- Available memory. Which applications are the biggest memory users?
- Compare system performance shortly after the RTView Server is started to system performance at the onset of the problem. Has the overall CPU load increased noticeably? Has the available memory decreased significantly?

Note: Heap usage might go up and down between garbage collection intervals but, overall, the heap size should be stable. An exception to this is an RTView server that runs the RTVIew cache data source. In that case, heap usage grows until each cache history table reaches its maximum row counts or the value specified for the **timeSpan** property. See "Caches" for more information.

---

3. Obtain RTView application logs. See "Log Files" for more information.
4. If the issue involves the RTView Display Server, obtain Tomcat logs.

   Note: Tomcat logs can also be useful if the RTView Data Servlet is used to provide HTTP access to the Data Server.

5. Investigate using the "RTView:name=Troubleshooting" **MBean**.
6. At this point you can contact SL Support (support@sl.com) or continue to the next step.
7. If an RTView server is still running, perform multiple application "Thread Dumps".
8. If an RTView server is still running and memory consumption is an issue, perform multiple Jmap log reports. See "Jmap Utility" for more information.
9. Provide all information to SL support and, if possible, wait to cycle servers until a review has been made.

RTView Monitor

The RTView Monitor enables you to monitor the performance, memory allocation, and utilization for all licensed RTView Data Servers, Display Servers, Historian applications, and Tomcat applications in your system. Use the RTView Monitor to determine how an RTView application is utilizing its memory.

The RTView Monitor is a demo application (located in the RTView **demos/rtvmonitor** directory) that you can customize for your own system, including adding alerts. To start the RTView Monitor demo, see "Starting the RTView Monitor". To customize the RTView Monitor, see "Configuring the RTView Monitor".

The RTView Monitor uses the RTView JMX Data Adapter to display all connectivity information available via JMX, as well as all detailed operational and management features currently available via JMX.

Monitoring Tips:

- If your RTView system is unexpectedly slow, use the RTView Monitor to see how much memory is allocated and determine how it is being utilized. If the amount of memory used is close to the amount designated, consider making more memory available for the application.
- When viewing metrics for application usage of JVM heap and non-heap memory, note that heap usage might go up and down between garbage collection intervals but, overall, the heap size should be stable. See "Using RTView Monitor" for more information. An exception to this is an RTView server that runs the RTVIew cache data source. In that case, heap usage grows until each cache history table reaches its maximum row counts or timeSpan. See "Caches" for more information.
- Obtain performance metrics for both before and after a failure.
- If an RTView application is not performing as expected, investigate the following most critical and commonly encountered issues:

  Display Server: Check CPU usage. This is the most typical issue for the Display Server as the application processes many user requests.

  Data Server and Historian: Check both the amount of available memory and memory used. The amount of memory used grows as these applications run, so the amount of memory originally designated might no longer be adequate. If the amount of memory used is close to the amount designated, consider making more memory available.

This section contains the following:

Configuring the RTView Monitor

This section provides step-by-step instructions on how to setup your servers to be monitored by the RTView Monitor.

Basic Steps To Configure the RTView Monitor

Perform the following steps in the order given to configure the RTView Monitor. Steps 2 and 3 are optional.

1. Setup your servers. Configure your servers to be monitored by the RTView Monitor, and to expose their metrics through JMX. See "Setup your servers for RTView Monitor" for more information.

2. Configure Application Groups (optionally). Application groups allow you to organize your application servers into categories for ease of access. You can create a group name and assign one or more servers to that group. Application servers can belong to multiple groups. See "Configure Application Groups for the RTView Monitor" for more information.

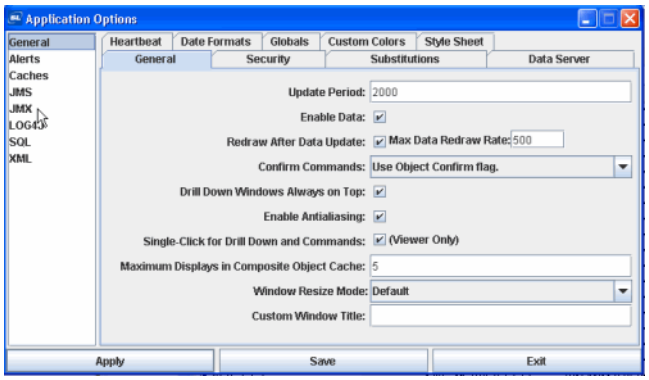
3. "Modify the default settings for the RTView Monitor" (optionally). You can edit default settings, such as the update period, in the **OPTIONS.ini** file.

4. Start the RTView Monitor. See "Starting the RTView Monitor" for information.

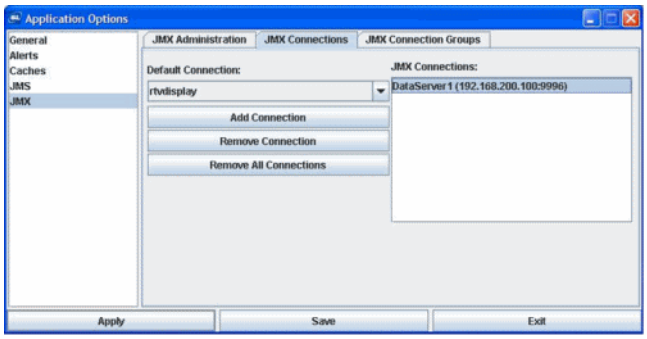
5. Verify your setup. See "Verifying Your Setup" for information.

Setup your servers for RTView Monitor

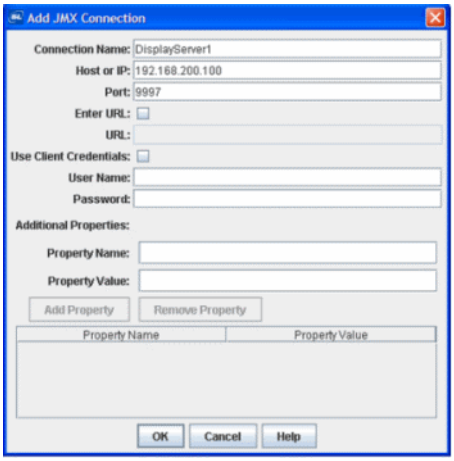
1. To set up your RTView servers for monitoring, execute **run_configutil.bat**, located in the **/bin** directory. The **Application Options** dialog opens.



2. Select **JMX** from the list of options in the left pane.



3. Select the **JMX Connections** tab, then click **Add Connection** to open the **Add JMX Connection** dialog.



4. In the **Add JMX Connection** dialog, make the following entries:

   Connection Name: Enter the connection name as you want it to appear in the RTView Monitor.

   Host or IP: Enter the server host or IP address.

   Port: Enter the JMX port number assigned to that server.

   Note: Use the additional fields only if needed for your specific configuration.

   The settings entered in the **Add JMX Connection** dialog are stored in the local **JMXOPTIONS.ini** file, located in the **demos/rtvmonitor** directory (the **JMXOPTIONS.ini** resides in the same directory as the RTView Monitor application).

   You can also use a text editor to edit the **JMXOPTIONS.ini** file, using the following syntax:

   ```
   jmxconn <servername> <IPaddress> <port> URL:- - - 'false'
   ```

   Where:

   <servername> is a name of your choosing for the application.

   <IPaddress> is the IP address of the host on which the application runs.

   <port> is the port number of the host on which the application runs. The port number must be the same port number used when launching the respective server.

   URL:- - - 'false'

   For example:

   ```
   jmxconn MyDataserver localhost 9952   URL:- - - ‘false’
   jmxconn MyDisplayServer localhost 9953 URL: - - - ‘false’
   jmxconn slhost1_OCM_HISTORIAN 192.168.200.101 9913   URL:- - - ‘false’
   ```

5. Click **OK** when you are finished.

6. To configure your servers to expose their metrics through JMX, add the following parameter to the command line string of each server:

-jmxport:xxxxx

Where **xxxxx** is the same JMX port number used to configure the "RTView Monitor".

Note: SL recommends port numbers above 10000 to avoid conflicts with existing SL server application port numbers.

To (optionally) configure Application Groups for the RTView Monitor, continue to the next steps (below). To (optionally) modify the RTView Monitor default settings, see "Modify the default settings for the RTView Monitor". Otherwise, proceed to "Starting the RTView Monitor".

Configure Application Groups for the RTView Monitor

1. Edit the **appgroups.xml** file, located in the **demos/rtvmonitor** directory, using the following format:

```
<tr>
<td>ApplicationGroupName</td>
<td>NameOfAnApplication</td>
</tr>
```

Where each row (**<tr>**) contains two columns (**<td>**). The first column contains the name of the Application Group, and the second column contains the name of an application to be contained in that group.

Application groups allow you to organize your application servers into categories for ease of access. You can assign one or more servers to a group. Application servers can belong to multiple groups.

In the following example, an Application Group named **DataServers** contains two applications, **slhost1_OCM_DATASERVER** and **slhost3_OCM_DATASERVER**:

```
<?xml version="1.0"?>
 <table key="app_groups">
            <tc name="GROUPNAME"
                        type="string"
                        index="true"/>
            <tc name="APPNAME"
                        type="string"
                        index="false"/>
            <tr>
                        <td>DataServers</td>
                        <td>slhost1_OCM_DATASERVER</td>
            </tr>
            <tr>
                        <td>DataServers</td>
                        <td>slhost3_OCM_DATASERVER</td>
            </tr>
 </table>
 </table>
 </dataset>
```

The XML code preceding the first **<tr>** and following the last **</tr>** must not be altered.

To (optionally) modify the RTView Monitor default settings, see "Modify the default settings for the RTView Monitor," next. Otherwise, proceed to "Starting the RTView Monitor".

Modify the default settings for the RTView Monitor

Optionally, you can edit the default performance settings for the RTView Monitor, such as the update period, in the **OPTIONS.ini** file. The **OPTIONS.ini** file settings override settings made in the **Application Options** dialog.

Proceed to "Starting the RTView Monitor," next.

Starting the RTView Monitor

This section provides step-by-step instructions on how to start the RTView Monitor in the "Display Viewer" and the "Thin Client".

Display Viewer

To start the RTView Monitor:

1. In an initialized command prompt (see "Initializing a Command Prompt or Terminal Window"), navigate to the **core/demos/rtvmonitor** directory and type:

   run_viewer

   The RTView Monitor opens.

2. Verify that the **Connected** check box in the table is checked for the applications you are running. If an application that should be running is not checked, do the following:
   - Verify that the non-connected application is running.
   - Verify that it was started with the **–jmxport:xxxx** parameter.
   - Verify that the same application port number is used for the **–jmxport:xxxx** parameter and the **JMXOPTIONS.ini** file entry. If they do not match, edit the **JMXOPTIONS.ini** entry and restart the RTView Monitor.

Thin Client

To start the RTView Monitor:

1. In a Web browser, navigate to:

   http://localhost:xxxx/rtvmonitor/

   where **xxxx** is the Web server port number.

   The RTView Monitor opens.

2. Verify that the **Connected** check box in the table is checked for the application/s you are running. If an application that should be running is not checked, do the following:
   - Verify that the non-connected application is running.
   - Verify that it was started with the **–jmxport:xxxx** parameter.
   - Verify that the same application port number is used for the **–jmxport:xxxx** parameter and the **JMXOPTIONS.ini** file entry. If they do not match, edit the **JMXOPTIONS.ini** entry and restart the RTView Monitor.

Proceed to "Verifying Your Setup," next.

Verifying Your Setup

Perform the following steps to verify the RTView Monitor is setup properly.

To verify your RTView Monitor setup:

1. Verify that new application servers appear in the RTView Monitor:

   Display Viewer

   - Open the **JMXOPTIONS.ini** file in a text editor (located in the **demos/rtvmonitor** directory) and add application servers using the following syntax: **jmxconn DisplayServer1 192.168.200.100 9997 URL:- - - 'false'**
   - Start the RTView Monitor.
   - Verify that the application servers you added appear in the Host and Application lists.

   Thin Client

   - Open the **JMXOPTIONS.ini** file in a text editor (located in the **demos/rtvmonitor** directory) and add application servers using the following syntax: **jmxconn DisplayServer1 192.168.200.100 9997 URL:- - - 'false'**
   - Start (or stop and the restart) the Display Server.
   - Verify that the application servers you added appear in the Host and Application lists.

2. Verify that new Application Groups appear in the RTView Monitor:

   Display Viewer

   - Create a new Application Group in the appgroups.xml file. See "Configure Application Groups for the RTView Monitor" for more information.
   - Save the appgroups.xml file and start (or restart) the RTView Monitor.
   - Verify that the Application Group you added appears in the Host and Application lists.

   Thin Client

   - Create a new Application Group in the appgroups.xml file. See "Configure Application Groups for the RTView Monitor" for more information.
   - Save the appgroups.xml file.
   - Verify that the Application Group you added appears in the Host and Application lists.

Note: Any change to the Thin Client JMXOPTIONS.ini file requires that the Display Server be restarted (not the application). Changes to the appgroups.xml file does not require anything to be restarted as it is read with each refresh, and updated automatically.

---

3. Verify RTView Monitor GUI functionality:
   - Click the **Show Only Connected** check box so that only connected application servers are displayed. Clear the check box and verify that both connected and non-connected servers appear.
   - Click entries in the **Applications** list and verify that the detail pane is populated with associated detail data.

4. Verify detail data displays properly by clicking:
   - **Memory**

a) Verify that fields are populated.

b) Click the **Time Rage** drop-down and verify that both graphs reflect the selected time range.

- **JVM**

  Verify that fields are populated and that the data is correct.

- **App > Data Server**

  a) Verify that fields are populated and that the data is correct.

  b) Verify that **Start Serving Data** and **Stop Serving Data** work as expected.

  c) Verify that **Start Serving Data** and **Stop Serving Data** reflect their current state: enabled or disabled.

  d) Verify that **Serving Data** reflects its current state.

  e) Verify that **Current Number of Clients** matches the number of client rows displayed in the Client Data table.

- **App > Display Server**

  a) Verify that fields are populated and that the data is correct.

  b) Verify that Clear Display Cache does clear the cache.

- **App > Historian**

  a) Verify that fields are populated and that the data is correct.

  b) Verify that **Start Storing Data** and **Stop Storing Data** work as expected.

  c) Verify that **Start Storing Data** and **Stop Storing Data** reflect their current state: enabled or disabled.

  d) Verify that **Storing Data** reflects its current status.

  e) Verify that **Purge Stored Data** does purge the stored data.

  f) Verify that the **Primary Server** indicator reflects its current status.

- **App > Tomcat**

  Verify that fields are populated and that the data is correct.

- **App > (none)**

  Verify that this panel only appears when App is selected and there is either no application server selected, or the server is not currently connected.

Your RTView Monitor setup is complete. For details about using the RTView Monitor, see "Using RTView Monitor" (next).

Using RTView Monitor

The RTView Monitor main page lists all of the RTView Data Servers, Display Servers, Historian applications and Tomcat applications in your system. The RTView Monitor provides information about the current status of RTView applications. Hosts available are specified in the **JMXOPTIONS.ini** file. See "Setup your servers for RTView Monitor" for more information.

Select a specific host or All Hosts, then select a specific group or All Groups, to populate the RTView Applications table. Select an application from the RTView Application table to populate the lower pane of the screen. Choose one of the following to drill down to detailed information for the application:

- **Memory** - View metrics for JVM heap and non-heap memory usage by the application. Note that heap usage might go up and down between garbage collection intervals but, overall, the heap size should be stable. An exception to this is an RTView server that runs the RTVIew cache data source. In that case, heap usage grows until each cache history table reaches its maximum row counts or timeSpan. See the table below for information.
- **JVM** - Monitor Java Virtual Machine memory usage, get JVM system information, application performance metrics, and input arguments for the application. See "JVM Details" for information.
- **App** - Monitor application load and client connections and manage the Data Server, Display Server, Historian or Tomcat applications. See "App Details" for information.
- **RTView** - View performance metrics for functions and Web applications. See "RTView Details" for information.

Field Name	Description	
Host	Select a host. This list is populated by the entries int he **JMXOPTIONS.ini** file. For details, see "Configuring the RTView Monitor".	
Group	Select a group of applications or All Groups. Groups are available if they are created in the **appgroups.xml** file. For details, see "Configure Application Groups for the RTView Monitor".	
Show Only Connected	When checked, excludes non-connected servers from the list in the RTView Applications table.	
RTView Applications	The table lists all RTView applications for the selected group/s. This list is populated by the entries in the **JMXOPTIONS.ini** file.	
	Application	The host name on which the application resides and the name of the application.
	Group	The name of the Group, if one was created, in which the application resides.

Connected Indicates that the application is connected when checked.

Host The IP address of the host on which the application resides.

Port The port number on the host that the application uses.

Memory Select to view details about JVM memory usage for an application. Fields are populated by the selection made in the RTView Applications table.

Time Range Select the time range for the trend charts range varying from 2 Minutes to Last 7 Days, or display All Data.

Heap Displays data pertaining to JVM heap memory allocation. That is, the allocation of memory storage for use by an application at runtime. These fields are populated by the selection made in the RTView Applications table.

Further JVM details are available in the "JVM Details" display.

Note: Heap usage might go up and down between garbage collection intervals but, overall, the heap size should be stable. An exception to this is an RTView server that runs the RTVIew cache data source. In that case, heap usage grows until each cache history table reaches its maximum row counts or timeSpan. See "Caches" for information.

Maximum The maximum amount of memory used for memory management by the application in the time range specified. This value may change or be undefined.

Note: A memory allocation can fail if the JVM attempts to set the Used memory allocation to a value greater than the Committed memory allocation, even if the amount for Used memory is less than or equal to the Maximum memory allocation (for example, when the system is low on virtual memory).

Committed The amount of memory guaranteed to be available for use by the JVM. The amount of committed memory can be a fixed or variable size. If set to be a variable size, the amount of committed memory can change over time, as the JVM may release memory to the system. This means that the amount allocated for Committed memory could be less than the amount initially allocated. Committed memory will always be greater than or equal to the amount allocated for Used memory.

Used The amount of memory currently used by the application. Memory used includes the memory occupied by all objects including both reachable and unreachable objects.

Heap Trend Graph Displays data pertaining to JVM allocation of memory storage for use by an application at runtime. The graph is populated by the selection made in the RTView Applications table. The memory pools available for display are determined by the JVM being used.

Further JVM Details are available in the "JVM Details" display.

Eden Space Traces the amount of memory used by the JVM eden pool in the time range specified.

Eden refers to the JVM eden pool, which is used to initially allocate memory for most objects.

Old Gen Traces the amount of memory used by objects that have survived some number of young generation collections, as well as some large objects that may be allocated directly in the old generation.

	PS Survivor Space	Traces the amount of memory used by the JVM survivor pool in the time range specified. The JVM survivor pool holds objects that survive the eden space garbage collection.
	Total Heap Memory Used	Traces the total amount of heap memory used.
Non-Heap		Displays data pertaining to JVM non-heap memory allocation. That is, the allocation of memory storage for the method area shared by all threads, and the memory required for JVM internal processing. The graph is populated by the selection made in the RTView Applications table. Further JVM Details are available in the "JVM Details" display.
	Maximum	The maximum amount of memory used for JVM non-heap memory management by the application in the time range specified.
	Committed	The amount of memory guaranteed to be available for use by JVM non-heap memory management. The amount of committed memory can be a fixed or variable size. If set to be a variable size, it can change over time, as the JVM may release memory to the system. This means that the amount allocated for Committed memory could be less than the amount initially allocated. Committed memory will always be greater than or equal to the amount allocated for Used memory.
	Used	The amount of memory currently used by the application. Memory used includes the memory occupied by all objects including both reachable and unreachable objects.
Non-Heap Trend Graph		Displays data pertaining to JVM non-heap memory allocation. The graph is populated by the selection made in the RTView Applications table. The memory pools available for display are determined by the JVM being used. Further JVM details are available in the "JVM Details" display.
	PS Perm Gen	Traces the amount of memory used by the pool containing reflective data of the virtual machine, such as class and method objects. With Java virtual machines that use class data sharing, this generation is divided into read-only and read-write areas.
	Total Used	Traces the total amount of memory used.

JVM	Select to view JVM details. Fields are populated by the selection made in the RTView Applications table. See "JVM Details", below.
App	Select to view application details. Fields are populated by the selection made in the RTView Applications table. See "App Details", below.
RTView	Select to view RTView details. Fields are populated by the selection made in the RTView Applications table. See "RTView Details", below.

JVM Details

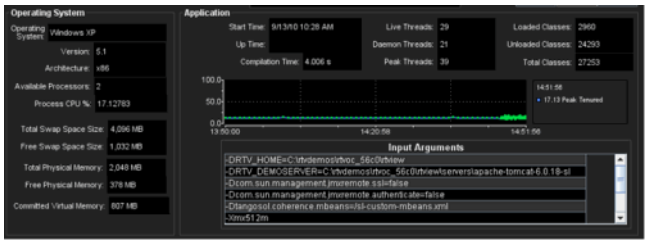
Use the **JVM Details** page to monitor Java Virtual Machine memory usage, get JVM system information, application performance metrics, and input arguments for the application. Verify whether the memory usage has reached a plateau. Or, if usage is getting close to the limit, determine whether to allocate more memory.

To view, select **JVM**. Fields are populated by the selection made in the **RTView Applications** table.

JDK 5.0

Some data is only available for JDK v5.0+. JDK v5.0+ extends the Operating System MBean to include the following operating system resource information:

- CPU processing time
- total amount of memory
- total amount of free physical memory
- total amount of committed virtual memory (virtual memory guaranteed to be available to the running process)
- total amount of swap space
- total amount of free swap space



Field Name	Description	
Operating System	Displays data pertaining to The operating system running on the host on which the JVM resides. Fields are populated by the selection made in the RTView Applications table.	
	Operating System	The name of the operating system running on the host on which the JVM resides.
	Version	The operating system version.
	Architecture	The ISA used by the processor.
	Available Processors	The total number of processors available to the JVM.
	Process CPU %	The average amount of CPU usage by the JVM, in percent.
	Total Swap Space Size	The total amount of swap space, in megabytes, allocated for the JVM (visible with JDK v5.0+).
	Free Swap Space Size	The total amount of currently free swap space, in megabytes, available to the JVM (visible with JDK v5.0+).

	Total Physical Memory	The total amount of physical memory, in megabytes, allocated for the JVM (visible with JDK v5.0+).
	Free Physical Memory	The total amount of currently free physical memory, in megabytes, available to the JVM.
	Committed Virtual Memory	The total amount of virtual memory guaranteed to be available to the running process, in megabytes, allocated for the JVM (visible with JDK v5.0+).
Application	Displays JVM data for the application selected in the RTView Applications table. Start Time The date and time that the application started running.	
	Up Time	The amount of time the application has been running, in the following format: **0d 00:00** **<days>d <hours>:<minutes>** For example: **30d 22:30**
	Compilation Time	The approximate accumulated elapsed time (in milliseconds) spent in compilation. If multiple threads are used for compilation, then this value is a summation of the approximate time that each thread spent in compilation.
	Live Threads	The total number of live threads.
	Daemon Threads	The total number of live daemon threads.
	Peak Threads	The total number of peak live threads since the Java virtual machine started or the peak was reset.
	Loaded Classes	The total number of classes that are currently loaded in the JVM.
	Unloaded Classes	The total number of classes unloaded since the JVM started executing.
	Total Classes	The total number of loaded and unloaded classes since the JVM started execution.
Trend Graph	Traces the amount of memory used by the specific Java space (server or client) being published at the time. For example: Peak Tenured: Traces the amount of memory used by tenured JVM objects in the time range specified. Tenured refers to JVM objects contained in a pool that holds objects that have avoided garbage collection and reside in the survivor space. Peak tenured refers to the maximum value of the tenured memory over a specified period of time.	
Input Arguments	The JVM arguments in the **RuntimeMXBean InputArguments** attribute.	

App Details

Use the **App Details** page for Data Server, Display Server, Historian and Tomcat applications to monitor processing load, the number of client connections, and determine whether the number of client connections or the load is getting close to the limit (and subsequently whether these need to be reduced).

Fields are populated by the type of application selected in the RTView Applications table.

To view, select **App** from the **RTView Monitor** main page.

- "Data Server" - Monitor Data Server application load, client connections and start or stop the application.
- "Display Server" - Displays information for Display Servers, such as the number of active displays, functions, Web application load, and client connections. You can also clear the display cache.
- "Historian" - Displays the caches that are archived by the Historian application. Also included are any substitution variables associated with the history cache configuration file, as well as the history cache status.
- "Tomcat" - Monitor the performance of Tomcat application sessions and get Tomcat hosting and connection details.
- "RTView Details" - Monitor performance between the RTView application and the Data Servers, and monitor how long it takes for an RTView application to execute functions.

Data Server

Monitor Data Server application load, the amount of data sent by the Data Server and connected clients, and start or stop the application.

To view Data Server application select **App**, then select a Data Server application from the RTView Applications table.

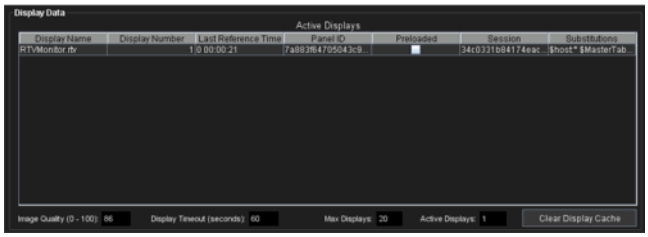
Field Name	Description	
Client Data	Displays metrics for data being sent to clients connected to the application selected in the RTView Applications table.	
	Host	The host machine for the client.
	Address	The client IP address.
	Client ID	The client ID.
	Duration	The length of the client-application session, with the following format: **00:00:00** **hours:minutes:seconds** For example: **24:00:00**
	Last Data Sent	The amount of data, in kilobytes, last sent to the client.
	Total Data Sent	The total amount of data, in kilobytes, sent to the client since the session started.

Current Number of Clients	The total number of clients currently connected to the application.
Serving Data	When green, indicates that the application is currently serving data to clients. When red, indicates that the application is not currently serving data to clients.
Start Serving Data	Click to start or resume serving data.
Stop Serving Data	Click to stop serving data.

Display Server

View the number of active displays, functions, Web application load, and client connections on the Display Server. You can also clear the display cache.

To view Display Server application select **App**, then select a Display Server application from the **RTView Applications** table.



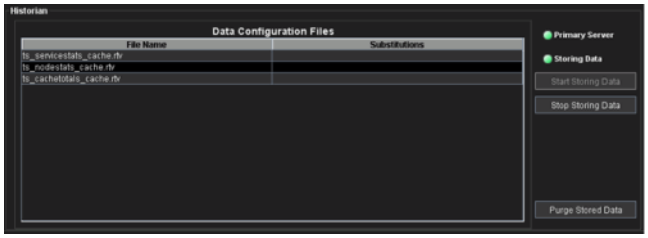
Field Name	Description	
Display Data	Fields are populated by the selection made in the RTView Applications table.	
	Active	Displays Lists the displays currently operational for the application selected in the RTView Applications table. Display Name The display (**.rtv**) file name.
	Display Number	The display number is a value that ranges from 1 to N, where N is the number of displays currently in the cache. This number is not tied to a particular display, and can change as displays are automatically removed from the cache.
	Last Reference Time	The amount of time that has elapsed since the display was last requested by a client.
	Panel ID	A unique string identifier assigned to each panel. The Display Server loads each display requested by each client into a panel. This ID can be useful in troubleshooting.
	Preloaded	When checked, indicates that the display (**.rtv**) file is configured in the **DISPLAYSERVER.ini** file to be preloaded. The **history_config** option is used to configure display preloading. Preloading a display makes data immediately available. Preloaded displays are not unloaded unless the Display Server is restarted or the display cache is cleared via JMX. This option can be used multiple times to specify multiple displays to preload. See "Display Server Configuration" for more information.
	Session	The browser session ID for the client that requested the display.

	Substitutions	Lists the substitutions used for the display which are specified in the **DISPLAYSERVER.ini** file. See "Display Server Configuration" for more information.
Image Quality (0 - 100)	A value between 0 and 100, set in the **DISPLAYSERVER.ini** file, which controls the quality of the generated images. If the value is 100, the Display Server outputs the highest quality image with the lowest compression. If the value is 0, the Display Server outputs the lowest quality image using the highest compression. The default is **75**.	
Display Timeout (seconds)	The amount of time, in seconds, that a display can be kept in memory after the Display Servlet has stopped requesting it. This value is set in the **DISPLAYSERVER.ini** file. Default is **60 seconds** (to allow faster load time when switching between displays).	
Max Displays	The number of displays kept in memory. This value is set in the **DISPLAYSERVER.ini** file. Default is **20** (to optimize memory used by the Display Server).	
Active Displays	The total number of displays currently being viewed by a user.	
Clear Display Cache	Click to unload and reload the preloaded (shared) displays. Use this when a shared display is modified while the Display Server is running.	

Historian

View the caches that are archived by the Historian application, substitution variables associated with the history cache configuration file, as well as the history cache status. You can also stop and start the Historian, and purge data.

To view Historian data select **App**, then select a Historian application from the **RTView Applications** table.



Field Name	Description	
Data Configuration Files	Lists the configuration files that are used by the history cache.	
	File Name	The name of the history cache configuration file.
	Substitutions	Lists the substitutions specified in the history cache configuration file.
Primary Server	When green, indicates that this Historian, when used within a group of Historians, is the primary group member. If the primary member fails or shuts down, the standby member with the highest priority becomes the primary group member. When red, indicates that this Historian is a secondary server.	
Storing Data	When green, indicates that this Historian is actively archiving data. When red, indicates that this Historian is not actively archiving data.	
Start Storing Data	Click to start or resume archiving data on the history cache.	

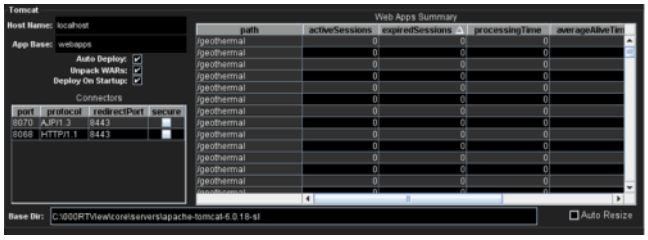
Stop Storing Data	Click to stop archiving data on the history cache.
Purge Stored Data	Click to remove all archived data on the history cache.

Tomcat

Monitor the performance of Tomcat application sessions and get Tomcat hosting and connection details. Use this data to verify response times of your Web applications.

To view Tomcat application data select **App**, then select a Tomcat application from the **RTView Applications** table.

For more information about Tomcat, see The Apache Software Foundation documentation, located at: *http://tomcat.apache.org/*.



Field Name	Description
Tomcat	The Tomcat pane displays hosting information for the Tomcat application, as well as performance metrics.
Host Name	The name of the host where the application resides.
App Base	The directory in which Tomcat is installed.
Auto Deploy	When checked, indicates that Tomcat option, automatic application deployment, is enabled. **Note:** This Tomcat option is set using the **autoDeploy** property in the server.xml file, located in the Tomcat **conf** directory. **autoDeploy=true** enables the option.
Unpack WARs	When checked, indicates that the option to unpack WAR files located in the application base directory is enabled. **Note:** This Tomcat option is set using the **unpackWARs** propert in the **server.xml** file, located in the Tomcat **conf** directory. When enabled (**unpackWARs=true**), WAR files are unpacked into the appropriate folder off of the **webapps** folder (the default base directory), and the applications run from that directory. When disabled (**unpackWARs=false**), the applications run directly from the WAR file. For more information about the Tomcat settings, refer to: **http://tomcat.apache.org/tomcat-5.5-doc/config/host.html#Introduction** and: **http://tomcat.apache.org/tomcat-5.5-doc/config/index.html**

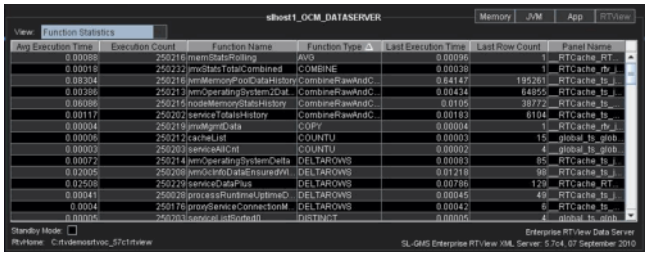
	Deploy On Startup	When checked, indicates that the option to deploy the application on Tomcat startup is enabled. **Note:** This Tomcat option is set using the **deployOnStartup** property in the **server.xml** file, located in the Tomcat **conf** directory. When enabled (**deployOnStartup=true**), applications from the host are automatically deployed.
Connections	Displays Tomcat application connection information. Port The port number used by the Tomcat application on the host.	
	protocol	The protocol used by the Tomcat application on the host.
	redirectPort	The redirect port number used by the Tomcat application on the host.
	secure	When checked, specifies that the Tomcat application uses a secure connection on the host.
Web Apps Summary	Displays servlet information. For more information about Tomcat, see The Apache Software Foundation documentation, located at: **http://tomcat.apache.org/**.	
	path	The path of the Web application.
	activeSessions	The number of clients currently in session with the servlet.
	expiredSessions	The number of clients currently in session with the servlet.
	processingTime	The amount of time, in milliseconds, for the servlet to process client requests.
	averageAliveTime	The average amount of time, in milliseconds, of client sessions.
Base Dir	The webapps folder, where applications are run from.	
Auto Resize	When checked, automatically resizes the Web Apps Table columns.	

RTView Details

Monitor the performance of specific RTView application functions and get connection information for Data Servers. From the **View** drop-down menu, select:

- **Function Statistics:** to verify execution time for functions and expressions. Check for slow Avg Execution Time values, and consider modifications to run the application run more efficiently.
- **Dataserver Connections:** to monitor the number of requests received by the Data Server, the last time a request was received and the current status.

To view RTView Details data select **App**, select an application from the **RTView Applications** table, then select **Function Statistics** or **Dataserver Connections** from the **View** drop-down menu.



Field Name	Description	
Function Statistics	Displays statistics for RTView functions executed by the application.	
	Avg Execution Time	The average amount of time, in milliseconds, for the function to execute.
	Execution Count	The number of times that the function executed.
	Function Name	The name of the function.
	Last Column Count	The number of columns in the most recent query result. A value of 0 indicates a scalar result.
	Last Execution Time	The amount of time, in milliseconds, the RTView function took to execute.
	Last Row Count	The number of rows in the most recent query result. A value of 0 indicates a scalar result.
	Panel Name	The name of the panel within the RTView display in which the function executed.
Dataserver Connections	Displays connection information for the Data Server.	
	Connected	The number of clients currently connected to the Data Server.
	ConnectionString	The host and port number of the connected Data Server.
	Name	The Data Server name.
	ReceiveCount	The total number of requests received by the Data Server.
	ReceiveTime	The time the last message was received.
	Status	The command execution status (OK, inactive, no service or no connection).

Java Tools

This section describes how to use Java tools to troubleshoot RTView applications.

SL recommends JDK v1.6 on systems running RTView server applications. JDK v1.6 includes tools such as "JConsole", "Jmap Utility", and JPS (which are not included with the JRE - see "Linux"). These tools are not required but make it easier to collect more complete information for troubleshooting. Documentation for the tools are located at:

http://www.oracle.com/technetwork/java/javase/tools6-unix-139447.html.

This section assumes you are using JDK v1.6.

Several of the steps in this section require an RTView application process ID (PID). To obtain a PID, see "Obtaining the RTView PID".

Check CPU and Memory Usage

There are several Java tools you can use to check CPU and memory usage: "JConsole" and the "Jmap Utility".

JConsole

Use the JConsole tool, included with JDK, to monitor CPU and memory usage of a Java application.

To use the JConsole:

Use the **-jmxport** option to specify the port number on which RTView accepts a JConsole connection. For example:

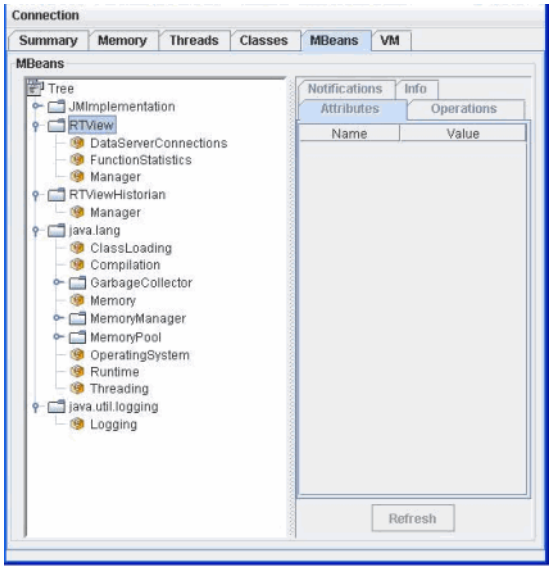
run_displayserver -jmxport:6789 ...

JConsole is then connected to the RTView application as:

jconsole localhost:6789

Note: -jmxport must specify a unique port number for each RTView application on a single host. If no **jmxport** was specified when the RTView application was started, connect JConsole by specifying the RTView application PID: **jconsole PID**

The JConsole opens.



View the following in the JConsole:

Memory Tab

View application heap usage trends. Usage might go up and down between garbage collection intervals but, overall, the heap size should be stable. An exception to this is an RTView server that runs the RTVIew cache data source. In that case, heap usage grows until each cache history table reaches its maximum row counts or timeSpan. See "Caches" for information.

MBeans Tab

See the **RTView** node which contains data generated by RTView applications. See the **RTViewHistorian** node, which is the base class for the Historian, the Data Server, and the Display Server. Also see the **java.lang** node, which contains standard Java application JMX metrics.

- **VersionInfoDetails** - Use this MBean for reporting problems to SL Technical Support. This table contains detailed information about the version of each jar used in an RTView application. The table contains the following columns:

  ApplicationName - The name of the application. For example, RTView Data Server.

  ApplicationConfiguration - The configuration string for the application. This string contains the main application version that corresponds to the version information that is printed to the console at startup.

  JarName - The name of the jar.

  JarConfiguration- The configuration string for the jar.

  JarVersionNumber - The version number for the jar.

  JarVersionDate - The version date for the jar.

  JarReleaseType - The release type for the jar.

  JarMicroVersion - The micro version for the jar.

Note: JConsole v1.6+ has a **Detect Deadlock** button on the **Threads** tab. Click this if the server has become unresponsive and a thread deadlock is suspected.

Jmap Utility

To use the JConsole:

Use the **Jmap** utility, included with JDK, to measure heap memory usage as follows:

jmap -histo:live PID > heap.txt

where **PID** is the RTView application process ID.

This produces a memory usage histogram. If a memory leak is suspected, run **Jmap** several times, several minutes apart (or hours apart, depending on the rate of growth), with the output sent to different files (specified by the file option) for comparison.

To pinpoint memory leaks, produce a complete memory dump of the application, as follows:

jmap -dump:live,format=b,file=heap.dump PID

The JDK **Jhat** utility is used to analyze the dump files. The **Jhat** analysis can be performed at SL.

Note: The dump file can be very large. Again, if a memory leak is suspected, **Jmap** should be run several times several minutes (or hours, depending on the rate of growth) apart with the output sent to different files (specified by the **file** option) for comparison.

Check Application and CPU Bottlenecks

Use thread dumps to identify application deadlocks and CPU bottlenecks.

Thread Dumps

A thread dump of a Java application shows a stack trace of the methods being executed by each thread. Take multiple thread dumps of the application: take one when the application is healthy and behaving as expected, and three or four additional thread dumps, about 1 minute apart, when the application is not performing as expected.

You can produce thread dumps on "Linux", "Windows", and "JDK".

To produce a thread dump:

Linux

Use **kill -3** PID where PID is the application process ID.

The thread dump is sent to the application standard output stream. Or, if that stream has been redirected to a log file, the thread dump is sent to that file.

Windows

If the application was started with a custom script (and not the standard **run_*.bat** script), type Ctrl-C in the application console window.

If the application was started with the standard **run_*.bat** script the thread dump only shows the threads for the RTView launcher application (which is not useful).

JDK

Use **jstack PID > threads.dump**. This sends the thread dump to the specified file.

Each thread dump begins with the following:

```
2011-01-29 14:19:28
Full thread dump Java HotSpot(TM) Client VM (...version info ...)
```

Obtaining the RTView PID

Windows

Open the Windows Task Manager. If several Java applications are running on the same host, examine the command-line arguments for each in order to distinguish between them. Use the check box in the Windows Task Manager **View / Select_Columns** menu.

Note: This feature is not available on Windows XP.

Note: The Display Server, Data Server, and Historian command lines look very similar. In RTView 5.8+, there is a **PROCESS_NAME** argument that is unique for each of those applications.

Linux

Use top or ps. If the JDK is installed, **jps -v** can be used on either operating system. The JPS tool only lists Java applications. If several Java applications are running on the same host, examine the command line arguments for each in order to distinguish between them. Consult the man pages for top and ps, if those utilities do not show the command line by default.

Note: The Display Server, Data Server, and Historian command lines look very similar. In RTView 5.8+, there is a **PROCESS_NAME** argument that is unique for each of those applications.

Log Files

There are several types of log files available for troubleshooting your RTView system. Display Server and Data Server log files contain output stream and error stream content. If your Display Server or Data Server is not operating properly, search for errors and exceptions in the RTView log file as well as the Tomcat application (or other application server) log file. If the RTView Data Servlet is used to provide HTTP access to your Data Server, also search for errors and exceptions in both types of log files.

This section contains:

"RTView Log Files": Describes how to use Log4j. This section includes:

- "Obtaining Log Files"
- "Formatting Log Files"
- "Rolling Log Files"
- "Windows Service Viewer"
- "Using JMX Access to Log Trace Levels"

"Tomcat Log Files" (or other application server): Describes how to obtain Tomcat log files.

RTView Log Files

This section describes how to use Apache Log4j v.1.2.x to generate log files for troubleshooting and debugging RTView applications. You can view RTView application output and error streams while testing or while operating in the production environment. Log4j enables you to format log files and make them searchable, and also allows log file output to multiple destinations.

By default, RTView processes (Builder, Viewer, Data Server, Display Server, or Historian) print log messages to the console. To obtain log files, redirect the RTView application output and error streams to a log file using Log4j.

The log file format is specified by a configuration file. By default, the sl.log4j.properties configuration file is used which is suitable for most use cases. We recommend using the **sl.log4j.properties** configuration file as it ensures that the Log4j features are available for RTView applications. You can modify the log file format by editing the **sl.log4j.properties** configuration file.

Note: To run the application as a background process use the **sl-bg.log4j.properties** configuration file (which only outputs to a log file rather than to a console) and the **-bg** command line argument. See "Obtaining Log Files", next.

This section includes:

Obtaining Log Files

To obtain Log4j log files, turn on Log4j using the **-log4j** option when you start the RTView application from a Windows Command Prompt or UNIX terminal window. The following example illustrates how to start Log4j for the Data Server.

RTView v.6.0.1+

Windows:

In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), type:

run_dataserver –log4j

To run the application as a background process, type:

run_dataserver –bg –log4j –log4jprops:sl-bg.log4j.properties

UNIX:

In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), type:

run_dataserver –log4j

To run the application as a background process, type:

run_dataserver –bg –log4j –log4jprops:sl-bg.log4j.properties

The following arguments are also available:

-showlogcat	Turns on the **Category** column in the log file output.	
-log4jprops	Specifies the **.properties** file to use to format the Log4j log file. By default, **sl.log4j.properties** is used. Use this to provide a different property file name (as with the **-bg** example, above). The .properties file is searched for inside a .jar/.war file, then searched for in the current directory, and lastly searched for in the **%RTV_HOME%/lib** directory. The filename can have a path preceding it. For example, **C:\mydir\my.log4j.properties**.	
-log4jlevel	Specifies the Log4j Level. INFO is the default. Valid values are:	
	FATAL	Indicates a severe error that likely causes the application to abort.
	ERROR	Indicates an event that might not cause the application to abort.
	WARN	Indicates a potentially harmful event.
	DEBUG	Indicates detailed informational about events for debugging the application.
	INFO	Indicates informational messages about the progress of the application at coarse-grained level.

For example:

run_dataserver –log4j

run_dataserver –log4j –log4jlevel:INFO –showlogcat

After executing this command, the first time-stamped row in the log file appears as follows:

```
2012-02-02 14:00:54,693 INFO – [rtview] Log4j is being used with
sl.log4j.properties as the configuration file.
```

When Log4j is not in use, the first time-stamped row in the log file appears as follows:

```
2012-02-03 10:40:31.866 [rtview] Logging redirected for System.out and
System.err. Log4j is not in use.
```

(Note the missing **INFO** column when Log4j is not in use.)

Note: The logging method from previous versions of RTView does not use Log4j. This previous method of logging is enabled with -logfile and –logdir and is still supported. Do not use both the previous logging method and Log4j or you receive the following error message: ERROR: log4j configuration ERROR - com.sl.rtview.useLog4j is set to true but -logfile redirection is in use. Log4j will not be used.

Formatting Log Files

To modify log file settings edit the **.properties** configuration file, located in the **RTV_HOME\lib** directory. To format log files, edit appenders and layouts in the **.properties** file.

- **Appenders:** Specify the output destination for a log file. Appenders exist for the console, files, GUI components, remote socket servers, JMS, NT Event Loggers, and remote UNIX Syslog daemons. Logging can also be performed asynchronously. Multiple appenders can be attached to a logger. By default, the **.properties** file has two appenders. The first appender is output to the console and the second appender is output to the log file.
- **Layouts:** Specify the output format by associating layouts with appenders. Use the PatternLayout to specify the output format according to conversion patterns similar to the C language **printf** function.

  For example, the **PatternLayout** with the conversion pattern:

  %r [%t] %-5p %c - %m%n

  outputs something similar to:

  ```
  176 [main] INFO org.foo.Bar - Located nearest gas station.
  The first field is the number of milliseconds elapsed since the start of the
  program. The second field is the thread making the log request. The third field is
  the level of the log statement. The fourth field is the name of the logger
  associated with the log request. The text after the '-' is the message of the
  statement.
  ```

For details about **PatternLayout** specifications, see: **http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/PatternLayout.html**

The following is an example sl.log4j.properties configuration file with default settings. The log file generated by these settings follows.

sl.log4j.properties With Default Settings

```
log4j.rootLogger=ALL
log4j.logger.com.sl.gmsjrtview.RTVLog=ALL, rtv_stdout, R
log4j.additivity.com.sl.gmsjrtview.RTVLog=false
log4j.appender.rtv_stdout=org.apache.log4j.ConsoleAppender
log4j.appender.rtv_stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.rtv_stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss.SSS} %-5p -
%m%n
log4j.appender.rtv_stdout.threshold=info
log4j.appender.R=org.apache.log4j.RollingFileAppender
log4j.appender.R.File=rtview.log
log4j.appender.R.MaxFileSize=1000KB
 log4j.appender.R.MaxBackupIndex=1
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern =%d{yyyy-MM-dd HH:mm:ss.SSS} %-5p %t - %m%n
log4j.appender.R.threshold=info
```

The following is the log file generated by the default **sl.log4j.properties** file.

Example Log File With Default Settings

The default settings have the following fields (listed in bold):

```
(Date            Time          Level   Category     Logging Message)
2012-03-08 16:19:44.702 INFO - [rtview] Log4j is being used with
C:\rtview_svn_trunk\core\lib\sl.log4j.properties as the configuration file.
 2012-03-08 16:19:45.882 INFO - [rtview] ... processMainArgument: jmx -jmxdstrace:5
 2012-03-08 16:19:45.882 INFO - ... traceLevel: 3 false
```

```
 2012-03-08 16:19:45.882 INFO - [rtview] ... processMainArgument: cache -cachedstrace:5
 2012-03-08 16:19:45.882 INFO - [cache] ... cacheDs.traceLevel: 3 false
 2012-03-08 16:19:45.891 INFO - [rtview] ERROR: GmsRtViewXmlDs -- jmx_constants.xml not
found.
 2012-03-08 16:19:45.892 INFO - [jmx] *** JmxDs.initializeData()
 2012-03-08 16:19:45.892 INFO - [jmx] ... Connection <Logging> connecting to server via
host:port: localhost:9990
```

Formatting Notes:

- **appender.R=org.apache.log4j.RollingFileAppender** specifies to rollover log file according to the **MaxFileSize**. For details, see **http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/DailyRollingFileAppender.html**.
- **appender.R.File=rtview.log** specifies the name of the log file in use (**rtview.log**).
- **appender.R.MaxFileSize=1000KB** specifies log file rolling to occur when the log file reaches a size of 1000KB.
- **appender.R.layout**, the line in bold in the above **.properties** file example, specifies the **PatternLayout** that formats the log file:

  %d{yyyy-MM-dd HH:mm:ss.SSS} %-5p %t - %m%n

  The converted code creates the following format:

  Date, Time, Level, Category, Logging Message.

  For details about code conversion for **PatternLayout**, see **http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/PatternLayout.html**.
- To setup log file rollover to occur every 24 hours at midnight, replace **RollingFileAppender** with **DailyRollingFileAppender**, and remove the following two lines:

  ```
  log4j.appender.R.MaxFileSize=1000KB
  log4j.appender.R.MaxBackupIndex=1 (keep only one older file)
  ```

For details about code conversion for **DailyRollingFileAppender**, see **http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/DailyRollingFileAppender.html**.

Log File Fields

For more information, see **http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/Level.html**.

Field	Description
Date	The date the row of data was received. Example: **2012-02-14**
Time	The time the row of data was received. Example: **15:12:08.425**
Level	The Log4j level to be used. Example: **INFO**

Category	The logging category the row of data originated from. See "Logging Categories" for more information. Example: **[rtview]**
Logging Message	The text for the log message. Example: `SQLException: Connection is broken: java.net.SocketException: Connection reset by peer: socket write error, update=insert into "aggregateTest" values('2012-02-14 15:12:08.0','ccccc',75,69,65.17,71.41,1,'2012-02-14 15:12:08.0','conn1','server2')`

For more information about Log4j, see **http://logging.apache.org/log4j/1.2/manual.html**, and *Log4j tutorial*.

Rolling Log Files

Log files grow large very quickly with obsolete data. There are two methods for managing log file size, one method is time-based and the other is file size-based:

- **Rolling File Appender** - With this method the log file rolls over when it reaches a certain size. For example, if you specify a maximum size of 1024kb for a log file, the oldest line is removed when that size is exceeded. The log file always contains only the latest entries. For details, see **http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/RollingFileAppender.html**.
- **Daily Rolling File Appender** - With this method log files store data for a specified period of time and subsequently saved with a new file name. For example, if you specify the time for midnight, every day at 11:59:59 PM the date and time is appended to the log file name. The original log file continues logging for the next 24 hours and the process repeats. Each log file contains entries for a 24 hour period. For details, see **http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/DailyRollingFileAppender.html**

Windows Service Viewer

For details about how to use Log4j with Windows Service, see "Log4j and Windows Service".

Using JMX Access to Log Trace Levels

This section describes how to use JMX Access to log trace levels. JMX Access enables you to increase logging information output while an application is running.

If you are using Log4j as your logger you can dynamically set logging trace levels for the different "Logging Categories". This is accomplished with the RTView Manager JMX MBean **RTView:name=Manager**. To gain access to the RTView process MBeans you must first start the process with a defined JMX port (see the **jmxport** command line properties documentation for the RTView process you want to set up logging for in Appendix C).

You can connect to this MBean via the JConsole utility. The screen shots show JConsole for JDK 1.5.n (J2SE 5.0). You can also connect to the MBean from your own Java code.

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), turn on Log4j JMX Access by adding **-jmxport:9991** to the script that starts the RTView application. Choose a different port number if using multiple applications (such

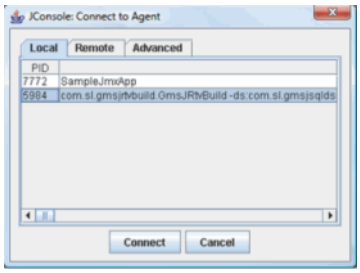
as **9990** for the Display Viewer, **9991** for the Builder and so forth). The following example uses the Data Server **run_builder.bat** file:

run_builder –log4j –jmxport:9991

2. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), open JConsole by typing:

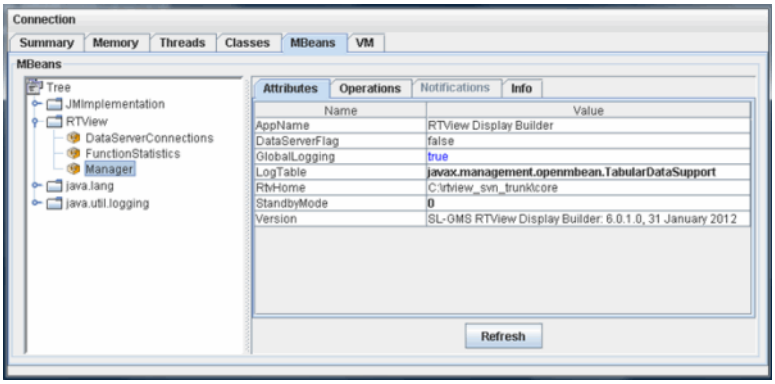
   C:> jconsole

   The **JConsole Connect to Agent** dialog opens.



3. Choose the RTView process that you want to control and click **Connect** (note that this example is connecting to an RTView Builder process). The following JConsole connections dialog shows the two MBean Servers that should appear.

The **JConsole Connection** dialog opens.

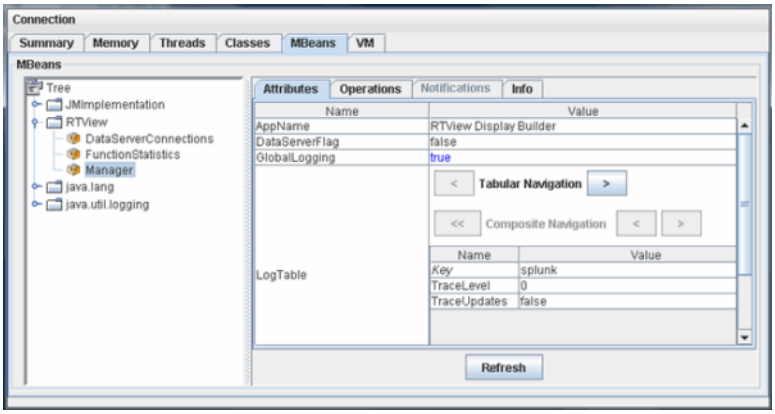


4. Select the **MBeans** tab, then select the **Attributes** tab from the table.

5. In the **JConsole** tree, open the **RTView** folder and select the **Manager**.

6. In the **Attributes** table, double-click the **LogTable Value** (for example, **javax.management.openmbean.TabularDataSupport**).

The **LogTable Value** field expands. If there is more than one **LogTable**, click the **Tabular Navigation** arrows to view them.

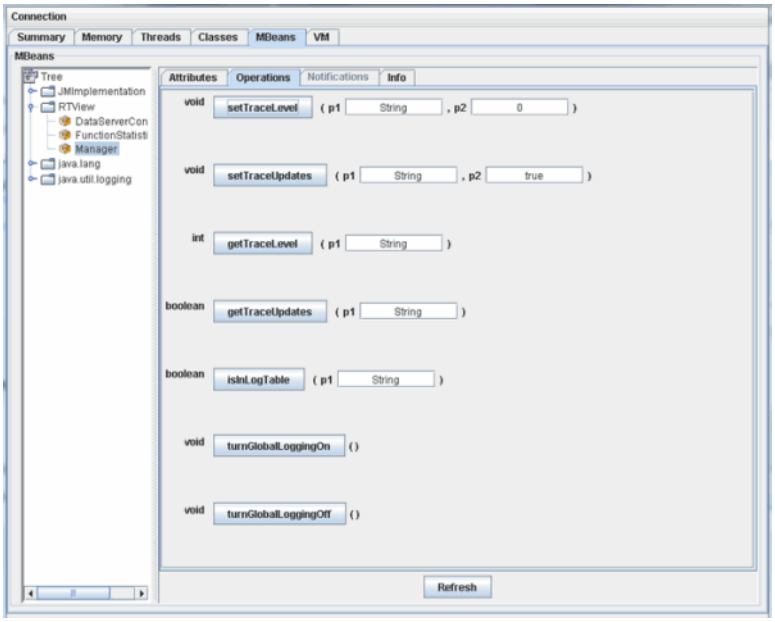


7. Click the **Operations** tab and make the following entries:
   - **Key** - Specifies the logging category to be logged (see "Logging Categories"). For example, **jmx**.
   - **TraceLevel** - Specifies the amount of logging to perform for the logging category. Valid values are **0**, **1**, **2**, and **3** where **0** equals no logging and **3** is the maximum possible logging.
   - **TraceUpdates** - Specifies whether to provide log messages containing details about logging category. Valid values are true and false.
8. Click **Refresh** when finished.

Logging Categories

Category Key	Description
alert	Alert Data Source
cache	Cache Data Source
cmdb	CMDB Data Source
function	Function Data Source
hawk	Hawk Data Source
ibmadm	IBM ADM MQ Data Source
ibmmq	IBM MQ Data Source
jms	JMS Data Source
jmsadm	TIBCO EMS Data Source
jmx	JMX Data Source
lbm	LBM Data Source
local	Local Variable Data Source
log4j	Log4j Data Source
ocd	Oracle Coherence Data Source
pi	PI Data Source
pipe	Pipe Data Source
rrd	RRD Data Source
rtvagent	Agent Data Source
rvd	TIBCO Rendezvous Data Source
sb	Streambase Data Source
snmp	SNMP Data Source
splunk	Splunk Data Source
sql	SQL Data Source
xml	XML Data Source

9. Modify **TraceLevel** or **TraceUpdates** for a category by selecting the **MBeans Operations** tab.



Operation	Description
setTraceLevel	Specifies the **TraceLevel** for the specified logging category. Clear the descriptive text in the field before making an entry. Enter the logging category in the p1 field. Values are case-sensitive. Enter the logging level in the p2 field, then click the **setTraceLevel**. Valid values are **0**, **1**, **2**, and **3** where **0** equals no logging and **3** is equals the maximum possible logging. See "Logging Categories" for more information.
setTraceUpdates	Specifies the **TraceUpdates** for the specified logging category. **TraceUpdates** log messages provide details about RTView application processes and are used for debugging and troubleshooting. Clear the descriptive text in the field before making an entry. Enter the logging category in the p1 field. Values are case-sensitive. Enter true or false in the p2 field, then click **setTraceUpdates**. See "Logging Categories" for more information.
getTraceLevel	Specifies to get the **TraceLevel** for the specified logging category. Clear the descriptive text in the field before making an entry. Enter the logging category in the p1 field. Values are case-sensitive. Click **getTraceLevel**. A new dialog opens to confirm the level. See "Logging Categories" for more information.
getTraceUpdates	Specifies to get the **TraceUpdates** for the specified logging category. Clear the descriptive text in the field before making an entry. Enter the logging category in the p1 field. Values are case-sensitive. Click **getTraceUpdates**. A new dialog opens to confirm the value. See "Logging Categories" for more information.
isInLogTable	Specifies to return whether the specified logging category is in the table. The log table list of logging categories is populated by the data sources available. Clear the descriptive text in the field before making an entry. Enter the logging category in the p1 field. Values are case-sensitive. Click **isInLogTable**. See "Logging Categories" for more information.

turnGlobal LoggingOff	Turns off all logging for the application. When selected, redirect logging is used.
turnGlobal LoggingOn	Turns on all logging for the application at current trace levels. When selected, Log4j logging is used.

Tomcat Log Files

If your Display Server or Data Server is not operating properly, search for errors and exceptions in the Tomcat application (or other application server) log file.

If an application server other than Tomcat is used, consult that application server documentation to determine where the log files are located. Typically, Tomcat log files are located in **CATALINA_HOME/logs**, where **CATALINA_HOME** is the Tomcat installation directory.

Display Server

The log file with relevant information for the RTView Display Servlet is output by the servlet when it starts. For example:

```
rtvdisplay servlet: <version_string>
rtvdisplay: DisplayServerHost=localhost:3279, timeout=15000
```

Data Server

The output from the RTView Data Servlet (used to provide HTTP access to the Data Server) is output when the first client connects. For example:

```
RTVDataServlet: <version_string>
RTVDataServlet: ServiceHost=localhost:3278, timeout=15000
```

Tomcat and Linux

Tomcat logging is configurable, as described in Tomcat documentation. On Linux, by default the log file of interest is **catalina.out**. Every run of Tomcat appends its output to the same **catalina.out** file. Because the file can get very large, consider rotating this log file periodically, or when Tomcat is restarted.

Tomcat and Windows

On Windows, if the default Tomcat startup script is used, the Tomcat output simply goes to the console window from which the script is run. It might be necessary to redirect this output to a file, use a custom startup script, or customize the Tomcat logging.

If Tomcat is run as a Windows service, look for files named **stdout*.log** and **stderr*.log** in the Tomcat logs directory.

CHAPTER 20 Customization

This section contains the following:

Overview

RTView allows you to extend built-in functionality by writing Java code to create custom commands, functions, objects, security managers and data adapters.

Note: This section is intended for developers familiar with writing, compiling, and deploying Java classes.

Define Custom Command

Note: This section assumes you have a working knowledge of JavaScript and writing, compiling and deploying Java classes.

The Custom Command Handler class extends the functionality of RTView by allowing you to write code that will get called when the custom commands you defined are executed. For the Display Builder and Display Viewer, you must write your custom commands in Java. With the Display Server, you may write your custom commands in Java or JavaScript. Java commands are executed by the Display Server on the application server and JavaScript commands are executed by the web browser on the client. For more information on how commands are executed, see the

section. See the "Define System Command" section for information on defining a command property to execute a custom command.

Java Custom Command Handler

To implement your own Java Custom Command Handler, create a Java class named **MyCommandHandler.java** that extends **GmsRtViewCustomCommandHandler**.

In **MyCommandHandler.java**, define the following method:

public GmsRtViewCommandStatus invokeCommand (GmsRtViewCommand cmd)

Every time a custom command is executed in RTView this method will get called.

Add the **gmsjrtview.jar** file, located in the **lib** directory (found in your installation directory,) to your classpath when you compile your Custom Command Handler. The compiled Custom Command Handler class must be included in the RTView classpath by adding it to the definition for the **RTV_USERPATH** environment variable.

The following example is a Custom Command Handler that will print the command argument to the console when the custom command **my_command** is executed.

```
import com.sl.gmsjrtview.*;
      /** This method is called each time a custom command is executed in RTView.
        *
        * @param cmd the command to be executed
        * @return a GmsRtViewCommandStatus object, indicating the status of
        * the custom command execution.
        */
      public GmsRtViewCommandStatus invokeCommand (GmsRtViewCommand cmd)
{
      GmsRtViewCommandStatus sts = new GmsRtViewCommandStatus();
       if (cmd.commandString.equals("my_command")){
      sts.status = GmsRtViewCommandStatus.OK;
      System.out.println(cmd.commandArg);
       } else {
      sts.status = GmsRtViewCommandStatus.ERROR;
       sts.message = "unrecognized custom command: " + cmd.commandString;
      }
       return sts;
}
```

The **GmsRtViewCommand** object is passed as the argument to each custom command and is defined as follows:

```
public class GmsRtViewCommand
{
      /**
        * The name of the custom command to be executed.
        */
      public String commandString;
      /**
        * The argument for the custom command.
        */
      public Object commandArg;

}
```

The **GmsRtViewCommandStatus** object is returned from the custom command and is defined as follows. The custom command code must assign a value to the status field and, optionally, to the message field.

```
public class GmsRtViewCommandStatus implements java.io.Serializable
{
      /**
        * Value for status field, indicating successful command completion.
        */
      public final static int OK;
      /**
        * Value for status field, indicating error during command execution.
        */
      public final static int ERROR;
      /**
        * Status (OK or ERROR) of command execution.
        * Default value is OK.
        */
      public int status = OK;
      /**
        * Message to be displayed in dialog from Viewer.
        * This can be an error message or an information message.
        * If null, no dialog is opened. Default is null.
        */
      public String message = null;
}
```

JavaScript Custom Command Handler

To implement your own JavaScript Custom Command Handler, modify **rtv_custom.js** file, which is located in **servlets\rtvdisplay**.

The **rtv_custom.js** file contains two JavaScript functions: **rtvGetInvokeCommandOnClient** (commandString) and **rtvInvokeCommand** (commandString, valueString).

1. rtvGetInvokeCommandOnClient (**commandString**)

This function must return true if the commandString should be executed on the client by the web browser, using the **rtvGetInvokeCommandOnClient** function. All commandStrings for which this function returns true must be implemented in **rtvInvokeCommand**. It must return false if the specified commandString should be executed by the Display Server on the application server. The default returns false.

2. rtvInvokeCommand (commandString, valueString)

This function must implement each commandString where **rtvGetInvokeCommandOnClient** (commandString) returns true, as described above. The **rtvInvokeCommand** function is invoked in a hidden IFrame that is a child of the Frame containing the Display Server display, which can be referenced in JavaScript as **"window.parent"**. The default implementation of **rtvInvokeCommand** does nothing.

To deploy JavaScript custom commands you must pack your **rtv_custom.js** into **rtvdisplay.war** and redeploy the servlet on your application server. Custom commands that are to be executed by the Display Server must be implemented in MyCommandHandler.java, and MyCommandHandler.class must be found in the Display Server's classpath. See "Deploying War Files" and "Java Custom Command Handler" for more information.

The following example is a JavaScript Custom Command Handler that will execute the custom command **client_echo** on the client:

```
function rtvGetInvokeCommandOnClient (commandString)
```

```
     {
  switch (commandString) {
case 'client_echo':
     // the custom "client_echo" command should
     // be run in the client browser, so
     // return true here
    return true;
default:
     // all other custom commands should be
     // run on the server
    return false;
  }
     }
function rtvInvokeCommand (commandString, valueString)
     {
  switch (commandString) {
case 'client_echo':
    alert("ECHO: " + valueString);
    break;
  }
     }
```

Custom Encryption/Decryption Handler

RTView supports a Custom File Crypto Handler for encrypting and decrypting **.rtv**, **.properties**, and **.ini** files. The Custom File Crypto Handler class extends the functionality of RTView by allowing you to write Java code that is called when **.rtv**, **.ini**, and **.properties** files are read, and when **.rtv** and **.ini** files are written. RTView does not write **.properties** files so you are responsible for encrypting them in such a way that they can be decrypted by the Custom Crypto Handler.

- To implement your own Java Custom File Crypto Handler, create a Java class named **MyFileCryptoHandler.java** that extends **GmsCustomFileCryptoHandler**. See **docs/javadocs** for more information on the **GmsCustomFileCryptoHandler**.
- In **MyFileCryptoHandler.java**, define the following methods:

```
 /**
  * Check if a file should be handled by the custom encoder. This is
  * called immediately before a call to <code>encryptString</code>
  * or <code>decryptString</code>. If <code>true</code> is returned, then
  * the custom handler will be used to invoke <code>encryptString</code>
  * or <code>decryptString</code>.
  * @param filename the name of the file to be checked
  * @return true if the custom handler will handle this file
  */

public boolean isFileHandled (String filename)

/**
  * Encrypt the string before it is saved to the specified filename. This is only
called if
  * isFileHandled returned true for the filename.
  * @param filename the name of the file being saved
  * @param stringToEncrypt the string value to encrypt. This will be the contents
of the saved file.
```

```
  * @return the encrypted string value
  */

public String encryptString (String filename, String stringToEncrypt)

/**
  * Decrypt the string after it is read in from the specified filename before it
is processed by
  * RTView. This is only called if isFileHandled returned true for the filename.
  * @param filename the name of the file being opened
  * @param stringToDecrypt the string value to decrypt. This will be the contents
of the file.
  * @return the decrypted string value
  */

public String decryptString (String filename, String stringToDecrypt)
```

- Every time a **.rtv**, **.ini**, or **.properties** file is read or written by RTView, the **isFileHandled** method will be called. If it returns true for the specified file, the **encryptString** (if the file is being saved) or **decryptString** (if the file is being written) will be called. Add the **gmsjrtview.jar** file, located in the **lib** directory (found in your installation directory) to your classpath when you compile your Custom File Crypto Handler. The compiled Custom File Crypto Handler class must be included in the RTView classpath by adding it to the definition for the **RTV_USERPATH** environment variable. To specify a different Custom File Crypto Handler class name, add the following to **RTV_JAVAOPTS**:

  ```
  -Dcom.sl.rtview.customCryptoClassName=FullyQualifiedClassName.
  ```

- A sample Custom File Crypto Handler that uses the **javax.crypto.Crypto** class is included in the RTView installation under **custom\rtvfilecrypto**. To build it, run **make_demo** (**.bat** or **.sh**). When you run from that directory, all **.rtv** and **.ini** files will be saved using the encryption defined in the **MyFileCryptoHandler** class and all **.rtv**, **.ini**, and **.properties** files will be decrypted using the same encryption. You can also encrypt and decrypt files in the command line using the **encrypt_file** (**.bat** or **.sh**) script.

  To encrypt:

  encrypt_file fileName

  To decrypt, run

  encrypt_file fileName -decrypt

Custom Functions

It is possible to perform calculations on your data before it is displayed. RTView comes with an array of built-in, pre-defined "Function Types" or you can create your own Custom Functions. You can define functions in the **Edit Function** dialog and attach your objects to the results of these functions using the **Attach to Function Data** dialog.

Extend the functionality of RTView by creating your own custom functions. Just like built-in functions, they can be attached to any object or used as input to another function.

In order to implement your custom functions, you'll need to create, compile and deploy a custom function class as described below. See the **docs/javadocs** directory for more information on the custom function API.

Creating a Custom Function Class

1. Create a class named **MyFunctionHandler** that subclasses **com.sl.gmsjrtview.GmsRtViewCustomFunctionHandler**. RTView will create one instance of this class for each function in your application.

2. Implement the **getFunctionDescriptors()** method to return a Vector of **GmsRtViewFunctionDescriptors**, one for each custom function in this class. For example:

```
public Vector getFunctionDescriptors ()
{
// vector of function descriptors
Vector<GmsRtViewFunctionDescriptor> v = new
Vector<GmsRtViewFunctionDescriptor>();
// Round
GmsRtViewFunctionArgument roundArgs[] = new
GmsRtViewFunctionArgument[1];
roundArgs[0] = new GmsRtViewFunctionArgument("Value", GMS.G_DOUBLE);
v.addElement(new GmsRtViewFunctionDescriptor("Round",  roundArgs, GMS.G_INTEGER,
null,
"This function rounds the value to the nearest integer.\n",
"This is the extended Help description.\n",false));
// return the vector full of functions
return v;
}
```

3. Implement the **get*Result** method for each **GmsRtViewFunctionDescriptor** returned in **getFunctionDescriptors()**. Four return types are supported for custom functions and each has a corresponding callback method:

Return Type	Description	Method
GMS.G_INTEGER	int	getIntResult()
GMS.G_DOUBLE	double	getDoubleResult()
GMS.G_STRING	String	getStringResult()
GMS.G_TABLE	GmsTabularData	getTabularResult()

The **get*Result** method will get called once when the display containing the function is loaded and the function has received values for all of the arguments listed in the **GmsRtViewFunctionDescriptor** and again any time the argument values change.

The example in Step 2 returns a **GMS.G_INTEGER**, so for the Round function we'll implement the **getIntResult()** method:

```
public int getIntResult (String functionName,
      GmsRtViewFunctionDescriptor functionDesc,
      GmsModelVariables functionIcon)
{
      int ret = 0;
```

```
      //-----------------------------------------
      // Add an element to this if for each custom function
      //-----------------------------------------
      if (functionName.equals("Round")) {
      double num = Double.NaN;
      try {
      // get the argument value via the descriptor
      num = functionDesc.getArgDoubleValue("Value", functionIcon);
      } catch (Exception e) {
      // show error message, if argument not found
      System.out.println(e.getMessage());
      return 0;
      }
      // round to the nearest integer
      ret = (int) Math.rint(num);
      }
      // return the result to the caller
      return ret;
}
```

Compiling a Custom Function Class

Add the **gmsjrtview.jar** file, located in the lib directory (found in your installation directory,) to your classpath when you compile your **MyFunctionHandler.java**.

Deploying a Custom Function Class

The compiled **MyFunctionHandler** class must be included in the RTView classpath by adding it to the definition for the **RTV_USERPATH** environment variable.

Note: By default, RTView adds a jar named **myclasses.jar** in the directory where you start up to the classpath, so you can add your custom function class to the RTView classpath by including it in this jar.

Accessing a Custom Function in the Display Builder

Custom functions will be listed in the **Edit Function** dialog in the **Function Type** drop down menu below the built in functions. The custom function will be listed using the name you passed into the **GmsRtViewFunctionDescriptor** constructor. When you select a custom function from the **Function Type** list, the **Edit Function** dialog will update with one text entry field for each argument in the **GmsRtViewFunctionDescriptor**. The labels will display the names you passed into the **GmsRtViewFunctionArgument** constructor. The user can enter static values or attach any of the arguments to any available data source.

Limitation

Variables are assigned to function arguments in the order that the arguments are added. These variables are then saved to your display (**.rtv**) file when a functions is added to a display. Therefore, once you have created a display for a custom function, it will not be compatible with versions of the custom function where the argument order has changed.

Example: Adding a Custom Function to the Display Builder

These instructions will lead you through the process of adding a custom function to the Display Builder.

1. In the **demos** directory (located in your installation directory), create a new directory and name it custom.

2. In the custom directory, create a new file called **MyFunctionHandler.java** and paste the following information into that file:

```
import  java.util.*;
import  com.sl.gmsjrtview.*;
import  com.sl.gmsjrt.*;
public class MyFunctionHandler extends GmsRtViewCustomFunctionHandler
{
public Vector getFunctionDescriptors ()
{
// vector of function descriptors
Vector<GmsRtViewFunctionDescriptor> v = new
Vector<GmsRtViewFunctionDescriptor>();
// Round
GmsRtViewFunctionArgument roundArgs[] = new
GmsRtViewFunctionArgument[1];
roundArgs[0] = new GmsRtViewFunctionArgument("Value", GMS.G_DOUBLE);
v.addElement(new GmsRtViewFunctionDescriptor("Round",  roundArgs, GMS.G_INTEGER,
null,
"This function rounds the value to the nearest integer.\n",
"This is the extended Help description.\n",false));
// return the vector full of functions
return v;
}
public int getIntResult (String functionName,
GmsRtViewFunctionDescriptor functionDesc,
GmsModelVariables functionIcon)
{
int ret = 0;
//-----------------------------------------
// Add an element to this if for each custom function
//-----------------------------------------
if (functionName.equals("Round")) {
double num = Double.NaN;
try {
// get the argument value via the descriptor
num = functionDesc.getArgDoubleValue("Value", functionIcon);
} catch (Exception e) {
// show error message, if argument not found
System.out.println(e.getMessage());
return 0;
}
// round to the nearest integer
ret = (int) Math.rint(num);
}
// return the result to the caller
return ret;
}
}
```

3. In the custom directory, create a new batch file named **make_function.bat** and paste the following into that file:

```
javac -classpath .;"%RTV_HOME%/lib/gmsjrtview.jar" MyFunctionHandler.java
jar -cf myclasses.jar MyFunctionHandler.class
```

4. In an initialized command window, go to **demos\custom** and type:

   run make_function

The files **MyFunctionHandler.class** and **myclasses.jar** should now appear in the custom directory.

5. Add **myclasses.jar** to the "RTV_USERPATH" environment variable.

6. Type:

   run_builder.

7. In the Display Builder, select **Tools>Functions** and then click **Add** to open the **Edit Function** dialog.

8. The custom function you added (**Round**) should now appear in the **Function Type** drop down menu.

Custom Objects

Users can customize the Display Builder's Object Palette using SL's J-Developer product to generate dynamic Java based objects.

Creating Custom Java Objects with SL-GMS J-Developer

Use SL-GMS J-Developer to create a wide variety of primitive graphic objects or import object drawings from other programs like Microsoft® Visio.

SL-GMS J-Developer comes with SL-GMSDraw, a dynamic graphic editor that provides over fifty animated behaviors (color, size, rotation, fill, movement, etc.) that change based on real-time data input. Once objects are defined, SL-GMSDraw outputs Java code that can be compiled with J/Developer libraries. When objects generated with J/Developer are added to the Object Palette, their properties automatically appear in the Display Builder's Object Property list.

Objects generated with J/Developer are supported in all deployments, including Java application and thin client.

Refer to the "Creating Custom Objects" tutorial for an example of how to create custom objects with SL-GMS J-Developer and add them to the Object Palette.

Creating Custom Objects

This tutorial will teach you how to create custom objects with SL-GMS J-Developer and add them to the Object Palette in RTView.

Learn to:

- Use SL-GMSDraw to create customized dynamic graphic objects
- Convert objects to Java classes
- Add your custom objects to the Object Palette in RTView

Get Started

To get started you will need to install SL-GMS J-Developer in a separate directory from RTView. In order to proceed, you will need to register for a J-Developer license key.

Note: The key you received for RTView will not work for J-Developer.

Register for a License Key

Open the J-Developer registration dialog and follow the instructions in the dialog to receive your license key.

On Windows

1. Select:

   Start-->Programs-->SL-GMS J-Developer-->Registration

On UNIX

1. In a GMS command prompt, type:

   GmsRegister

Creating Customized Dynamic Graphic Objects

Start SL-GMSDraw

On Windows

1. Select:

   Start-->Programs-->SL-GMS J-Developer-->GMSDraw_mfc

On UNIX

1. Open a UNIX terminal window.
2. In the terminal window, go to your J-Developer installation directory.
3. Initialize the terminal window:

   type **source gms_init**

Note: Leave this initialized terminal window open, you will need to use it again later in this tutorial.

---

4. Start SL-GMSDraw:

   type **gmsdraw_xm**

You are now ready to create a dynamic graphic object.

Create a Dynamic Graphic Object

At this point you have:

- Registered for a license key
- Started SL-GMSDraw

1. Select **File>New**
2. Click on the filled rectangle tool from the toolbar on the left of the window.

   As you move the cursor over the drawing window, the position of the cursor is listed in the status bar in the bottom right corner.
3. Click at approximately 5, 10, and then click again at approximately 10, 5.

   You should now have a blue rectangle.
4. Right-click and select **Done** to exit the rectangle mode.
5. If the rectangle is not selected, select it, then select **Dynamics>Object Dynamic Properties**.

   The **Object Dynamic Properties** dialog displays, which allows you to enter dynamics for this object.
6. In this dialog type:

   *****

   fpercent value
7. Click **Apply**, then **Close**.

   You have now defined a dynamic that will cause the rectangle fill to move up and down according to the value of a variable called value.
8. If the rectangle is not selected, select it, then select **Object>Move**.

   The dialog should already be set up to move the object to the point 0, 0.
9. Click Apply, then Close.
10. Select **File>Change Directory** and change your current directory to your RTView installation directory.
11. Change your current directory again to **demos** (located in your RTView installation directory) and then click **Close**.
12. Save your file in the **demos** directory and name it **tutorial.m1**.
13. To test the dynamics before you convert the graphic to a Java class, select **Dynamics>Edit Data File**.

    In this window type:

    value step 0. 100. 0. 10.

14. Click **Save File**, then **Close**.

You have now created a file called tutorial.dat that will simulate data changes to a variable called value, which will increment between 0 and 100.

15. If necessary, scroll the window so you can see the rectangle.

16. Select **Dynamics>Preview Options** and in the **Preview Options** dialog, click **Start**.

The rectangle should initially be unfilled and then the fill level will move up and down.

17. Click **Stop** to end the preview.

18. Exit SL-GMSDraw.

You are now ready to convert your object to a Java class.

Convert Object to Java Class

In order to add your object to the RTView Object Palette you must convert it to a Java class.

On Windows

1. Select **Start-->Programs-->SL-GMS J-Developer-->GMS Command Prompt**.

2. In the Command Prompt window, go to your RTView installation directory and type:

rtv_init

3. In the initialized Command Prompt window, go from your installation directory to the **demos** directory and confirm that your **tutorial.m1** file is in this directory.

4. In the same initialized window, type:

make_rtvobject tutorial

You have now created a class file called tutorial.class. You are ready to add your object to the RTView Object Palette.

On UNIX

1. Open a UNIX terminal window.

2. In the terminal window, go to your RTView installation directory.

3. Initialize the terminal window.

csh bsh

type **source rtv_init type . ./rtv_init.ksh**

Note: You must initialize each new terminal window you open. See the "Setup" section for more details about setting up your environment.

---

4. In the initialized terminal window, go from your installation directory to the **demos** directory and confirm that your **tutorial.m1** file is in this directory.

5. In the same initialized window, type:

make_rtvobject tutorial

You have now created a class file called tutorial.class. You are ready to add your object to the RTView Object Palette.

Add an Object to the RTView Object Palette

At this point you have:

- Created an object in SL-GMSDraw
- Converted your object to a Java class

Each tab in the **Object Palette** window is defined by an XML file. You will need to create this file and then modify the palette initialization file so that RTView will load your tab in the **Object Palette**. (A future version of RTView will provide an interface for modifying the **Object Palette**.)

1. In a text editor, such as Notepad or vi, create a new file and enter the following:

```
<?xml version="1.0"?>
<palette>
<object name="Tutorial" class="tutorial" value="25">
</object>
</palette>
```

Note: The object name is arbitrary, but the class must match the name of the class file you created (without the .class extension). The value will set the initial value of the fill percent to 25.

---

2. Save this text file in the **demos** directory as tutorial.rtp.
3. Copy the **PALETTE.ini** file from the **lib** directory into the **demos** directory.
4. In a text editor, edit the new copy of **PALETTE.ini**:

   Add the following line to the bottom of the file:

```
Tutorial tutorial.rtp
```

5. Save and close **PALETTE.ini**.

   You are now ready to use your new object in RTView.

Use Custom Object in RTView

To use your custom object in RTView, you need to start the XML data simulator and login to the Display Builder.

Start the XML Data Simulator

In this exercise you start the XML "Data Simulator" which is the XML source used in this tutorial.

On Windows

1. In an initialized Windows command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos** directory, and type:

   start run_simdata

   The XML data simulator is ready when dots appear across the screen.

On UNIX

1. In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos** directory, and start the XML "Data Simulator" by typing:

   run_simdata &

   The XML data simulator runs as a background process and is ready when dots appear in the console.

Start the Display Builder

On Windows

1. Start the Display Builder in your original, initialized terminal window by typing:

   start run_builder

2. Login to the Display Builder. By default, the Display Builder does not require a login. "Login" can be enabled at setup to support "Role-based Security". The default user name and password are:

   User Name: **admin**

   Password: **admin**

Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role.

You are now ready to create a display.

On UNIX

1. Start the Display Builder in your original, initialized terminal window:

   type **run_builder &**

2. Login to the Display Builder. By default, the Display Builder does not require a login. "Login" can be enabled at setup to support "Role-based Security". The default user name and password are:

   User Name: **admin**

   Password: **admin**

Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role. See "Role-based Security" for more information.

You are now ready to create a display.

Create A Display

At this point you have:

- Created an object in SL-GMSDraw
- Converted your object to a Java class
- Modified the **PALETTE.ini** file
- Started the XML data simulator
- Started the Display Builder

Add an XML Source to XML Source List

To animate your custom object, you will need to add an XML source, **update.xml**, to the XML Source List.

1. Select **Tools>Options** to open the **Application Options** dialog.
2. Select the "XML Tab" and click **Add** to open the **Add XML Source** dialog.
3. In the **Add XML Source** dialog:

   XML Source Name - Enter **update.xml**

Note: The **update.xml** source is generated by the XML data simulator. See "Creating XML Sources" for technical details on creating and formatting your own XML source.

---

4. Click **OK** to close the **Add XML Source** dialog. The XML source appears in the list of available XML Data Sources.
5. Click **OK** to apply and close the **Application Options** dialog.

   The XML source is now available for animating objects.

Attach Your Custom Object to Data

To attach your object to the XML data source, you will need to add it to a display and attach it to a Data Key called **element1_load**.

1. In the **Object Palette**, select the **Tutorial** tab and add your object to the display.

   In the **Object Properties** dialog one of the available properties is value; this corresponds to the dynamic that you added to the object when you created your object in SL-GMSDraw.
2. In the **Object Properties** dialog:

   value (category: Data) - Right-click in the **Property Name** field and select **Attach to Data>XML**
3. In the "Attach to XML Data" dialog:

   XML Source - **update.xml** should already be selected.

   Data Key - Select **element1_load** from the drop down menu.
4. Click **OK** to apply these values and close the **Attach to XML Data** dialog.

The rectangle is now animated with real-time data updates provided by the value of the **element1_load** Data Key.

Make Custom Palettes and Objects Available Throughout RTView

The palette and object that you created in this example will be available only when you start the Display Builder from the demos directory. In this exercise, you will replace the standard RTView files with your customized versions to make your customized palette and object available from other directories.

RTView looks for **PALETTE.ini** in the directory where you start the Display Builder. If it is not there, RTView searches under **lib** in your installation directory. Similarly, RTView looks for palette XML files (**.rtp**) and object classes (**.class**) in the directory where you start the Display Builder. If they are not there, RTView searches in the classpath. To add a .class file to the classpath, pack it into a .jar file and add it to the "RTV_USERPATH" environment variable.

Replace the standard **PALETTE.ini** with your custom version

1. Go to the **lib** directory, located in your RTView installation directory.
2. Rename **PALETTE.ini** to **PALETTE.ini0**. This will allow you to revert to the original **PALETTE.ini** if necessary.
3. Copy your customized version of **PALETTE.ini** into the **lib** directory.

Add your palette (.rtp) files and object class (.class) files to the classpath

1. In a GMS command window, go to the **demos** directory and type:

   jar –cvf myclasses.jar *.rtp *.class

2. Add **RTV_HOME\demos\myclasses.jar** to the **RTV_USERPATH** environment variable.

   Your customized palette and objects are now available throughout your version of RTView.

More Information About SL-GMSDraw

For more information on creating customized dynamic graphics with SL-GMSDraw, refer the J-Developer documentation in **Start-->Programs-->SL-GMS J-Developer-->Documentation (PDF)**. On Unix, open **jdoc\JDEV.pdf**.

Custom Security Managers

RTView offers role based security that allows you to limit access to displays based on the user's role. To use this security feature, you must define the users that can access RTView and assign each one a password and one or more associated roles. The role definitions allow you to specify which displays users can access. By default, user and role definitions are stored in XML files. Alternatively, you can define your users using the **GmsCustomUserManager** (see "Custom User Manager" for more information) and your roles using the **GmsCustomRoleManager** (see "Custom Role Manager" for more information). For general information on RTView role based secutiry, see "Role-based Security".

Custom User Manager

Defining Users in a Custom User Manager Class

Note: This section assumes you have a working knowledge of writing, compiling and deploying Java classes.

The Custom User Manager class extends the functionality of RTView by allowing you to write Java code that gets called when the user logs in. You can use this class to implement your own mechanism for validating users, such as getting user information from a database or LDAP server. To implement your own Custom User Manager, create a Java class named MyUserManager.java that extends **com.sl.gmsjrtview.GmsCustomUserManager**. The default java class name (**MyUserManager**) can be changed using the **customUserManagerClassName** command line option. See "MyUserManager.java Example" for more information.

The following methods are available to overwrite in the **MyUserManager**:

Method	Description
public boolean activate ()	This method is called after the constructor. Subclasses should overwrite this method to initialize any data needed to validate the **userName** and **password**. This method should return true if the initialization was successful, otherwise return false. If this method returns false, the login will be considered invalid and RTView will exit.
public boolean validateLogin (String userName, String password)	This method is called from the login dialog to validate the **userName** and password. Subclasses should overwrite this method to return true if the **userName** and password are valid, otherwise return false. If there are no roles defined for the specified user, the login will fail, even if this method returns true.
public void clientLogin (String userName, String role, String sessionID)	This method is called when a client user logs in to Display Server and Data Server applications. There is a unique **sessionID** for each client login session.
public void clientLogout (String userName, String role, String sessionID)	This method is called when a client user logs out from Display Server and Data Server applications. There is a unique **sessionID** for each client login session.
public void invalidateLogin ()	This method is called if the user presses Reset in the login dialog. The Reset button clears the fields so the user can set a new **userName** and password. If subclasses stored the **userName** or password from **validateLogin()** or the status of whether they were valid these values should be cleared.
public boolean userHasMultipleRoles (String userName)	This method must return true if the specified user has multiple roles, otherwise false.
public Vector getRoleNamesForUser (String userName)	This method must return a Vector containing the role names for the specified user. This method will only be called if **userHasMultipleRoles()** returns true for the specified user. Each user is required to have at least one role.

public int validateLoginOnDefaultDataServer (String userName, String password)	This method is called to allow the Custom User Manager to request validation from the default Data Server. This method returns **0** if **userName** and **password** are valid, **-1** if login fails and **-2** if the default Data Server is undefined or unavailable.
getSubstitutions (String userName)	This method should return substitutions to set on the RTView application when the specified user is logged in. If this user does not have any associated substitutions, this method should return null. Otherwise, return a pair of Vectors. Both Vectors must be the same length, with the first Vector containing the substitution strings and the second Vector containing the corresponding substitution values.
getPassword (String userName)	This method must return the password for the specified user, or null if the user is not recognized.

This method is called for data sources that have the **Use Client Credentials** option selected. If the **Use Client Credentials** feature is not used in your application, this method can return null for all users.

Note: Some data sources do not support this feature. For information on Application Options for your data source, refer to the "RTView Data Sources" section of this documentation.

Add **gmsjrtview.jar** located in the **lib** directory (found in your installation directory) to your classpath when you compile your Custom User Manager. You must include your compiled Custom User Manager class in the classpath for RTView by adding it to the definition for the **RTV_USERPATH** environment variable or by packing it into **myclasses.jar**.

MyUserManager.java Example

The following is an example of a Java class named MyUserManager.java:

```
//**************************************************************************
//
// SL-GMS Graphical Modeling Application, RTView Package
// MyUserManager.java
// Copyright (c) 1998-2006 Sherrill-Lubinski Corporation. All Rights Reserved.
// May 2005
//
//**************************************************************************

import java.util.*;
import com.sl.gmsjrtview.*;

/** The MyUserManager class implements a custom user login.
 *  <P>
 */

public class MyUserManager extends GmsCustomUserManager
{


//*************************************************************
// Constructor methods

public MyUserManager ()
{
```

```
}

//*****************************************************************
// Instance methods

/** This method is called after the constructor. Subclasses should
 *  overwrite this method to initialize any data needed to validate the
 *  userName and password. This method should return true if the initialization
 *  was successful, otherwise return false. If this method returns false,
 *  the login will be considered invalid and RTView will exit.
 *  <P>
 */

public boolean activate ()
{
 return true;
}

//*****************************************************************
// Get substitutions method

/** This method should return substitutions to set on the RTView
 *  application when the specified user is logged in. If this user does not
 *  have any associated substitutions, this method should return null.
 *  Otherwise, return a pair of Vectors. Both Vectors must be the same length,
 *  with the first Vector containing the substutition strings and the second
 *  Vector containing the corresponding substitution values.
 * <P>
 */

public Vector[] getSubstitutions (String userName)
{
        if (userName.equals("admin")) {
                Vector subStrs = new Vector();
                Vector subVals = new Vector();
                subStrs.addElement("sub1");
                subVals.addElement("value1");
                subStrs.addElement("sub2");
                subVals.addElement("value2");
                return new Vector[] {subStrs, subVals};

        } else if (userName.equals("user2")) {
                Vector subStrs = new Vector();
                Vector subVals = new Vector();
                subStrs.addElement("user2sub1");
                subVals.addElement("user2value1");
                subStrs.addElement("user2sub2");
                subVals.addElement("user2value2");
                return new Vector[] {subStrs, subVals};

        } else
                return null;
}

//*****************************************************************
// methods called from the login dialog

/** This method is called from the login dialog to validate the userName
 *  and password. Subclasses should overwrite this method to return true
```

```
 *  if the userName and password are valid, otherwise return false.
 *  If there are no roles defined for the specified user, the login will
 *  fail, even if this method returns true.
 *  <P>
 */

public boolean validateLogin (String userName, String password)
{
 if (userName == null || password == null)
  return false;
 if (!userName.equals("admin") && !userName.equals("user1") &&
  !userName.equals("user2") && !userName.equals("user3"))
  return false;
 if (!userName.equals(password))
  return false;
 return true;

}

/** This method is called if the user presses Reset in the login
 *  dialog. The Reset button clears the fields so the user can set
 *  a new user name and password. If subclasses stored the userName
 *  or password from validateLogin() or the status of whether they
 *  were valid these values should be cleared.
 *  <P>
 */

public void invalidateLogin ()
{
}

/** This method must return true if the specified user has multiple role,
 *  otherwise false.
 *  <P>
 */

public boolean userHasMultipleRoles (String userName)
{
 if (userName == null)
  return false;
 if (userName.equals("user2") || userName.equals("user3"))
  return true;
 return false;
}

/** This method must return a Vector containing the role names for the
 *  specified user. Each user is required to have at least one role.
 *  <P>
 */

public Vector getRoleNamesForUser (String userName)
{
        if (userName == null)
                return null;
        Vector v = new Vector();
        if (userName.equals("admin")) {
                v.addElement("admin");
                return v;
        }
```

```
        if (userName.equals("user1")) {
                v.addElement("role1");
                return v;
        }
        if (userName.equals("user2")) {
                v.addElement("role2");
                v.addElement("role3");
                return v;
        }
        if (userName.equals("user3")) {
                v.addElement("role2");
                v.addElement("role3");
                v.addElement("role4");
                return v;
        }
        return null;
}


//*****************************************************************
// methods called for pass client credentials

/** This method must return the password for the specified user, or null if
 *  if the user is not recognized. This method is called for the sql
 *  data source to use if databases are defined that have Use Client
 *  Credentials selected.  If the Use Client Credentials feature is not used in
 *  your application, this method can return null for all users.
 */

public String getPassword (String userName)
{
        return null;
}
}
```

Custom Role Manager

Defining Roles in a Custom Role Manager Class

Note: This section assumes you have a working knowledge of writing, compiling and deploying Java classes. The Custom Role Manager class extends the functionality of RTView by allowing you to write Java code that will get called when the user is logging in. To implement your own Custom Role Manager, create a Java class named **MyRoleManager.java** that extends **com.sl.gmsjrtview.GmsCustomRoleManager**. The default java class name (**MyRoleManager**) can be changed using the **customRoleManagerClassName** command line option. See "MyRoleManager.java Example" for more information.

The following methods are available to overwrite in the **MyRoleManager**:

Method Name	Description
public boolean activate ()	Activate the role manager. Subclasses should overwrite this method to perform any initialization necessary to implement the **isAllowed()** method. This method should return true if initialization was successful, otherwise return false. RTView will exit if this method returns false.
public boolean isAllowed (String displayName, String role)	Returns true if the specified display is allowed, otherwise return false. Subclasses should overwrite this method to return the appropriate value based on the **displayName** and **role**.
public boolean isAllowed (String displayName, String role, String user)	Returns true if the specified display is allowed, otherwise return false. Subclasses should overwrite this method to return the appropriate value based on the **displayName**, role and user. **Note:** This method is only invoked by the Display Server, which may call this method simultaneously on different threads.
getSubstitutions (String role)	This method should return substitutions to set on the RTView application when the specified role is logged in. If this role does not have any associated substitutions, this method should return null. Otherwise, return a pair of Vectors. Both Vectors must be the same length, with the first Vector containing the substitution strings and the second Vector containing the corresponding substitution values. **Note:** The Display Server calls **GmsRoleManager.getSubstitutions(roleName)** every time a user logs in or changes roles. This option allows a custom role manager to change the substitutions returned for a specific role without restarting the Display Server.

Add **gmsjrtview.jar** located in the lib directory (found in your installation directory) to your classpath when you compile your Custom Role Manager. You must include your compiled Custom Role Manager class in the classpath for RTView by adding it to the definition for the **RTV_USERPATH** environment variable or by packing it into myclasses.jar.

MyRoleManager.java Example

The following is an example of a Java class named MyRoleManager.java:

```
//************************************************************************
//
// SL-GMS Graphical Modeling Application, RTView Package
// MyRoleManager.java
// Copyright (c) 1998-2006 Sherrill-Lubinski Corporation. All Rights Reserved.
// May 2005
//
//************************************************************************

import com.sl.gmsjrtview.*;
import java.util.*;

/** The MyRoleManager class provides a custom role manager class.
 *  <P>
 */
```

```
public class MyRoleManager extends GmsCustomRoleManager
{

//************************************************************************
// ACTIVATE METHOD

/** Activate the role manager. Subclasses should overwrite this method to
 *  perform any initialization necessary to implement the isAllowed() method.
 *  This method should return true if initialization was successful, otherwise
 *  return false. RTView will exit if this method returns false.
 *  <P>
 */

public boolean activate ()
{
 return true;
}

//************************************************************************
// GET SUBSTITUTIONS METHOD

/** This method should return substitutions to set on the RTView
 *  application when the specified role is logged in. If this role does not
 *  have any associated substitutions, this method should return null.
 *  Otherwise, return a pair of Vectors. Both Vectors must be the same length,
 *  with the first Vector containing the substutition strings and the second
 *  Vector containing the corresponding substitution values.
 * <P>
 */

public Vector[] getSubstitutions (String role)
{
        if (role.equals("role1")) {
                Vector subStrs = new Vector();
                Vector subVals = new Vector();
                subStrs.addElement("role1sub1");
                subVals.addElement("role1value1");
                subStrs.addElement("role1sub2");
                subVals.addElement("role1value2");
                return new Vector[] {subStrs, subVals};

        } else if (role.equals("role4")) {
                Vector subStrs = new Vector();
                Vector subVals = new Vector();
                subStrs.addElement("role4sub1");
                subVals.addElement("role4value1");
                subStrs.addElement("role4sub2");
                subVals.addElement("role4value2");
                return new Vector[] {subStrs, subVals};

        } else
                return null;
}

//************************************************************************
// IS ALLOWED METHOD

/** Returns true if the specified display is allowed, otherwise return
 *  false. Subclasses should overwrite this method to return the appropriate
```

```
 *  value based on the displayName and for the specified role.
 *  <P>
 */

public boolean isAllowed (String displayName, String role)
{

 System.out.println("isAllowed role: " + role +
      " display: " + displayName);
 if (role == null || displayName == null)
  return false;
 if (role.equals("admin"))
  return true;

 if (displayName.equals("title_panel") ||
    displayName.equals("operations"))
  return true;

 if (role.equals("role1")) {
  if (displayName.equals("ems_administration") ||
   displayName.startsWith("manage"))
   return false;
  else
   return true;
 }
 if (role.equals("role2")) {
  if (displayName.startsWith("manage") &&
   !displayName.equals("manage_routes"))
   return false;
  else
   return true;
 }
 if (role.equals("role3")) {
  if (displayName.equals("ems_administration") ||
   displayName.startsWith("manage"))
   return true;
  else
   return false;
 }
 if (role.equals("role4")) {
  if (displayName.equals("ems_monitoring") ||
   displayName.equals("ems_allservers_api") ||
   displayName.equals("ems_server_info") ||
   displayName.equals("host_details"))
   return true;
  else
   return false;

 }
 return false;
}

}
```

Customization - RTVAgent

An RTVAgent publishes data to an RTView application (i.e. Display Builder, Data Server, Display Server, Historian or Display Viewer Application) via the RTVAgent Data Adapter. The RTVAgent provides tabular data via socket, http, or https to the receiving RTView application. Multiple RTVAgents can provide data differentiated by a unique agent name. The receiving application only needs to enable and define the port via the RTVAgent Data Adapter. See "RTVAgent Data Source" for more information.

Note: Communication is one way from the RTVAgent to the receiving RTView application.

RTVAgents need to designate the port where they are publishing data, the agent name and the agent class. For example, suppose you created an RTVAgent that gathers CPU usage for a particular platform and reports that data to an RTView application. The agent class name might be CPU agent. When you install on multiple computers the agent class name will remain the same, but you will need to specify a unique agent name for each computer. The Agent Name, Agent Class and Table Name are all exposed in the Display Builder's "Attach to RTVAgent Data" dialog when active (i.e. connected) RTVAgents are publishing data.

Creating a Custom RTVAgent

There are two ways to create a custom RTVAgent: Create a standalone Java application or use a full RTView application.

Standalone Java Application

Creating a standalone Java application is advised if a smaller footprint agent is required and data is easily accessible via a Java API.

Using the API in the **com.sl.gmsjagent** package, refer to the **docs/javadocs** directory. There is a sample custom agent display (**RTCache.rtv**) file in **custom/rtvagent** (located in your RTView installation directory).

Review the README file (in that same directory) for directions on how to build the sample display, run an agent, and see the published data from inside the Display Builder.

Note: This sample creates simulated data and publishes it in two named tables every two seconds. An actual RTVAgent would be gathering information of interest and publishing it at its own periodic rate.

RTView Application

Using a RTView application is ideal when you want to install RTView as the RTVAgent and access data from one of RTView's standard data sources.

Note: This option requires that RTView be installed and act as the RTVAgent which will be publishing data to the Data Server. The Data Server to which the data tables are sent should be configured to receive the data.

A simple display (**.rtv**) file can be used to provide tabular data to a remote instance of Data Server via socket or http. Using the **Send Table** function, create an RTVAgent at each site that has data tables to send:

1. Configure one or more instances of the **Send Table** function to send the data tables of interest. The function's **Table to Send** argument can be attached to tabular data from any of RTView's data sources. See "Tabular Functions" for more information.
2. Save the function instances in an **.rtv** file (e.g. **SendTables.rtv**).
3. Use the **run_agent** script to start the RTVAgent and load the **.rtv** file. For example:

   run_agent SendTables.rtv

Note: Data tables from RTVAgents in the same agent class can be cached by attaching the **valueTable** property of a Cache object to RTVAgent data. See "Caches" for more information.

Customization - RTVPipe Handler

The RTVPipe data adapter supports attachments that launch an external process. The connection between RTView and the output stream of that process is referred to as a pipe. Using a Java class known as a pipe handler, it is possible to create a custom RTVPipe Handler to parse the output of specific programs or scripts. See "RTVPipe Data Source" for more information.

Note: Custom RTVPipe Handlers must be defined on the "RTVPipe Handlers Tab" of the **Application Options** dialog and the Java Class Name(s) assigned must be added to the "RTV_USERPATH" environment variable.

Using the API in the **com.sl.gmsjpipeds** package, refer to the **docs/javadocs** directory. There are two sample display files (**WindowsExample.rtv** and **LinuxExample.rtv**) in **custom/rtvpipe** (located in your RTView installation directory).

Review the README file (in that same directory) for directions on how to create two basic custom RTVPipe Handlers: the **Windows Tasklist Handler** and the **Linux Top Handler**. These custom handlers are designed to convert the output of a **tasklist** command, or **top** command, to a table.

Custom Web Applications

The rtvquery Servlet allows custom client applications to retrieve data tables from the RTView Data Server via a REST interface. The client sends an HTTP GET to the servlet specifying the query parameters, and the servlet returns the query result in XML, JSON, JSONP, or plain text format.

The rtvquery Servlet is intended for use by custom browser-based applications to display tables from the RTView Data Server at relatively low data volumes. For high data volume displays, the RTView Viewer application or Thin Client should be used.

Note: For examples of client applications, see the **README.txt** file in the **\custom\rtvquery-samples** directory.

Query Format

The rtvquery Servlet supports queries for tables from either the RTView cache data source or the SQL data source. The query parameters are specified in the URL and the headers of an HTTP GET request.

Note: Rather than formulating HTTP GETs and processing the responses, Ajax clients can make queries more simply by using the *JavaScript library* included with the servlet.

Cache Table Queries

If the rtvquery Servlet is deployed at **http://host/rtvquery**, the URL for a cache query is:

http://host/rtvquery/cache/<cacheName>/<tableName>

For example, the URL to request the history table from a cache named Production is:

http://host/rtvquery/cache/Production/history

The cache and table names should be URL encoded if they contain special characters. For example, the URL to request the current table from a cache named CPU Usage is:

http://host/rtvquery/cache/CPU+Usage/current

In addition to the cache and table names, several optional parameters can be specified in the URL to refine the query. For example:

http://host/rtvquery/cache/<cacheName>/<tableName>?<param1>=<value1>&<param2>=<value2>...

Parameter values must be URL encoded if they contain special characters.

Cache Query Optional Parameters

The following are optional parameters for cache queries. Each parameter corresponds to a filter option in a cache data source attachment in RTView. If a parameter is not specified, its default value is used in the query.

Note: The optional parameters can also be specified as custom headers in the HTTP GET request. For example, the name of the custom header for a query parameter named "P" is **x-sl-P**. The name of the custom header for specifying the time range is **x-sl-tr**. Unlike URL parameter values, if a query parameter is specified by a custom header, the value should not be URL encoded. Specifying query parameters in custom headers rather than URL parameters avoids the browser URL length limits (about 2000 characters in some IE versions). If the **fmt** parameter is not specified in the URL or in the **x-sl-fmt** header, the response format is determined by the value of the standard **http "Accept"** header.

Parameters	Description	Default Value
arr	If **fmt**=**jsonp** or **json** and this option is set to **1**, then the JavaScript array response will be used for data rows in the response. The intended use for this parameter is to set **fmt**=**jsonp** and **arr**=**1** to reduce the size of the response. If this option is set to **0**, then the JavaScript array response will not be used.	**0**
callback	The name of the JavaScript function to be called with json data. This parameter is only used when **fmt**=**jsonp**.	**jsonpCallback**
cols	The names of table columns to include in the query response, separated by semicolons (;).	*****
fcol	The names of the filter columns, separated by semicolons (;). The **fcol** and **fval** parameters are used together to specify one or more filters. Only rows that pass the filter(s) are included in the query result. The values of columns specified by **fcol** are compared with the values listed in the **fval** parameter. In the simplest case, **fcol** specifies the name of a single filter column, **fval** specifies a single filter value, and the result includes only rows in which the filter column matches the filter value. If multiple values are acceptable for the filter, they must be separated by commas in **fval**, and rows in which the filter column matches any of the filter values are included in the query result. If multiple filter columns are required, they should be separated by semicolons in **fcol**. The corresponding filter values for each filter column should be separated by semicolons in **fval**.	**none**
fmt	Specifies the desired response format: **text**, **xml**, **js**, **json**, or **jsonp**. (For details, see "Response Formats"). If this parameter is not specified in the URL or in the **x-sl-fmt** header, the response format is determined by the value of the standard **http "Accept"** header.	**xml**
fval	The filter value(s) for each filter column. This parameter is used with the **fcol** parameter. See the **fcol** parameter for further details.	**none**
nanok	When set to **true**, this option encodes **NaN**, **Infinity**, and **-Infinity** as "**NaN**", "**Infinity**", and "**-Infinity**" when the desired response format is **json** or **jsonp**. **Note:** This only impacts **json** and **jsonp** response formats.	**false**

pn	Specifies the page number to be returned. The **rp** and **pn** parameters must be used together and have the following behavior: • If **rp** and **pn** are specified, the first row in the query result is row number **pn * rp** (where row zero is the top row of the table) and the last row in the result is row number **pn * rp + rp**. • If only **rp** is specified, **pn** defaults to zero. • If only **pn** is specified, it is ignored.	**0**
rp	Specifies the maximum number of rows per page to be returned. The **rp** and **pn** parameters must be used together and have the following behavior: • If **rp** and **pn** are specified, the first row in the query result is row number **pn * rp** (where row zero is the top row of the table) and the last row in the result is row number **pn * rp + rp**. • If only **rp** is specified, **pn** defaults to zero. • If only **pn** is specified, it is ignored.	**none**
sqlex	Specifies whether to enable the **Extend with SQL** option for a query on a cache history table. This option only applies when time range (**tr**), begin time (**tb**), and end time (**te**) are also specified. For details, see "Caches".	**false**
tr	The time range, in seconds. This parameter is valid only if **table=history**. The **tr**, **tb**, and **te** parameters are time parameters and return the following: • If only **tr** is specified, all rows with **timestamps >= currentTime - tr** are returned. • If only **tb** is specified, all rows with **timestamps >= tb** are returned. • If **tb** and **tr** are specified, all rows in the range of **tb** to **tb + tr** are returned. • If only **te** is specified, all rows with **timestamps <= te** are returned. • If **te** and **tr** are specified, all rows in the range of **te - tr** to **te** are returned. • If **te**, **tr**, and **tb** are specified, **tr** is ignored and all rows with **timestamps >= tb** and **<= te** are returned.	**30**
tb	Specifies the begin time (in milliseconds since 1970) for the rows to be retrieved. This parameter is valid only if **table=history**. See the tr parameter for details about behavior with other time parameters (**tr** and **te**).	**none**
te	Specifies the end time (in milliseconds since 1970) for the rows to be retrieved. This parameter is valid only if **table=history**. See the tr parameter for details about behavior with other time parameters (**tr** and **tb**).	**none**
to	Specifies the query timeout, in seconds.	**15**

Cache Query URL Examples

The following are Cache Query URL examples.

Note: For brevity, the base URL (for example, **http://host/rtvquery/**) is omitted from the following examples.

Get the Customer, Symbol and Purchase Price columns from the current table of a cache named trades, in xml format:

cache/trades/current?cols=Customer;Symbol;Purchase+Price&fmt=xml

Get all columns of the most recent 5 minutes of data from the **trades.history** table, for a customer named John Doe, in JavaScript array format:

cache/trades/history?tr=300&fcol=Customer&fval=John+Doe&fmt=js

Get all columns from the trades.current table, for customers named John Doe or Alice Chen and for symbol = IBM or GE, in json format:

cache/trades/
current?fcol=Customer;Symbol&fval=John+Doe,Alice+Chen;IBM,GE&fmt=json

Get all columns from **trades.history** table, with indicated begin and end times, but no more than 1500 rows, in text format:

cache/trades/
history?tb=June+17,2010+10:00:00&te=June+17,2010+10:15:00&rp=1500&fmt=text

SQL Table Queries

If the rtvquery Servlet is deployed at **http://host/rtvquery**, the URL for an SQL query is:

http://host/rtvquery/sql/<dbName>?sql=<sqlQueryString>

where **dbName** is the name of the RTView database connection.

The SQL query string must be URL encoded. For example, the URL to perform the SQL query select * from production_table on an RTView database connection named SampleDB is:

http://host/rtvquery/sql/sampleDB?sql=select+*+from+production_table

Alternatively, the SQL query string can be omitted from the URL and specified in a custom http header named **x-sl-sql**.

SQL Query Optional Parameters

The following are optional parameters for SQL queries.

Parameter	Default Value	Description
maxrows	none	The maximum number of rows to be returned for the query.
to	15	The amount of time, in seconds, for the query to timeout.
fmt	xml	The query response format: **text**, **xml**, **js**, **xmlrtv**, **json**, or **jsonp**. See "Response Formats" for more information.

Responses

This section describes supported response formats and response status for the rtvquery Servlet.

Response Formats

The supported response formats are named **xml**, **json**, **jsonp**, **js**, **xmlrtv**, and **text**.

XML Response Format

The following is an XML response format, where **<DataType>** is one of the following strings: **string**, **int**, **long**, **double**, or **date**.

```
<dataset>
      <metadata>
      <column name="column 1 name" type=DataType/>
      ... metadata for other columns ...
      </metadata>
      <data>
      <row>
      <column_1_name>row 1, column 1 value</column_1_name>
      ... data for other columns in row 1 ...
      </row>
      ... data for other rows ...
      </data>
</dataset>
```

JSON Response Format

The following is a **json** response format, where <DataType> is one of the following strings: string, int, long, double, or date.

```
{
   "metadata":[
      {"name":"column 1 name","type":DataType},
       ... metadata for other columns ...
     ],
     "data":[
       {"column 1 name":"row1, column1 value", ... data for other columns in row 1},
        ... data for other rows ...
     ]
 }
```

JSONP Response Format

The following is a response in **jsonp** format:

```
jsonpCallback (json object using format described above>);
```

The **jsonp** format can be used to make cross-domain requests to the rtvquery servlet. The name of the JavaScript callback function can be specified via the **callback** parameter in the query URL. If not specified, then **jsonpCallback** is used. The callback function must be defined before the query is made. For example, the following uses **jsonp** to make a cache query and write the result to the JavaScript console:

```
<!doctype html>
<html>
<body>
<script>
```

```
var url = "http://somehost/rtvquery/cache/myCache/
current?fmt=jsonp&callback=myCallback";

// callback function for jsonp response
var myCallback = function (queryResultAsJson) {
    console.log("query result:" +
        JSON.stringify(queryResultAsJson));
}
// create script element to make jsonp query
var scr = document.createElement("script");
scr.setAttribute("type", "text/javascript");
scr.src = url;
document.body.appendChild(scr);
</script>
<body>
<html>
```

The **arr=1** parameter can be specified along with **fmt=jsonp** to make a cross-domain query, but have the data rows in the response formatted as a JavaScript array rather than an array of **json** objects, for conciseness.

JavaScript Array Response Format

The following is a JavaScript Array response format, where **<DataType>** is one of the following strings: **string**, **int**, **long**, **double**, or **date**. The first row in the array contains the column names, the second row contains the column data types, and the remaining rows are the data rows from the data table. This format is the most compact format.

```
[
   ["column 1 name", "column 2 name", ...],
   [DataType, DataType, ...],
   [value for column 1 in row 1, value for column 2 in row 1, ...],
   [row 2 values, ...]
   ...
 ]
```

XMLRTV Response Format

The following is an XMLRTV response format, where **<DataType>** is one of the following strings: **string**, **int**, **long**, **double**, or **date**. The XMLRTV format is the traditional XML dataset format used in RTView.

```
<table key="test">
   <tc name="column 1 name" type=DataType index="false"/>
   <tc name="column 2 name" type=DataType index="false"/>
   ...more column definitions ...
   <tr name="">
      <td>value for column 1 in row 1</td>
      <td>value for column 2 in row 1</td>
      ...
   </tr>
   ... other rows ...
 </table>
```

Text Response Format

The following is a text response format, where **<DataType>** is one of the following strings: **string**, **int**, **long**, **double**, or **date**. The text format uses tabs to separate columns.

```
column 1 name <tab> column 2 name <tab> ...
```

```
row 1, col 1 value <tab> row 2, col 2 value <tab> ...
...
```

Response Status

The following describes the response status indicated by the integer value of the custom header **x-sl-status**, and the corresponding string value of the header **x-sl-status-text**.

x-sl-status	x-sl-status-text	Description
0	**OK**	The query was successful.
-1	**Not connected to data server.**	The query failed and the response is empty.
-2	**Query is missing one or more required parameters.**	The query failed and the response is empty.
-3	**No data received before timeout, query may be invalid.**	The query failed and the response is empty.
-4	**Error, reason unknown.**	The query failed and the response is empty.
-5	**Item not found.**	The query failed and the response is empty. The cache query specified a cache table that does not exist.

Servlet Configuration Files

The **rtvquery.war** file contains two files that determine servlet behavior: **web.xml** and **rtvquery.properties**. Within **rtvquery.war**, the paths for these files are **WEB-INF/web.xml** and **WEB-INF/classes/com/sl/rtvquery/rtvquery.properties**, respectively.

rtvquery.properties defines the port number of the RTView Data Server to which the servlet connects. The default value is **3278**. The servlet supports a backup Data Server connection. To specify a backup Data Server connection, open the rtvquery.properties file, located in the **servlets\rtvquery** directory, and add a line as follows:

```
DataServerBackup=host:port
```

where **host:port** are the hostname and port number of the backup Data Server. After editing the file, run the **make_war** script and redeploy the **rtvquery.war** file. Several other servlet properties are also defined in rtvquery.properties. See the comments in that file for a description of each.

rtvquery.properties also contains an optional parameter named **AllowOrigins**. The value you enter into this property is a string that specifies which domains are allowed to send requests to the servlet. By default, **AllowOrigins** is blank, which indicates that requests are allowed only from the same domain that hosts the rtvquery servlet (as in all prior releases). Specify a value of * to indicate that requests are allowed from any domain. For example:

```
AllowOrigins=*
```

Or, you can specify a comma-separated list of the domains that may make requests using the following format:

```
protocol://hostname_or_address:port
```

For example:

```
AllowOrigins=http://192.168.20.11:3456,https://SomeHost:6789
```

web.xml should be edited only if it is necessary to change the servlet authentication. By default, authentication is disabled, so any client can submit queries to the servlet. To enable authentication, edit the **web.xml** file, uncommenting the authentication section at the end of the file, and remake and redeploy the rtvquery.war file. When authentication is enabled, the browser prompts the user for login information when the first request for a session is sent to the servlet. The user must enter a username and password that are valid for the application server (for example, Tomcat). For details, see the comments in the **web.xml** file.

The source for these two files are in **\servlets\rtvquery**. Use the **make_war.bat** and **make_war.sh** scripts to rebuild **rtvquery.war** after changing either file.

JavaScript Library

This section is intended for readers familiar with JavaScript, HTML and Ajax.

The rtvquery Servlet includes a JavaScript library to simplify development of Ajax client applications. Examples of client applications that use the library are available in the **\custom\rtvquery-samples** directory.

Note: Use of the JavaScript library is optional. Alternatively, an application could compose its own HTTP GET requests using the URL and header formats previously described, send them to the rtvquery Servlet using **XMLHttpRequest**, and process the response itself.

The library is contained in a file named **rtvquery.js**. Assuming that the client application is deployed in a web directory that is a sibling of the directory in which the rtvquery Servlet is deployed, the following line would typically be used to load the library into an HTML page:

```
<script src='../rtvquery/rtvquery.js'></script>
```

The library defines a single JavaScript class named rtvQuery. An instance of that class can be created as follows:

```
var rtvquery = new rtvQuery();
```

The **rtvQuery** constructor takes no arguments.

rtvQuery Class Fields

The following fields are defined by the **rtvQuery** class.

Field	Type	Value
responseStatus	Number	The status of the last query. If the HTTP request failed, the value is the HTTP status code (for example, **404**). If the HTTP request succeeded, the value indicates the query status: either **0** for success or a negative error value as described for **x-sl-status** in "Response Status".

responseStatusText	String	The status of the last query. If the HTTP request failed, the value indicates the HTTP status (for example, **404 Not Found**). If the HTTP request succeeded, the value indicates the query status: either OK for success or one of the error messages described for **x-sl-status-text** in "Response Status".
response	String or Object	The response result of the last query. If the HTTP request for the query failed, the value is undefined. If the query succeeded and the requested format is **text**, **xml**, or **xmlrtv** the value is a string. If the format is **js**, **json**, or **jsonp** the response is a JavaScript object. See "Response Status" for details.

rtvQuery Class Functions

The following functions are defined by the **rtvQuery** class.

startQuery(args) Function

Call this function to send a query to the Data Server. The function is returned immediately after the request is sent. The query result is returned asynchronously via the user-defined function specified by the **doneCB** field using arguments. The **startQuery** function expects a single argument containing the following fields.

Field	Description
baseURL	A string indicating the prefix to be prepended to the URL used to access the rtvquery Servlet. For example, if the rtvquery Servlet is located on the same application server as the calling application, the value would typically be **../rtvquery**. The default is **""**.
format	A string indicating the desired response format: **xml**, **json**, **jsonp**, **js**, **xmlrtv**, or text as described in "Response Formats". The default is **xml**.
timeout	The query timeout, in seconds. The default is determined by the rtvquery Servlet properties file and is typically set to **15 seconds**.
doneCB	The function to be called when the query result is received. There is no default value. The **rtvQuery** object that invoked the **startQuery** function is passed as the first (and only) argument to the **doneCB** function. The function can access the query result via the **rtvQuery** response* fields. See "rtvQuery Class Fields" for more information.
noJSConvert	A boolean indicating if a **js**, **json**, or **jsonp** response should be parsed and converted to a JavaScript object. If false, the response is a string. The default is true. Typically this would only be set to false for debugging purposes.

startQuery For Cache Queries

The following fields in the **startQuery** argument pertain to cache queries. The values should not be URL encoded.

Field	Description
cache	The name of the RTView cache. There is no default value.
table	The name of the table. Typically, this is either current or history. The default is **none**.

columns	A string containing the names of the cache table columns. The default is *****. For details, see the **cols** parameter in "Cache Query Optional Parameters".
filterColumns	A string containing the names of the cache table columns to be used to filter the result. There is no default value. For details, see the **fcol** parameter in "Cache Query Optional Parameters".
filterValues	A string containing the values that the filter column must match for a row to be included in the result. No default value. For details, see the **fval** parameter in "Cache Query Optional Parameters".
timeRange	The time range, in seconds, for a history query. The default is **30**. For details, see the **tr** parameter in "Cache Query Optional Parameters".
timeBegin	The begin (minimum) time for a history query. There is no default value. For details, see the **tb** parameter in "Cache Query Optional Parameters".
timeEnd	The end (maximum) time for a history query. There is no default value. For details, see the **te** parameter in "Cache Query Optional Parameters".
rowsPerPage	The maximum number of rows to be returned. There is no default value. For details, see the **rp** parameter in "Cache Query Optional Parameters".
pageNumber	The page number. The default is **0**. For details, see the **pn** parameter in "Cache Query Optional Parameters".

startQuery For SQL Queries

The following fields in the **startQuery** argument pertain to SQL queries. The values should not be URL encoded.

Field	Description
database	The RTView database name. There is no default value. For details, see "SQL Table Queries".
sql	The SQL query string. For details, see "SQL Table Queries".
maxRows	The maximum number of rows to be returned. There is no default value.

startQuery Example

The following HTML page calls **startQuery** to request the current table from a cache named **prod_cache**, in text format, and displays it in a text area component.

```
<html>
 <head>
 <title>Simple cache query using rtvquery servlet</title>
 <script src='../rtvquery/rtvquery.js'></script>
 <script>
    // callback from 'Run Query' button
 function doQuery ()
 {
         document.body.style.cursor = 'wait';
         var ta = document.getElementById('ResultArea');
         ta.value = 'Submitted query, waiting for response ...';
         var rtvquery = new rtvQuery();
         rtvquery.startQuery({
                 baseURL : '../rtvquery',
```

```
                cache : 'prod_cache',
                table : 'current',
                format : 'text',
                doneCB : function(rtvquery) {
                       document.body.style.cursor = 'auto';
                        if (rtvquery.responseStatus == 0) {
                                // query successful, show result
                                ta.value = rtvquery.response;
                        } else {
                                // query failed, show error msg
                                ta.value = rtvquery.responseStatusText;
                        }
                }
        });
}
</script>
</head>
<body>
<button onclick='doQuery()' id='queryButton'>Run Query</button>
<br><br>
<textarea id='ResultArea' wrap='off' cols='80' rows='20' readonly></textarea>
</body>
</html>
```

getCacheNames(args) Function

Call this function to get the names of all RTView caches available from the Data Server. The function returns immediately after the request is sent. The cache names are returned asynchronously via the user-defined function specified by the **doneCB** argument.

The **getCacheNames** function expects a single argument containing the following fields.

Field	Description
baseURL	A string indicating the prefix to be prepended to the URL used to access the rtvquery Servlet. If the rtvquery Servlet is located on the same application server as the calling application, the value typically would be ../rtvquery. The default value is **""**.
doneCB	The function to be called when the result is received. It is called with two arguments. If the query fails, the first argument is null. If the query succeeds, the first argument is a JavaScript array whose first element is an array of the column names, and whose second element is an array of the column types. The column types are the strings int, string, etc. (see "Response Formats"). The rtvQuery object that invoked **getCacheColumns** is passed as the second argument to the function. The function can access the query result via the "rtvQuery Class Fields".

getCacheNames Example

The following script calls to **getCacheNames** populates a drop-down list with the available cache names.

```
var namesQuery = new rtvQuery();
 namesQuery.getCacheNames ({
        baseURL : '../rtvquery',
        doneCB: function(cacheNames, rtvquery) {
                var list = document.getElementById('cacheNameList');
                list.options.length = 0;
                if (!cacheNames || !cacheNames.length) {
```

```
                        alert(rtvquery.responseStatus == 0 ?
                                'no caches found' :
                                rtvquery.responseStatusText);
                } else {
                        for (var i=0; i < cacheNames.length; ++i) {
                                list.options[i] = new Option(cacheNames[i],
                                        cacheNames[i]);
                        }
                }
        }
});
```

getCacheColumns(args) Function

Call this function to get the name and type of the columns in a cache table. The function returns immediately after the request is sent. The column information is returned asynchronously via the user-defined function specified by the **doneCB** argument.

The **getCacheNames** function expects a single argument containing the following fields:

Field	Description
baseURL	A string indicating the prefix to be prepended to the URL used to access the rtvquery Servlet. Typically, if the rtvquery Servlet is located on the same application server as the calling application, the value would be **../rtvquery**. The default value is **""**.
cache	The name of the RTView cache. There is no default value.
table	The name of the table. Typically, this is either current or history.
doneCB	The function to be called when the result is received. It is called with two arguments. If the query fails, the first argument is null. If the query succeeds, the first argument is a JavaScript array whose first element is an array of the column names, and whose second element is an array of the column types. The column types are the strings int, string, etc. (see "Response Formats".) The rtvQuery object that invoked **getCacheColumns** is passed as the second argument to the function. The function can access the query result via the "rtvQuery Class Fields".

getCacheColumns Example

The following script calls to **getCacheColumns** populates a drop-down list with the column names for the current table of a cache named **prod_cache**:

```
var colNamesQuery = new rtvQuery();
 colNamesQuery.getCacheColumns ({
        baseURL : '../rtvquery',
        cache : 'prod_cache',
        table : 'current',
        doneCB: function(colInfo, rtvquery) {
                // colInfo[0] has column names, colInfo[1] has column types
                var colNames = colInfo ? colInfo[0] : null;
                var list = document.getElementById('cacheNameList');
                list.options.length = 0;
                if (!colNames || !colNames.length) {
                        alert(rtvquery.responseStatus == 0 ?
                                'no columns found' :
                                rtvquery.responseStatusText);
                } else {
```

```
                        for (var i=0; i < colNames.length; ++i) {
                                list.options[i] = new Option(cacheNames[i],
                                        cacheNames[i]);
                        }
                }
        }
});
```

Custom Data Adapter

The RTView Custom Data Adapter API lets you create a custom adapter to an external source of data, making the data available for use in RTView applications. The API provides support for event-driven and polled data updates, connection management, and automatic creation of the **Options** Tab, **Connections** Tab, and **Attach To Data** dialog in the Display Builder.

The Custom Data Adapter API is defined by the classes in the com.sl.gmsjrtviewds package, most importantly the **GmsRtViewDs** and **GmsRtViewDsAdapter** classes. Because the data is stored in tabular form, the **GmsTabularData** class in the **com.sl.gmsjrt package** is also considered an important part of the API.

All custom data adapters contain a class that extends the **com.sl.gmsjrtviewds.GmsRtViewDs** class. This class, referred to as the **custom ds** or simply the **ds**, defines the data source in a declarative manner, including the properties of the data source and the formats of the objects used to create data attachments, define connections, and specify options. In addition, the **GmsRtViewDs** class contains methods to store and retrieve data tables.

All custom data adapters also contain a class that extends the **com.sl.gmsjrtviewds.GmsRtViewDsAdapter** class. This class, referred to as the **custom ds adapter** or simply the **adapter**, defines methods that are invoked during the execution of an RTView application, including methods to initialize the data source, start data collection, establish connections, process options, update data attachments, validate information entered into dialogs and shut down the data source.

This section contains the following:

Custom Data Adapter - Data Adapter Characteristics

The Custom Data Adapter API includes features for handling a wide variety of data sources. This section presents some common characteristics of data adapters and discusses the features in the API to support them.

Lifecycle Callbacks

The Data Adapter API provides support for both polled and event-driven data updates through a collection of lifecycle callbacks, invoked as an application runs. A custom adapter may override any of these methods to provide support for polled or event-driven data updates, or a combination of both. The default implementations of these methods do nothing, so a custom adapter is free to implement only the methods it requires.

- The initialize data method, **GmsRtViewDsAdapter.initializeData()**, is called at application startup to allow the custom adapter to perform set-up operations on the external data source.
- The start data method, **GmsRtViewDsAdapter.startData()**, is called after all RTView Data Adapters have been initialized, but before any displays have been loaded. A custom data adapter can override this method to begin data generation.
- The register data method, **GmsRtViewDsAdapter.registerData(GmsRtViewDataObject)**, is called whenever a new data attachment is activated, normally by loading a display. This method is typically used by an event-driven data adapter to register interest in data with the external data source.
- The update data object method, **GmsRtViewDsAdapter.updateDataObject(GmsRtViewDataObject)**, is called once every update period for each active data attachment. This is the primary method that a polled data adapter would override to update data.
- The unregister data method, **GmsRtViewDsAdapter.unregisterData(GmsRtViewDataObject)**, is called whenever the last attachment to a data item is removed, normally by unloading a display. This method is typically used by an event-driven data adapter to remove interest in data with the external data source.
- The terminate data method, **GmsRtViewDsAdapter.terminateData()**, is called when the application shuts down to allow the custom adapter to perform cleanup operations.

In addition to the lifecycle callbacks listed above, an event-driven data adapter would also include an event handler, which would receive notifications of data changes from the external data source and would store the new data.

Data Attachments

In the Display Builder, graphical objects are attached to data retrieved from the data source using the **Attach To Data** dialog. Each data attachment contains enough information to identify which item of data should be accessed and how the data should be stored for later retrieval. The data object class, **com.sl.gmsjrtviewds.GmsRtViewDataObject**, defines a data attachment.

During initialization the Custom Data Adapter defines an exemplar, or template, for its data attachments. On this exemplar, the Custom Data Adapter adds the fields that make up a data attachment and identifies which fields make up the data key. The data key determines how the data is stored. In the Display Builder, the **Attach To Data** dialog is automatically constructed from the information provided in the exemplar.

Connection Management

The Custom Data Adapter API provides support for automatic connection management. The **conninfo** class, **com.sl.gmsjrtviewds.GmsRtViewDsConnInfo**, defines a connection. Each connection has a name and an arbitrary number of fields that define the connection parameters (e.g. host, port, username, password).

During initialization, the Custom Data Adapter defines the information needed for the connection objects by adding fields to an exemplar. In the Display Builder, the **Connections** tab of the **Options** dialog is automatically constructed from the information provided in the exemplar. At runtime, the **GmsRtViewDsAdapter.attemptConnection(GmsRtViewDsConnInfo)** method is invoked to signal that a connection should be made to the external data source. The return status from the method indicates whether the connection attempt was successful. Unsuccessful connection attempts are retried on the next update cycle. Connection definitions are stored in the options file for the Custom Data Adapter, which is specified in the method **GmsRtViewDsProperties.setOptionFileName(String)**.

Data Source Options

The Custom Data Adapter API provides support for custom options which can be used in any way at runtime. The **dsoptions** class, **com.sl.gmsjrtviewds.GmsRtViewDsOptions**, defines the options available to the Custom Data Adapter.

During initialization, the Custom Data Adapter defines these options by adding fields to an exemplar. In the Display Builder, the **Options** tab of the **Options** dialog is automatically constructed from the information provided in the exemplar. At runtime, the **GmsRtViewDsAdapter.applyDsOption(GmsRtViewDsOptions)** method is invoked once for each option. Options are stored in the options file for the Custom Data Adapter, which is specified in the method **GmsRtViewDsBaseProperties.setOptionFileName(String)**.

Row Filtering

In the Display Builder, the **Attach To Data** dialog is automatically created by classes included in the Custom Data Adapter API. Normally this dialog contains fields to enable a row filter on an individual data attachment. A row filter specifies the subset of rows from the data table which are used in a data attachment. The dialog includes a check box, a combo-box to identify the column for row filtering, and a text field to specify the filter value to match.

In some data sources, row filtering is not relevant. An example might be a streamed data source, where each data table has only one record containing the most recent streamed event. To simplify the **Attach To Data** dialog, the Custom Data Adapter may choose to disable row filtering.

The Custom Data Adapter API includes the method **GmsRtViewDsControlProperties.setRowFilteringEnabled(boolean)** to enable or disable the creation of the row filtering fields in the **Attach To Data** dialog. This characteristic must be set at initialization time in the method **GmsRtViewDs.initDsControlProperties(GmsRtViewDsControlProperties)**. This property cannot be changed after initialization.

Column Filtering

In the Display Builder, the **Attach To Data** dialog is automatically created by classes included in the Custom Data Adapter API. Normally this dialog contains fields to enable a column filter on an individual data attachment. A column filter specifies the subset of columns from the data table which are used in a data attachment. The dialog includes a combo-box to select the columns from the data table.

In some data sources, column filtering is not relevant. An example might be a data source which provides time-series data, where all tables include only a timestamp and value. To simplify the **Attach To Data** dialog, the Custom Data Adapter may choose to disable column filtering.

The Custom Data Adapter API includes the method **GmsRtViewDsControlProperties.setColumnFilteringEnabled(boolean)** to enable or disable the creation of the column filtering fields in the **Attach To Data** dialog. This characteristic must be set at initialization time in the method **GmsRtViewDs.initDsControlProperties(GmsRtViewDsControlProperties)**. This property cannot be changed after initialization.

Update Modes

In the Display Builder, the **Attach To Data** dialog is automatically created by classes included in the Custom Data Adapter API. Normally this dialog contains a combo-box to allow the polled update mode to be set on an individual data attachment. The polled update mode controls how frequently the method **GmsRtViewDataAdapter.updateDataObject(GmsRtViewDataObject)** is called while the display is loaded. Available options include polling at the RTView update interval, an interval specific to the Custom Data Adapter, an interval which can be specified for an individual data attachment, only once whenever a display is loaded or reloaded, or only once the first time a display is loaded.

In some data sources, the polled update behavior is not relevant. An example might be a data source that is completely event-driven. In this case, the Custom Data Adapter may choose to disable support for this feature, to simplify the **Attach To Data** dialog.

The Custom Data Adapter API includes the method **GmsRtViewDsControlProperties.enableUpdateModes(Set<UpdateMode>)** to control which update modes are shown in the **Attach To Data** dialog. This characteristic must be set at initialization time in the method **GmsRtViewDs.initDsControlProperties(GmsRtViewDsControlProperties)**. This property cannot be changed after initialization.

Custom Data Adapter - Data Adapter Properties

The methods **GmsRtViewDs.initDsBaseProperties(GmsRtViewDsBaseProperties)** and **GmsRtViewDs.initDsControlProperties(GmsRtViewDsControlProperties)** are invoked during application initialization to define properties and behaviors for the Custom Data Adapter. The **GmsRtViewDsBaseProperties** and **GmsRtViewDsControlProperties** objects contain methods for setting properties and behaviors. Note that all properties and behaviors must be set in the **GmsRtViewDsAdapter.initDsBaseProperties(GmsRtViewDsBaseProperties)** and **GmsRtViewDs.initDsControlProperties(GmsRtViewDsControlProperties)** methods. Properties cannot be changed after initialization.

The base properties are initialized from the **GmsRtViewDs** constructor, before any command-line arguments or options are processed. The control properties are initialized after command-line arguments and options have been processed.

This section describes the properties that can be configured for a Custom Data Adapter. Of the properties, only the DS Key and the Adapter Class Name are required, although we recommend that you set each of the properties to clearly document the intended behavior of your Custom Data Adapter.

Base Property: DS Key (required)

The DS Key, or dskey, is a short string that uniquely identifies the data source. The dskey should not contain any spaces or colon characters, and should not conflict with other dskeys already in use by RTView data adapters (e.g. sql, xml, jms, jmx). SL recommends using a prefix like **custom-** at the beginning of your dskey to avoid conflicts with other RTView dskeys. For example, a custom XML data adapter might use the **custom-xml** dskey.

In an RTView display, each data attachment is saved as a string, known as the **dsstring**. The dskey is the first word in the **dsstring**, and is used to identify the data adapter which will provide the data. The dskey is followed by a series of fields which contain additional information to specify the data that will be accessed.

To set the dskey for a Custom Data Adapter, use the method **GmsRtViewDsBaseProperties.setDsKey(String)**. For example:

```
public class MyDs extends GmsRtViewDs
 {
      protected void initDsBaseProperties ( GmsRtViewDsBaseProperties properties )
      {
      properties.setDsKey( “custom-xml” );
      ...
      }
}
```

Base Property: Adapter Class Name (required)

This property specifies the fully qualified class name of the adapter class for this Custom Data Adapter. The adapter class is a subclass of **com.sl.gmsjrtviewds.GmsRtViewDsAdapter**. This property is set using the method **GmsRtViewDsBaseProperties.setAdapterClassName(String)**. For example:

```
public class MyDs extends GmsRtViewDs
 {
      protected void initDsBaseProperties ( GmsRtViewDsBaseProperties properties )
      {
      // identify the class MyCustomEventDsAdapter in the package
      // com.mydomain.mypackage
      properties.setAdapterClassName(
      “com.mydomain.mypackage.MyCustomEventDsAdapter” );
      }
}
```

Base Property: Custom Data Adapter Description

This property is used to identify the Custom Data Adapter in the RTView About Dialog. If this property is not set, the description will default to RTView DS (Generic). The description is set using the method **GmsRtViewDsBaseProperties.setDsDescription(String)**. For example:

```
public class MyDs extends GmsRtViewDs
 {
      protected void initDsBaseProperties ( GmsRtViewDsBaseProperties properties )
      {
      properties.setDsDescription( “My Custom Event DS” );
      ...
      }
}
```

Base Property: Option File Name

This property identifies the name of the file used to store the Options and Connection definitions for the Custom Data Adapter. If this property is not set, the option file **REVIEWDSOPTIONS.ini** will be used. The option file name is set using the method **GmsRtViewDsBaseProperties.setOptionFileName(String)**. For example:

```
public class MyDs extends GmsRtViewDs
 {
      protected void initDsBaseProperties ( GmsRtViewDsBaseProperties properties )
      {
      properties.setOptionFileName( “MYCUSTOMDSOPTIONS.ini” );
      ...
      }
}
```

Control Property: Row Filtering

This property indicates whether the Custom Data Adapter will permit row filtering in data attachments. Row filtering allows a data attachment to return a subset of rows from a data table by specifying a column and value to match against.

By default, row filtering is enabled. When row filtering is disabled, the **Attach To Data** dialog will not contain the **Filter Rows** check box or the two associated text fields **Filter Column** and **Filter Value**. To enable or disable row filtering, use the method **GmsRtViewDsControlProperties.setRowFilteringEnabled(boolean)**. For example:

```
public class MyDs extends GmsRtViewDs
 {
      protected void initDsControlProperties ( GmsRtViewDsControlProperties properties
      )
      {
      properties.setRowFilteringEnabled( false );
      ...
      }
}
```

Control Property: Column Filtering

This property indicates whether the Custom Data Adapter will permit column filtering in data attachments. Column filtering allows a data attachment to specify which columns will be returned from a data table.

By default, column filtering is enabled. When column filtering is disabled, the **Attach To Data** dialog will not contain the Column(s) combo-box and column chooser. To enable or disable column filtering, use the method **GmsRtViewDsControlProperties.setColumnFilteringEnabled(boolean)**. For example:

```
public class MyDs extends GmsRtViewDs
 {
```

```
      protected void initDsControlProperties ( GmsRtViewDsControlProperties properties
      )
      {
      properties.setColumnFilteringEnabled( false );
      ...
      }
}
```

Control Property: Update Modes

This property indicates the available polled update modes the Custom Data Adapter will allow for data attachments. The following modes are supported:

- **UpdateMode.DS_OPTION_PERIOD**: Poll for data at the interval specified in the **Ds Options** tab for the custom data adapter. If this interval is zero, poll for data at the RTView data update rate.
- **UpdateMode.DATA_ATTACHMENT_PERIOD**: Poll for data at the interval specified for this data attachment. When this update mode is selected, the **Poll Interval** field in the **Attach To Data** dialog is enabled to allow the interval to be specified.
- **UpdateMode.ON_DEMAND_UPDATE**: Poll for data once when the display is loaded, and again whenever the display is reloaded.
- **UpdateMode.STATIC_UPDATE_ONCE**: Poll for data once, when the display is first loaded.

By default, all of the update modes are enabled. A custom data adapter may disable some or all of the update modes using the method **GmsRtViewDsControlProperties.enableUpdateModes(Set<UpdateMode>)**. For example:

```
public class MyDs extends GmsRtViewDs
 {
      protected void initDsControlProperties ( GmsRtViewDsControlProperties properties
      )
      {
      Set<UpdateMode> modes =
      Collections.unmodifiableSet(
      EnumSet.of( UpdateMode.DS_OPTION_PERIOD,
      UpdateMode.ON_DEMAND_UPDATE ));
      properties.enableUpdateModes( modes );
      ...
      }
}
```

For convenience, the GmsRtViewDs base class defines the following sets of update modes:

- **GmsRtViewDs.DEFAULT_UPDATE_MODES**: The set of update modes which are enabled by default.
- **GmsRtViewDs.NO_UPDATE_MODES**: A set containing none of the update modes. A custom data adapter which is completely event-driven and does no polling could use this to disable all polled updates. When no update modes are enabled, the attach to data dialog will not contain the Polling Mode combo-box or the Poll Interval text field.

Control Property: Commands

This property indicates whether the Custom Data Adapter supports commands in data attachments.

By default, commands are disabled. When commands are enabled, the dskey will appear in the list of Data Adapters for the **Define Command** menu option. Selecting the Data Adapter will bring up the **Define Command** dialog. To enable or disable commands, use the method **GmsRtViewDsControlProperties.setCommandsEnabled(boolean)**. For example,

```
public class MyDs extends GmsRtViewDs
 {
      protected void initDsControlProperties ( GmsRtViewDsControlProperties properties
      )
      {
      properties.setCommandsEnabled( true );
      ...
      }
}
```

Control Property: Option Tab Class Name

This property specifies the fully-qualified class name of the **Option** Tab for the data adapter. If this property is not set or the class name provided is null, the default option tab class will be used. The default option tab class, **com.sl.gmsjrtviewds.GmsJRtViewOptionTab**, automatically builds the **Option** tab from the fields defined in the **com.sl.gmsjrtviewds.GmsRtViewDsOptions** exemplar.

Note: This version of the Custom Data Adapter API does not support custom dialog classes. SL recommends using the default class by specifying null for this property, as shown in the following example.

To specify the name of the **Option** tab class, use the method **GmsRtViewDsControlProperties.setOptionTabClassName(String)**. For example,

```
public class MyDs extends GmsRtViewDs
{
      protected void initDsControlProperties ( GmsRtViewDsControlProperties properties
      )
      {
      // use default class
      properties.setOptionTabClassName( null );
      ...
      }
}
```

Control Property: Attach To Data Dialog Class Name

This property specifies the fully-qualified class name of the **Attach To Data** dialog for the data adapter. If this property is not set or the class name provided is null, the default dialog class will be used. The default dialog class, **com.sl.gmsjrtviewds.GmsJRtViewDataDialog**, automatically builds the **Attach To Data** dialog from the fields defined in the **com.sl.gmsjrtviewds.GmsRtViewDataObject** exemplar.

Note: This version of the Custom Data Adapter API does not support custom dialog classes. SL recommends using the default class by specifying null for this property, as shown in the following example.

To specify the name of the **Attach To Data** dialog class, use the method **GmsRtViewDsControlProperties.setAttachDataDialogClassName(String)**. For example:

```
public class MyDs extends GmsRtViewDs
{
      protected void initDsControlProperties ( GmsRtViewDsControlProperties properties
      )
      {
      // use default dialog class
      properties.setAttachDataDialogClassName( null );
      ...
      }
}
```

Control Property: Command Dialog Class Name

This property specifies the fully-qualified class name of the **Command** dialog for the data adapter. If this property is not set or the class name provided is null, the default dialog class, **com.sl.gmsjrtvbuild.GmsJRtViewCommandDialog**, will be used.

Note: This version of the Custom Data Adapter API does not support custom dialog classes. SL recommends using the default class by specifying null for this property, as shown in the following example.

To specify the name of the **Command** dialog class, use the method **GmsRtViewDsControlProperties.setCommandDialogClassName(String)**. For example:

```
public class MyDs extends GmsRtViewDs
{
      protected void initDsControlProperties ( GmsRtViewDsControlProperties properties
      )
      {
      // use default dialog class
      properties.setCommandDialogClassName( null );
      ...
      }
}
```

Control Property: Resource Bundle Name

This property identifies the name of the file that contains localization resources for the data adapter. These resources are used to customize the text displayed in the **Attach To Data** dialog, the **Options** tab, and the **Connections** tab in the Display Builder. If this property is not set or the bundle name is null, the default resource bundle, **com.sl.gmsjrtviewds.rtviewds**, will be used.

To specify the resource bundle name use the method **GmsRtViewDsControlProperties.setResourceBundleName(String)**. For example,

```
public class MyDs extends GmsRtViewDs
 {
      protected void initDsControlProperties ( GmsRtViewDsControlProperties properties
      )
      {
```

```
      // look for localization resources in the
      // file “mycustomds.properties”
      properties.setResourceBundleName( “mycustomds” );
      ...
      }
}
```

Custom Data Adapter - Data Management

The primary responsibility of a Custom Data Adapter is to retrieve data from an external data source and to make the data available for use in RTView applications. This section describes the classes and methods in the Custom Data Adapter API which are used for data management.

In RTView, data is stored in tabular form as a collection of named **com.sl.gmsjrt.GmsTabularData** objects. The table name, or data key, is used by the Custom Data Adapter to store the data. The data key is also used by RTView data attachments to access and display the data.

The Custom Data Adapter defines the data key when it initializes the **com.sl.gmsjrtviewds.GmsRtViewDataObject** class exemplar. The data key then determines the organization of the data tables within RTView.

Data Storage and Retrieval

In the Custom Data Adapter, all data is stored in named tables. The **com.sl.gmsjrt.GmsTabularData** class defines the tabular data format used by the Custom Data Adapter API. This class includes methods to define a table, to add rows and columns, and to set and query values in individual cells.

The **com.sl.gmsjrtview.GmsRtViewDs** class contains several overloaded methods to store tabular data, as well as the method **GmsRtViewDs.getStoredDataTable(String)** to retrieve a data table.

It is also important to note that the Custom Data Adapter stores only current data values. When new data is stored, the previous data is replaced. If necessary, historical data can be saved through the use of the RTView Cache.

Data Key

In the Custom Data Adapter, each data table is identified by a unique name, the **data key**. When data is stored or retrieved, the data key can either be specified directly or determined automatically from the fields of a **data object**. Data objects are described in the section Data Attachments and Data Objects.

The information that determines the data key is specified during initialization, in the method **GmsRtViewDs.initDataObjectExemplar(GmsRtViewDataObject)**. This method defines the fields that make up a data object. After defining the fields, one or more fields are chosen to uniquely identify the data when it is stored, using the method **GmsRtViewDataObject.setDataKeyColumnFields(String[])**. The data key is built from the values of these fields.

The following example shows a data object containing three fields, **SalesRegion**, **Product**, and **Model**. In this example, the **SalesRegion** and **Product** are defined as the data key fields. This means that each unique combination of **SalesRegion** and **Product** will be stored in a new data table. Within each data table, records for any number of Models might be stored.

Consider the following data set:

SalesRegion	Product	Model
East	Radio	A100
East	Radio	A101
East	Heater	A100
South	Radio	A220
South	Radio	A104
South	Radio	A130

The definition of **SalesRegion** and **Product** as the data key would result in three tables:

SalesRegion=East Product=Radio

SalesRegion=East Product=Heater

SalesRegion=South Product=Radio

If instead, we had only defined **SalesRegion** as the data key, the data would have been stored in two tables:

SalesRegion=East

SalesRegion=South

And if we wanted all of the data to be stored in a single table, we could have added a field named **TableName**, designated it as the only data key field, and set its value to **SalesData** in all of our data attachments. We then would have only a single table, with the data key:

TableName=SalesData

In our custom ds class, we define the data key fields:

```
public class MyDs extends GmsRtViewDs
{

public void initDataObjectExemplar (GmsRtViewDataObject exemplar)

{

                                // define the fields that will appear in

                                // the ds string


      exemplar.addField( “SalesRegion” );

      exemplar.addField( “Product” );

      exemplar.addField( “Model” );


                                // define the data key fields – these are

                                // the fields that uniquely identify a
```

```
                                    // data table


        String[] dataKeyFields = new String[]{ “SalesRegion”, “Product” };

        exemplar.setDataKeyColumnFields( dataKeyFields );

    }

    }
```

When we build displays and make data attachments, we specify which **SalesRegion**, **Product**, and **Model** to query. Then, when a display is loaded into RTView, the data attachment (data object) triggers a call to **GmsRtViewDsAdapter.updateDataObject(GmsRtViewDataObject)**. In the Custom Data Adapter, this method would examine the fields in the data object to determine which data is needed, get the data from the data source, put the data into a **GmsTabularData** object, and store the data.

Indexed Data

The Custom Data Adapter API also provides support for data indexing. Index columns are defined during the initialization of the data object exemplar in the method **GmsRtViewDs.initDataObjectExemplar(GmsRtViewDataObject)**. In this method, one or more of the data object fields can be identified as index columns using the method **GmsRtViewDataObject.setIndexColumnFields(String[])**.

When data is stored in an indexed data table, the Custom Data Adapter provides the index values for the new data. The methods **GmsRtViewDs.storeData(GmsRtViewDataObject**, **GmsTabularData, String[])** and **GmsRtViewDs.storeData(String, GmsTabularData, String[])** are used to specify the index values. The third argument in these methods is a String array containing the values for each index column, in the order they were defined in the data object exemplar.

The following example adds a Connection field to the example from the previous section. In this example, the data key remains the same, so that each unique combination of **SalesRegion** and **Product** is stored in a separate table. However, we also define the Connection as an index column. This means that a given table will contain data for several different connections, and a new column called **Connection** will be added to each table:

```
public class MyDs extends GmsRtViewDs
 {

public void initDataObjectExemplar (GmsRtViewDataObject exemplar)

{

                        // define the fields that will appear in

                        // the ds string


        exemplar.addField( “Connection” );

        exemplar.addField( “SalesRegion” );
```

```
        exemplar.addField( “Product” );
        exemplar.addField( “Model” );

                        // define the data key fields—these are
                        // the fields in the dsstring that uniquely
                        // identify a data table

        String[] dataKeyFields = new String[]{ “SalesRegion”, “Product” };
        exemplar.setDataKeyColumnFields( dataKeyFields );

                        // define the index fields—these fields
                        // will automatically be added into new columns
                        // in the data table whenever new data is
                        // received; this additional data can be used
                        // to uniquely identify the source of the data

        String[] indexFields = new String[]{ “Connection” };
        exemplar.setIndexColumnFields( indexFields );
    }
     }
```

In our custom ds adapter class, we specify the index values when we store the data. This example is from a polled data source, where the **GmsRtViewDsAdapter.updateDataObject(GmsRtViewDsObject)** method is called periodically for each active data object. The **SalesRegion** and **Product** specified in the data object will determine which table contains the data (because these two fields were defined to be the data key), and the value of the **Connection** will be added to the table.

```
    public class MyDsAdapter extends GmsRtViewDsAdapter
     {
    @Override
    protected void updateDataObject (GmsRtViewDataObject dataObject)
    {
                        // determine the connection, sales region,
                        // product, and model required for this
```

```
                        // data object


        String conn = dataObject.getFieldValue( “Connection” );

        String region = dataObject.getFieldValue( “SalesRegion” );

        String product = dataObject.getFieldValue( “Product” );

        String model = dataObject.getFieldValue( “Model” );


                        // get the data from the external

                        // data source


        GmsTabularData table = getSalesData(conn, region, product, model);


                    // store the data, indexed by Connection


        String[] indexValues = new String[] { conn };

        ds.storeData( dataObject, table, indexValues );

    }

     }
```

In an event-driven data source, an event handler would typically be invoked when data is changed. The following example shows how a custom adapter might store indexed data when an event is received. Note that in this example we do not have a **GmsRtViewDataObject**, so we must build the data key to identify the data table from the values sent to the event handler:

```
    public class MyDsAdapter extends GmsRtViewDsAdapter
     {

    private void onDataChanged (String conn, String region,

                                String product, GmsTabularData table)

    {

                        // Using the SalesRegion and Product,

                        // build the data key to identify the

                        // table to be stored
```

```
        String[] dataKeyValues = new String[] { region, product };

        String dataKey = ds.buildDataKey( dataKeyValues );


                        // store the data, indexed by Connection


        String[] indexValues = new String[] { conn };

        ds.storeData( dataKey, table, indexValues );

    }

     }
```

Data Attachments and Data Objects

In the Display Builder, the **Attach To Data** dialog allows you to bind a property of an RTView object to a table or to an element within a table. In the Custom Data Adapter API, each data attachment is represented by a data object, an instance of the **com.sl.gmsjrtviewds.GmsRtViewDataObject** class.

The data object is a general purpose object that contains an arbitrary number of named fields. A Custom Data Adapter configures the data object at initialization time by adding fields to a class exemplar. These fields are used to automatically build the **Attach To Data** dialog.

Data Object Exemplar

A Custom Data Adapter contains a single data object exemplar, which serves as a template for all of its data attachments. The data object exemplar is initialized in the method **GmsRtViewDs.initDataObjectExemplar(GmsRtViewDataObject)**. In this method, the Custom Data Adapter calls **GmsRtViewDataObject.addField()**, **GmsRtViewDataObject.setDataKeyColumnFields()**, and optionally **GmsRtViewDataObject.setIndexColumnFields()** to define the exemplar.

Note: Data object field names must not begin with two underscore characters (i.e. __) because this prefix is reserved for internal use by RTView.

In the following example we define a data object containing the fields **Plant Name**, **Tag Name**, and **Retrieval Method**. The **Attach To Data** dialog will display a combo-box for each of these fields. We designate the **Plant Name** as the data key, so we will have a separate table for each **Plant**. Finally, we designate the **Tag Name** as an index column. RTView will add a **Tag Name** column to the data table when it is stored, if a column named **Tag Name** does not already exist.

```
public class MyDs extends GmsRtViewDs
{

public void initDataObjectExemplar (GmsRtViewDataObject exemplar)

{
```

```
                        // define the fields that will appear in
                        // the ds string

        exemplar.addField( “Plant Name” );
        exemplar.addField( “Tag Name” );
        exemplar.addField( “Retrieval Method” );

                        // define the data key fields—these are
                        // the fields in the dsstring that uniquely
                        // identify a data table

        String[] dataKeyFields = new String[]{ “Plant Name” };
        exemplar.setDataKeyColumnFields( dataKeyFields );

                        // define the index fields—these fields
                        // will automatically be added into new columns
                        // in the data table whenever new data is
                        // received; this additional data can be used
                        // to uniquely identify the source of the data

        String[] indexFields = new String[]{ “Tag Name” };
        exemplar.setIndexColumnFields( indexFields );
    }
     }
```

Scrambled Fields

The Custom Data Adapter API provides support for scrambled fields within the data object. Scrambled fields are not stored in plain-text, so they are useful for passwords or other sensitive information. To add a scrambled field, use the method **GmsRtViewDataObject.addScrambledField(String)** when initializing the data object exemplar. For example,

```
public class MyDs extends GmsRtViewDs
 {
```

```
public void initDataObjectExemplar (GmsRtViewDataObject exemplar)

{

                        // define the fields that will appear in

                        // the ds string


      exemplar.addField( “user” );

      exemplar.addScrambledField( “password” );

      ...

}

 }
```

Localization

The labels that are displayed for each field in the **Attach To Data** dialog can be configured by adding properties to the Resource File. For information about specifying the resource file, please refer to the section Resource Bundle Name.

The property name is formed by appending the field name to the string **dataKeyField_**. For example, the property name for the user field would be **dataKeyField_user**. Here is an example of the entries in the resource file for the data object defined above:

```
dataKeyField_user=User Name:
dataKeyField_password=Password:
```

In addition to the properties for custom data fields, the resource file also contains entries for the other buttons and labels in the **Attach To Data** dialog:

```
# Attach To Data Dialog
# The Property Name label and the OK, Apply, Reset, Clear and Cancel
# buttons are shared by all Attach to Data dialogs. The localized
# values for these components are defined in rtview.properties.
# The Select Columns dialog which comes up when the user selects
# Select Columns is also shared by all of the Attach To Data dialogs.
# The localized values for this dialog’s components are defined in
# rtview.properties.

d_AttachToCustomData_Title=Attach To Custom Event DS Data
d_AttachToCustomData_Label_Column=Column(s):
d_AttachToCustomData_Label_FilterRows=Filter Rows:
d_AttachToCustomData_Label_FilterColumn=Filter Column:
d_AttachToCustomData_Label_FilterValue=Filter Value:
d_AttachToCustomData_Label_UpdateOnce=Update Once:
d_AttachToCustomData_Popup_SelectColumns=Select Columns
```

DS String

When a data object is stored in an RTView display, it is serialized into a string called the dsstring. The Custom Data Adapter API generates dsstrings with the following format:

```
dskey {token1 token2 token3}
```

where **dskey** is the unique identifier defined in the **GmsRtViewDs.initDsBaseProperties(GmsRtViewDsBaseProperties)** method and each token represents a field in the data object. The tokens have the following format:

```
fieldname=fieldvalue
```

where **fieldname** is a URL-encoded representation of the field name, and **fieldvalue** is a URL-encoded and optionally scrambled representation of the field value.

Note that RTView uses field names beginning with two underscore characters to store internal properties of a data object, including row and column filtering information, update once behavior, and the encoding format.

Dialog Field Type

By default, each field defined in the data object is represented by a combo-box in the **Attach To Data** dialog. You can choose a different UI element when adding a field by using the overloaded forms of the **GmsRtViewDataObject.addField()** and **GmsRtViewDataObject.addScrambledField()** methods.

In the following example, the server will appear as a text field, the query will appear as a multi-line text area, the user will appear as a combo-box (the default), and the password will appear as a password text field.

Note: Currently the combo-box, text field, text area, and password text field are the only dialog field types supported.

```
public class MyDs extends GmsRtViewDs
{

public void initDataObjectExemplar (GmsRtViewDataObject exemplar)

{

                        // define the fields that will appear in

                        // the dsstring


      exemplar.addField( “server”,  DialogFieldType.FIELD_TEXTFIELD );

      exemplar.addField( “query”,  DialogFieldType.FIELD_TEXTAREA );

      exemplar.addField( “user” );

      exemplar.addScrambledField( “password”,

                              DialogFieldType.FIELD_PASSWORD );

      ...

}

 }
```

Accessing Data Objects

The **com.sl.gmsjrtviewds.GmsRtViewDs** class maintains a collection of active data objects. Data objects are added to the collection when a display is loaded, and freed when there are no more references to them. Data objects are passed to the lifecycle methods **GmsRtViewDsAdapter.registerData(GmsRtViewDataObject)**, **GmsRtViewDsAdapter.unregisterData(GmsRtViewDataObject)**, **GmsRtViewDataAdapter.updateDataObject(GmsRtViewDataObject)**, and **GmsRtViewDataAdapter.validate(GmsRtViewDataObject)**.

You can query the value of a field in a data object using the method **GmsRtViewDataObject.getFieldValue(String)**, and you can set the value of a field using the method **GmsRtViewDataObject.setFieldValue(String, String)**. Field values are normally only set during validation of the information entered into the **Attach To Data** dialog. For more information, please refer to the Dialogs section.

The following example shows how an adapter for a polled data source would query the fields of the data object to determine the data to retrieve from the external data source:

```
public class MyDsAdapter extends GmsRtViewDsAdapter
 {

@Override

protected void updateDataObject (GmsRtViewDataObject dataObject)

{

                        // determine the data required for this

                        // data object


      String server = dataObject.getFieldValue( “server” );

      String query = dataObject.getFieldValue( “query” );

      String user = dataObject.getFieldValue( “user” );

      String password = dataObject.getFieldValue( “password” );


      // NOTE: You would get the data from the external data source

      // here...


      GmsTabularData table = getData(server, query, user, password);


                        // store the data


      ds.storeData( dataObject, table );
```

```
}

 }
```

Custom Data Adapter - Connections

The Custom Data Adapter API provides support for named connections through the **com.sl.gmsjrtviewds.GmsRtViewDsConnInfo**, or **conninfo** object. Like the data object, the **conninfo** object is a general purpose object that you configure at initialization time by adding fields. The **conninfo** object contains one predefined field, the name of the connection. This section describes **conninfo** objects and the features in the Custom Data Adapter API to support them.

Conninfo Object Exemplar

A Custom Data Adapter contains a single **conninfo** object exemplar, which serves as a template for all named connections. The **conninfo** exemplar is configured in the method **GmsRtViewDs.initConnInfoExemplar(GmsRtViewDsConnInfo)**, which calls **GmsRtViewDsConnInfo.addField(String)** and **GmsRtViewDsConnInfo.addScrambledField(String)** to add fields to the exemplar. For a discussion of plain-text and scrambled fields, please refer to the "Data Object Exemplar" section.

If your Custom Data Adapter does not require the use of named connections, just add an empty **GmsRtViewDs.initConnInfoExemplar(GmsRtViewDsConnInfo)** method and return without adding any fields. The **Connections** tab in the **Options** dialog will not be created. For example,

```
public class MyDs extends GmsRtViewDs
 {

public void initConnInfoExemplar (GmsRtViewDsConnInfo exemplar)

{

      // this Custom Data Adapter does not use

      // named connections



      return;

}

 }
```

If your Custom Data Adapter uses connections, then configure the exemplar by adding fields.

Note: Field names must not begin with two underscore characters (i.e. __) because this prefix is reserved for internal use by RTView.

In the following example, we define a **conninfo** object containing the fields host, port, user, and password. The password field is defined to be a scrambled field. We also specify the UI element to be used for each field in the **Connections** tab of the **Options** dialog. The host field will appear as a combo-box (the default UI element), the port and user fields will appear as text fields, and the password field will appear as a password text field.

```
public class MyDs extends GmsRtViewDs
 {

public void initConnInfoExemplar (GmsRtViewDsConnInfo exemplar)

{

                        // define the fields that will appear in

                        // the conninfo object



      exemplar.addField( “host” );

      exemplar.addField( “port”, DialogFieldType.FIELD_TEXTFIELD );

      exemplar.addField( “user”, DialogFieldType.FIELD_TEXTFIELD );

      exemplar.addScrambledField( “password”,

                              DialogFieldType.FIELD_PASSWORD );

}

 }
```

Localization

The labels that are displayed for each field in the **Connections** tab can be configured by adding properties to the Resource File. For information about specifying the resource file, please refer to the “Control Property: Resource Bundle Name” section.

The property name is formed by appending the field name to the string **connInfoField_**. For example, the property name for the host field would be **connInfoField_host**. Here is an example of the entries in the resource file for the **conninfo** object defined above:

```
connInfoField_host=Host Name:
connInfoField_port=Port Number:
connInfoField_user=User Name:
connInfoField_password=Password:
```

In addition to the properties for custom **conninfo** fields, the resource file also contains entries for the other buttons and labels on the **Connection** tab and the **Connection Setup** dialog:

```
# Connection Tab
d_Options_CustomDs_Connection_TabName=Connections
d_Options_CustomDs_Label_DefaultConnection=Default Connection:
d_Options_CustomDs_Button_Add=Add Connection
d_Options_CustomDs_Button_Remove=Remove Connection
d_Options_CustomDs_Label_Connections=Connections:

# Connection Setup Dialog
d_Options_CustomDs_AddConn_Title= Add Connection
```

```
d_Options_CustomDs_AddConn_Label_ConnName=Connection Name:
d_Options_CustomDs_AddConn_Button_OK=OK
d_Options_CustomDs_AddConn_Button_Clear=Clear
d_Options_CustomDs_AddConn_Button_Reset=Reset
d_Options_CustomDs_AddConn_Button_Cancel=Cancel
d_Options_CustomDs_AddConn_Button_Help=Help
```

Conninfo String

The **conninfo** objects defined in the **Connections** tab of the **Options** dialog are stored as strings in the Options File. For details about specifying the options file, please refer to the "Base Property: Option File Name" section.

Each **conninfo** object is written to the options file with the following format:

```
<dskey>conn token1 token2 token3 ...
```

where **<dskey>** is the unique identifier defined in the **GmsRtViewDs.initDsProperties(GmsRtViewDsProperties)** method and each token represents a field in the **conninfo** object. The tokens have the following format:

```
fieldname=fieldvalue
```

where **fieldname** is a URL-encoded representation of the field name, and **fieldvalue** is a URL-encoded and optionally scrambled representation of the field value.

Note: RTView uses field names beginning with two underscore characters to store internal properties of a conninfo object.

The following example shows a **conninfo** object named **conn2** in a Custom Data Adapter with the **custom-eventds** dskey. The connection refers to the host **server1** and port **3333**:

```
custom-eventdsconn __name=conn2 host=server1 port=3333
```

Connection Management

Establishing Connections

At program startup, the method **GmsRtViewDsAdapter.attemptConnection(GmsRtViewDsConnInfo)** is invoked for each defined connection. Each connection attempt is made on a different thread.

The **attemptConnection** method returns a **ConnectStatus** indicating whether or not the attempt was successful. There are three possible values:

ConnectStatus.SUCCEEDED	The connection attempt was successful.
ConnectStatus.FAILED	The connection attempt failed (this time).
ConnectStatus.INVALID	The connection is invalid as configured. It will never connect.

If the adapter returns **ConnectStatus.FAILED**, the **attemptConnection** method will be invoked again to try to establish the connection. If the adapter returns **ConnectStatus.INVALID**, the **conninfo** is assumed to be configured improperly, and the **attemptConnection** method will not be invoked again.

Here is an example of the **attemptConnection** method for the **conninfo** object defined above:

```
public class MyDsAdapter extends GmsRtViewDsAdapter
 {

@Override

protected ConnectStatus attemptConnection (GmsRtViewDsConnInfo connInfo)

{

      String host = connInfo.getFieldValue( “host” );

      String port = connInfo.getFieldValue( “port” );

      String user = connInfo.getFieldValue( “user” );

      String password = connInfo.getFieldValue( “password” );


      if (host == null || host.length() == 0 ||

                        port == null || port.length() == 0) {

            return ConnectStatus.INVALID;

      }


      // NOTE: your “openConnection” method would establish a

      // connection to your external data source


      boolean success = openConnection(host, port, user, password);


      if (success) {

            return ConnectStatus.SUCCEEDED;

      } else {

            return ConnectStatus.FAILED;

      }

}

 }
```

Closing Connections

At program termination, the **GmsRtViewDsAdapter.closeConnection(GmsRtViewDsConnInfo)** method is invoked for each open connection. A Custom Data Adapter should override this method to close the connection to the external data source. For example,

```
public class MyDsAdapter extends GmsRtViewDsAdapter
 {

@Override

protected void closeConnection (GmsRtViewDsConnInfo connInfo)

{

      String host = connInfo.getFieldValue( “host” );

      String port = connInfo.getFieldValue( “port” );

      String user = connInfo.getFieldValue( “user” );

      String password = connInfo.getFieldValue( “password” );


      // NOTE: your “closeConnection” method would close the

      // connection to your external data source


      closeConnection(host, port, user, password);


      return;

}

 }
```

A Custom Data Adapter can also inform RTView that a connection has been closed by calling the method **GmsRtViewDsConnInfo.setDisconnected()**. RTView will then try to re-establish the connection as described above in the section Establishing Connections. The following example shows how a custom adapter for a polled data source might detect a broken connection when trying to update data:

```
public class MyDsAdapter extends GmsRtViewDsAdapter
 {

@Override

protected void updateDataObject (GmsRtViewDataObject dataObject)

{

                              // get connection name from the

                              // data object
```

```
String connName = dataObject.getFieldValue( “conn” );
if (connName = null || connName.length() == 0) {
      return;
}
                        // get conninfo object and extract
                        // fields

GmsRtViewDsConnInfo connInfo = ds.getConnInfoFromName( connName );
if (connInfo == null) {
      return;
}

String host = connInfo.getFieldValue( “host” );
String port = connInfo.getFieldValue( “port” );
String user = connInfo.getFieldValue( “user” );
String password = connInfo.getFieldValue( “password” );

// NOTE: your “isConnected” method would check the
// connection to your external data source

boolean connected = isConnected(host, port, user, password);

                        // if not connected, alert DS
                        // to attempt to reconnect
if (!connected) {
      connInfo.setDisconnected();
      return;
}
```

```
                                    // get data...

            ...

      }

       }
```

Custom Data Adapter - Data Source Options

The Custom Data Adapter API provides support for user-supplied options through the **com.sl.gmsjrtviewds.GmsRtViewDsOptions**, or **ds options object**. Like the data object and **conninfo** object, the **ds options object** is a general purpose object that you configure at initialization time by adding fields. However, unlike those objects, there is only one **ds options object** for the entire Custom Data Adapter. All options are stored as fields on this object.

This section describes the **ds options** object and the features in the Custom Data Adapter API to support it.

DS Options Object Exemplar

A Custom Data Adapter contains a single **ds options object exemplar**, which is configured in the method **GmsRtViewDs.initDsOptionsExemplar(GmsRtViewDsOptions)**. In this method, the Custom Data Adapter uses the methods **GmsRtViewDsOptions.addField(String)** and **GmsRtViewDsOptions.addScrambledField(String)** to add fields to the exemplar. For a discussion of plain-text and scrambled fields, please refer to the "Data Object Exemplar" section.

If your Custom Data Adapter does not require any user-supplied options, just add an empty **GmsRtViewDs.initDsOptionsExemplar(GmsRtViewDsOptions)** method and return without adding any fields. The **Options** tab in the **Options** dialog will not be created. For example:

```
public class MyDs extends GmsRtViewDs
 {

public void initDsOptionsExemplar (GmsRtViewDsOptions exemplar)

{

      // this Custom Data Adapter does not define any

      // user-supplied options



      return;

}

 }
```

If your Custom Data Adapter requires user-supplied options, then configure the exemplar by adding fields.

Note: Field names must not begin with two underscore characters (i.e. __) because this prefix is reserved for internal use by RTView.

In the following example, we define a ds options object containing three fields. The **defaultUser** and **defaultPassword** fields will be used for connection management, in case the user leaves these fields blank when defining **conninfo** objects. The field **maxPoints** will be used for data management, to limit the size of data returned from the external data source:

```
public class MyDs extends GmsRtViewDs
 {

public void initDsOptionsExemplar (GmsRtViewDsOptions exemplar)

{

                        // define the fields that will appear in

                        // the ds options object


      exemplar.addField( “defaultUser”, DialogFieldType.FIELD_TEXTFIELD );

      exemplar.addScrambledField( “defaultPassword”,

                                  DialogFieldType.FIELD_PASSWORD );

      exemplar.addField( “maxPoints”, DialogFieldType.FIELD_TEXTFIELD );


}

 }
```

Localization

The labels that are displayed for each field in the **Options** tab can be configured by adding properties to the Resource File. For information about specifying the resource file, please refer to the “Control Property: Resource Bundle Name” section.

The property name is formed by appending the field name to the string **dsOptionsField_**. For example, the property name for the **defaultUser** field would be **dsOptionsField_defaultUser**. Here is an example of the entries in the resource file for the **ds options object** defined above:

```
dsOptionsField_defaultUser=Default User Name:
dsOptionsField_defaultPassword=Default Password:
dsOptionsField_maxPoints=Maximum Number of Data Points:
```

In addition to the properties for custom ds options fields, the resource file also contains entries for the label on the **Options** tab, and for a Dummy Tab Message. The dummy tab message is displayed only if you do not define any fields in both the ds options and **conninfo** object exemplars:

```
# Options Tab
d_Options_CustomDs_Options_TabName=Options
# Dummy Tab message
d_Options_CustomDs_Label_NoOptions=This data source does not \
 have any customizable options.
```

DS Option Strings

All DS Options are stored in the Options File. For details about specifying the options file, please refer to the "Base Property: Option File Name" section.

Each field in the **ds options object** is written to the options file on a separate line, using the following format:

```
<dskey>option token1
```

where **<dskey>** is the unique identifier defined in the **GmsRtViewDs.initDsBaseProperties(GmsRtViewDsBaseProperties)** method and the token represents a field in the ds options object. The token has the following format:

```
fieldname=fieldvalue
```

where **fieldname** is a URL-encoded representation of the field name, and **fieldvalue** is a URL-encoded and optionally scrambled representation of the field value.

The following example shows the ds options defined above, in a Custom Data Adapter with the **dskey custom-eventds**. The **defaultPassword** field is scrambled:

```
custom-eventdsoption defaultUser=sales
custom-eventdsoption defaultPassword=8i539k5fa47e864l75032
custom-eventdsoption maxPoints=1000
```

Accessing DS Options

During initialization, each DS Option is passed to the custom ds adapter in a separate call to the method **GmsRtViewDsAdapter.applyDsOption(String, String)**. A Custom Data Adapter would override this method to process the options for later use. The return value from this method indicates whether the option was successfully processed.

```
public class MyDsAdapter extends GmsRtViewDsAdapter
 {

@Override

protected boolean applyDsOption (String name, String value)

{

      if (name.equals( "defaultUser" )) {

            if (value.length() == 0) return false;

            defaultUserName = value;

            return true;
```

```
        } else if (name.equals( “defaultPassword” )) {
              if (value.length() == 0) return false;
              defaultPassword = value;
              return true;

        } else if (name.equals( “maxPoints” )) {
              if (value.length() == 0) return false;
              try {
                    maxPoints = Integer.valueOf( value );
              } catch( NumberFormatException nfe ) {
                    return false;
              }
              return true;
        }
              return false;
  }
   }
```

Custom Data Adapter - Dialogs

The Custom Data Adapter API includes classes to automatically create dialogs for the objects defined by the Custom Data Adapter. The **Attach To Data** dialog is created for the **com.sl.gmsjrtviewds.GmsRtViewDataObject**, the **Connections** tab is created for the **com.sl.gmsjrtviewds.GmsRtViewDsConnInfo** object, and the **Options** tab is created for the **com.sl.gmsjrtviewds.GmsRtViewDsOptions** object.

This section describes the features available in the Custom Data Adapter API to control the appearance and contents of these dialogs. The procedures for configuring the dialogs and validating user input are almost identical for each of the objects, so the concepts and techniques described in each subsection may be applied to any of the dialogs, except where noted.

Dialog Field Types

In the dialog, each field in an object is represented by a Swing control and a corresponding label. The default control for a field is a combo-box, but this may be changed by specifying the dialog field type (**com.sl.gmsjrtview.DialogFieldType**) when adding the field to the exemplar during initialization.

The Custom Data Adapter API currently supports the following dialog field types:

DialogFieldType.FIELD_COMBOBOX	Combo-box
DialogFieldType.FIELD_TEXTFIELD	Single-line text field
DialogFieldType.FIELD_TEXTAREA	Multi-line text field
DialogFieldType.FIELD_PASSWORD	Text field for entering passwords

In the following example, a **conninfo** object is defined and dialog field types are specified as fields are added. In the **Connections** tab, the host field will be represented by a combo-box, user will be a text field, and password will be a password text field:

```
public class MyDs extends GmsRtViewDs
 {

public void initConnInfoExemplar (GmsRtViewDsConnInfo exemplar)

{

                        // define the fields that will appear in

                        // the conninfo object


      exemplar.addField( “host” );

      exemplar.addField( “user”, DialogFieldType.FIELD_TEXTFIELD );

      exemplar.addScrambledField( “password”,

                              DialogFieldType.FIELD_PASSWORD );

}

 }
```

Default Values

When the exemplar is defined, default values can be specified for each field. The following example sets default values for each of the fields in the **conninfo object exemplar**:

```
public class MyDs extends GmsRtViewDs
 {

public void initConnInfoExemplar (GmsRtViewDsConnInfo exemplar)

{

                        // define the fields that will appear in

                        // the conninfo object


      exemplar.addField( “host” );
```

```
        exemplar.addField( “user”, DialogFieldType.FIELD_TEXTFIELD );

        exemplar.addScrambledField( “password”,

                                DialogFieldType.FIELD_PASSWORD );


                        // set default values for each field

        exemplar.setFieldValue( “host”, “SERVER1” );

        exemplar.setFieldValue( “user”, “guest” );

        exemplar.setFieldValue( “password”, “guest” );

    }

     }
```

Field Choices

A default list of choices for a combo-box can be specified when the exemplar is defined, if the choices are known at initialization time. If the choices are not known, for example if they depend on a value entered into another field, the list of choices can also be set during dialog validation, as described in the “Validation” section.

The following example initializes a field named **retrievalMode** to contain three choices: **Current Data**, **Historical Data**, and **Current and Historical Data**. Because these choices are set on the **data object exemplar**, they will appear in the **retrievalMode** combo-box for each data attachment:

```
import java.util.Set;
import java.util.LinkedHashSet;
public class MyDs extends GmsRtViewDs
 {
static final Set<String> retrievalModeChoices;
static {

        retrievalModeChoices = new LinkedHashSet<String>();

        retrievalModeChoices.add( “Current Data” );

        retrievalModeChoices.add( “Historical Data” );

        retrievalModeChoices.add( “Current and Historical Data” );
}
public void initDataObjectExemplar (GmsRtViewDataObject exemplar)

{

                        // define the fields that will appear in

                        // the ds string, in order

        exemplar.addField( “conn” );

        exemplar.addField( “pointName”, DialogFieldType.FIELD_TEXTFIELD );
```

```
        exemplar.addField( “retrievalMode” );

                        // set the list of choices for the

                        // retrievalMode combo-box

        exemplar.setFieldChoices( “retrievalMode”, retrievalModeChoices );
}

 }
```

UI State

The UI state controls whether a control is visible, invisible, disabled, or editable. By default, each field in a dialog is visible and editable. The UI state is typically set during data validation, but the UI state for a field can also be changed in the exemplar during initialization to provide a different default state.

The UI State can be set to the following values:

DialogFieldState.HIDDEN	Field is not visible.
DialogFieldState.DISABLED	Field is grayed-out
DialogFieldState.ENABLED	Field is visible but not editable
DialogFieldState.EDITABLE	Field is editable
DialogFieldState.UNKNOWN	The UI state has not been set

In the following example, the default UI state for the **pointName** and **retrievalMode** fields is set to disabled. When the **Attach To Data** dialog is displayed, these fields will be grayed-out. Presumably, the validate method in this Custom Data Adapter would make these fields editable when a valid connection was chosen:

```
public class MyDs extends GmsRtViewDs
 {

public void initDataObjectExemplar (GmsRtViewDataObject exemplar)

{

                        // define the fields that will appear in

                        // the ds string, in order

        exemplar.addField( “conn” );

        exemplar.addField( “pointName”, DialogFieldType.FIELD_TEXTFIELD );

        exemplar.addField( “retrievalMode” );

                        // disable (gray-out) the pointName and

                        // retrievalMode fields

        exemplar.setFieldUIState( “pointName”, DialogFieldState.DISABLED );

        exemplar.setFieldUIState( “retrievalMode”,
```

```
                                        DialogFieldState.DISABLED );
    }
     }
```

Field Validity

The field validity controls the background color of the field in the dialog. Valid fields are white, invalid fields are red, and unknown fields are gray. The field validity can be set to the following values:

DataFieldValidity.GOOD	Field is valid
DataFieldValidity.BAD	Field is invalid
DataFieldValidity.UNKNOWN	Field validity is unknown

In the following example, a data object is validated in the **GmsRtViewDsAdapter.validate(GmsRtViewDataObject)** method. If the **retrievalMode** field has not been set to one of the allowable values, the field is marked as invalid:

```
import java.util.Set;
import java.util.LinkedHashSet;
public class MyDsAdapter extends GmsRtViewDsAdapter
 {
static final Set<String> retrievalModes;
static {

      retrievalModes = new LinkedHashSet<String>();

      retrievalModes.add( “Current Data” );

      retrievalModes.add( “Historical Data” );

      retrievalModes.add( “Current and Historical Data” );
}
@Override
protected void validate (GmsRtViewDataObject dataObject)
{

      String mode = dataObject.getFieldValue( “retrievalMode” );

      if (mode.length() > 0) {

            if (retrievalModes.contains(mode))

                  dataObject.setFieldValidity( “retrievalMode”,

                                          DataFieldValidity.GOOD );

            else

                  dataObject.setFieldValidity( “retrievalMode”,

                                          DataFieldValidity.BAD );

      }
```

```
      ...

}
}
```

Handling Substitutions in Fields

When validating objects, you may need to apply substitutions in order to determine whether a field is valid. For example, you might find the value **$conn** entered into a connection field. If a substitution is available for **$conn**, you can get the substituted value by using the **getFieldSubstitutionValue()** method. For example:

```
public class MyDsAdapter extends GmsRtViewDsAdapter
 {
@Override
protected void validate (GmsRtViewDataObject dataObject)
{
                        // get the value entered into the conn

                        // field after applying substitutions

      String conn = dataObject.getFieldSubstitutionValue( “conn” );

      GmsRtViewDsConnInfo conninfo = ds.getConnInfoFromName( conn );

      ...
}
}
```

Validation

The **GmsRtViewDsAdapter** class provides a **validate()** method for each of the automatically generated dialogs. A custom subclass can override these methods to add validation logic. Using methods in the Custom Data Adapter API, the validation methods can query and set field values, modify the UI state to hide or gray-out a field, update the available field choices, or set the field validity to change the background color of the field.

Data Object Validation

When a data object is edited in the **Attach To Data** dialog, the **GmsRtViewDsAdapter.validate(GmsRtViewDataObject)** method is called when the dialog is first displayed, whenever the focus changes, and when the dialog is dismissed. A subclass can override this method to perform custom validation.

In the following example, we query the value of the server field. If the user has entered a server name, we check to see whether the server name is valid, and set the field validity. If the server name is valid, we also set the available choices for the **pointName** combo-box.

```
public class MyDsAdapter extends GmsRtViewDsAdapter
 {
@Override
protected void validate (GmsRtViewDataObject dataObject)
{

      String server = dataObject.getFieldValue( “server” );
```

```
        if (server == null || server.length() == 0)
              return;
        // you would check whether the server name was valid here ...
        boolean isValid = MYCLASS.isServerValid(server);
        if (isValid) {
              dataObject.setFieldValidity( “server”,
                                    DataFieldValidity.GOOD );
              // ... you would get the available point names
              // for the server here ...
              Set<String> pointNames = MYCLASS.getPointNames(server);
              dataObject.setFieldChoices( “pointName”, pointNames );
        } else {
              dataObject.setFieldValidity( “server”,
                                    DataFieldValidity.BAD );
        }
  }
   }
```

Conninfo Object Validation

When a **conninfo object** is edited in the **Connections** tab, the **GmsRtViewDsAdapter.validate(GmsRtViewDsConnInfo)** method is called when the dialog is first displayed, whenever the focus changes, and when the dialog is dismissed. A subclass can override this method to perform custom validation. In the following example, we check that the value in the port field is numeric:

```
public class MyDsAdapter extends GmsRtViewDsAdapter
 {
@Override
protected void validate (GmsRtViewDsConnInfo connInfo)
{
      String port = connInfo.getFieldValue( “port” );
      DataFieldValidity validity = DataFieldValidity.GOOD;
      if (port.length() > 0) {
            try {
                  int i = Integer.valueOf(port);
```

```
                } catch(NumberFormatException nfe) {

                      validity = DataFieldValidity.BAD;

                }

          }

          connInfo.setFieldValidity( “port”, validity );

    }
    }
```

DS Options Object Validation

When the **ds options object** is edited in the **Options** tab, the **GmsRtViewDsAdapter.validate(GmsRtViewDsOptions)** method is called when the dialog is first displayed, whenever the focus changes, and when the dialog is dismissed. A custom ds adapter can override this method to perform custom validation. In the following example, we check that the value in the **maxPoints** field is a positive number:

```
public class MyDsAdapter extends GmsRtViewDsAdapter
 {
@Override
protected void validate (GmsRtViewDsOptions dsOptions)
{

      String maxPoints = dsOptions.getFieldValue( “maxPoints” );

      DataFieldValidity validity = DataFieldValidity.GOOD;

      if (maxPoints.length() > 0) {

            try {

                  int i = Integer.valueOf(maxPoints);

                  if (i <= 0)

                        validity = DataFieldValidity.BAD;

            } catch(NumberFormatException nfe) {

                  validity = DataFieldValidity.BAD;

            }

      }

      dsOptions.setFieldValidity( “maxPoints”, validity );
}

 }
```

Localization and Customization

In each of the dialogs, the field label defaults to the field name. These labels can be customized by adding properties to the Resource File. For information about specifying the resource file, please refer to the "Control Property: Resource Bundle Name" section. The property name consists of a prefix followed by the field name, as shown below (where xxxxx is the field name):

```
dataKeyField_xxxxx
connInfoField_xxxxx
dsOptionsField_xxxxx
```

The following example shows custom labels for the **machine** and **pointName** fields in the data object, for the server and port fields in the **conninfo object**, and for the **maxPoints** and **defaultServer** fields in the **ds options object**:

```
dataKeyField_machine=Machine Name:
dataKeyField_pointName=Point Name:
connInfoField_server=Server Name:
connInfoField_port=Port Number:
dsOptionsField_maxPoints=Maximum Number of Points:
dsOptionsField_defaultServer=Default Server Name:
```

The resource file also contains entries for the labels shown on standard buttons and controls in the dialogs, allowing these objects to be customized or localized as well. For specific information please refer to the Localization section for the appropriate object.

Row and Column Filtering Choices

Note: This section only applies to the **Attach To Data** dialog, and only when row or column filtering has been enabled for the Custom Data Adapter. For more information about enabling row and column filtering, please refer to the "Column Filtering" section.

If row filtering has been enabled for the data adapter, the **Attach To Data** dialog will contain a Filter Column combo-box. If column filtering has been enabled for the data adapter, the **Attach To Data** dialog will contain a Column(s) combo-box and column chooser. You can set the available columns for these controls using the method **GmsRtViewDataObject.setAvailableColumns(Set<String>)**.

The available columns are typically set in the method **GmsRtViewDsAdapter.validate(GmsRtViewDataObject)**, as shown in the following example.

In this example, we use the data object to locate the corresponding data table. If data has previously been stored, we query the column names from the data table to set the available columns. If data has not previously been stored for this table, we assume we know nothing about the table structure and we make the set of available columns empty.

```
public class MyDsAdapter extends GmsRtViewDsAdapter
 {
@Override
protected void validate (GmsRtViewDataObject dataObject)
{

                        // Make empty Set

      Set<String> dataColumns = new HashSet<String>();
```

```
                        // Try to locate a data table for this
                        // data object
        String dataKey = ds.buildDataKey(dataObject);
        if (dataKey != null) {
              GmsTabularData tabularData = ds.getStoredDataTable(dataKey);
                        // If the table is found, add the column names
                        // to the set of available columns
              if (tabularData != null) {
                    int numColumns = tabularData.getNumColumns();
                    String[] columnNames = tabularData.getColumnNames();
                    for (int i = 0 ; i < numColumns; i++) {
                          dataColumns.add(columnNames[i]);
                    }
        }
                          // Set the available columns for the combo-box
                          // and column chooser
        dataObject.setAvailableColumns(dataColumns);
    }
     }
```

Customization API

For information on the Customization API, see the **docs/javadocs** directory for more information.

CHAPTER 21 Examples

This section contains the following:

RTView Demos

Introduction

Except where noted, all demos can be run in three ways, as an application, or via rich or thin client in a browser. If your RTView package features additional data sources, further Demos are located in the "RTView Data Sources" section.

Note: Some data sources do not have corresponding demos.

This section contains the following:

Before You Begin

RTView Demo Server

An Apache Tomcat application server is included with your RTView installation for prototyping and testing your deployment prior to deploying in the production environment. Several RTView demo applications are installed on the Demo Server as well. Once the Demo Server is started, you can access both rich and thin client demos.

The Demo Server runs on the HTTP/1.1 Connector port 8068. The shutdown port is 8069 and the AJP Connector port is 8070. The Demo Server also requires that you have JDK1.7+, and that your JAVA_HOME environment variable is set to the JDK installation directory.

Starting the Demo Server

In an initialized command window (see "Initializing a Command Prompt or Terminal Window") type:

run_startup_demoserver

Stopping the Demo Server

In an initialized command window (see "Initializing a Command Prompt or Terminal Window") type:

run_shutdown_demoserver

Initializing a Command Prompt or Terminal Window

Before running any RTView application from a Windows Command Prompt or UNIX terminal window, you must first initialize that window to set **RTV_HOME** and append **RTV_HOME\bin** to PATH. Alternatively, see "Setup" for information on how to set **RTV_HOME** and append **RTV_HOME\bin** to PATH globally.

Windows

To initialize a Command Prompt window, select **Start-> Programs-> Accessories-> Command Prompt**, go to your installation directory, and type:

rtv_init

UNIX

The script used to initialize a terminal window depends on whether you are in csh or bsh (e.g. Linux, Mac OS X).

csh

Open a terminal window, go to your installation directory, and type:

source rtv_init

bsh

Open a terminal window, go to your installation directory, and type:

../rtv_init.ksh

Data Source Demo

This demo illustrates how to use the RTView data source. Refer to the "RTView Data Sources" section of this documentation for further demonstrations specifically designed to illustrate your data source.

Note: You may not be licensed to run all RTView data sources.

---

1. Start the Simulators

Start the simulators for each data source you will be using. Refer to the "RTView Data Sources" section of this documentation for information on how to start a data simulation for your data source.

2. Run Demos - "Application Demo" or "Thin Client Browser Demo"

Application Demo

1. In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory.
2. To view the demo, type:

   run_viewer
3. To edit the demo, type:

   run_builder

Thin Client Browser Demo

Start the Demo Server if it is not running. See "Starting the Demo Server".

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/dstutorial** directory and start the Display Server by typing:

   run_displayserver
2. Open a browser and navigate to **http://localhost:8068/dstutorial**.

Self Service Alert Demo

The Self Service Alerts feature makes it easy to set and persist threshold, duration, and enabled settings for your alerts in a database. For details, see "Alerts".

The Self Service Alerts demo is located in your RTView installation directory under **demos\selfservicealerts**. The demo can be modified and used stand-alone or integrated into your RTView application in order to view and administrate alerts. The demo contains some demo alerts and a pre-configured hsqldb database to store your alert settings. See "Modify the Stand-Alone Demo" and "Integrate the Demo into an RTView Application" for more information.

Note: For other database types, the **demos\selfservicealerts\dbconfig** directory contains .sql files with the correct table schemas and a README.txt that explains how to use them.

The demo contains three main displays (Alert Detail Table, Alert, Administration and Administration Audit). In the top right corner, each display shows the current time, a ***** button that opens the display in a new window and a **?** button that opens Help. These objects are customizable and can be modified, removed, or replaced. See the "Customization Options" section for details.

Running the Demo

1. In an initialized command/terminal window (see "Initializing a Command Prompt or Terminal Window"), navigate to the **demos/selfservicealerts** directory:

2. "Start the XML Data Simulator"

3. Start the hsqldb database by typing:

   start run_hsqldb

   Note: The hsqldb database will open a new runtime window that shows you the state of the database. You can stop the hsqldb database by typing **Ctrl^C** in this new hsqldb window or by typing **stop_hsqldb** in your **demo** directory.

4. Confirm that the demo server is running. If not, start it by typing: **run_startup_demoserver**.

5. You can view the demo in the Display Viewer or the Thin Client.

   To view the demo in the Display Viewer, type the following:

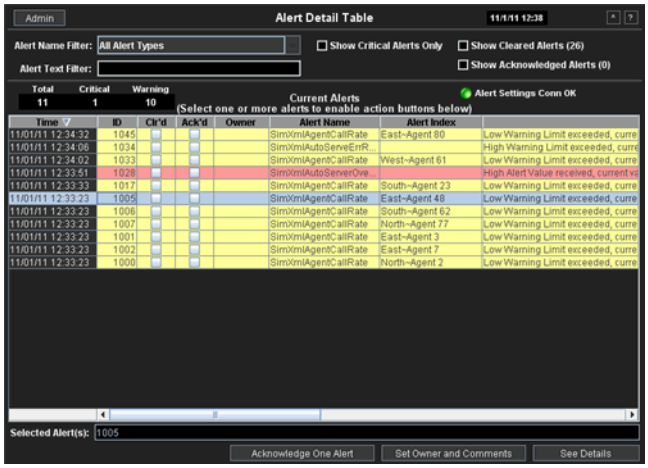
   run_viewer

   To view the demo in the Thin Client, type the following, and then open a browser and navigate to **http://localhost:8068/ssa/panels.html**:

   run_displayserver

Alert Detail Table

This display shows all of your current alerts:



Field Name	Description
Admin Button	Click on this button to open the **Alert Administration** display in a new window.
Alert Name Filter	Select an alert name from the list to filter the table by the **Alert Name** column.
Alert Text Filter	Enter a value to filter the table by the **Alert Text** column. For example, a value of *__High__* will filter to rows where the alert text contains the word **High**. A value of **High*** will filter to rows where the alert text starts with **High**.
Show Critical Alerts Only	Only show alerts with a severity greater than 1.
Show Cleared Alerts	Show cleared alerts. The number following the **Show Cleared Alerts** label indicates the number of cleared alerts that match the selected **Alert Name Filter** and **Show Critical Alerts Only** values.
Show Acknowledged Alerts	Show Acknowledged alerts. The number following the **Show Acknowledged Alerts** label indicates the number of active (not-cleared) acknowledged alerts that match the selected **Alert Name Filter** and **Show Critical Alerts Only** values.
Total	Total number of alerts that match the filter criteria.
Critical	Number of alerts that match the filter criteria with a severity greater than 1.
Warning	Number of alerts that match the filter criteria with a severity of 1.
Alert Settings ConnOK	Indicates the connection status of the Alert Settings table.

Current Alerts Table

Displays all alerts that match the selected filters.

- Red rows indicate the alert has a severity greater than 1.
- Yellow rows indicate the alert has a severity of 1.

Select one or more alerts from the table to enable the action buttons below the table. To select more than one alert:

- Click on one alert, then hold the Shift key while clicking on another alert. All alerts in between will be selected.
- Click on one alert, then hold the Control key while clicking on one or more other alerts. Only the alerts you click on will be selected.
- Click on one alert, then hold the Shift key while pressing the up or down arrow on your keyboard. All alerts in between will be selected.
- Click on one alert, then press Control+A to select all alerts in the table.

Selected Alert(s)

Lists ID of currently selected alerts.

Acknowledge One Alert/Acknowledge Multiple Alerts

Acknowledge the selected alert(s). If more than one alert is selected, you will be asked to confirm before the alerts are acknowledged. If only one alert is selected, it will be acknowledged without confirmation.

Set Owner and Comments

Click to open the **Set Owner and Comments** display. The ID field at the top of the display lists the IDs for all selected alerts. Any action will be applied to all alerts in that list.

Set the Owner - The value of the **Enter Owner** field is filled in as the user name used when you logged in. If you are logged in with a role of admin or login is disabled, you can specify a different value for the owner.

Click **Set Owner on One Alert**/**Set Owner on Multiple Alerts** to set the owner field for the selected alert(s) to the value specified in the **Enter Owner** field.

Note: If more than one alert is selected, you will be asked to confirm before the owner is set on the alerts. If only one alert is selected, the owner will be set without confirmation.

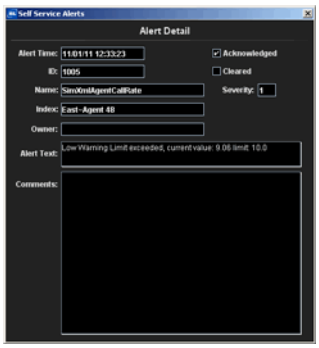
Add a Comment - Type a comment in the **Enter Comment** field, then click **Add Comment to One Alert**/**Add Comment to Multiple Alerts** to add the comment to the selected alert(s). The comment will be added to the alert(s) along with the timestamp and the user name from your login.

Note: If more than one alert is selected, you will be asked to confirm before the comment is added to the alerts. If only one alert is selected, the comment will be added without confirmation.

Clear All Comments - Click **Clear Comments on One Alert**/**Clear Comments on Multiple Alerts** to clear all comments from the selected alert(s).

Note: If more than one alert is selected, you will be asked to confirm before the comments are cleared from the alerts. If only one alert is selected, the comments will be cleared without confirmation.

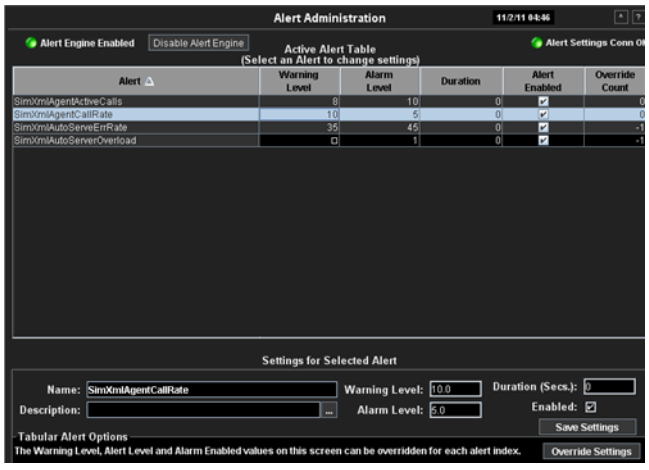
Details	Click to open the **Alert Detail** display to view details about the selected alert. **Note:** If more than one alert is selected, this display will show the details for the last alert in the selection list. See the "Customization Options" section for details on customizing this display.



Options	Click to open Options display showing options for the selected alert. **Note:** By default, the **Options** button in the **Alert Details** table display is hidden. See the "Customization Options" section for details on showing this button and customizing this display.

Alert Administration

This display shows you the current values in the Alert Settings Table in the Self Service Alerts database.



Field Name	Description
Alert Engine Enabled / Disabled	Indicates the status of the alert engine. If disabled, then click **Enable Alert Engine** to enable. If enabled, click **Disable Alert Engine** to disable. **Note:** The Enable and Disable Alert Engine buttons are only active if you are logged in with a role of admin or login is disabled.
Alert Settings ConnOK	Indicates the connection status of the **Alert Settings Table**.

Active Alert Table

Warning Level - Warning level threshold for the alert. If this value is NaN or a square, this means the alert doesn't support this threshold.

Alarm Level - Alarm level threshold for the alert. If this value is NaN or a square, this means the alert doesn't support this threshold.

Duration - Amount of time in seconds the value needs to meet the alert condition before an alert is generated. If the value is -1, this means the alert doesn't support duration.

Alert Enabled - Enabled value for the alert. If an alert is disabled, no alerts of that type will be generated.

Override Count - If this is a tabular Limits, Discrete or Multi state alert, the settings in this table can be overridden on a per-index basis. This column lists the number of indexes in that alert that override the settings in this table. A value of -1 indicates the alert doesn't support overriding the settings on a per-index basis.

Active - Indicates whether the alert definition is loaded in the current instance of RTView. If false, then there is a row in the Alert Settings Table in the database for this alert, but the alert is not loaded in the system. This may indicate a problem with your configuration.

By default, the **Active** column in the **Alert Administration** display is hidden. See the "Customization Options" section for details on showing this column.

To configure RTView to remove alerts that are not loaded in the system on startup, select the **Clean Alert Settings Table on Start** option on the Self Service Alerts tab of the Application Options dialog or use the -**cleansettingstable:true** command line option.

Settings for Selected Alert

To modify an alert setting, select a row in the **Active Alert** table.

Make changes to the **Warning Level**, **Alarm Level**, **Duration**, and **Enabled** fields, then click **Save Settings**.

Note: The **Save Settings** button is only enabled if an alert is selected and the connection to the **Alert Settings Table** is OK and you are logged in with a role of admin or login is disabled.

Name - Reflects value of alertName property.

Description - If the description is longer than the field, click on the ellipsis (...) button to see the whole description. NOTE: Alert definitions may or may not contain a description.

Enabled - If false, the selected alert will not be evaluated.

Warning Level - Threshold that the value of the selected alert must cross in order to generate a severity 1 alert.

Alarm Level - Threshold that the value of the selected alert must cross in order to generate a severity 2 alert.

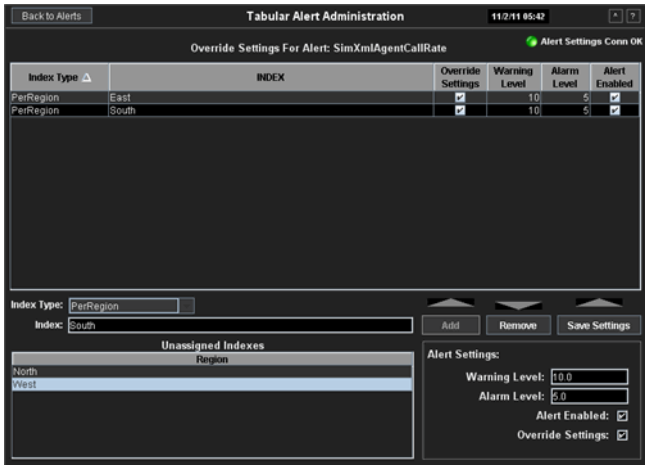
Duration (secs) - Amount of time that the selected alert must be in a Warning or Alarm state before an alert is generated.

Tabular Alert Options

If an alert is a tabular Limits, Discrete or Multi state alert, you can override the Warning Level, Alarm Level and Alert Enabled values specified on this screen. When you select an alert that supports this feature, an **Override Settings** button will appear at the bottom of the display. Click **Override Settings** to enter index-specific thresholds and enabled values. See Tabular Alert Administration section (below) for details.

Tabular Alert Administration

This display allows you to override the default warning, alarm and enabled settings on a per-index bases.



Field Name	Description
Back to Alerts	Click to return to the main Alert Administration page.
Alert Settings Conn OK	Indicates the connection status of the Alert Settings Table.
Override Settings Table	Shows all of the per-index settings that you have saved. All indexes that are not overridden here will use the settings specified for this alert on the main Alert Administration page. **Index Type** - The index type of this setting. For alerts with no **indexTypes** defined, this will be All. Otherwise this will list the **indexTypes** defined for that alert. In the case of alerts with multiple index columns, the **indexTypes** allow you to set threshold and enabled values on a subset of index columns instead of specifying each index column in the index value. For example, if you have an alert that is indexed on Region and Agent, you can either set a threshold for a specific Region and Agent or you can set a threshold for all Agents in that Region. **Index** - Index value. **Override Settings** - If selected, use these settings for the specified index rather than the default settings specified on the main Alert Administration page. If not selected, this row of settings will be ignored. **Warning Level** - The warning level threshold for the alert index. If this value is NaN or a square, this means the alert doesn't support this threshold. **Alarm Level** - The alarm level threshold for the alert index. If this value is NaN or a square, this means the alert doesn't support this threshold. **Alert Enabled** - The enabled value for the alert index. If an alert index is disabled, no alerts for that index will be generated.

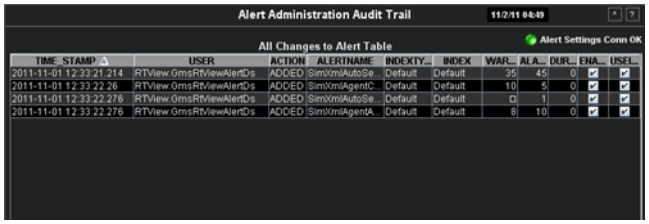
The **Add**, **Remove**, and **Save Settings** buttons are only enabled if you have a valid selection, the connection to the Alert Settings Table is OK, and you are logged in with a role of admin or login is disabled.

In addition, the **Add** button is disabled if the selected index is already defined. In that case **Remove** and **Save Settings** are enabled. If the selected index is not yet defined, the **Add** button is enabled and **Remove** and **Save Settings** are disabled.

Add	To add a new per-index setting, select an Index Type from the list and the table below will list all unassigned indexes available for that Index Type. Select an index from that list and fill in the **Warning Level**, **Alarm Level**, **Alert Enabled**, and **Override Settings** fields. Click **Add** to add to the **Override Settings** table. NOTE: If the **Override Settings** check box is not selected, these settings will be ignored and settings specified in the main **Alert Administration** page for this alert will be used.
Remove	To delete the settings for an index, select it from the table and click **Remove**.
Save Settings	To modify the settings for an index, select it from the table. Make changes and click **Save Settings**.

Administration Audit

This display shows a history of alert administration actions executed in the **Alert Administration** and **Tabular Alert Administration** displays. The Self Service Audit Table contains the TIME_STAMP of the change, the USER that made the change, the ACTION that was done, plus row information.



Field Name	Description
Alert Settings ConnOK	Indicates the connection status of the **Alert Settings Table**.
TIME_STAMP	Time the action was executed.
USER	User name from the RTView login or No Login if login was disabled. If the **USER** listed is **RTView.GmsRtViewAlertDs**, this indicates that RTView made the change. This happens when: • a row is added for a new alert, • when the threshold enabled flag for an alert that was already in the database changes, and • when a row is removed due to a selected **Clean Settings Table On Startup** option.
ACTION	Action taken.

Alert Action Audit Trail

This display shows a history of alert actions executed in the Alert Detail display.

Field Name	Description
Action Audit ConnOK	Indicates the connection status of the Self Service Audit Table.
TIME_STAMP	Time the action was executed.
USER	User name from the RTView login or **No Login** if login was disabled.
ACTION TYPE	Type of action taken. The **ACTION TYPE** depends on the command.
ACTION	Action taken. This will be the name of the command, with the exception of the **Set Custom Alert Event Attribute** command which will be **Set <Attribute Name>** (e.g. **Set MyCustomAttrField**).
TARGET	Target of the action. The **TARGET** depends on the command.
VALUE	The value of the action. The **VALUE** depends on the command.

Modify the Stand-Alone Demo

Replace Demo Alerts

To use the Self Service Alerts demo to monitor your own alerts, you must first replace the demo alerts with your own. There are two ways to do this:

1. Save your alert definitions in a file named **rtv_alertdefs.rtv**, and replace the **rtv_alertdefs.rtv** in the **selfservicealerts** demo with yours.

   OR

2. Save your alert definitions in one or more **.rtv** files in the **demos/selfservicealerts** directory.
   1. "Start the Display Builder" from the **selfservicealerts** directory.
   2. In the Display Builder, select **Tools>Options**.
   3. In the **Application Options** dialog, select **Alerts>**"Alert Definitions Tab", remove **rtv_alertdefs.rtv**, and add your own alert definition files.
   4. Save and exit the Display Builder.

In either case, your alerts must be constructed as documented for self service alerts.

Replace Demo (hsqldb) Database

To use a different database to store your alert settings, modify the ALERTDB SQL Database Definition.

1. "Start the Display Builder" from the **selfservicealerts** directory.

2. In the Display Builder, select **Tools>Options**.

   The **Application Options** dialog displays.

3. Select **SQL** in the **Application Options** dialog.

4. Edit the ALERTDB database connection to point to your database.

5. If necessary, change the **Settings Table Name** and **Audit Table Name** entries on the **Alerts>**"Self Service Alerts Tab" of the **Application Options** dialog.

6. Save and exit the Display Builder.

The **dbconfig** directory, located in the **selfservicealerts** directory, contains SQL schema files for several databases. You can use these files to create the necessary tables in your database. See the README.txt file in that directory for instructions.

Clear Demo (hsqldb) Database

You can also clear the included hsqldb database by running the **DATA\reset_alerts** script while the database is not running. This will remove all entries from the **Alert Settings Table** and the **Self Service Audit Table**.

Modify Alert Options

The Self Service Alerts demo is setup to limit the number of active alerts in the Alert Detail table to 2000, which removes cleared alerts every 5 minutes. If you anticipate a larger number of active alerts in the Alert Table at one time, you will need to increase this value. You can also adjust the rate at which cleared alerts are removed. Both of these options are available on the "Alerts Tab" of the **Application Options** dialog.

This demo is setup to use a Custom Alert Definition Property named **DrillDownSuffix**. You may add additional Custom Alert Definition Properties. See "Custom Alert Fields Tab" for more information.

Customization Options

The Self Service Alerts demo is set up to be easily customized in a few ways:

Modify the time label, new window button, and help buttons in the header

- These objects are all contained in **rtv_alerts_header_include.rtv**. Edit this display to add or modify the objects displayed in the header of the displays.

Note: Do not add objects to the center or left top as they will be obscured by objects in the displays.

Use the $rtvAlertDataServer substitution

- If you have deployed your alerts to a data server, specify the name of that data server for the **$rtvAlertDataServer** substitution on the **General>Substitutions** tab in the **Application Options** dialog.

Use the $rtvUserAlertOptionsEnabled substitution

- By default, the **Options** button in the **Alert Details** table display is hidden. To make it visible, specify a value of 1 for the **$rtvUserAlertOptionsEnabled** substitution on the **General>Substitutions** tab in the **Application Options** dialog.

Override the Options display and do a per-alert-type override

- Modify the **user_alert_options.rtv** file to add your own custom alert options. This is useful if you have added Custom Alert Event Attributes that you want your user to set. This file will be displayed when you click on **Options** in the **Alert Detail Table** for an alert that has no value for the **DrillDownSuffix** property in the alert definition. To create a different Options display for different alert definitions, specify a value for the **DrillDownSuffix** on the alert(s) where you don't want to use the default display. When Options is selected for an alert where the **DrillDownSuffix** is specified, it will open a file named **user_alert_optionsDrillDownSuffix.rtv** where **DrillDownSuffix** is the value you specified in the **DrillDownSuffix** property for that alert definition.

  For example, let's say you have the following alert definitions:

  MyAlert1 - This has no value for the **DrillDownSuffix** property. When you select an instance of **MyAlert1** from the Alert Detail Table and click the **Options** button, it will open **user_alert_options.rtv**.

  MyAlert2 - This has the value "**_alert2**" for the **DrilldownSuffix** property. When you select an instance of **MyAlert2** from the Alert Detail Table and click the **Options** button, it will open **user_alert_options_alert2.rtv**.

  MyAlert3 - This has the value "**_alert3**" for the **DrilldownSuffix** property. When you select an instance of **MyAlert3** from the Alert Detail Table and click the **Options** button, it will open **user_alert_options_alert3.rtv**.

Override the Details display and do a per-alert-type override

- Modify the **user_alert_ details.rtv** file to add your own custom alert details. This is useful if you've added Custom Alert Definition Properties or Custom Alert Event Attributes that you want to display. This file will be displayed when you click on Details in the Alert Detail Table for an alert that has no value for the **DrillDownSuffix** property in the alert definition. To create a different details display for different alert definitions, specify a value for the **DrillDownSuffix** on the alert(s) where you don't want to use the default display. When Details is selected for an alert where the **DrillDownSuffix** is specified, it will open a file named **user_alert_detailsDrillDownSuffix.rtv** where **DrillDownSuffix** is the value you specified in the **DrillDownSuffix** property for that alert definition.

  For example, let's say you have the following alert definitions:

  MyAlert1 - This has no value for the **DrillDownSuffix** property. When you select an instance of **MyAlert1** from the Alert Detail Table and click the **Details** button, it will open **user_alert_details.rtv**.

  MyAlert2 - This has the value "**_alert2**" for the **DrilldownSuffix** property. When you select an instance of **MyAlert2** from the Alert Detail Table and click the **Details** button, it will open **user_alert_details_alert2.rtv**.

  MyAlert3 - This has the value "**_alert3**" for the **DrilldownSuffix** property. When you select an instance of **MyAlert3** from the Alert Detail Table and click the **Details** button, it will open **user_alert_details_alert3.rtv**.

Use the $rtvUserAlertActiveColumnEnabled substitution

- By default, the **Active** column in the **Alert Administration** display is hidden. To make it visible, specify a value of **1** for the **$rtvUserAlertActiveColumnEnabled** substitution on the **General>Substitutions** tab of the **Application Options** dialog.

Help files

- The help button (**?**) in the top left corner of the display is configured to display help files located in the docs directory of the demo. To modify this button to look for help files in another location, add a substitution named **$displayHelpURL** and set it to the url where you are hosting your help files.
- To change the name of the help file for each display, modify **ssa_displays.xml**. This xml file has an entry for each **.rtv** file in the demo with the name of the corresponding html file to load. This file will be loaded from the url specified in **$displayHelpURL**.

Integrate the Demo into an RTView Application

Before integrating this demo into your RTView application, you must configure it as described above to monitor your alerts and store the settings in the database of your choice. See "Replace Demo Alerts" and "Replace Demo (hsqldb) Database" for more information.

Once you have done this, proceed with the following instructions:

1. Copy all of the **.rtv** files from the **demos/selfservicealerts** directory to your RTView application directory. If you will be using the built-in help files, also copy **ssa_displays.xml** and the **selfservicealerts\docs** directory to your RTView application directory.

2. Copy **ALERTOPTIONS.ini** and **CACHEOPTIONS.ini** files to the directory where you will be running the alerts. This might be your application directory, or it might be the directory where you are running the Data Server if you want the alerts to run in the Data Server.

   Note: If this directory is not your application directory, move your alert configuration files (**rtv_alertdefs.rtv** and/or your other alert configuration files) and **rt_alerts_cache.rtv** to the same directory as **ALERTOPTIONS.ini** and **CACHEOPTIONS.ini**. Then, copy values from those files in the **selfservicealerts** directory into the corresponding **OPTIONS.ini** files in your application.

3. Add the **ALERTDB** option to the options file in the directory where you put the **ALERTOPTIONS.ini** and **CACHEOPTIONS.ini**:

   Note: If the directory where you put **ALERTOPTIONS.ini** and **CACHEOPTIONS.ini** already contains an **OPTIONS.ini** file, then add the lines that start with sqldb ALERTDB and dbretry from **demos/selfservicealerts/OPTIONS.ini** file to that file.

   Note: If the directory where you put **ALERTOPTIONS.ini** and **CACHEOPTIONS.ini** does not contain an **OPTIONS.ini** file, copy the **OPTIONS.ini** file from **demos/selfservicealerts** to there.

4. In your application directory, add the following display (**.rtv**) files to your panel configuration file:

   ```
   rtv_alerts_table.rtv (Alert Detail View)
   rtv_admin_alerts.rtv (Alert Administration)
   rtv_alerts_audit.rtv (Administration Audit)
   ```

5. The Self Service Alerts demo uses the style sheets **rtv_darkstyles.rts** and **rtv_flat.rtv** to set the look and feel of the displays. If your application already uses these style sheets or if you do not want to apply a style sheet, you are finished.

a. If you want to apply the style sheets to your whole application:

In the Display Builder select **Tools>Options**. In the **Application Options** dialog, select **General>Style Sheet**. Click **Add Built-in Styles** and select **rtv_darkstyles.rts** and **rtv_flat.rts**. If you do not want these style sheets applied to the main **Display Builder** window, deselect **Apply Style Sheets to Main Builder Window**. Save your options and exit the Display Builder.

b. If you only want to apply the style sheets to the Self Service Alerts display files:

Open each of these files in the Display Builder and in each display, select **Tools>Style Sheets**. Click **Add Built-in Styles** and select **rtv_darkstyles.rts** and **rtv_flat.rts**. Click **OK** and **Save** each display.

6. If you will be deploying the Self Service Alerts in a thin client application and you will be using the built-in help files, include **selfservicealerts\docs** in the **.war** file for your thin client application. See the **make_war.bat** script, located in the **selfservicealerts** directory, as an example of how to include this in your war file.

ElectroSphere Demo

The ElecroSphere demo, a Business Information Application, allows you to navigate through multiple types of business data.

Note: This demo requires use of the SQL data source and only runs on Windows with Microsoft Access. Your version of RTView may not be licensed to run the SQL data source.

---

1. **Run Demos** - "Application Demo" or "Thin Client Browser Demo"

Application Demo

1. In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/esphere** directory.

2. To view the demo, type:

   run_viewer

3. To edit the demo, type:

   run_builder

Thin Client Browser Demo

Start the Demo Server if it is not running. See "Starting the Demo Server".

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/esphere** directory.

2. Start the Display Server by typing:

   run_displayserver

3. Open a browser and navigate to **http://localhost:8068/esphere**.

Features Demo

Presents an overview of the many features of RTView.

1. **Run Demos -** "Application Demo" or "Thin Client Browser Demo"

Application Demo

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/features** directory.
2. "Start the XML Data Simulator"
3. To view the demo, type:

   run_viewer
4. To edit the demo, type:

   run_builder

Thin Client Browser Demo

Start the Demo Server if it is not running. See "Starting the Demo Server".

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/features** directory.
2. "Start the XML Data Simulator"
3. Start the Display Server by typing:

   run_displayserver
4. Open a browser and navigate to **http://localhost:8068/features**.

Alert Demo

Presents an overview of RTView Alert functionality.

1. **Run Demos** - "Application Demo" or "Thin Client Browser Demo"

Application Demo

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/alertdemo** directory.
2. "Start the XML Data Simulator"
3. To view the demo, type:

   run_viewer
4. To edit the demo, type:

run_builder

Thin Client Browser Demo

Start the Demo Server if it is not running. See "Starting the Demo Server".

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/alertdemo** directory.
2. "Start the XML Data Simulator"
3. Start the Display Server by typing:
   run_displayserver
4. Open a browser and navigate to **http://localhost:8068/alertdemo**.

Geothermal Demo

A real-time operational application that allows you to drill down to various levels of detail.

1. **Run Demos -** "Application Demo" or "Thin Client Browser Demo"

Application Demo

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/geothermal** directory.
2. "Start the XML Data Simulator"
3. To view the demo, type:
   run_viewer
4. To edit the demo, type:
   run_builder

Thin Client Browser Demo

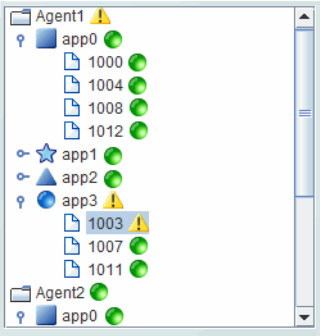
Start the Demo Server if it is not running. See "Starting the Demo Server".

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos/geothermal** directory.
2. "Start the XML Data Simulator"
3. Start the Display Server by typing:
   run_displayserver
4. Open a browser and navigate to **http://localhost:8068/geothermal/panels.jsp**.

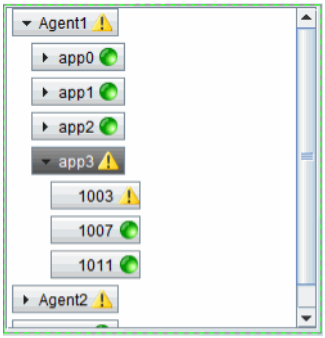
Navigation Control Demo

Presents a multi-panel display in which a navigation control is used to navigate elements in a separate display. This demo allows you to view the tree control or the accordion control:

"Tree Control"



"Accordion Control"



Note: The navigation control demo does not illustrate all of the control features. For example, the number of nodes does not change.

---

1. In an initialized command/terminal window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos\treecontrol** directory.
2. "Start the XML Data Simulator":
3. Start the Viewer by typing:

   run_viewer to view the tree control demo.

   run_viewer -panelconfig:PANELS_accordion.ini to view the accordion control demo.
4. To edit the demo, type:

   run_builder

Quick Start Tutorial

This Quick Start Tutorial provides you with the fundamentals on how to use RTView. Once completed, you can swiftly apply this knowledge to building your own real-time dashboard displays that give you comprehensive business information at a glance.

Learn to:

- Animate graphs, tables and meters
- Create drill down displays that show more detail with a mouse click
- Create a display that is reused for any number of data sets
- Create alerts for critical data
- Perform automated calculations on your data and display results
- Automatically highlight critical data in tables
- View your displays in their deployed form

In order to illustrate RTView features, this tutorial uses an XML source - the XML data simulator. Even if you will not be using XML data, we suggest you complete these exercises in order to learn basic RTView concepts. In the Data Sources section, there are links to additional exercises that address topics specific to other data sources.

Get Started

To start RTView, you need to get a license key, start the XML data simulator and login to the Display Builder.

Register for a License Key

In order to proceed, you need to register for a license key if you have not already done so. See "Registration" for more information.

Start the XML Data Simulator

In this exercise you start the XML data simulator which is the XML source used in this tutorial.

On Windows

1. In an initialized Windows command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos** directory and type:

   start run_simdata

   The XML data simulator is ready when dots appear across the screen.

Note: You must initialize each new terminal window you open. See the "Setup and Registration" section for more details about setting up your environment.

On UNIX

1. In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos** directory and start the XML "Data Simulator" by typing:

   run_simdata &

   The XML data simulator runs as a background process and is ready when dots appear across the screen.

Note: You must initialize each new terminal window you open. See the "Setup and Registration" section for more details about setting up your environment.

Start the Display Builder

1. Start the Display Builder in your original, initialized command/terminal window by typing:

   On Windows

   start run_builder

   On Unix

   run_builder &

2. Login to the Display Builder. By default, the Display Builder does not require a login. "Login" can be enabled at setup to support "Role-based Security". The default user name and password are:

   User Name: admin

   Password: admin

   Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role. See "Role-based Security" for more information.

You are now ready to create a display.

Create A Display

At this point you have:

- Registered for a license key. See "Registration" for more information.
- Started the XML data simulator
- Started the Display Builder

In addition to illustrating the fundamentals of RTView, we also want to provide you with a conceptual understanding of its many uses. Therefore, in the following exercises, you will use the XML simulator as a data source to create a dashboard display for a fictitious company with plants in multiple cities. The display will track live production numbers, such as Units Completed and Units in Production. The dashboard will have several animated objects showing real-time updates, including:

- A table that shows production data for fifteen plants, highlights rows when production status goes offline, and drills down to plant-specific information with a mouse click
- An automatically activated, audible alert
- A graphic showing the sum total of Units Completed for all fifteen plants
- A bar graph comparing Units Completed versus Units in Production for each plant

Set Display Background Properties

The first step in creating your dashboard display is to set the size and background color as it is to appear in your deployment.

1. In the Display Builder, select **File> Background Properties**.
2. In the "Background Properties" dialog:

   Model Width - Increase to **800**.

   Model Height - Increase to **600**.

   Model Properties - Click the button to open the **Model Properties** window:

   bgColor - Click in the right column, click the ellipsis button and choose a background color from the palette.

   Close the Color Chooser.
3. Click **OK** to apply and close the **Background Properties** dialog.
4. Click **Yes** to add space to the top of the display.

Your display size and background color are now set. You now need to setup access to an XML source.

Add an XML Source to XML Source List

In this exercise, you add an XML source, **update.xml**, to the XML Source List to make it available for animating objects.

1. Select **Tools> Options** to open the **Application Options** dialog.
2. Select the "XML Tab" and click **Add** to open the "Edit XML Source" dialog.
3. In the **Edit XML Source** dialog:

   XML Source Name - Enter **update.xml**

   Note: The **update.xml** source is generated by the XML data simulator. See "Creating XML Sources" for technical details on creating and formatting your own XML source.
4. Click **OK** to close the **Edit XML Source** dialog.

The XML source appears in the list of available XML Data Sources.

5. Click **OK** to apply and close the **Application Options** dialog.

The XML source is now available for animating objects.

Animate a Graphic Object

Certain objects are suitable for displaying certain types of data. For example, you could not display an entire table of data in a single meter object, but you could display a single cell from that table in the meter object.

In this exercise, you create a meter that shows the plant load for a single plant. You animate the meter simply by attaching it to a Data Key in the XML source. You will first add a meter object to your display, edit the meter label, and then attach the meter to the **element1_load** Data Key.

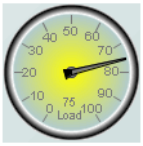
1. Select the **Meters** tab in the Object Palette.
2. Click on the first meter in the palette.
3. Move the cursor to the Working Area. A **+** symbol appears next to the cursor which means that the Display Builder is in Add mode. Click in the Working Area to place the meter. The meter is ready to be edited.
4. In the **Object Properties** dialog:

   label (category: Label) - Change to **Load**. Press <Enter> to apply the label.

   value (category: Data)- Right-click in the **Property Name** field and select **Attach to Data>XML**.
5. In the "Attach to XML Data" dialog:

   XML Source - **update.xml** should already be selected.

   Data Key - Select **element1_load** from the drop down menu.
6. Click **OK** to apply these values and close the **Attach to XML Data** dialog.



The meter is now animated with real-time data updates provided by the value of the **element1_load** Data Key.

Create an Alert

In this exercise, you create an audible alert using the **element1_load** Data Key. You setup an alert by defining the value at which it is to be activated, and what happens when it gets activated. You will first copy and paste the meter's data attachment properties (from the previous exercise) to a new object, label it Plant Load, and then set the alert to beep when **element1_load** is 75 or greater.

1. In the Object Palette, select the **General** tab and add the **Range Dynamic** object shown here (class name: **obj_rect_ilvx_ra4**) to your display.

2. Select the meter you previously added (class name: **obj_meter20**) and click on the Copy button in the toolbar.

3. Select the **Range Dynamic** object and click the Paste Data Attachments button. The new object is animated by real-time data updates provided by the value of **element1_load**.

4. In the **Object Properties** dialog:

   label (category: Label) - Change to **Plant Load**. Press <Enter> to apply the label.

   valueHighAlarmEnabledFlag (category: Alert) - Select the check box to enable.

   valueHighAlarm (category: Alert) - Decrease to 75. Press <Enter>. This will activate the alert when the value of **element1_load** is 75 or greater.

   valueHighAlarmCommand (category: Alert) - Right-click in the **Property Name** field and select **Define Command>SYSTEM**.

5. In the **Define System Command** dialog:

   Command Type - Select **Beep** from the drop down menu.

6. Click **OK** to apply these values and close the "Define System Command" dialog.

The alert now automatically beeps when the **Plant Load** is 75 or greater. You will also notice that the color of the meter changes to the default **valueHighAlarmColor** setting, red, when the alert is activated.

Display Data in a Table

In this exercise, you add a table to your display that shows real-time production numbers for fifteen plants, and highlights rows when the status of a plant changes to offline. This is done by attaching data to the table, then creating a filter for the table. You will notice that you do not have to specify the number of columns and rows for the table - RTView automatically creates the exact number needed based on the data. During this exercise you will also add a table and change the label.

1. In the Object Palette, click on the **Tables** tab and add the first table in the palette (class name: **obj_table02**) to your display.

2. In the Object Properties dialog:

   label (category: Label) - Change to **Production Table**. Press <Enter>.

3. In the **Object Properties** dialog:

   valueTable (category: Data) - Right-click in the **Property Name** field and select **Attach to Data>XML**.

4. In the "Attach to XML Data" dialog:

XML Source - **update.xml** should already be selected.

Data Key - Select **production_table** from the drop down menu.

Column(s) - Select ***** to display all columns available in the Data Key. RTView automatically generates the correct number of columns and rows for the table.

5. Click **OK** to apply these values and close the **Attach to XML Data** dialog.

   The table now displays real-time data updates provided by the value of **production_table**.

6. In the **Object Properties** dialog:

   filterProperties (category: Alert) - Double-click in the **Property Name** field to open the **Filter Properties** dialog.

7. In the **Filter Properties** dialog, click the **Add** button to open the **Edit Filter** dialog.

8. In the **Edit Filter** dialog:

   Condition - Select **Status**.

   Select = (equals).

   Click in the text field and enter **offline**.

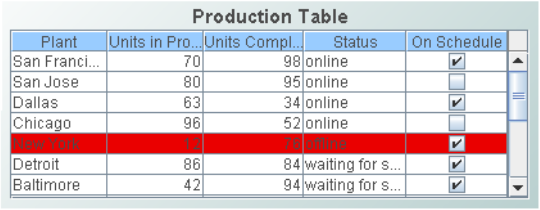
   Action - **Set Background Color To** should already be selected.

   Click in the next field to open the **Color Chooser**. Click to select the filter color and close the Color Chooser.

   Target - **Rows** should already be selected.

9. Click **OK** to close the **Edit Filter** dialog.

10. Click **OK** to apply and close the **Filter Properties** window.



The table now highlights rows where the Status cell value is offline. You can easily create an animated bar graph version of this Production Table. See the next exercise for instructions.

Animate a Bar Graph

In this exercise, you display the Production Table data in a bar graph. This is done by copying and pasting properties from the Production Table to a bar graph object. You will also edit the label.

1. In the Object Palette, click on the **Graphs** tab and add the first bar graph (class name: **obj_bargraph**) to your display.

2. Right-click on the Production Table and select **Copy** from the popup menu.

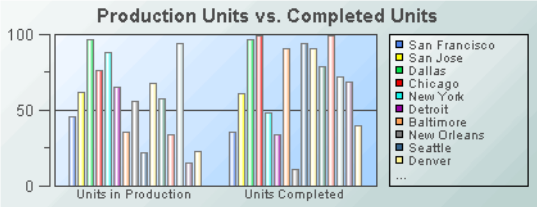
3. Right-click on the Bar Graph and select **Paste All Properties** from the popup menu.

   The bar graph is now animated by the Production Table data.

4. In the **Object Properties** dialog:

   label (category: Label) - Change to Production Units vs. Completed Units. Press <Enter>.

   xAxisFlag (category: X-Axis)- Click to select the check box to display labels at the bottom of the graph.



   If you cannot read the labels below the graphs, scale the graph horizontally.

The bar graph is now animated by three columns of the **production_table** data. You can easily create a graphic that displays an automated calculation. See the next exercise for instructions.

Display an Automated Calculation

In this exercise, you create an object that displays total Units Completed for all production sites. You do this by creating a function that calculates a sum, attaching the function to the Units Completed column of the **production_table** Data Key, creating an object to display the sum, and attaching the object to the function.

1. Select **Tools>Functions** and click the **Add** button to open the **Edit Function** dialog.

2. In the **Edit Function** dialog (see "Editing Functions"):

   Function Name - Enter **total_units_comp**. (This must be a unique name.)

   Function Type - Select **Add All Rows or Columns** from the drop down menu.

   Table - Right-click in the text field and select **Attach to Data>XML**.

   Return Column - **0**

   The function is ready to calculate - it now needs data to perform calculations on.

3. In the "Attach to XML Data" dialog:

   XML Source - **update.xml** should already be selected.

   Data Key - Select **production_table** from the drop down menu.

   Column(s) - Select **Units Completed** from the drop down menu.

4. Click **OK** to close the **Attach to XML Data** dialog.

5. Click **OK** to apply these values and close the **Edit Function** dialog.

   The function now performs calculations on the **Units Completed** column of the **production_table** Data Key. It now needs a place to display the totals.

6. In the Object Palette, click on the **General** tab and add the oval object shown here (class name: **obj_circ2d_ilv**) to your display.

To be sure that you have added the correct object to your display, you can verify the class name listed at the top of the **Object Properties** window.

7. In the **Object Properties** dialog:

label (category: Label) - Change to **Total Units Completed**. Press <Enter>.

labelTextPosY (category: Label) - Select **Outside Bottom** from the drop down list to position the label.

8. In the **Object Properties** dialog:

value (category: Data) - Right-click in the **Property Name** field and select **Attach to Data>FUNCTION**.

9. In the "Attach to Function Data" dialog:

Function Name - Select **total_units_comp** from the drop down menu.

Column(s) - Select **Units Completed** from the drop down menu.

10. Click **OK** to apply and close the **Attach to Function Data** dialog.

The oval object now displays the sum of Units Completed for all production sites.

Create a Drill Down

In this exercise, you create a drill down in the Production Table. A drill down enables you to navigate through your data in many different ways.

Background Information

RTView has a Substitution feature that allows you to build open-ended displays in which data attachments and commands depend on values defined at the time the display is run. In this way, a single display can be reused to show data and execute commands from a number of different sources. When data attachments are created in the Display Builder, generic values are used instead of the actual value of any field in the **Attach To Data** and **Define Command** dialogs. Later, when the display is running, these generic values are defined.

The data structure of tables and graphs (tabular data) enables RTView to automatically create several data source specific, built-in Substitutions for you. You will see these built-in Substitutions used in the target display when you create the drill down. For more information on Substitutions, see "Substitutions".

In this exercise, you create a drill down using the previously created display, **xml_dd_qs.rtv**, as the target display. First you will set the Production Table to display two columns, the **Plant** and **Status** columns. Then you will create a drill down that will open a bar graph that shows production numbers for each plant.

1. Select the Production Table and set the following in the **Object Properties** dialog:

valueTable (category: Data) - Double-click in the **Property Name** field to open the **Attach to XML Data** dialog.

2. In the **Attach to XML Data** dialog:

   XML Source - **update.xml** should already be selected.

   Data Key - **production_table** should already be selected.

   Column(s) - Click on the ellipsis button to open the **Select Columns** dialog.

3. In the **Select Columns** dialog:

   Select **Plant** in the **Available Columns** list and click **Add**.

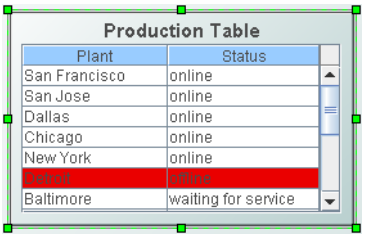
   Select **Status** in the **Available Columns** list and click **Add**.

   Click **OK** to close the **Select Columns** dialog.

4. Click **OK** to apply and close the **Attach to XML Data** dialog.

5. In the **Object Properties** dialog:

   autoResizeFlag (category: Column) - Click to select the check box and set the width of all columns to fit the visible area.



The table now displays only two columns: **Plant** and **Status**.

6. Select the **Production Table** and in the **Object Properties** dialog:

   drilDownTarget (category: Interaction) - Double-click in the **Property Name** field to bring up the **Drill Down Properties** dialog.

7. In the "Drill Down Properties" dialog:

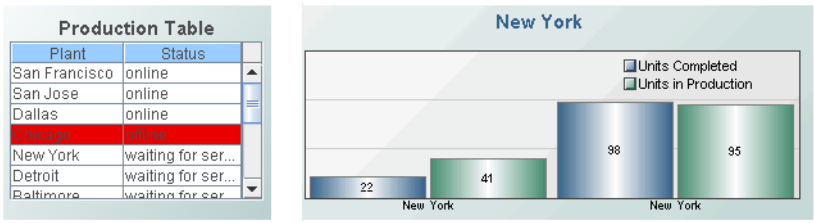
   Apply Drill Down To - Select **Named Window** from the drop down menu. This option lets you re-use the window when you drill down multiple times.

   Window Name - Enter **xml**. This name should be unique unless the display is to open in an existing window.

   Drill Down Display Name - Select **dstutorial\xml_dd_qs.rtv**, the previously created display for this tutorial, from the drop down menu.

8. Click **OK** to set the drill down target and close the **Drill Down Properties** dialog.

9. Double-click on any row in the table to drill down. The previously created display, **xml_dd_qs.rtv**, opens.



10. Double-click on another row in the table and the same display, **xml_dd_qs.rtv**, is reused to show different data based on the row you selected.

11. Close the drill down display.

Select **File>Save**, name this display **mydisplay.rtv**, and save it in the **demos** directory.

For more information on creating drill down displays, see "Drill Down Displays". For more information on Substitutions, see "Substitutions".

Deploy Your Display

In this exercise, you access your displays in their deployed form. Depending on whether you deploy RTView as an Application or a Thin Client Browser, you will access your displays either with the Display Viewer or a web browser, respectively. For the purposes of this tutorial, we show you one way to access your display for each of the three types of deployment. See "Deployment" for more information.

Application with Direct Data Connection

On Windows

1. With the XML Simulator running and in an initialized Windows command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos** directory and type:

start run_viewer

2. Login to the Display Viewer. By default, the Display Viewer does not require a login. "Login" can be enabled at setup to support "Role-based Security". The default user name and password are:

User Name: admin

Password: admin

Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role. See "Role-based Security" for more information.

---

3. Select **File>Open**, choose **mydisplay.rtv**, and click **Open**.

Your display opens with real-time data updates.

4. Single-click on any row in the table to drill down. The previously created display, **xml_dd_qs.rtv**, opens.

5. Single-click on another row in the table and the same display, **xml_dd_qs.rtv**, is reused to show different data based on the row you selected.

6. Close the drill down display.

7. Exit the Display Viewer and, if you are not going to view your display again, exit the XML data simulator.

If your RTView package features additional data sources, see the "Work with Additional Data Sources" section at the end of this tutorial.

On UNIX

1. With the XML simulator running and in an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos** directory and type:

   run_viewer mydisplay

2. Login to the Display Viewer. By default, the Display Viewer does not require a login. "Login" can be enabled at setup to support "Role-based Security". The default user name and password are:

   User Name: admin

   Password: admin

Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role. See "Role-based Security" for more information.

Your display opens with real-time data updates.

3. Single-click on any row in the table to drill down. The previously created display, **xml_dd_qs.rtv**, opens.

4. Single-click on another row in the table and the same display, **xml_dd_qs.rtv**, is reused to show different data based on the row you selected.

5. Close the drill down display.

6. Exit the Display Viewer and, if you are not going to view your display again, exit the XML data simulator.

If your RTView package features additional data sources, see the "Work with Additional Data Sources" section at the end of this tutorial.

Thin Client Browser

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos** directory and start the demo server by typing:

   run_startup_demoserver

2. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **demos** directory and start the display server by typing:

   run_displayserver

3. Open a browser and navigate to **http://localhost:8068/rtvdisplay**.

4. Select **mydisplay.rtv** from the left frame in the browser.

   Your display opens with real-time data updates.

5. Single-click on any row in the table to drill down. The previously created display, **xml_dd_qs.rtv**, opens.

6. Single-click on another row in the table and the same display, **xml_dd_qs.rtv**, is reused to show different data based on the row you selected.

7. Close the drill down display.

8. Exit the Demo Server and the Display Server and, if you are not going to view your display again, exit the XML data simulator.

Work with Additional Data Sources

You may continue this tutorial by working with XML via "Java Server Pages". If your RTView package features additional data sources, further Quick Start exercises are located in the"RTView Data Sources" section.

Display Server

Requirements

- Your web server is running and you have a standard working knowledge of that server
- Standard working knowledge of JSP and web server deployment

Objective

- Use the Display Server to download display (**.rtv**) files as HTML in a web browser

---

Note: A web server with a Java servlet container, such as Tomcat, is required for this example.

---

1. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the **servlets\rtvdisplay** directory.

2. If the web server you are using is not the same machine where RTView was installed, then open the **rtvdisplay.properties** file (located in the **rtvdisplay** directory) and change the **DisplayServerHost** property to the name of the host running the web server you are using.

3. Before running the Display Server you must install the Display Servlet, a JSP servlet that will run on your web server. If you are using a JSP servlet container other than Tomcat, install the files contained in the **rtvdisplay** directory and classes from **lib\gmsjcs_client.jar** on your web server according to instructions given with that product. Otherwise, type the following in your initialized command window:

   make_war - This script creates a web archive (**.war**) named rtvdisplay.war.

install_to_tomcat rtvdisplay - This script installs the web archive **rtvdisplay.war** to your Tomcat server.

Note: This script will shutdown and restart Tomcat and requires administrative permissions.

The **install_to_tomcat** script will not work if the **CATALINA_HOME** environment variable has not been set. If you did not set this variable following product installation, then return to the "Setup" section for details before continuing with this example.

4. In the command window, go to the **demos\features** directory.
5. "Start the XML Data Simulator"
6. Start the Display Server by typing **run_displayserver** in your original command window.
7. Open a web browser and enter the following URL (where host is web server hosting the Display Servlet):

   http://host:8080/rtvdisplay/index.html

   The left frame of index.html lists the display (**.rtv**) files available in the directory where you started the Display Server. If an error message appears, check the command window where you started the Display Server as well as the servlet log file on your web server.
8. Select a display from the left frame of **index.html**. The selected display will load in the main frame.

The Historian

Requirements

The steps described in this section are for a Historian configuration in which RTView display (**.rtv**) files were used for the Historian Data Configuration file. For information, see "Configuring the Historian".

The SQL data source is required to view data from the Historian in RTView. Verify that the SQL data source is licensed in your installation in the Display Builder **About** dialog.

This example requires that you have a working knowledge of RTView. If you would like to work through a basic example, please complete the "Quick Start Tutorial" before continuing.

Objectives

- Create a configuration file for the Historian
- Run the Historian with this configuration file
- Create and view a display file which shows data from the Historian
- Create a configuration file that will cause the Historian to store your tabular data to a new table in the history database
- Run the Historian with this configuration file to create and view a display file which shows the new table of data

---

Note: This example uses the XML data source to illustrate the features listed below. Even if you will not be using XML data, we suggest you complete this exercise in order to learn basic concepts of archiving and viewing historical data.

Getting Started

This example assumes that you have a database to which the Historian will store data. If you have not already completed "Database Connection Configuration", you must do so before continuing.

In an initialized command/terminal window (see "Initializing a Command Prompt or Terminal Window"):

1. "Start the XML Data Simulator"
2. "Start the Display Builder"

Note: You must initialize each new terminal window you open. See the "Setup and Registration" section for more details about setting up your environment.

Application Options

1. In the Display Builder, select **Tools>Options** to open the **Application Options** dialog.
2. Select the "XML Tab" and click on **Add**.
3. In the "Edit XML Source" dialog:

   XML Source Name - Enter **update.xml**
4. Click **OK** to add the XML source.
5. Click on the "SQL Tab" and select **Add Database**.
6. In the **Add Database** dialog:

Database Name - Enter **RTVHISTORY**

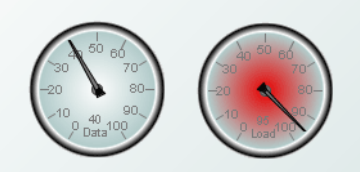
Note: Enter a JDBC Driver Class Name and JDBC Database URL. See the "Application Options - SQL" page in the Data Sources section of this documentation for more details.

7. Click **OK** to add this database.
8. Click **Save** to save these options and select **No** in the confirmation dialog.

Create a Configuration File to Store Historical Data

In this section you will add two meters to a display. The data you attach to these meters will be used by the Historian to store data in the history database.

1. Select **Edit>Add**.
2. In the **Object Palette** window, select the **Meters** tab and add the first meter in the palette (class name: **obj_meter20**) to the display.
3. In the **Object Properties** dialog:
   label - change to **Data**. Press <Enter>.
   value - Right-click in the **Property Name** field and select **Attach to Data>XML**.
4. In the "Attach to XML Data" dialog:
   XML Source - **update.xml** should already be selected.
   Data Key - Select **element1_data** from the drop down menu.
5. Click **OK** to apply these values and close the **Attach to XML Data** dialog.
6. Select the Element 1 Data meter and click the Copy button .
7. Click the Paste button .
8. Click to place the new meter.
9. In the **Object Properties** dialog:
   label - change to **Load**. Press <Enter>.
   value - Right-click in the **Property Name** field and select **Attach to Data>XML**.
10. In the "Attach to XML Data" dialog:
    XML Source - **update.xml** should already be selected.
    Data Key - Select **element1_load** from the drop down menu.
11. Click **OK** to apply these values and close the **Attach to XML Data** dialog.



12. Select **File>Save** and name this file **tutorial_history_config.rtv**.

Store Historical Data

1. Start the Historian:

   On Windows

   In an initialized Windows command window (see "Initializing a Command Prompt or Terminal Window"), type:

   start run_historian

   On UNIX

   In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"), type:

   run_historian &

2. Click **Add** on the **Configuration** tab.
3. Select **tutorial_history_config.rtv**
4. Click **Open** to insert into the **Data Configuration Files** list.
5. Select the **Show Data in Console** check box.

   This prompts the Historian to print a line to the console for each record it stores to the history database.

   Note: If you are using a direct JDBC connection, enter a JDBC Driver Class Name and JDBC Database URL. See "Historian Application Configuration" for details. You must enter the same configuration settings you entered in the **Add Database** dialog.

6. Click **Save Configuration** to save these settings.
7. Select the **Console** tab.
8. Click **Start Storing Data**.

   Data will immediately start printing to the console window.

Note: Let the Historian run as you go through the next section.

View Historical Data

In this section you will create a display with a graph that loads initial data from the history database and then updates with live XML data. You will also add a table that will show all data from a table in the history database.

1. In the Display Builder, click on the New button.
2. In the **Object Palette** window, select the **Graphs** tab and add the second graph in the palette (class name: **obj_trendgraph02**) to the display.
3. In the **Object Properties** dialog:

   objWidth - Increase to **500**. Press <Enter>.

   label - Change to **History Graph**. Press <Enter>.

   trace1ValueHistoryFlag - Select the check box.

   trace2ValueHistoryFlag - Select the check box.

   Note: Selecting these check boxes indicates that, if available, initial values for the graph should be read from the Historian.

   ---

   trace1Value - Right-click in the **Property Name** field and select **Attach to Data>XML**.
4. In the "Attach to XML Data" dialog:

   XML Source - **update.xml** should already be selected.

   Data Key - Select **element1_data** from the drop down menu.
5. Click **OK** to apply these values and close the **Attach to XML Data** dialog.
6. In the **Object Properties** dialog:

   trace2Value - Right-click in the **Property Name** field and select **Attach to Data>XML**. This attaches the second trace to data.
7. In the "Attach to XML Data" dialog:

   XML Source - **update.xml** should already be selected.

   Data Key - Select **element1_load** from the drop down menu.
8. Click **OK** to apply these values and close the **Attach to XML Data** dialog.

   Data attachments in the History Graph now match the data attachments on the meters in the configuration display file you saved as **tutorial_history_config.rtv**.
9. Select **File>Save** and name this file **tutorial_history_display.rtv**.

   When you open this display, the graph will load initial data from the history database and will continue to update as live XML data comes in.
10. Select **File>Open** to reopen the display (**.rtv**) file you saved as **tutorial_history_display.rtv**.
11. In the **Object Palette** window, select the **Tables** tab and add the first table in the palette (class name: **obj_table02**) to the display.

12. In the **Object Properties** dialog:

objWidth - Increase to **500**. Press <Enter>.

label - Change to **History Table**. Press <Enter>.

valueTable - Right-click in the **Property Name** field and select **Attach to Data>SQL**.

13. In the **Attach to SQL Data** dialog:

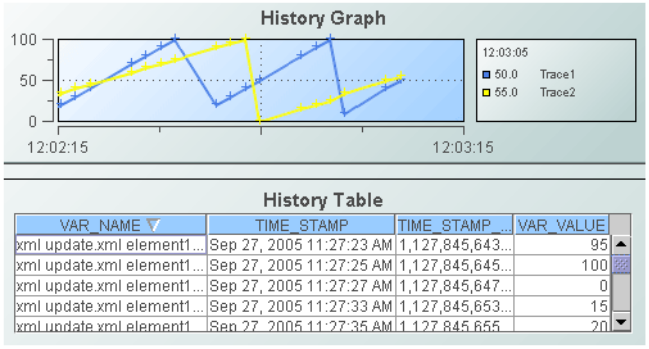
Database Name - **RTVHISTORY** should already be selected.

Table Name - Select **HISTORY** from the drop down menu.

Column(s) - Select ***** from the drop down menu.

14. Click **OK** to apply these values and close the **Attach to SQL Data** dialog.

The table will now display the entire HISTORY table from the history database.



Setup Database to Store Historical Tabular Data

In this section, you will need to add a table to the history database. If necessary, contact your database administrator to complete the next section.

1. Exit the Historian and Display Builder.

2. Add a table named **production_archive** to the history database.

3. Add the following columns to that table:

Field Name	Field Type
RowName	Text (10 characters)
Plant	Text (30 characters)
UnitsInProduction	Number
UnitsCompleted	Number
Status	Text (30 characters)
OnSchedule	Text (10 characters)

TimeStamp	Text (30 characters)
TimeStampLs	Text (30 characters)

Create a Configuration File to Store Historical Tabular Data

In this section, you will add a table to a display. The data you attach to this table will be used by the Historian to store data in the **production_archive** table you created in the history database.

1. "Start the Display Builder" in your original, initialized command/terminal window.
2. Select **Edit>Add** to open the Object Palette.
3. Select the **Tables** tab and add the first table in the palette (class name: **obj_table02**) to the display.
4. In the **Object Properties** dialog:

   label - Change to **Production Data**. Press <Enter>.

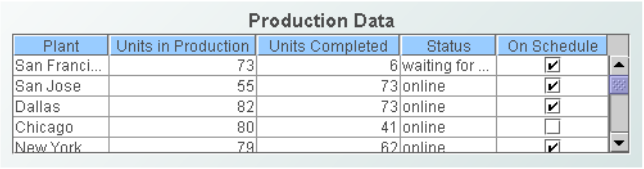
   objWidth - Increase to **500**. Press <Enter>.

   valueTable - Right-click in the **Property Name** field and select **Attach to Data>XML**.
5. In the "Attach to XML Data" dialog:

   XML Source - **update.xml** should already be selected.

   Data Key - Select **production_table** from the drop down menu.

   Column(s) - Select ***** from the drop down menu.
6. Click **OK** to apply these values and close the **Attach to XML Data** dialog.



7. In the **Object Properties** window:

   historyTableName - Type **production_archive** in the **Property Value** field. Press <Enter>.

   historyTableRowNameFlag - Select the check box.

Note: When you enter **production_archive** in the **Property Value** field, you are telling the Historian which table in your history database to store the tabular data in. Selecting the **historyTableRowNameFlag** check box tells the Historian to store the data from the row name field into the first column in the **production_archive** table in your history database.

---

8. Select **File>Save** and name this file **tutorial_history_table_config.rtv**.

Store Historical Tabular Data

1. **Restart the Historian:**

   On Windows

   In an initialized Windows command window (see "Initializing a Command Prompt or Terminal Window") type:

   start run_historian

   On UNIX

   In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window") type:

   run_historian &

2. Click **Add** on the **Configuration** tab.

3. Select **tutorial_history_table_config.rtv**

4. Click **Open** to insert into the **Data Configuration Files** list.

   The **Show Data in Console** flag should already be checked. This prompts the Historian to print a line to the console for each record it stores to the database.

5. Click **Save Configuration** to save these settings.

6. Click on the **Console** tab.

7. Click **Start Storing Data**.

   Data will immediately start printing to the console window.

Note: Let the Historian run as you go through the next section.

View Historical Tabular Data

1. In the Display Builder, click on the New button.

2. In the **Object Palette** window, select the **Tables** tab and add the first table in the palette (class name: **obj_table02**) to the display.

3. In the **Object Properties** dialog:

   label - Change to **Historical Production Data**. Press <Enter>.

   objWidth - Increase to **500**. Press <Enter>.

   objHeight - Increase to **350**. Press <Enter>.

   valueTable - Right-click in the **Property Name** field and select **Attach to Data>SQL**.

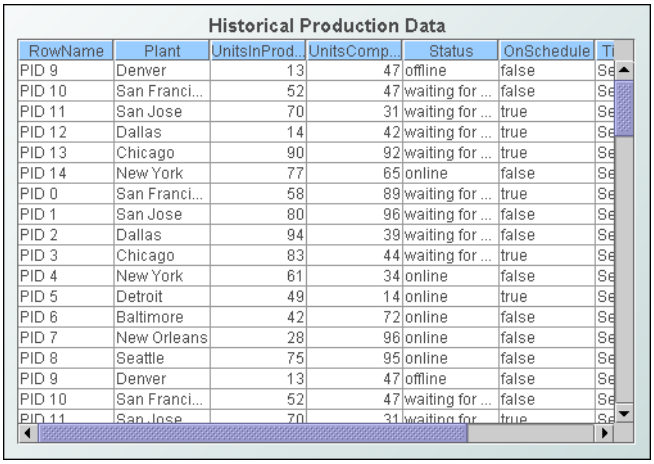
4. In the **Attach to SQL Data** dialog:

   Database Name - **RTVHISTORY** should already be selected.

   Table Name - Select **production_archive** from the drop down menu.

Column(s) - Select ***** from the drop down menu.

5. Click **OK** to attach data and close the **Attach to SQL Data** dialog.



6. Select **File>Save** and name this file **tutorial_history_table_display.rtv**.

7. Exit the Historian, Display Builder, and XML data simulator.

Serving Data

Objectives

- Configure the Data Server using a data configuration (**.rtv**) file (created in the Quick Start) as input
- Run the Data Server with this data configuration file
- View data redirected by the Data Server in the Display Viewer Application

---

Note: If you have not completed the "Quick Start Tutorial", please do so before continuing.

---

1. In an initialized command/terminal window (see "Initializing a Command Prompt or Terminal Window"):

2. "Start the XML Data Simulator"

3. Start the Data Server by typing:

   run_dataserver

If **mydisplay.rtv** is not listed in the **Data Configuration Files** list then:

4. Select **Add**.

5. Select **mydisplay.rtv** and click **Open**.

6. Check the **Show Data in Console** check box to print a line to the **Console** tab for each piece of data being served to the XML file.

7. Click the **Save Configuration** button to save these settings.

8. Select the **Console** tab and click the **Start Serving Data** button.

   Data should immediately output to the Data Server Console. If data is not appearing in the console, select the **Configuration** tab in the **Data Server** window and make sure that the **Show Data Console** check box is marked. If the box is marked and data is still not appearing in the console, then check that the data simulators (started in the Quick Start Example) are still running.

If you would like to view data in the Display Viewer Application continue with Step 9.

9. Start the Display Viewer Application:

   In an initialized command/terminal window (see "Initializing a Command Prompt or Terminal Window") type:

   run_viewer

10. Login to the Display Viewer. By default, the Display Viewer does not require a login. "Login" can be enabled at setup to support "Role-based Security". The default user name and password are:

    User Name: admin

    Password: admin

    Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role. See "Role-based Security" for more information.

11. In the Display Viewer, open the file **mydisplay.rtv**.

    Objects in the display should reflect data redirected by the Data Server.

Java Server Pages

Note: You may not be licensed to run all RTView data sources.

Requirements

This example requires that you have a working knowledge of RTView. If you would like to work through a basic example, please complete the "Quick Start Tutorial" before continuing. This exercise also assumes that you have a web server running and that you have a standard working knowledge of that server. In addition, you must be familiar with deploying Java Sever Pages on your web server.

Objective

- To understand how to reference a JSP file on an active server instead of directly referencing an XML file.

Using Java Server Pages is a convenient way to provide data to RTView. Instead of directly referencing an XML file, you can reference a JSP file on an active server. Via the JSP file, the server returns formatted XML data to each RTView client that requests it. The advantage of retrieving XML data from Java Server Pages is that an intermediary XML file is not necessary and all client server communication is managed automatically by the HTTP server. Using Java Server Pages also provides a way to limit the data transferred to only what is requested by the individual clients.

Setup

The **demos\dstutorial** directory contains the following files:

make_demo.bat	Script that compiles the JSP Data Simulator. A sample JSP Data Simulator (**gmsjsp/JspDemoSimulator.java**) has already been compiled. If the JSP Data Simulator is recompiled, you must set the environment variable **CATALINA_HOME** to be your web server root directory and run the **make_demo.bat** script.
make_war.bat	Script called by **make_demo.bat** to create a web application archive.
sample.jsp	The Java Server Page that references the JSP Data Simulator class.
xml_jsp_*.rtv	RTView Display files.
gmsjsp/JspDemoSimulator.java	Sample JSP Data Simulator that outputs XML data. This file can be modified to output and update your own data. If you modify the file, you must set the environment variable **CATALINA_HOME** to be your web server root directory and recompile the simulator using the **make_demo.bat** script.

1. In an initialized command window, go to the **demos\dstutorial** directory.
2. Copy the HTML files, displays files, and JSP Data Simulator class to your web server directory:

make_war	This script creates a web archive (**.war**) named **jspdata.war**.
install_to_tomcat jspdata	This script installs the web archive jspdata.war to your Tomcat server. **Note:** This script will shutdown and restart Tomcat and requires administrative permissions.

Note: If you are using a JSP servlet container other than Tomcat, install these files on your web server according to instructions given with that product.

---

*The **install_to_tomcat** script will not work if the **CATALINA_HOME** environment variable has not been set. If you did not set this variable following product installation, then return to the product "Setup" section for details before continuing with this example.*

Create a Display in the Display Builder

1. "Start the Display Builder" in your initialized command window.

   In the Display Builder, begin by setting up substitutions for the JSP path.

2. Select **Tools>Options** to open the **Application Options** dialog and click on the **Substitutions** tab.

3. In the String field enter **$jsp_path**.

4. Enter the location of the **jspdata** directory when run through your web server (e.g.: **http://myhost:myport/jspdata**) in the **Value** field.

5. Click **Add** to insert this into your list of substitutions.

Next, setup the XML Sources.

6. Click on the **XML** tab.

7. If **UnitTable** is already defined in the XML Sources display, double-click on it. Otherwise, click **Add**.

   XML Source Name - Enter **UnitTable**

   XML Source Path - Enter **$jsp_path/sample.jsp?unit=***

   Contains Substitutions - Select the check box

   Click **OK** to close the dialog.

8. Click **Save**

9. Select **No** when prompted to save the initialization file in the **lib** directory.

Next, add a label to the Working Area:

10. In the **Object Palette** window, select the **Labels** tab and click on the fifth label in the palette (class name: **obj_label05**).

11. To attach to data from the JSP Data Simulator, in the **Object Properties** dialog set the following properties for the object's label:

    label - Right-click in the Property Name field and select **Attach to Data>XML**.

12. In the "Attach to XML Data" dialog, enter the following:

    XML Source - Select **UnitTable** from the drop down menu

    Data Key - Select **Units** from the drop down menu

    Column(s) - Select **Name**

    Filter - Select the check box

    Filter Column - Select **Name**

    Filter Value - Type **A**

    Click **OK** to attach data and close the dialog.

13. To attach data to the object's value, set the following properties

value - Right-click in the **Property Name** field and select **Attach to Data>XML**.

14. In the **Attach to XML Data** dialog, enter the following:

XML Source - Select **UnitTable** from the drop down menu.

Data Key - Select **Units** from the drop down menu.

Column(s) - Select **Status**

Filter - Check the check box.

Filter Column - Select **Name**

Filter Value - Type **A**

Click **OK** to attach data and close the dialog.

The object's label should display the name of unit A and the object's value should display the status of unit A.

Next, set the command to update unit A:

15. In the **Object Properties** dialog, set the following:

command - Right-click in the **Property Name** field and select **Define Command>System**.

16. In the "Define System Command" dialog, enter the following:

Command Type - Select **Execute URL** from the drop down menu.

URL String - Enter **$jsp_path/sample.jsp?unit=A?key=status?value=2**

Click **OK** to apply the command and close the dialog.

17. To execute the command, double-click on the object.

The status should update to 2.

18. Select **File>Save** and name this display file as **tutorial_jsp.rtv**

19. Exit the Display Builder.

View Sample Displays in the Display Viewer

Sample displays (**xml_jsp_data.rtv** and **xml_jsp_controls.rtv**) are provided in the **demos\dstutorial** directory. These displays illustrate how to drill down to detailed displays, as well as how to use controls to update values on the server.

1. Open the Display Viewer by typing **run_viewer** in your initialized command window.

2. Login to the Display Viewer. By default, the Display Viewer does not require a login. "Login" can be enabled at setup to support "Role-based Security". The default user name and password are:

   User Name: admin

Password: admin

Note: It is possible that your system administrator may have configured another user name and password. In this case, you may also need to select a role. See "Role-based Security" for more information.

---

3. Double-click on the **XML Displays** folder in the tree.
4. Double-click on the **Java Server Pages** folder.
5. Select **Get Data Via JSP** in the tree to view the **xml_jsp_data.rtv** display.

If you did not complete the Setup procedures provided at the beginning of this example, you must select Setup in the tree before attempting to view other displays.

6. Click on a row in any of the tables to view the default drill down substitutions and to see more detailed information on these units.
7. Select **Set Data Via JSP** in the tree to view a display (**xml_jsp_controls.rtv**) that illustrates the use of controls to update values on the server.
8. Exit the Display Viewer.

APPENDIX A XML Tags and Attributes for Display (.rtv) Files

Display (.rtv) files are saved in an XML format. The following is a list of supported tags and attributes:

rtview Top level tag that includes the namespace attribute xmlns and the attribute version, which must be defined as: (xmlns="www.sl.com" version="3.0")

Attribute	Description
xmlns	Must be defined as www.sl.com
version	Must be defined as 3.0

model Describes the background model. Attributes of the model tag include:

Attributes	Description
file	Name of the background model class
width	Width of the background model (in pixels)
height	Height of the background model (in pixels)
bgColor	Index of the background color
doBevel	Indicates whether to bevel edge of the background
doGradient	Indicates whether to display a gradient in the background

image Describes the background image. Attributes of the image tag include:

Attribute	Description
file	Name of the background .gif, .jpeg or .png

object Describes an object in the display. Attributes of the object tag include:

Attribute	Description
class	The object's Java class name
name	Assigned and used internally
x	The x coordinate for the object
y	The y coordinate for the object
scale	Scale factor for the object
dd	Drill down target for the object
command	Command for the object

	confirmText	Text for the command confirmation dialog
	cmdClose	Value of the commandCloseWindowOnSuccess property for this object
	confirm	Command confirm policy for the object
	vardef	Describes a data attachment for a property of the object
	varname	Name of the property attached to the data
	dsstring	Data source key to use in updating the variable

link Describes a link object in the display. Attributes of the link tag include:

Attribute	**Description**
class	The link's Java class name
name	Assigned and used internally
source	Source object of the link
target	Target object of the link
dd	Drill down target for the link
command	Command for the link
confirm	Command confirm policy for the link
vardef	Describes a data attachment for a property of the link
varname	Name of the property attached to the data
dsstring	Data source key to use in updating the variable

localvar Describes a local variable. Attributes of the local variable tag include:

Attribute	**Description**
name	Local variable name
initValue	Initial value of the local variable

All properties not attached to data will be saved out as an attribute value pair.

The following is an example of a display (**.rtv**) file:

```
<?xml version="1.0"?>
    <rtview xmlns="www.sl.com" version="3.0">
        <model file="com.sl.gmsjrtview.m_basemodel"
               width="570" height="428"
                bgColor="13"
                doBevel="1"
                doGradient="0"/>
        <object name="N1" class="label12"
                y="45.375" x="15.625" scale="1.6"
                fieldHeight="3"
                label="label12"
                labelTextColor="7"
                labelTextHeight="1.5"
                value="0.0"
                fieldWidth="12"
                bgColor="0"
                raisedFlag="0"
                borderFlag="1"
```

```
            edgeWidth="2"
            valueTextColor="7"
            valueTextHeight="1.25">
    </object>
</rtview>
```

APPENDIX B Command Line

This section contains the following:

Command Line Options: Display Builder and Display Viewer

The following command line arguments are enabled when you run the Display Builder (see "Running the Display Builder") or Display Viewer (see "Display Viewer Application") from a Windows Command Prompt or UNIX terminal window. Options specified using command line arguments override values saved in initialization (e.g. **OPTIONS.ini**) files.

For command line options for your data source, refer to the "RTView Data Sources" section of this documentation.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: **"-sub:$data:my Data"**).

Name	Description
-bg	Set the RTView application to run as a background process. When this option is specified, the **GmsLauncher** process and run scripts exit immediately after the RTView application is started, rather than continuing to run, thereby reducing the host system process count. However, note that: • The RTView application output and error messages will not appear in the command/shell window from which it was launched. • Ctrl-c cannot be used to terminate the application. **Note:** This option is only recognized on the command line and is not read from, or saved to, any RTView options (**.ini**) files. Example: **-bg**

-confirm:(policy value)	Set the confirm policy for all commands, overriding the confirm policy on individual objects. Default is **0**. Values: **-1** - do not confirm any commands **1** - confirm all commands **0** - follow individual object confirm policy Example: **-confirm:-1**	
-customwindowtitle:(title)	Specify a custom window title. To specify an empty window title, enter a single space. By default, window titles contain the name of the application followed by the name of the display (**.rtv**) file (e.g. **RTView mydisplay.rtv**). A Custom Window Title: • Takes precedence over the title specified in your panel configuration file for "Multiple Display Panels". • Is superseded by the Window Title option in the **Drill Down Properties** dialog (see "Drill Down Displays"). Example: **-customWindowTitle:myTitle**	
-customRoleManagerClassName	Override default java class name (MyRoleManager) for Custom Role Manager. Example: **run_viewer -customRoleManagerClassName:com.xyz.RoleMgr**	
-customUserManagerClassName	Override default java class name (**MyUserManager**) for Custom User Manager. Example: **run_viewer -customUserManagerClassName:com.xyz.UserMgr**	
-dataserver:	(filename)	Read data from Data Server output file instead of directly from data sources. If no file name is specified, default output file (rtvdata.xml) will be used. If necessary, include local directory path or http URL. Example: **-dataserver:rtvdata.xml**
	remote:primary, backup1	Read data from Data Server instead of directly from data sources. Specify primary and backup servers. If no host is specified, local host will be used. If no port is specified, default port (3278) will be used. **Note:** Designation of a backup server is optional; one or multiple backup servers can be specified. Example: **-dataserver:remote:host:8723,host:8080**

	remote:http://host:port/rtvdata,http://host:port/rtvdata_backup1	Read data from Data Server via servlet instead of directly from data sources. Specify primary and backup servers. The host is web server hosting the servlet. The port is port used by the web server. **Note:** Designation of a backup server is optional; one or multiple backup servers can be specified. Example: **-dataserver:remote:http://host:8723/rtvdata,http://host:8080/rtvdata_backup**
	name=Name;connect=primary,backup1	Specify primary and backup named data server(s). The name is the Name specified when this data server was configured and connect is either host:port or, for servlet, *http://host:port/rtvdata*. **Note:** Designation of a backup server is optional; one or multiple backup servers can be specified. Example: **-dataserver:name=MyDataServer;connect=localhost:56789,host:8080** **-dataserver:name=London;connect=https://londonServer:8080/rtvdata,http://host:8080/rtvdata_backup**
-displayTemplate	Specify a display (**.rtv**) file to serve as a template for new displays. You can edit the template as desired (e.g. add objects). The Builder prompts you for a new file name when you save the changes so that the (original template) remains unchanged. Example: **run_builder -displayTemplate:MyTemplate.rtv** Property File Example: **sl.rtview.displayTemplate=MyTemplate.rtv**	
-dsenable:(dskey)	Enable data source(s) for data attachments and defined commands that have been configured to bypass data being redirected through the specified data server(s). The **dskey** is the abbreviation for the data source as listed in the **Attach to Data** and **Define Command** drop down menus, but in all lower case. Example: **-dsenable:sql**	
filename	Open a specific file in the Display Builder or Display Viewer. **Note:** If your "Login" doesn't allow you to view a particular display, the display will not open when you use the filename option. Example: **sample.rtv**	
-historytablename:(tablename)	Specify the table name (e.g., **MY_TABLE**) to use when loading historical data into graphs. **Note:** Table names cannot contain spaces. Example: **-historytablename:MY_TABLE**	

-log4j

Turns on Log4j logging for the RTView application. By default, RTView processes (Builder, Viewer, Data Server, Display Server, or Historian) print log messages to the console. To obtain log files, you redirect the RTView application output and error streams to a log file using Log4j.

After executing this command, the first time-stamped row in the log file appears as follows:

```
2012-02-02 14:00:54,693 INFO – [rtview] Log4j is being used with sl.log4j.properties as the configuration file.
```

When Log4j is not in use, the first time-stamped row in the log file appears as follows:

```
2012-02-03 10:40:31.866 [rtview] Logging redirected for System.out and System.err. Log4j is not in use.
```

(Note the missing INFO column when Log4j is not in use.)

For example:

```
run_builder –log4j
run_builder –log4j –log4jlevel:INFO –showlogcat
```

To run an RTView application as a background process using the **-bg** command line argument, use the **sl-bg.log4j.properties** configuration file (which only outputs to a log file rather than to a console).

-bg (background) example:

run_dataserver –bg –log4j –log4jprops:sl-bg.log4j.properties

Note: The logging method from previous versions of RTView does not use Log4j. This previous method of logging is enabled with **-logfile** and **–logdir** and is still supported. Do not use both the previous logging method and Log4j or you receive the following error message:

```
ERROR: log4j configuration ERROR - com.sl.rtview.useLog4j is set to true but -logfile redirection is in use. Log4j will not be used.
```

-showlogcat

Turns on the **Category** column in the log file output. When not in use, the **Category** column is not shown in the log file. When not in use, the **Category** column is not shown in the log file.

For example:

-showlogcat

-log4jprops

Specify the .properties file to use to format the Log4j log file. By default, sl.log4j.properties is used. Use this to provide a different property file name. The .properties file is searched for inside a .jar/.war file, then searched for in the current directory, and lastly searched for in the %RTV_HOME%/lib directory. The filename can have a path preceding it. For example, **C:\mydir\my.log4j.properties**.

You can also use Log4j configuration files in the XML format. For example, log4j.xml. For details, see *http://wiki.apache.org/logging-log4j/Log4jXmlFormat*.

For example:

-log4jprops:mylog4jfile.properties

	-log4jlevel	Specify the Log4j Level. INFO is used by default. Valid values are: **FATAL:** Indicates a severe error that likely causes the application to abort. **ERROR:** Indicates an event that might not cause the application to abort. **WARN:** Indicates a potentially harmful event. **DEBUG:** Indicates detailed informational about events for debugging the application. **INFO:** Indicates informational messages about the progress of the application at coarse-grained level. For example: **-log4jlevel:INFO**
-logdir:(dirname)		Specify to prefix the log file name that is set in the **-logfile** option to the directory name in which the log file is stored. If the **-logfile** option is not specified, this option is ignored. **Note:** This option is only recognized on the command line and is not read from, or saved to, any RTView options (**.ini**) files. Example: **-logdir:ABCcompany**
-logfile:(filename)		Specify the redirection of output and error messages to a file. The RTView application output and error message streams are redirected to the specified file. The file is created if it does not exist. By default, if the file does exist, its previous contents are cleared. If the name of the log file contains the string DDDD (four upper case D characters), the string is replaced with the current local date and time using the format yyMMdd_HHmmss. For example, if we execute the following command on Sep 27 2012 at 3:55:43 PM: **run_dataserver -logfile:dataserver_DDDD.log** a log file named dataserver_120927_155543.log is produced. In most cases, this is a unique filename so that the previous log file, if any, remains unchanged. Over time, a large number of log files can accumulate so it is advisable to periodically purge the old files. On Linux, the logrotate utility can be used to automate this. **Note:** The **-logfile** option is only recognized on the command line and is not read from, or saved to, any RTView options (**.ini**) files. Example: **run_dataserver -logfile:dataserver.log**
	-logappend	Appends new log file output to the previous file content. That is, if the dataserver.log file already exists, output from the new log process is added to the file, preserving pre-existing content. The file size can grow quite large so it is advisable to periodically rotate the file. On Linux, the logrotate utility can be used to automate this. For example: **run_dataserver -logfile:dataserver.log -logappend**
-login		Turns on role based security. A login dialog will come up at startup. Example: **-login**

-max_displays_in_cache	Sets the maximum number of display (**.rtv**) files with composite objects to cache. Default is **5**. If value is set to **0**, no displays are cached. Example: **-max_displays_in_cache:50**
-noedit	Display Builder only. Run with editing disabled. Example: **-noedit**
-nohistory	Supress historical data in graphs. Example: **-nohistory**
-nomenus	Display Viewer only. Run without menus. Example: **-nomenus**
-nosingleclick	Disables the default setting. Double-click to open drill down windows or execute commands. **Note:** This option applies to the Display Viewer only. Example: **-nosingleclick**
-panelconfig:(filename)	Specify the name of the panel configuration file for "Multiple Display Panels". Example: **-panelconfig:PANELS_GRID.ini**
-processName	Specify to identify applications running as background processes. This option tags a unique identifier onto RTView server instances, enabling you to differentiate between multiple instances of those RTView applications. This option allows you to stop a particular instance without eliminating the other instances. If no process name is specified, the RTView application name is used as the process name. For example, **run_builder-processName:XX** adds the following JVM option to the Java call: **-DPROCESS_NAME=XX** Where **XX** is the value you specified for the -**processName** argument. **Note:** Values with spaces cannot be used for this option on Unix. Example: **-processName:XX**
-resetlayout	Display Builder only. Starts with the default window layout. Example: **-resetlayout**

-resizemode:(mode)

Globally controls object layout when a display window is resized. It is also possible to set a specific Resize Mode for each particular display (**.rtv**) file using the "Background Properties" dialog.

In the Display Builder, the selected Resize Mode is only applied to drill down windows. The main window of the Display Builder is always in Crop mode.

All three resize modes support zooming the display (right-click -> zoom). In both Scale and Layout modes if the window is resized while the display is zoomed, then the resize will further zoom the display.

Values:

crop When the window is resized, the display stays the same size. If the window is bigger than the display, empty space will show around the display. If the window is smaller than the display, scrollbars will be added. The window is not forced to maintain its aspect ratio. This is the default for the Thin Client.

scale When the window is resized, the display and all of the objects in it are scaled to fit the available space. The window is forced to maintain its aspect ratio. This is the default for the Display Builder and Display Viewer Application.

layout When the window is resized, the display is resized to fit the available space. The objects in the display are positioned according to their anchor and dock properties. The window is not forced to maintain its aspect ratio.

Objects that are not docked or anchored will move relative to their offset from the top left corner of the display. For example, if the object is centered on the display, the object will move 50% of the resize amount. If the object is centered at 3/4 of the display, it will move 75% of the resize amount.

Example:

-resizemode:layout

-rtvpass

If "Login" is enabled, specify the password in plain text to use for the login. This parameter must be used in conjunction with **rtvuser** and will bypass the login dialog. If the **rtvrole** parameter is not specified for a user with multiple roles, the first role will be used. Use the **rtvsign** parameter instead to specify an encoded user name and password.

Note: If the user name or password specified is not valid, the login dialog will appear.

Example:

-rtvpass:admin

-rtvrole

If "Login" is enabled, specify the role to use for the login. This parameter must be used with **rtvsign** or **rtvuser** and **rtvpass**. If this parameter is not specified for a user with multiple roles, the first role will be used.

Example:

-rtvrole:admin

-rtvsign	If "Login" is enabled, specify an encoded user name and password to use for the login, and bypass the login dialog. Contact SL Technical Support at support@sl.com to request a copy of the utility to create the encoded strings. If the **rtvrole** parameter is not specified for a user with multiple roles, the first role will be used. **Note:** If the user name or password specified is not valid, the login dialog will appear. Example: **-rtvsign:8I559A5NA8A5864J6J924N0B2**
-rtvuser	If "Login" is enabled, specify the user name in plain text to use for the login. This parameter must be used in conjunction with **rtvpass** and will bypass the login dialog. If the **rtvrole** parameter is not specified for a user with multiple roles, the first role will be used. Use the **rtvsign** parameter instead to specify an encoded user name and password. **Note:** If the user name or password specified is not valid, the login dialog will appear. Example: **-rtvuser:admin**
-saveusers	Saves the user definition file with encoded passwords. The file is only saved if you are logged in the admin role and you are not using the Custom User Manager. See "Configuration" for more information. Example: **-saveusers**
-singleclick	Single-click to open drill down windows or execute commands. This is the default setting. **Note:** This option applies to the Display Viewer only. Example: **-singleclick**
-stylesheet:(filename)	Specify style sheet(s) to apply to all displays in your applications. **Note:** Style sheets are applied at startup. If you edit a style sheet, then you need to restart. Example: **-stylesheet:rtv_darkstyles.rts** When multiple style sheet (**.rts**) files are applied, they are processed in the order specified. Therefore if the same property is specified in multiple style sheets, the value in the last style sheet applied (e.g. **stylesheet3.rts**) will take precedence. Example: -**stylesheet:stylesheet1.rts,stylesheet2.rts,stylesheet3.rts**.
-sub:(substring:subvalue)	Add a substitution string/value pair. Multiple substitution pairs can be specified on the command line. **Note:** Substitution strings cannot contain the following: : | . tab space , ; = < > ' " & / \ { } [ ] ( ) If your substitution value contains single quotes, you must escape them using a /. Example: **-sub:$1:myValue** **-sub:$filter:Plant=/'SanFrancisco/'**

-tabcomposites:	If true, enables the **Tab** key to switch focus to controls that are inside composite objects (see "Composite Object"). If false or not specified, the **Tab** key excludes controls that are inside composite objects. This option applies to the Display Viewer and the Builder preview window only. Property File Example: `sl.rtview.tabcomposites=true` Command Line Example: **run_viewer -tabcomposites:true**
-timezone	Set the default timezone for interpreting and displaying dates. Include a Java timezone ID or a custom ID, such as "GMT-8:00". Unrecognized IDs will be treated as GMT. See "Timezone ID Values" for more information. If you run the RTView Builder with a valid timezone parameter and then save Application Options, the timezone information will be persisted. To prevent the persisted timezone value from being used, pass "none" as the timezone ID. Example: **-timezone:US/Eastern** **-timezone:none**
-u(milliseconds)	Set update rate in milliseconds. Default is **2000**. Example: **-u5000 (updates every 5 seconds)**
-version	Prints the RTView application version information and exits immediately. Example: **run_builder -version**

Options Enabled with Alerts

In addition to the General Options, the following command line arguments are enabled with the Alert data source. See "Alerts" and "Audit Alert Action" for more information.

Name	Description
-actionauditdb:(database)	Specifies name of a database connection, as defined on the SQL tab, in which to store "Audit Alert Action" information. Example: **-actionauditdb:ALERTBD**
-actionaudittable:(table)	Specifies name of the table in the Alert Action Audit Database in which to store the "Audit Alert Action" information. Example: **-actionaudittable:ACTION_AUDIT_TABLE**
-alertcleartime:(number of seconds)	Specifies the rate, in seconds, to remove cleared alerts. Example: **-alertcleartime:3**

-alertds:alertdef:(filename)	Adds an alert definition file. Cannot specify substitutions. To specify substitutions, use the "Application Options - Alerts" tab. Example: **-alertds:alertdef:myalerts.rtv**
-alertds:enabled:(true or false)	Enables/disables all alerts in the active alert definition files. Example: **-alertds:enabled:false**
-alertds:history:(size of table)	Sets the number of rows that are stored in the "AlertTable". Example: **-alertds:history:1000**
-alertinitdelay:(number of seconds)	Specifies the duration, in seconds, to wait after startup to begin executing alerts. Example: **-alertinitdelay:5**
-cleansettingstable:(true or false)	If true, delete entries from the Alert Settings Table for alert names that are not defined in RTView. **Note:** This is done at startup after alert configuration files are processed and all of the alerts are loaded. Example: **-cleansettingstable:true**
-enableactionaudit:(true or false)	If true and configured, alert actions will be stored to the specified database table. Example: **-enableactionaudit:true**
-exitOnPersistInitFailed	Specifies what occurs when alert persistence is enabled but cannot be initialized due to a database problem or configuration issue. When **false** (the default setting), RTView initializes the alerts with persistence disabled. This is consistent with the behavior in releases previous to RTView 6.6.0 and Enterprise Monitor 2.2.0. When **true**, RTView exits after the persistence initialization has failed without initializing the alerts. Example: **-exitOnPersistInitFailed:true**
-ignorelutforcount:(true or false)	If true, the **AlertTable Count** column increments for an alert when new data is received even if the Last Update Time has not changed. This can cause invalid Counts for alerts attached to caches. If false or not specified, the **AlertTable Count** column increments for an alert only if the Last Update Time has also updated. This is the default behavior. Example: **-ignorelutforcount:true**
-lutupdatesnewdata:(true or false)	Enables\disables updates to the AlertTable when **New Data Only** is selected and to the "Alert Persistence" database when the only columns that contain changes for that row are **Last Update Time** and **Count**. By default, the **Last Update Time** and **Count** columns are not tracked by the **Row Update Time** column. To track the updates of the two columns in the **Row Update Time** column, use the **-lutupdatesnewdata** command line option. Example: **-lutupdatesnewdata:true**

-multipleindexdelim:(string)	For alerts with multiple index columns, create a unique alert index by concatenating all of the index column values. Value can be any string, except the following: • comma (,) • semi-colon (;), or • empty string. Default is tilde (~). Example: **-multipleindexdelim:~**
-persistInitDelayTime:(number of seconds)	Specify the amount of time, in seconds, to delay a backup Data Server from reading the alert persistence database during a failover. The default is 5 seconds. Increase the amount of time if the persistence database is slow or if you expect a large number of alerts to change on each update period. Otherwise, there might not be enough time for the failing Data Server to write all the alerts to the database before the backup server reads them. **Note:** Even with high availability configurations, there are cases in which some alerts might not be persisted. For example: • The persistence database fails. In this case, alerts cannot be written to, or read from, the database. If a failover occurs while the database is down, the backup server will not be able to read persisted alerts from the database. This will also happen if persistence is configured to use a Persistence Data Server to access the database, and the Persistence Data Server is down during failover. • The Alert Server is terminated in a non-orderly shutdown. Alerts are written out once per update period and once during orderly termination. If there is a non-orderly shutdown, some alerts might not be written to the database. In cases where alerts are not persisted, the new primary Data Server generates new alerts if the data is still in an alert state. The new primary Data Server might also re-use ID's that were used by the failed Data Server. Example: **-persistInitDelayTime:10**
-purgepersistedalerts	Clears all alerts for the alert engine from the "Alert Persistence" database on startup and no persisted alerts will be loaded. **Note:** If you are persisting alerts for more than one alert engine in the same database, alerts for other alert engines will not be removed.
-printssawarnings:(true or false)	If false, the Self Service Alerts warnings about extra unmapped thresholds will be suppressed. **Note:** This option only applies to Self Service Alerts. Example: **-printssawarnings:false**

Command Line Options: Display Server

The following command line arguments are enabled when you run the Display Server from a Windows Command Prompt or UNIX terminal window. Options specified using command line arguments override values saved in initialization (e.g. **OPTIONS.ini**) files.

For command line options for your data source, refer to the "RTView Data Sources" section of this documentation.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: **"-sub:$data:my Data"**).

Name	Description
-bg	Set the RTView application to run as a background process. When this option is specified, the **GmsLauncher** process and run scripts exit immediately after the RTView application is started, rather than continuing to run, thereby reducing the host system process count. However, note that: • The RTView application output and error messages will not appear in the command/shell window from which it was launched. • Ctrl-c cannot be used to terminate the application. **Note:** This option is only recognized on the command line and is not read from, or saved to, any RTView options (**.ini**) files. Example: **-bg**
-cellsperexport	Specify to limit the number of table cells included in HTML/Excel exports requested by the Thin Client. This option avoids out-of-memory exceptions and timeouts when exporting tables with many rows. This option is typically used in conjunction with the **cellsperpage** option, and has the following behavior: • If the **cellsperexport** option is not specified, or if a value of less than a 1000 is specified, the Display Server attempts to export all rows for all tables, regardless of the table size. • If a table contains fewer cells than the **cellsperexport** setting, the Display Server exports all rows for that table. • The exported HTM/Excel table starts with the same first row (or near it) that is visible in the Thin Client. That is, if you scroll to row 900 in the Thin Client and perform an HTML/Excel export, the exported table will begin near line 900. • If the rows included in an export to HTML/Excel are limited by the **cellsperexport** option, the first row in the exported file is the same as the first row currently visible in the Thin Client. • An export to HTML/Excel requires less CPU and memory than an export to PDF (see **cellsperreport**), therefore the value of the **cellsperexport** option is typically larger than the value of the **cellsperreport** option. For example, if an export to HTML/Excel was performed on a table with 5 columns and 100,000 rows, and the option **-cellsperexport:30000** was specified when the Display Server was launched, 6000 rows (30000/5) would be included in an exported HTML/Excel file for that table. If the **-cellsperreport:5000** option was specified, an export to PDF would include 1000 rows from the table. **Note:** This option is not recognized by the Builder or the Viewer. Example: **run_displayserver -cellsperpage:10000 -cellsperexport:30000**

-cellsperpage

Specify server-side table paging and sorting mode, also referred to as paging mode, for the Thin Client. Paging mode improves the performance of displays containing table objects (**obj_table02**) with many rows. In paging mode, the Display Server sends a specified maximum number of table data rows at a time to the Thin Client, rather than sending all table data rows at once. This option avoids out-of-memory exceptions, timeouts and sluggish performance that can otherwise occur from processing and transmitting all of the rows at once. This option is typically used in conjunction with the **cellsperexport** and **cellsperreport** options.

The page of rows sent from the Display Server to the Thin Client includes all of the rows currently visible in the Thin Client plus additional rows above and/or below the visible rows. If the user scrolls beyond the rows contained in the current page or clicks on a column header to change the sorting order, the Display Server sends another page of rows in response.

The **cellsperpage** option can also be specified in the **DISPLAYSERVER.ini** file. See "Display Server Configuration" for more information.

Command line example:

run_displayserver -cellsperpage:NNNN

where NNNN specifies the number of table cells the Display Server includes per page. Typical values for **cellsperpage** are 10000 to 30000.

The number of cells in a table is equal to the number of rows times the number of columns. For example, if **cellsperpage = 10000**, and the display contains a table with 5 columns, the Display Server uses a page size of 2000 rows for the table. This means the Display Server sends a maximum of 2000 table rows to the Thin Client at a time. If the table contains 40,000 rows the Display Server sends rows 1 through 2000 to the Thin Client when a user opens the display. If the user then scrolls to the bottom of the table, the Display Server sends rows 38,001 through 40,000 to the Thin Client. Similarly, if the user clicks on a column header to sort by that column, the Display Server sorts the table and sends the first 2000 sorted rows to the Thin Client. After the user scrolls or sorts a table in paging mode, the Thin Client displays '...' in each cell and an hourglass cursor appears over the table while it waits to receive the new page from the Display Server. The **cellsperpage** option also has the following behavior:

- A smaller value for **cellsperpage** reduces the memory requirements for processing large tables in the Display Server, Application Server, and browser. A larger value makes for smoother scrolling in the Thin Client because it increases the number of rows through which the Thin Client can scroll before it needs to request another page from the server.
- If the **cellsperpage** option is not specified, or if a value less than a 1000 is specified, paging mode is disabled and the Display Server sends all data rows to the Thin Client for all tables regardless of the table size.
- If a table contains fewer cells than the **cellsperpage** setting, the Display Server sends all rows for that table.
- Paging mode only affects the behavior of **obj_table02** objects, and only in the Thin Client.
- The **maxNumberOfHistoryRows** property on **obj_table02** limits the maximum number of rows that are shown in the table, regardless of the **cellsperpage** setting.
- After scrolling to a new page, if the user immediately drags the scrollbar again, the table may not react. The user may need to nudge the scrollbar knob or click on the scroll up/down arrows to make the table react.

- The Thin Client Export Table to Excel/HTML/PDF feature is not affected by paging mode. Attempts to export a table with many rows may result in timeouts or out-of-memory exceptions in the Display Server or in the Display Servlet. To avoid those errors, see the **cellsperexport** and **cellsperreport** Display Server options.
- When displaying a table with more than 75,000 rows in Internet Explorer version 8, the last row in the table may be partially obscured even if the scrollbar knob is dragged to the bottom position. The last row can be made visible by using the mouse wheel but that may cause unused space to appear at the bottom of the table.
- After the user clicks a column to sort a table, the table vertical scrollbar is reset to the top position.
- The Thin Client ignores the **scrollToSelectionFlag** property on a table that is in paging mode.
- A table row selection is cleared when a different page of rows is received from the Display Server. If a table is in paging mode, only rows that are on the current page can be selected, even if the table **multiSelectFlag** property is checked.

-cellsperreport

Specify to limit the number of table cells included in PDF exports requested by the Thin Client. This option avoids out-of-memory exceptions and timeouts when exporting tables with many rows. This option is typically used in conjunction with the **cellsperpage** option, and has the following behavior:

- If the **cellsperexport** option is not specified, or if a value of less than a 1000 is specified, the Display Server attempts to export all rows for all tables, regardless of the table size.
- If a table contains fewer cells than the **cellsperreport** setting, the Display Server exports all rows for that table.
- In an exported PDF file, the scroll position of the Thin Client has no effect on the starting row in the PDF file, nor any effect on the rows that are included in the PDF report.
- An export to PDF requires more CPU and memory than an export to HTML/Excel (see **cellsperexport**), therefore the value of the **cellsperreport** option is typically smaller than the value of the **cellsperexport** option. For example, if an export to HTML/Excel was performed on a table with 5 columns and 100,000 rows, and the option **-cellsperexport:30000** was specified when the Display Server was launched, 6000 rows (30000/5) would be included in an exported HTML/Excel file for that table. If the **-cellsperreport:5000** option was specified, an export to PDF would include 1000 rows from the table.

Note: This option is not recognized by the Builder or the Viewer.

Example:

run_displayserver -cellsperpage:10000 -cellsperreport:5000

-clearsubs:

Set to false in order to preserve the value of local variables, for which the Use as Substitution option is selected, when navigating to different displays in the same panel. This behavior is consistent with that of the Display Viewer.

By default (or if set to true), the Display Server will reset all local variables to their initial values when navigating to a new display regardless of the variable's value in the current display.

Note: Global variables are not affected by this option.

Example:

run_displayserver -clearsubs:false

-daemon	Run the Display Server as a daemon process. Example: **run_displayserver -daemon**
-dataserver:	**(filename)** - Read data from Data Server output file instead of directly from data sources. If no file name is specified, default output file (**rtvdata.xml**) will be used. If necessary, include local directory path or http URL. Example: **-dataserver:rtvdata.xml** **//(host:port)** - Read data from Data Server via socket instead of directly from data sources. If no host is specified, local host will be used. If no port is specified, default port (3278) will be used. Example: **-dataserver://remotehost:8723** **remote:primary,backup1** - Read data from Data Server instead of directly from data sources. Specify primary and backup servers. If no host is specified, local host will be used. If no port is specified, default port (**3278**) will be used. **Note:** Designation of a backup server is optional; one or multiple backup servers can be specified. Example: **-dataserver:remote:host:8723,host:8080** **remote:http://host:port/rtvdata,http://host:port/rtvdata_backup1** - Read data from Data Server via servlet instead of directly from data sources. Specify primary and backup servers. The host is web server hosting the servlet. The port is port used by the web server. **Note:** Designation of a backup server is optional; one or multiple backup servers can be specified. Example: **-dataserver:remote:http://host:8723/rtvdata,http://host:8080/rtvdata_backup** **name=name;connect=primary,backup1** - Specify primary and backup named data server(s). The name is the Name specified when this data server was configured and connect is either host:port or for servlet http://host:port/rtvdata. **Note:** Designation of a backup server is optional; one or multiple backup servers can be specified. Example: **-dataserver:name=MyDataServer;connect=localhost:56789,host:8080** **-dataserver:name=London;connect=https://londonServer:8080/rtvdata,http://host:8080/rtvdata_backup**
-dsenable:(dskey)	Enable data source(s) for data attachments and defined commands that have been configured to bypass data being redirected through the specified data server(s). The **dskey** is the abbreviation for the data source as listed in the Attach to Data and Define Command drop down menus, but in all lower case. Example: **-dsenable:sql**

-exit_on_out_of_memory	Force the Display Server to terminate when an OutOfMemoryException occurs. This option is intended for use in a HA deployment where a backup server is available, otherwise the process may continue to run in a crippled state and prevent the backup from taking over. **Note:** The Display Server is not automatically restarted by this option and must be restarted manually. Example: **-exit_on_out_of_memory**
-imageformat	Specify image format: jpg or png. By default, the Display Server will automatically select the image that results in the fewer number of bytes for each display. Example: **-imageformat:jpg**
-imagequality:	Specify a value between 0 and 100 to control the quality of .jpg images. If the value is 100, the Display Server will output the highest quality image with the lowest compression. If the value is 0, the Display Server will output the lowest quality image using the highest compression. Default is 75. If the -verbose option is specified at startup, image creation time/size for each display refresh will be shown in the Display Server console output. Example: **-imagequality:75** **Note:** The Display Server chooses an image format (.jpg or .png) for each display, depending on which format produces the smallest image size (in bytes). The size of .png images is controlled by the -pngcompress option.
-jmxport:(port number)	The port number to use to expose JMX methods to monitor and manage the Display Server. There is no default port. If not specified, these JMX methods will not be accessible. See "Managing the Display Server Using JMX" for more information. Example: **-jmxport:9998**

-log4j

Turns on Log4j logging for the RTView application. By default, RTView processes (Builder, Viewer, Data Server, Display Server, or Historian) print log messages to the console. To obtain log files, you redirect the RTView application output and error streams to a log file using Log4j.

After executing this command, the first time-stamped row in the log file appears as follows:

```
2012-02-02 14:00:54,693 INFO – [rtview] Log4j is being used with sl.log4j.properties as the configuration file.
```

When Log4j is not in use, the first time-stamped row in the log file appears as follows:

```
2012-02-03 10:40:31.866 [rtview] Logging redirected for System.out and System.err. Log4j is not in use.
```

(Note the missing INFO column when Log4j is not in use.)

For example:

run_displayserver –log4j

run_displayserver –log4j –log4jlevel:INFO –showlogcat

To run an RTView application as a background process using the -bg command line argument, use the sl-bg.log4j.properties configuration file (which only outputs to a log file rather than to a console).

-bg (background) example:

run_displayserver –bg –log4j –log4jprops:sl-bg.log4j.properties

Note: The logging method from previous versions of RTView does not use Log4j. This previous method of logging is enabled with **-logfile** and **–logdir** and is still supported. Do not use both the previous logging method and Log4j or you receive the following error message:

```
ERROR: log4j configuration ERROR - com.sl.rtview.useLog4j is set to true but -logfile redirection is in use. Log4j will not be used.
```

-showlogcat - Turns on the Category column in the log file output. When not in use, the Category column is not shown in the log file. When not in use, the Category column is not shown in the log file.

For example:

-showlogcat

-log4jprops - Specify the .properties file to use to format the Log4j log file. By default, sl.log4j.properties is used. Use this to provide a different property file name. The .properties file is searched for inside a .jar/.war file, then searched for in the current directory, and lastly searched for in the %RTV_HOME%/lib directory. The filename can have a path preceding it. For example, C:\mydir\my.log4j.properties.

For example:

-log4jprops:mylogfile.properties

-log4jlevel - Specify the Log4j Level. INFO is used by default. Valid values are:

FATAL: Indicates a severe error that likely causes the application to abort.

ERROR: Indicates an event that might not cause the application to abort.

WARN: Indicates a potentially harmful event.

DEBUG: Indicates detailed informational about events for debugging the application.

INFO: Indicates informational messages about the progress of the application at coarse-grained level.

For example:

-log4jlevel:INFO

-logdir	Specify to prefix the log file name that is set in the **-logfile** option to the directory name in which the log file is stored. If the **-logfile** option is not specified, this option is ignored. **Note:** This option is only recognized on the command line and is not read from, or saved to, any RTView options (**.ini**) files. Example: **-logdir:ABCcompany**
-logfile	Specify the redirection of output and error messages to a file. The RTView application output and error message streams are redirected to the specified file. The file is created if it does not exist. If the file does exist, its previous contents are cleared. **Note:** This option is only recognized on the command line and is not read from, or saved to, any RTView options (**.ini**) files. Example: **-logfile:DisplayServer.log**
-passclientlogin	Pass RTView login information into all data sources that have the Use Client Credentials option enabled. **Note:** Some data sources do not support this feature. For information on Application Options for your data source, refer to the Data Sources section of this documentation. Example: **-passclientlogin**
-pngcompress	Specify a value between 1 and 9 to control the quality of .png images. If the value is 1, the Display Server will output the highest quality image with the lowest compression. If the value is 9, the Display Server will output the lowest quality image using the highest compression. Default is 9. A value of 3 is a good compromise between speed and quality. If the **-verbose** option is specified at startup, image creation time/size for each display refresh will be shown in the Display Server console output. Example: **-pngcompress:3** **Note:** The Display Server chooses an image format (.jpg or .png) for each display, depending on which format produces the smallest image size (in bytes). The size of .jpg images is controlled by the **-imagequality** option.
-processName	Specify to identify applications running as background processes. This option tags a unique identifier onto RTView server instances, enabling you to differentiate between multiple instances of those RTView applications. This option allows you to stop a particular instance without eliminating the other instances. If no process name is specified, the RTView application name is used as the process name. For example, **run_builder-processName:XX** adds the following JVM option to the Java call: **-DPROCESS_NAME=XX** Where **XX** is the value you specified for the **-processName** argument. **Note:** Values with spaces cannot be used for this option on Unix. Example: **-processName:XX**

-resizemode:(mode)

Globally controls object layout when a display window is resized. It is also possible to set a specific Resize Mode for each particular display (.rtv) file using the Background Properties dialog.

In the Display Builder, the selected Resize Mode is only applied to drill down windows. The main window of the Display Builder is always in Crop mode.

All three resize modes support zooming the display (right-click -> zoom). In both Scale and Layout modes if the window is resized while the display is zoomed, then the resize will further zoom the display.

Values:

crop - When the window is resized, the display stays the same size. If the window is bigger than the display, empty space will show around the display. If the window is smaller than the display, scrollbars will be added. The window is not forced to maintain its aspect ratio. This is the default for the Thin Client.

scale - When the window is resized, the display and all of the objects in it are scaled to fit the available space. The window is forced to maintain its aspect ratio. This is the default for the Display Builder and Display Viewer Application.

layout - When the window is resized, the display is resized to fit the available space. The objects in the display are positioned according to their anchor and dock properties. The window is not forced to maintain its aspect ratio.

Objects that are not docked or anchored will move relative to their offset from the top left corner of the display. For example, if the object is centered on the display, the object will move 50% of the resize amount. If the object is centered at 3/4 of the display, it will move 75% of the resize amount.

Example:

-resizemode:layout

-standby:warm

Run a backup Display Server without the overhead of maintaining the Alert and Cache data sources.

The following actions will be delayed until the backup server has become the primary:

- Loading definition files (i.e. Global, Alert, Cache)
- Preloading display files specified in initialization (.ini) files or on the command line

Note: Although the **-standby:warm** option reduces overhead because data sources do not provide data until a failover, it is important to note that Alert and Cache data definitions will not start collecting data until the first client connects. Therefore, any previous alert state or cached data from the primary server will not be available to the backup.

Example:

-standby:warm

-stylesheet:(filename)

Specify style sheet(s) to apply to all displays in your applications.

Note: Style sheets are applied at startup. If you edit a style sheet, then you need to restart.

Example:

-stylesheet:rtv_darkstyles.rts

When multiple style sheet (**.rts**) files are applied, they are processed in the order specified. Therefore if the same property is specified in multiple style sheets, the value in the last style sheet applied (e.g. stylesheet3.rts) will take precedence.

Example:

-stylesheet:stylesheet1.rts,stylesheet2.rts,stylesheet3.rts.

-sub:(substring:subvalue)	Add a substitution string/value pair. Multiple substitution pairs can be specified on the command line. **Note:** Substitution strings cannot contain the following: : | . tab space , ; = < > ' " & / \ { } [ ] ( ) If your substitution value contains single quotes, you must escape them using a /. Example: **-sub:$1:myValue** **-sub:$filter:Plant=/'SanFrancisco/'**
-substracelim:(number of characters)	For requests from the Thin Client, limits the total number of characters for substitution name:value that the verbose output includes when listing substitutions. This argument prevents the Display Server log file from quickly filling with requests that have very long substitution strings. If no value is specified, the default, 1000, is used. A value of 0 specifies that no limit is applied to the substitutions listed by -verbose (as in previous releases). For example, the following limits the total number of characters for substitution **run_displayserver -xmlonly -verbose -substracelim:200** In the Display Server console, the messages indicate that the list of substitutions is truncated, for example: 2012-04-23 15:17:30.649 request: **get_image_map, name=test1 subs=$currentDisplay: test1.rtv** `$celldata:0 col1:0;1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20;21;22;23;24;25;26;27;28;29;30;31;32;33;34;35;36;37;38;39;40;41;42;43;44;45;46;47;48;49;50;51;52;53;54;5 <truncated, 77666 more chars> panel=1c32a24bc56547cfb1d5d69fa60b4298, tid=20` Example: **-substracelim:200**
-timezone	Set the default timezone for interpreting and displaying dates. Include a Java timezone ID or a custom ID, such as "GMT-8:00". Unrecognized IDs will be treated as GMT. See "Timezone ID Values" for more information. If you run the RTView Builder with a valid timezone parameter and then save Application Options, the timezone information will be persisted. To prevent the persisted timezone value from being used, pass "none" as the timezone ID. Example: **-timezone:US/Eastern** **-timezone:none**
-u(milliseconds)	Set update rate in milliseconds. Default is **2000**. Example: **-u5000 (updates every 5 seconds)**
-version	Prints the RTView application version information and exits immediately. Example: **run_displayserver -version**

Options Enabled with Alerts

In addition to the General Options, the following command line arguments are enabled with the Alert data source. See "Alerts" and "Audit Alert Action" for more information.

Name	Description
-actionauditdb:(database)	Specifies name of a database connection, as defined on the SQL tab, in which to store "Audit Alert Action" information. Example: **-actionauditdb:ALERTBD**
-actionaudittable:(table)	Specifies name of the table in the Alert Action Audit Database in which to store the "Audit Alert Action" information. Example: **-actionaudittable:ACTION_AUDIT_TABLE**
-alertcleartime:(number of seconds)	Specifies the rate, in seconds, to remove cleared alerts. Example: **-alertcleartime:3**
-alertds:alertdef:(filename)	Adds an alert definition file. Cannot specify substitutions. To specify substitutions, use the Application Options dialog. Example: **-alertds:alertdef:myalerts.rtv**
-alertds:enabled:(true or false)	Enables/disables all alerts in the active alert definition files. Example: **-alertds:enabled:false**
-alertds:history:(size of table)	Sets the number of rows that are stored in the AlertTable. Example: **-alertds:history:1000**
-alertinitdelay:(number of seconds)	Specifies the duration, in seconds, to wait after startup to begin executing alerts. Example: **-alertinitdelay:5**
-cleansettingstable:(true or false)	If true, delete entries from the Alert Settings Table for alert names that are not defined in RTView. **Note:** This is done at startup after alert configuration files are processed and all of the alerts are loaded. Example: **-cleansettingstable:true**
-enableactionaudit:(true or false)	If true and configured, alert actions will be stored to the specified database table. Example: **-enableactionaudit:true**
-exitOnPersistInitFailed	Specifies what occurs when alert persistence is enabled but cannot be initialized due to a database problem or configuration issue. When **false** (the default setting), RTView initializes the alerts with persistence disabled. This is consistent with the behavior in releases previous to RTView 6.6.0 and Enterprise Monitor 2.2.0. When **true**, RTView exits after the persistence initialization has failed without initializing the alerts. Example: **-exitOnPersistInitFailed:true**

-ignorelutforcount:(true or false)	If true, the **AlertTable Count** column increments for an alert when new data is received even if the Last Update Time has not changed. This can cause invalid Counts for alerts attached to caches. If false or not specified, the **AlertTable Count** column increments for an alert only if the Last Update Time has also updated. This is the default behavior. Example: **-ignorelutforcount:true**
-lutupdatesnewdata:(true or false)	Enables\disables updates to the "AlertTable" when **New Data Only** is selected and to the "Alert Persistence" database when the only columns that contain changes for that row are **Last Update Time** and **Count**. By default, the **Last Update Time** and **Count** columns are not tracked by the **Row Update Time** column. To track the updates of the two columns in the Row Update Time column, use the -**lutupdatesnewdata** command line option. See "Attach to Alert Data Dialog" and "Viewing Alerts" for more information. Example: **-lutupdatesnewdata:true**
-multipleindexdelim:(string)	For alerts with multiple index columns, create a unique alert index by concatenating all of the index column values. Value can be any string, except the following: • comma (,) • semi-colon (;), or • empty string. Default is tilde (**~**). Example: **-multipleindexdelim:~**
-persistInitDelayTime:(number of seconds)	Specify the amount of time, in seconds, to delay a backup Data Server from reading the alert persistence database during a failover. The default is 5 seconds. Increase the amount of time if the persistence database is slow or if you expect a large number of alerts to change on each update period. Otherwise, there might not be enough time for the failing Data Server to write all the alerts to the database before the backup server reads them. **Note:** Even with high availability configurations, there are cases in which some alerts might not be persisted. For example: • The persistence database fails. In this case, alerts cannot be written to, or read from, the database. If a failover occurs while the database is down, the backup server will not be able to read persisted alerts from the database. This will also happen if persistence is configured to use a Persistence Data Server to access the database, and the Persistence Data Server is down during failover. • The Alert Server is terminated in a non-orderly shutdown. Alerts are written out once per update period and once during orderly termination. If there is a non-orderly shutdown, some alerts might not be written to the database. In cases where alerts are not persisted, the new primary Data Server generates new alerts if the data is still in an alert state. The new primary Data Server might also re-use ID's that were used by the failed Data Server. Example: **-persistInitDelayTime:10**

-purgepersistedalerts	Clears all alerts for the alert engine from the "Alert Persistence" database on startup and no persisted alerts will be loaded. **Note:** If you are persisting alerts for more than one alert engine in the same database, alerts for other alert engines will not be removed.
-printssawarnings:(true or false)	If false, the Self Service Alerts warnings about extra unmapped thresholds will be suppressed. **Note:** This option only applies to Self Service Alerts. Example: **-printssawarnings:false**

Command Line Options: Historian

The following command line arguments are enabled when you run the Historian from a Windows Command Prompt or UNIX terminal window. Options specified using command line arguments override values saved in initialization (e.g. **OPTIONS.ini**) files. See "Starting the Historian" for more information.

For command line options for your data source, refer to the "RTView Data Sources" section of this documentation.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: **"-sub:$data:my Data"**).

Name	Description	
	Historian and Advanced Historian	**Advanced Historian Only**
-bg	Set the RTView application to run as a background process. When this option is specified, the **GmsLauncher** process and run scripts exit immediately after the RTView application is started, rather than continuing to run, thereby reducing the host system process count. However, note that: • The RTView application output and error messages will not appear in the command/ shell window from which it was launched. • Ctrl-c cannot be used to terminate the application. **Note:** This option is only recognized on the command line and is not read from, or saved to, any RTView options (**.ini**) files. Example: **-bg**	

-cachelast	Store only the last (most recent) values in the cache for each unique data attachment. By default, the Historian stores all records in the cache each time the **-cachesize** or **-cachetime** limit is reached. This option allows the Historian to store less data than it receives, which can be useful in a configuration where the Historian receives data from the Data Server at a higher rate than necessary for historical storage. Example: **-cachelast**
-cachesize: (records)	Set the cache size. The Historian will cache the specified number of records and then commit them all to the database at one time. Example: **-cachesize:50**
-cachetime: (seconds)	Set the cache time. The Historian will cache records for the specified number of seconds and then commit them all to the database at one time. Example: **-cachetime:60**
-charlimit	Specify the maximum number of characters for a String column in a database table created by the Historian. The default is **50**. This option allows the Historian to generate tables with larger String columns, rather than manually configuring each table. For example, in the command line: **-charlimit:NNN** and in HISTORY.ini: `charlimit NNN` where **NNN** specifies the maximum number of characters (for example, **-charlimit:300**). The Historian creates a VARCHAR column with the specified limit (for example VARCHAR(300)) for all String columns. However, be aware that each database vendor has a different upper limit on the maximum VARCHAR length. If you specify a **charlimit** value that is larger than the database vendor limit, errors might occur when the Historian creates a database table. Consult your database documentation for details.

	To store very long strings, you can specify MAX as the **charlimit** value rather than a number. For example, on the command line: **-charlimit:MAX** and in HISTORY.ini: `charlimit MAX` When MAX is specified as the value for the **charlimit** option, the type of database column created by the Historian for text columns depends on the specific database, as follows: **Database**: Column Type Declaration for charlimit=MAX **HSQLDB**: VARCHAR **SQL Server**: VARCHAR(MAX) **Oracle**: VARCHAR2(4000) **DB2**: CLOB **MySQL**: TEXT Each of these column types allows for the storage of very long strings. Consult your database documentation for details.
-charlimitindex	Specifies the maximum character length for text (VARCHAR) columns that are also index columns in the corresponding cache. For example, if the maximum character length is 70 (-**charlimitindex**:70), the Historian adds a VARCHAR(70) column to the database table it creates and persists to the cache for each text column that is an index column in a cache. If the **charlimitindex** property is not specified all text columns, including index text columns, use the size specified by the **charlimit** property. If the **charlimit** property is not specified, all text columns are created as VARCHAR(50). In the following example all index text columns are created as VARCHAR(70) and all non-index text columns are created as VARCHAR(200): **run_historian -charlimit:200 -charlimitindex:70** Use the **charlimitindex** property in cases where the database limits the total width (in characters or bytes) that can be used in an index. Databases include DB2 and Oracle.
-compactiontimerinterval	Specifies the time interval, in seconds, for the compaction thread to sleep before checking for work. The default is 5 seconds. Example: **-compactiontimerinterval:5**

-compactionverbose:n		Specifies whether and how to output to the console. There are three options: **0** - No information is output to the console. **1** - Summary information is output to the console. **2** - Debug-level information is output to the console. Example: **-compactionverbose:1**
-daemon	Run the Historian as a daemon process. **Note:** This parameter must be the first command line argument given. Example: **run_historian -daemon**	
-dbname:(name)	Set the name of the history database. **Note:** This name must match the name of the history database configured in the Display Builder's **Application Options>** "SQL Tab". Example: **-dbname:RTVHISTORY**	
-exit_on_out_of_memory	Force the Historian to terminate when an OutOfMemoryException occurs. This option is intended for use in a HA deployment where a backup server is available, otherwise the process may continue to run in a crippled state and prevent the backup from taking over. **Note:** The Historian is not automatically restarted by this option and must be restarted manually. Example: **-exit_on_out_of_memory**	
filename	Add a data configuration (.rtv) file to the Historian. Example: **run_historian config.rtv**	
-index_history_tables	When this option is set to true, the Historian creates two indexes on each database table that it creates to persist cache history. One index uses the cache's timestamp column, the other index uses the cache's index columns plus its timestamp column. If the cache has no index columns defined, then only the first index is created. For large tables, this can improve the performance of SQL queries that RTView executes to retrieve cache history data from the database. Command Line Example: **-index_history_tables:true** HISTORY.ini File Example: `index_history_tables:true`	

-insertcolumnnames	Include column names in the database insert statements for user-defined tables. This is useful when the Historian is adding data to existing tables and the column order is not an exact match or the insert statement contains a subset of the existing table columns. **Note:** This option does not apply to the HISTORY and HISTORY_S tables. Example: **-insertcolumnnames**
-jmxport:(port number)	The port number to use to expose JMX methods to monitor and manage the Historian. There is no default port. If not specified, these JMX methods will not be accessible. See "Managing the Historian Using JMX" for more information. Example: **-jmxport:9996**
-local_backlog	If enabled, allows the Historian to use its local heap space to temporarily store a backlog of data that is subsequently stored in the database. If not enabled, the backlog of data is kept in the Data Server, where it might be discarded after a relatively brief period of time if the Historian is unable to process it. Use this property to help avoid loss of data during busy periods when the Historian receives data from a Data Server at a higher rate than it can be stored in the database. By default the local_backlog property is not enabled. The **local_backlog** property keeps a minimum of 50,000 rows of data. If the Oracle server JVM is used, the backlog grows as large as the available heap space allows.

	The following message appears in the Historian log each time the backlog increases by 1000 rows:
	`backlog: increased to N rows`
	`or decreases by 1000 rows:`
	`backlog: decreased to N rows`
	where **N** is the current number of rows in the backlog.
	If the incoming rate of data exceeds the rate at which data can be stored in the database for an extended period of time, the backlog may need to be trimmed in order to avoid an OutOfMemory exception in the Historian. If that occurs, the following message appears in the Historian log:
	`backlog: discarded N rows`
	If this message is seen frequently, it may be necessary to adjust the Data Server, Historian, and/or database configuration to either reduce the incoming data volume or increase database throughput.
	Example:
	run_historian -local_backlog
	Properties File Example:
	`sl.rtview.historian.local_backlog=true`

-log4j	Turns on Log4j logging for the RTView application. By default, RTView processes print log messages to the console. To obtain log files, redirect the RTView application output and error streams to a log file using Log4j.
	After starting Log4j, the first time-stamped row in the log file appears as follows:
	`2012-02-02 14:00:54,693 INFO – [rtview]`
	`Log4j is being used with`
	`sl.log4j.properties as the`
	`configuration file.`
	When Log4j is not in use, the first time-stamped row in the log file appears as follows:
	`2012-02-03 10:40:31.866 [rtview]`
	`Logging redirected for System.out and`
	`System.err. Log4j is not in use.`
	(Note the missing INFO column when Log4j is not in use.)
	For example:
	run_historian –log4j
	run_historian –log4j –log4jlevel:INFO –showlogcat

To run the Historian as a background process using the **-bg** command line argument, use the s**l-bg.log4j.properties** configuration file (which only outputs to a log file rather than to a console) and the **-daemon** argument.

-bg (background) example:

run_historian -daemon –bg –log4j –log4jprops:sl-bg.log4j.properties

Note: The logging method from previous versions of RTView does not use Log4j. This previous method of logging is enabled with **-logfile** and **–logdir** and is still supported. Do not use both the previous logging method and Log4j or you receive the following error message:

```
ERROR: log4j configuration ERROR -
com.sl.rtview.useLog4j is set to true
but -logfile redirection is in use.
Log4j will not be used.
```

-showlogcat - Turns on the **Category** column in the log file output. When not in use, the **Category** column is not shown in the log file. When not in use, the **Category** column is not shown in the log file.

For example:

-showlogcat

-log4jprops - Specify the .properties file to use to format the Log4j log file. By default, **sl.log4j.properties** is used. Use this to provide a different property file name. The .properties file is searched for inside a .jar/.war file, then searched for in the current directory, and lastly searched for in the **%RTV_HOME%/lib** directory. The filename can have a path preceding it. For example, **C:\mydir\my.log4j.properties**.

For example:

-log4jprops:mylogfile.properties

-log4jlevel - Specify the Log4j Level. **INFO** is used by default. Valid values are:

FATAL: Indicates a severe error that likely causes the application to abort.

ERROR: Indicates an event that might not cause the application to abort.

WARN: Indicates a potentially harmful event.

DEBUG: Indicates detailed informational about events for debugging the application.

INFO: Indicates informational messages about the progress of the application at coarse-grained level.

For example:

-log4jlevel:INFO

-logdir	Specify to prefix the log file name that is set in the **-logfile** option to the directory name in which the log file is stored. If the **-logfile** option is not specified, this option is ignored. **Note:** This option is only recognized on the command line and is not read from, or saved to, any RTView options (**.ini**) files. Example: **-logdir:ABCcompany**	
-logfile	Specify the redirection of output and error messages to a file. The RTView application output and error message streams are redirected to the specified file. The file is created if it does not exist. If the file does exist, its previous contents are cleared. **Note:** This option is only recognized on the command line and is not read from, or saved to, any RTView options (**.ini**) files. Example: **-logfile:Historian.log**	
nogui	Specifies to run the Historian as a daemon process. **Note:** This parameter must be the first command line argument given. Example: **run_historian -nogui**	
-noreset	Do not clear database tables before storing new data. Example: **-noreset**	This option does not apply to the Advanced Historian.

-persistCaches:true	Allows the Historian to receive rows of cache data to be stored from the cache data source (via the data server, if in use) after the processing and primary compaction have already been performed. The Historian also queries the cache data source (via the data server, if in use) for the caches that should be persisted, thus avoiding the need for the Historian to load the cached data files as Historian data configuration files. You can enable the this option by using one of the following options: • Specify the following command line option when selecting the Historian: **-persistCaches:true** • Add the following line to the **HISTORY.ini** file: **persistCaches true** **Note:** When this option is enabled, the Historian will continue to load any files that appear in the Data Configuration Files list or that are listed as **history_config** lines in the **HISTORY.ini** file, even if those files contain cache definitions. So, if you use **persistCaches** in an existing configuration, you should remove those files from the list of **Data Configuration Files** or from the **HISTORY.ini** file. Otherwise, data for caches defined in those files will get stored twice.
-persistInitTimeRange	Specify a time range for the historian to get cache history data from the Data Server. Normally, when the cache persistence feature is enabled in the historian, the historian begins collecting cache history data starting at the time when each Data Server connects. This is adequate in most situations. The **persistInitTimeRange** option specifies a time range, in seconds, back from the current time for the historian to get cache history data from the Data Server. This is useful if the historian is started after the Data Server. In this case the Data Server collects cache history that has not been sent to the historian. For example, if the Data Server was started an hour before the historian, you could start the historian so that it requests an hour of cache history from the Data Server. For example: **run_historian -persistCaches:true -persistInitTimeRange:3600**

-processName	Specify to identify applications running as background processes. This option tags a unique identifier onto RTView server instances, enabling you to differentiate between multiple instances of those RTView applications. This option allows you to stop a particular instance without eliminating the other instances. If no process name is specified, the RTView application name is used as the process name. For example, **run_builder -processName:XX** adds the following JVM option to the Java call: **-DPROCESS_NAME=XX** Where XX is the value you specified for the **-processName** argument. **Note:** Values with spaces cannot be used for this option on Unix. Example: **-processName:XX**	
-rebuildtables	On startup, delete all existing Historian tables (including HISTORY and HISTORY_S) and recreate them. Example: **run historian -rebuildtables** It is also possible identify specific tables to rebuild (i.e. delete and recreate). **Note:** Table names are case-sensitive and may not contain spaces. Example: **-rebuildtables:Table1,Table2,Table3** **Note:** User-defined tables will be rebuilt as data is received for each table.	Do not use **-rebuildtables** in conjunction with the **-smoothCompaction** option because older data will be lost when the Historian tables are rebuilt.
-retention	Specify (in minutes) the length of time records can exist before they are deleted. Example: **-retention:5**	In the Advanced Historian, **compactionRules** specify how long records can exist before they are compacted or deleted. Go to **Caches> Introduction> Historian Properties**.

-retentionChunkSize		This option allows you to delete excess accumulated history rows in "chunks" of time (oldest rows deleted first) rather than deleting the data all at one time. Valid values are: **y** = YEAR **M** = MONTH **w** = WEEK **d** = DAY **h** = HOUR **m** = MINUTE **s** = SECOND Values can be as short as 1m (minute) or as long as 99y (years). For example: **-retentionChunkSize:2d** In the example above, history rows marked for deletion are broken down into 2 day "chunks" and deleted one after the other until all the rows in the "chunks" have been deleted.
-retentionMax	Specify (in minutes) the maximum length of time that may pass before retention processing is performed. **Note:** If the -retention time set is shorter than the specified -retentionMax, then the record's retention span is used. Example: **-retentionMax:60**	Specify (in minutes) the maximum length of time that may pass before retention processing is performed -- this setting only applies to the compaction rule with the longest retention amount. For tables that do not have **compactionRules** set, this option defaults to basic Historian behavior. **Note:** If the span of the compaction rule with the longest retention amount is shorter than the specified **-retentionMax**, then the compaction rule span is used. Example:- **-retentionMax:60**

-smoothcompaction	Based on specified **compactionRules**, perform compaction on old data that currently exists in your database from prior executions of the Historian. Do not use the **-rebuildtables** option in conjunction with **-smoothCompaction** because older data will be lost when the Historian tables are rebuilt. **table1,table2** - Specifies which tables to apply smooth compaction, where **table1** and **table2** are the table names. Any number of valid tables can be specified. Example: **-smoothcompaction:table3,table4** **Note:** This option is not recommended if the Historian has a heavy load of data to process or the Historian database is being used by other applications.
-smoothCompactionRecent	Specify how far back in time the smoothing process is applied. The format is: **-smoothCompactionRecent:NN** Where **NN** is '**1d**' or '**1w**'. Example: **-smoothCompactionRecent:1w**
-smoothingonly	When **-smoothcompaction** is also specified, enables compaction to be performed while no data updates are available. Example: **-smoothingonly**

-stringColMaxLen	Specify a limit on the length of a string from a cache table column that is stored in the database. If a string is longer than the specified limit, it is truncated to the limit before being stored in the database table. This can avoid SQL exceptions encountered when the length of a string exceeds the capacity of the column's data type (for example, 4000 characters in an VARCHAR2 column in Oracle). This can be specified in **HISTORY.ini** as follows: **stringColMaxLen 3500** It can also be specified on the command line or a properties file. By default the property has no value (no limit is enforced). Example: **-stringColMaxLen:3500**
-sub: (substring:subvalue)	Add a substitution string/value pair. Multiple substitution pairs can be specified on the command line. **Note:** Substitution strings cannot contain the following: : | . tab space , ; = < > ' " & / \ { } [ ] ( ) If your substitution value contains single quotes, you must escape them using a /. Example: **-sub:$data:myData** **-sub:$filter:Plant=/'SanFrancisco/'**
-tablename: (tablename)	Specify the table name to use when archiving scalar data. The table is created if it does not exist. If this option is not specified, the table name HISTORY is used by default. If **__none** is specified, then no table is created for storing scalar data. (If the Historian is used only to persist RTView cache data, then no scalar data is stored anyway). Example: **-tablename:MY_TABLE** Example: **-tablename:__none**
-timestamp: (type)	Specify the type of timestamp. There are three options: **none** - No **TIMESTAMP** column is stored. **sql** - A single **TIMESTAMP** column is stored using a standard SQL TIMESTAMP data type. **str** - Two TIMESTAMP columns are stored with each record as strings. Example: **-timestamp:sql**

-timezone	Set the default timezone for interpreting and displaying dates. Include a Java timezone ID or a custom ID, such as "GMT-8:00". Unrecognized IDs will be treated as GMT. See "Timezone ID Values" for more information. If you run the RTView Builder with a valid timezone parameter and then save Application Options, the timezone information will be persisted. To prevent the persisted timezone value from being used, pass "none" as the timezone ID. Example: **-timezone:US/Eastern** **-timezone:none**
-u(milliseconds)	Set update rate in milliseconds. Default is **2000**. Example: **-u5000** - updates every 5 seconds
-verbose	Set the Show Data in Console flag to true so that a line is printed to the console for each record that is stored in the database. Example: **-verbose**
-version	Prints the RTView application version information and exits immediately. Example: **run_historian -version**

Options Enabled with Alerts

In addition to the General Options, the following command line arguments are enabled with the Alert data source. See "Alerts" and "Audit Alert Action" for more information.

Name	Description
-actionauditdb:(database)	Specifies name of a database connection, as defined on the SQL tab, in which to store "Audit Alert Action" information. Example: **-actionauditdb:ALERTBD**
-actionaudittable:(table)	Specifies name of the table in the Alert Action Audit Database in which to store the "Audit Alert Action" information. Example: **-actionaudittable:ACTION_AUDIT_TABLE**
-alertcleartime: (number of seconds)	Specifies the rate, in seconds, to remove cleared alerts. Example: **-alertcleartime:3**

-alertds:alertdef:(filename)	Adds an alert definition file. Cannot specify substitutions. To specify substitutions, use the "Application Options - Alerts" tab. Example: **-alertds:alertdef:myalerts.rtv**
-alertds:enabled:(true or false)	Enables/disables all alerts in the active alert definition files. Example: **-alertds:enabled:false**
-alertds:history:(size of table)	Sets the number of rows that are stored in the "AlertTable". Example: **-alertds:history:1000**
-alertinitdelay:(number of seconds)	Specifies the duration, in seconds, to wait after startup to begin executing alerts. Example: **-alertinitdelay:5**
-enableactionaudit:(true or false)	If true and configured, alert actions will be stored to the specified database table. Example: **-enableactionaudit:true**
-exitOnPersistInitFailed	Specifies what occurs when alert persistence is enabled but cannot be initialized due to a database problem or configuration issue. When **false** (the default setting), RTView initializes the alerts with persistence disabled. This is consistent with the behavior in releases previous to RTView 6.6.0 and Enterprise Monitor 2.2.0. When **true**, RTView exits after the persistence initialization has failed without initializing the alerts. Example: **-exitOnPersistInitFailed:true**
-ignorelutforcount:(true or false)	If true, the **AlertTable Count** column increments for an alert when new data is received even if the Last Update Time has not changed. This can cause invalid Counts for alerts attached to caches. If false or not specified, the **AlertTable Count** column increments for an alert only if the Last Update Time has also updated. This is the default behavior. Example: **-ignorelutforcount:true**
-lutupdatesnewdata:(true or false)	Enables\disables updates to the "AlertTable" when New Data Only is selected and to the "Alert Persistence" database when the only columns that contain changes for that row are Last Update Time and Count. By default, the **Last Update Time** and **Count** columns are not tracked by the **Row Update Time** column. To track the updates of the two columns in the **Row Update Time** column, use the **-lutupdatesnewdata** command line option. See "Attach to Alert Data" and "Viewing Alerts" for more information. Example: **-lutupdatesnewdata:true**

-multipleindexdelim:(string)	For alerts with multiple index columns, create a unique alert index by concatenating all of the index column values. Value can be any string, except the following: • comma (,) • semi-colon (;), or • empty string. Default is tilde (~). Example: **-multipleindexdelim:~**
-persistInitDelayTime: (number of seconds)	Specify the amount of time, in seconds, to delay a backup Data Server from reading the alert persistence database during a failover. The default is 5 seconds. Increase the amount of time if the persistence database is slow or if you expect a large number of alerts to change on each update period. Otherwise, there might not be enough time for the failing Data Server to write all the alerts to the database before the backup server reads them. **Note:** Even with high availability configurations, there are cases in which some alerts might not be persisted. For example: • The persistence database fails. In this case, alerts cannot be written to, or read from, the database. If a failover occurs while the database is down, the backup server will not be able to read persisted alerts from the database. This will also happen if persistence is configured to use a Persistence Data Server to access the database, and the Persistence Data Server is down during failover. • The Alert Server is terminated in a non-orderly shutdown. Alerts are written out once per update period and once during orderly termination. If there is a non-orderly shutdown, some alerts might not be written to the database. In cases where alerts are not persisted, the new primary Data Server generates new alerts if the data is still in an alert state. The new primary Data Server might also re-use ID's that were used by the failed Data Server. Example: **-persistInitDelayTime:10**

Command Line Options: Data Server

The following command line arguments are enabled when you run the "Data Server" from a Windows Command Prompt or UNIX terminal window. Options specified using command line arguments override values saved in initialization (e.g. **OPTIONS.ini**) files.

For command line options for your data source, refer to the "RTView Data Sources" section of this documentation.

Note: If a command line argument contains a space or a semicolon, then the entire argument must be enclosed in quotes (e.g.: **"-sub:$data:my Data"**).

Name	Description
-bg	Set the RTView application to run as a background process. When this option is specified, the **GmsLauncher** process and run scripts exit immediately after the RTView application is started, rather than continuing to run, thereby reducing the host system process count. However, note that: • The RTView application output and error messages will not appear in the command/shell window from which it was launched. • Ctrl-c cannot be used to terminate the application. **Note:** This option is only recognized on the command line and is not read from, or saved to, any RTView options (**.ini**) files. Example: **-bg**
-client_blacklist	Specifies the client access list to the Data Server. There are three access list options: **client_blacklist**, **client_graylist**, and **client_whitelist**. Clients listed on the Blacklist are denied access to the Data Server. By default, the client access lists are empty which means that any client can connect to the Data Server. **Note:** The client access list options can be specified as Data Server command line arguments, as options in the **DATASERVER.ini** file, or as entries in a properties file. Specify a client hostname or IP address which may contain * characters as wildcards, or a range of IPv4 addresses in the format of n.n.n.n-n.n.n.n, where each n is a number between 0 and 255. For details, see "Security". Example: **run_dataserver -client_blacklist:DMZ**
-client_graylist	Specifies the client access list to the Data Server. There are three access list options: **client_blacklist**, **client_graylist**, and **client_whitelist**. Clients listed on the Graylist are permitted access to the Data Server if they provide a trusted SSL certificate. By default, the client access lists are empty which means that any client can connect to the Data Server. **Note:** Graylisting should be used only when necessary since it involves certificate management, delays from SSL handshaking, and overhead from data encryption. **Note:** The client access list options can be specified as Data Server command line arguments, as options in the **DATASERVER.ini** file, or as entries in a properties file. Specify a client hostname or IP address which may contain * characters as wildcards, or a range of IPv4 addresses in the format of n.n.n.n-n.n.n.n, where each n is a number between 0 and 255. For details, see "Security". Example: **run_dataserver -client_graylist:192.168.1.***

-client_whitelist	Specifies the client access list to the Data Server. There are three access list options: **client_blacklist**, **client_graylist**, and **client_whitelist**. Clients listed on the Whitelist are permitted access to the Data Server, no SSL certificate is required. If the Whitelist is empty, then all clients that are not blacklisted are accepted. However, if the Whitelist has at least one entry all clients are rejected that have no match in the Whitelist or the Graylist. By default, the client access lists are empty which means that any client can connect to the Data Server. **Note:** The client access list options can be specified as Data Server command line arguments, as options in the **DATASERVER.ini** file, or as entries in a properties file. Specify a client hostname or IP address which may contain * characters as wildcards, or a range of IPv4 addresses in the format of n.n.n.n-n.n.n.n, where each n is a number between 0 and 255. For details, see "Security". Example: **run_dataserver -client_whitelist:192.168.1.* -client_whitelist:localhost**
-client_write_timeout:(seconds)	Prevent a deadlock condition that can occur in abnormal situations. If the server attempts to push data to a client that is connected via socket and the corresponding write operation on the client's socket stalls for more than the specified number seconds, the server will close the client's socket. Default is **0**, which means the write to a client socket will never timeout. This timeout avoids the potential for deadlock with other threads in the server that may be trying to access the same data table that was locked while being pushed to the client. The write to the client's socket may stall because of networking problems, or a deadlock or other error in the client that prevents it from reading its incoming data. Example: **run_dataserver -client_write_timeout:(60)**
-daemon	Run the Data Server as a daemon process. Example: **run_dataserver -daemon**
-dataout:(path)	Set the path of the XML output to a directory other than the local directory. You can specify both an absolute path or a relative path. Double quotes are required if the path contains spaces. Example: **-dataout:"c:\rtview files\rtvdata.xml"**
-exit_on_out_of_memory	Force the Data Server to terminate when an OutOfMemoryException occurs. This option is intended for use in a HA deployment where a backup server is available, otherwise the process may continue to run in a crippled state and prevent the backup from taking over. For example, if the Data Server experiences an OutOfMemoryException it may still remain connected to its clients, preventing failover, but may stop pushing any data to the clients. **Note:** The Data Server is not automatically restarted by this option and must be restarted manually. Example: **-exit_on_out_of_memory**
filename	Add a data configuration (**.rtv**) file to the Data Server. Example: **run_dataserver config.rtv**

-group_initial_wait:(number of seconds)	This argument is used for the backup server in a high availability (HA) server group. For details about HA server groups and configuring this in the Data Server GUI, see **Active / Standby with auto-reset mode** in the "High Availability" section. For an example of how to configure a server group on the command line, see "Server Group Configuration Example". Use this argument to lengthen the wait time for the backup server to connect to the group's primary server at startup. This prevents a failure when the primary and backup server are both started at nearly the same time and the backup server attempts to connect to the primary server before it is connection ready. Specify the amount of time (in seconds) that the backup server waits at startup for a connection to the primary. The default is zero (0). This argument is only used for the backup server. A value of 30 (seconds) is typically used. Example: **-group_initial_wait:30**
-group_member:(hostname:port)	This argument is used, in conjunction with the **-group_priority** argument, to designate the primary and backup servers for a high availability (HA)server group. For details about HA server groups and configuring this in the Data Server GUI, see **Active / Standby with auto-reset mode** in the "High Availability" section. Specify the hostname and port number of the other Data Server in this group. That is, when launching the primary server in the group, use the **-group_member** option to specify the hostname and port number of the backup server in the group. Conversely, when launching the backup server in the group, use the **-group_member** option to specify the hostname and port number of the primary server in the group. If a hostname is specified without a port number, port 3278 is assumed. Do not specify a URL for the rtvdata servlet. Specifying the **-group_member** argument mitigates the need to specify the **-standby:warm** option. The server remains in standby mode until the primary server is determined. If the **-group_member** argument is specified and **-group_priority** is not, the **-group_priority** value defaults to a value of **1** (the lowest priority). Example: **-group_member:otherhost:3278** **Server Group Configuration Example** The following command line examples illustrate how to use all the server group command line options (**-group_member**, **-group_priority**, **-group_initial_wait** and **-group_timeout arguments**) to run Data Servers in a server group. In this example, one Data Server runs on host A and is the primary, another runs on host B and is the backup. Both servers use port **5555**. # command line for the primary server, executed on host A **run_dataserver -daemon -port:5555 -group_priority:2 -group_member:B:5555** # command line for the backup server, executed on host B **run_dataserver -daemon -port:5555 -group_priority:1 -group_member:A:5555 -group_initial_wait:30 -group_timeout:20**

-group_ping_timeout:(number of seconds)	This argument is used for the backup server in a high availability (HA) server group. For details about HA server groups and configuring this in the Data Server GUI, see **Active / Standby with auto-reset mode** in the "High Availability" section. For an example of how to configure a server group on the command line, see "Server Group Configuration Example". The primary Data Server sends a ping message to the backup server every 5 seconds. The **group_ping_timeout** option determines the amount of time (in seconds) the backup server waits for a ping from the primary server before it assumes the primary server failed and takes over as primary. The default value is 30 seconds and 30 is also the minimum non-zero value. A value of 0 (zero) disables the ping timeout feature entirely. This timeout argument is intended to detect rare cases where the primary Data Server is running and connected to the backup server and the primary server becomes unresponsive. If the primary Data Server terminates or the host on which it is running shuts down, the backup server detects this immediately regardless of the value of the **group_ping_timeout** or **group_timeout** properties, and takes over as the primary. Example: **-group_ping_timeout:30**
-group_priority:(integer)	This argument is used, in conjunction with the **-group_member** argument, to designate the primary and backup servers for a high availability (HA) server group. For details about HA server groups and configuring this in the Data Server GUI, see **Active / Standby with auto-reset mode** in the "High Availability" section. For an example of how to configure a server group on the command line, see "Server Group Configuration Example". Specify the priority of the Data Server, where a value of 1 is the lowest priority. A larger value assigns a higher priority. The running server in the pair with the highest priority becomes the primary server. For example, let us say we have two redundant Data Servers, one running on host A and another running on host B. The server on host A is normally the primary server and the server on host B is the backup server. In this case, we set the server's **group_priority** value for host A to 2 and the server's **group_priority** value for host B to 1. The server on host A is designated as the primary server. The server on host B is designated as the backup server. The **-group_priorit**y defaults to a value of 1 (the lowest priority) if the **group_member** property is specified and the **-group_priority** is not specified. Example: **-group_priority:2**

-group_standby_mode

This argument is used, in conjunction with the **-group_member** argument, to configure the behavior of backup Data Servers in a redundant high availability (HA) server group. For details about HA server groups and configuring this in the Data Server GUI, see **Active / Standby with auto-reset mode** in the "High Availability" section. For an example of how to configure a server group on the command line, see "Server Group Configuration Example".

There are two modes for this argument:

passive: The backup server does not activate the RTView global, cache, or alert data sources at startup. In this mode, the backup server only activates those data sources if it leaves standby mode because the primary server is unavailable. When the primary server is available again, the backup server returns to passive standby mode by deactivating the alert, cache, and global data sources.

Note: Since the passive standby mode activates several data sources during the failover process, cache data can be missed (which the active standby mode avoids).

active: The backup server activates the global and cache data sources at startup and begins collecting and storing data in caches as configured in the global and cache definition files. However, the backup server does not activate the alert data source at startup. In this mode, the backup server only activates the alert data source when if it leaves standby mode because the primary server is unavailable. When the primary server is available again, the backup server returns to active standby mode by deactivating the alert data source, and leaving the global and cache data sources activated. Since the active standby mode activates only one data source, the alert data source, during the failover process, cache data is less likely to be missed.

There is some cost associated with active standby mode because both the primary and backup server collect and store data in caches. For example, if a cache definition file defines SQL queries to collect data, in active standby mode both servers perform those SQL queries. Whereas in passive standby mode only the primary server performs the queries.

Note: The alert data source is inactive in the backup server in both active and passive standby mode, to avoid generation of duplicate alerts by the primary and the backup servers.

If no value is specified the default value is passive.

The **group_standby_mode** property can be also be specified as follows, where X is either passive or active:

`DATASERVER.ini: group_standby_mode X`

In a properties file:
sl.rtview.dataserver.group_standby_mode=X

Example:

run_dataserver -group_standby_mode:active

-group_timeout:(number of seconds)	This argument is used for the backup server in a high availability (HA) server group. For details about HA server groups and configuring this in the Data Server GUI, see **Active / Standby with auto-reset mode** in the "High Availability" section. For an example of how to configure a server group on the command line, see "Server Group Configuration Example". Specify the amount of time (in seconds) for the backup server to wait for a read and write operation on the socket connection between the primary and backup server. The default is 5 seconds. In most cases, if the primary server terminates or the host on which it is running is shut down, the backup server detects this immediately. This timeout argument is intended to detect failures where the connection remains open but stalls, or the server is unresponsive. Example: **-group_timeout:10**
-jmxport:(port number)	The port number to use to expose JMX methods to monitor and manage the Data Server. There is no default port. If not specified, these JMX methods will not be accessible. See "Managing the Data Server Using JMX" Example: **-jmxport:9997**
kill_dataserver	Stop the Data Server. A value of 0 on success and a value of 1 on failure. By default, the Data Server running on port 9020 is stopped. If you have NOT specified the jmxport property for the Data Server in the appropriate properties file you must specify it using the command line option **-jmxport:xxxx** (where **xxxx** is the port number) with **run_dataserver** in order to run **kill_dataserver**. For example: **kill_dataserver -port:9995** shuts down the Data Server on the local host which was started with the **-jmxport** property set to 9995. **Note:** When the default port **9020** is NOT used, the port must be specified for both the **run_dataserver** and **kill_dataserver** commands.

Values:

-host - The name of the host. The default value is localhost. This value is overridden when **-url** is specified after it.

Example:

-host:localhost

-port - The port number for the Data Server. The default value is 9020. This value is overridden when **-url** is specified after it.

Example:

-port:9020

-silent - Specifies not to print out the progress of low level operations. **-silent** and **-verbose** are mutually exclusive.

Example:

-silent

-user - The user name for accessing the JMX Mbean with authentication. If you specify **-user**, also specify -password.

Example:

-user:fred

-password - The password for accessing the JMX Mbean with authentication. If you specify **-password**, also specify **-user**.

Example:

-password:secret

-url - The URL for accessing the (remote) JMX Mbean. If you specify **-url**, it must not contain spaces. URL strings are always used internally. Specifying **-url** overrides the use of **–host** and **-port**. The substitutions are similar to the following example.

Example:

-url:service:jmx:rmi:///jndi/rmi://localhost:9995/jmxrmi

-verbose - Specifies to print out the progress of low level operations. -silent and -verbose are mutually exclusive.

Example:

kill_dataserver -port:9995

-log4j

Turns on Log4j logging for the RTView application. By default, RTView processes (Builder, Viewer, Data Server, Display Server, or Historian) print log messages to the console. To obtain log files, you redirect the RTView application output and error streams to a log file using Log4j.

After executing this command, the first time-stamped row in the log file appears as follows:

```
2012-02-02 14:00:54,693 INFO – [rtview] Log4j is being used with sl.log4j.properties as the configuration file.
```

When Log4j is not in use, the first time-stamped row in the log file appears as follows:

```
2012-02-03 10:40:31.866 [rtview] Logging redirected for System.out and System.err. Log4j is not in use.
```

(Note the missing INFO column when Log4j is not in use.)

For example:

run_dataserver –log4j

run_dataserver –log4j –log4jlevel:INFO –showlogcat

To run an RTView application as a background process using the -bg command line argument, use the sl-bg.log4j.properties configuration file (which only outputs to a log file rather than to a console).

-bg (background) example:

run_dataserver –bg –log4j –log4jprops:sl-bg.log4j.properties

Note: The logging method from previous versions of RTView does not use Log4j. This previous method of logging is enabled with **-logfile** and **–logdir** and is still supported. Do not use both the previous logging method and Log4j or you receive the following error message:

```
ERROR: log4j configuration ERROR - com.sl.rtview.useLog4j is set to true but -logfile redirection is in use. Log4j will not be used.
```

-showlogcat - Turns on the **Category** column in the log file output. When not in use, the **Category** column is not shown in the log file.

For example:

-showlogcat

	-log4jprops - Specify the .properties file to use to format the Log4j log file. By default, sl.log4j.properties is used. Use this to provide a different property file name. The .properties file is searched for inside a .jar/.war file, then searched for in the current directory, and lastly searched for in the %RTV_HOME%/lib directory. The filename can have a path preceding it. For example, C:\mydir\my.log4j.properties. For example: **-log4jprops:mylogfile.properties** **-log4jlevel** - Specify the Log4j Level. INFO is used by default. Valid values are: **FATAL**: Indicates a severe error that likely causes the application to abort. **ERROR**: Indicates an event that might not cause the application to abort. **WARN**: Indicates a potentially harmful event. **DEBUG**: Indicates detailed informational about events for debugging the application. **INFO**: Indicates informational messages about the progress of the application at coarse-grained level. For example: **-log4jlevel:INFO**
-logdir	Specify to prefix the log file name that is set in the **-logfile** option to the directory name in which the log file is stored. If the **-logfile** option is not specified, this option is ignored. **Note:** This option is only recognized on the command line and is not read from, or saved to, any RTView options (**.ini**) files. Example: **-logdir:ABCcompany**
-logfile:(filename)	Specify the redirection of output and error messages to a file. The RTView application output and error message streams are redirected to the specified file. The file is created if it does not exist. By default, if the file does exist, its previous contents are cleared. If the name of the log file contains the string DDDD (four upper case D characters), the string is replaced with the current local date and time using the format yyMMdd_HHmmss. For example, if we execute the following command on Sep 27 2012 at 3:55:43 PM: **run_dataserver -logfile:DataServer_DDDD.log** a log file named DataServer_120927_155543.log is produced. In most cases, this is a unique filename so that the previous log file, if any, remains unchanged. Over time, a large number of log files can accumulate so it is advisable to periodically purge the old files. On Linux, the logrotate utility can be used to automate this. **Note:** The **-logfile** option is only recognized on the command line and is not read from, or saved to, any RTView options (.ini) files. Example: **run_dataserver -logfile:DataServer.log**
-logappend	Appends new log file output to the previous file content. That is, if the dataserver.log file already exists, output from the new log process is added to the file, preserving pre-existing content. The file size can grow quite large so it is advisable to periodically rotate the file. On Linux, the logrotate utility can be used to automate this. For example: **run_dataserver -logfile:DataServer.log -logappend**

-passclientlogin	Pass RTView login information into all data sources that have the Use Client Credentials option enabled. **Note:** Some data sources do not support this feature. For information on Application Options for your data source, refer to the Data Sources section of this documentation. Example: **-passclientlogin**
-port:(port number)	Specify port when Data Server is set to output data via socket. Default is **3278**. Example: **run_dataserver -socket -port:8723**
-processName	Specify to identify applications running as background processes. This option tags a unique identifier onto RTView server instances, enabling you to differentiate between multiple instances of those RTView applications. This option allows you to stop a particular instance without eliminating the other instances. If no process name is specified, the RTView application name is used as the process name. For example, **run_builder-processName:XX** adds the following JVM option to the Java call: **-DPROCESS_NAME=XX** Where **XX** is the value you specified for the **-processName** argument. **Note:** Values with spaces cannot be used for this option on Unix. Example: **-processName:XX**

-resizemode:(mode)	Globally controls object layout when a display window is resized. It is also possible to set a specific Resize Mode for each particular display (**.rtv**) file using the "Background Properties" dialog. In the Display Builder, the selected Resize Mode is only applied to drill down windows. The main window of the Display Builder is always in Crop mode. All three resize modes support zooming the display (right-click -> zoom). In both Scale and Layout modes if the window is resized while the display is zoomed, then the resize will further zoom the display. Values: **crop** - When the window is resized, the display stays the same size. If the window is bigger than the display, empty space will show around the display. If the window is smaller than the display, scrollbars will be added. The window is not forced to maintain its aspect ratio. This is the default for the Thin Client. **scale** - When the window is resized, the display and all of the objects in it are scaled to fit the available space. The window is forced to maintain its aspect ratio. This is the default for the Display Builder, and Display Viewer Application. **layout** - When the window is resized, the display is resized to fit the available space. The objects in the display are positioned according to their anchor and dock properties. The window is not forced to maintain its aspect ratio. Objects that are not docked or anchored will move relative to their offset from the top left corner of the display. For example, if the object is centered on the display, the object will move 50% of the resize amount. If the object is centered at 3/4 of the display, it will move 75% of the resize amount. Example: **-resizemode:layout**
-sendalldata	Send all data over the socket regardless of whether or not it has been updated. Example: **-sendalldata**
-socket	Set the Data Server to output data via socket. Example: **run_dataserver -socket**
-standby:warm	Run a backup Data Server without the overhead of maintaining the Alert and Cache data sources. The following actions will be delayed until the backup server has become the primary: • Loading definition files (i.e. Global, Alert, Cache) • Preloading display files specified in initialization (.ini) files or on the command line **Note:** Although the **-standby:warm** option reduces overhead because data sources do not provide data until a failover, it is important to note that Alert and Cache data definitions will not start collecting data until the first client connects. Therefore, any previous alert state or cached data from the primary server will not be available to the backup. Example: **run_dataserver -standby:warm**

-sub:(substring:subvalue)	Add a substitution string/value pair. Multiple substitution pairs can be specified on the command line. **Note:** Substitution strings cannot contain the following: : | . tab space , ; = ? > ' " ? / \ { } [ ] ( ) If your substitution value contains single quotes, you must escape them using a /. Example: **-sub:$data:myData** **-sub:$filter:Plant=/'SanFrancisco/'**
-timezone	Set the default timezone for interpreting and displaying dates. Include a Java timezone ID or a custom ID, such as "GMT-8:00". Unrecognized IDs will be treated as GMT. See "Timezone ID Values" for more information. If you run the RTView Builder with a valid timezone parameter and then save Application Options, the timezone information will be persisted. To prevent the persisted timezone value from being used, pass "none" as the timezone ID. Example: **-timezone:US/Eastern** **-timezone:none**
-u(milliseconds)	Set update rate in milliseconds. Default is **2000**. Example: **-u5000 (updates every 5 seconds)**
-verbose	Specifies to print out the progress of low level operations. **-silent** and **-verbose** are mutually exclusive. Example: **-verbose**
-version	Prints the RTView application version information and exits immediately. Example: **run_dataserver -version**

Options Enabled with Alerts

In addition to the General Options, the following command line arguments are enabled with the Alert data source. See "Alerts" and "Audit Alert Action" for more information.

Name	Description
-actionauditdb:(database)	Specifies name of a database connection, as defined on the SQL tab, in which to store "Audit Alert Action" information. Example: **-actionauditdb:ALERTBD**
-actionaudittable:(table)	Specifies name of the table in the Alert Action Audit Database in which to store the "Audit Alert Action" information. Example: **-actionaudittable:ACTION_AUDIT_TABLE**

-alertcleartime:(number of seconds)	Specifies the rate, in seconds, to remove cleared alerts. Example: **-alertcleartime:3**
-alertds:alertdef: (filename)	Adds an alert definition file. Cannot specify substitutions. To specify substitutions, use the "Application Options - Alerts" tab. Example: **-alertds:alertdef:myalerts.rtv**
-alertds:enabled:(true or false)	Enables/disables all alerts in the active alert definition files. Example: **-alertds:enabled:false**
-alertds:history:(size of table)	Sets the number of rows that are stored in the "AlertTable". Example: **-alertds:history:1000**
-alertinitdelay:(number of seconds)	Specifies the duration, in seconds, to wait after startup to begin executing alerts. Example: **-alertinitdelay:5**
-cleansettingstable:(true or false)	If true, delete entries from the Alert Settings Table for alert names that are not defined in RTView. **Note:** This is done at startup after alert configuration files are processed and all of the alerts are loaded. Example: **-cleansettingstable:true**
-enableactionaudit:(true or false)	If true and configured, alert actions will be stored to the specified database table. Example: **-enableactionaudit:true**
-exitOnPersistInitFailed	Specifies what occurs when alert persistence is enabled but cannot be initialized due to a database problem or configuration issue. When **false** (the default setting), RTView initializes the alerts with persistence disabled. This is consistent with the behavior in releases previous to RTView 6.6.0 and Enterprise Monitor 2.2.0. When **true**, RTView exits after the persistence initialization has failed without initializing the alerts. Example: **-exitOnPersistInitFailed:true**
-ignorelutforcount:(true or false)	If true, the **AlertTable Count** column increments for an alert when new data is received even if the **Last Update Time** has not changed. This can cause invalid Counts for alerts attached to caches. If false or not specified, the **AlertTable Count** column increments for an alert only if the **Last Update Time** has also updated. This is the default behavior. Example: **-ignorelutforcount:true**

-lutupdatesnewdata:(true or false)	Enables\disables updates to the "AlertTable" when **New Data Only** is selected and to the "Alert Persistence" database when the only columns that contain changes for that row are **Last Update Time** and **Count**. By default, the **Last Update Time** and **Count** columns are not tracked by the **Row Update Time** column. To track the updates of the two columns in the Row Update Time column, use the **-lutupdatesnewdata** command line option. See "Attach to Alert Data" and "Viewing Alerts" for more information. Example: **-lutupdatesnewdata:true**
-multipleindexdelim: (string)	For alerts with multiple index columns, create a unique alert index by concatenating all of the index column values. Value can be any string, except the following: • comma (,) • semi-colon (;), or • empty string. Default is tilde (~). Example: **-multipleindexdelim:~**
-persistInitDelayTime: (number of seconds)	Specify the amount of time, in seconds, to delay a backup Data Server from reading the alert persistence database during a failover. The default is 5 seconds. Increase the amount of time if the persistence database is slow or if you expect a large number of alerts to change on each update period. Otherwise, there might not be enough time for the failing Data Server to write all the alerts to the database before the backup server reads them. **Note:** Even with high availability configurations, there are cases in which some alerts might not be persisted. For example: • The persistence database fails. In this case, alerts cannot be written to, or read from, the database. If a failover occurs while the database is down, the backup server will not be able to read persisted alerts from the database. This will also happen if persistence is configured to use a Persistence Data Server to access the database, and the Persistence Data Server is down during failover. • The Alert Server is terminated in a non-orderly shutdown. Alerts are written out once per update period and once during orderly termination. If there is a non-orderly shutdown, some alerts might not be written to the database. In cases where alerts are not persisted, the new primary Data Server generates new alerts if the data is still in an alert state. The new primary Data Server might also re-use ID's that were used by the failed Data Server. Example: **-persistInitDelayTime:10**
-purgepersistedalerts	Clears all alerts for the alert engine from the "Alert Persistence" database on startup and no persisted alerts will be loaded. **Note:** If you are persisting alerts for more than one alert engine in the same database, alerts for other alert engines will not be removed.
-printssawarnings:(true or false)	If false, the Self Service Alerts warnings about extra unmapped thresholds will be suppressed. **Note:** This option only applies to Self Service Alerts. Example: **-printssawarnings:false**

APPENDIX C Third Party Notice Requirements

Please refer to the **LICENSES_thirdparty.txt** file from your product installation.

APPENDIX D Timezone ID Values

Timezone ID Values

America/Nome
America/Yakutat
Etc/GMT+9
Pacific/Gambier
SystemV/YST9
SystemV/YST9YDT
US/Alaska
America/Dawson
America/Ensenada
America/Los_Angeles
America/Tijuana
America/Vancouver
America/Whitehorse
Canada/Pacific
Canada/Yukon
Etc/GMT+8
Mexico/BajaNorte
PST
PST8PDT
Pacific/Pitcairn
SystemV/PST8
SystemV/PST8PDT
US/Pacific
US/Pacific-New
America/Boise
America/Cambridge_Bay
America/Chihuahua
America/Dawson_Creek
America/Denver
America/Edmonton
America/Hermosillo
America/Inuvik
America/Mazatlan
America/Phoenix
America/Shiprock
America/Yellowknife
Canada/Mountain
Etc/GMT+7
MST
MST7MDT
Mexico/BajaSur
Navajo
PNT
SystemV/MST7
SystemV/MST7MDT
US/Arizona
US/Mountain
America/Belize
America/Cancun
America/Chicago
America/Costa_Rica
America/El_Salvador
America/Guatemala
America/Managua
America/Menominee
America/Merida
America/Mexico_City
America/Monterrey
America/North_Dakota/Center
America/Rainy_River
America/Rankin_Inlet
America/Regina
America/Swift_Current
America/Tegucigalpa
America/Winnipeg
CST
CST6CDT
Canada/Central
Canada/East-Saskatchewan
Canada/Saskatchewan
Chile/EasterIsland
Etc/GMT+6
Mexico/General
Pacific/Easter
Pacific/Galapagos
SystemV/CST6
SystemV/CST6CDT
US/Central

Timezone ID Values

America/Bogota
America/Cayman
America/Detroit
America/Eirunepe
America/Fort_Wayne
America/Grand_Turk
America/Guayaquil
America/Havana
America/Indiana/Indianapolis
America/Indiana/Knox
America/Indiana/Marengo
America/Indiana/Vevay
America/Indianapolis
America/Iqaluit
America/Jamaica
America/Kentucky/Louisville
America/Kentucky/Monticello
America/Knox_IN
America/Lima
America/Louisville
America/Montreal
America/Nassau
America/New_York
America/Nipigon
America/Panama
America/Pangnirtung
America/Port-au-Prince
America/Porto_Acre
America/Rio_Branco
America/Thunder_Bay
Brazil/Acre
Canada/Eastern
Cuba
EST
EST5EDT
Etc/GMT+5
IET
Jamaica
SystemV/EST5
SystemV/EST5EDT
US/East-Indiana
US/Eastern
US/Indiana-Starke
US/Michigan
America/Anguilla
America/Antigua
America/Aruba
America/Asuncion
America/Barbados
America/Boa_Vista
America/Caracas
America/Cuiaba
America/Curacao
America/Dominica
America/Glace_Bay
America/Goose_Bay
America/Grenada
America/Guadeloupe
America/Guyana
America/Halifax
America/La_Paz
America/Manaus
America/Martinique
America/Montserrat
America/Port_of_Spain
America/Porto_Velho
America/Puerto_Rico
America/Santiago
America/Santo_Domingo
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Thule
America/Tortola
America/Virgin
Antarctica/Palmer
Atlantic/Bermuda
Atlantic/Stanley
Brazil/West
Canada/Atlantic
Chile/Continental
Etc/GMT+4 PRT
SystemV/AST4
SystemV/AST4ADT
America/St_Johns
CNT
Canada/Newfoundland
AGT
America/Araguaina
America/Belem
America/Buenos_Aires
America/Catamarca
America/Cayenne
America/Cordoba
America/Fortaleza
America/Godthab
America/Jujuy
America/Maceio
America/Mendoza
America/Miquelon
America/Montevideo
America/Paramaribo
America/Recife
America/Rosario

Timezone ID Values

America/Sao_Paulo
Brazil/East
Atlantic/South_Georgia
America/Scoresbysund
Etc/GMT+1
Africa/Bamako
Africa/Casablanca
Africa/El_Aaiun
Africa/Monrovia
Africa/Sao_Tome
Atlantic/Canary
Atlantic/Reykjavik
Etc/GMT
Etc/GMT0
Etc/UTC
Europe/Belfast
Europe/London
GMT
Iceland
UTC
Zulu
Africa/Brazzaville
Africa/Kinshasa
Africa/Luanda
Africa/Niamey
Africa/Windhoek
CET
Europe/Amsterdam
Europe/Berlin
Europe/Budapest
Europe/Ljubljana
Europe/Malta
Europe/Paris
Europe/San_Marino
Europe/Stockholm
Antarctica/Rothera
Etc/GMT+3
Brazil/DeNoronha
Atlantic/Azores
Africa/Abidjan
Africa/Banjul
Africa/Conakry
Africa/Freetown
Africa/Nouakchott
Africa/Timbuktu
Atlantic/Faeroe
Atlantic/St_Helena
Etc/GMT+0
Etc/Greenwich
Etc/Universal
Europe/Dublin
GB
GMT0
Portugal
Universal
Africa/Algiers
Africa/Ceuta
Africa/Lagos
Africa/Malabo
Africa/Porto-Novo
Arctic/Longyearbyen
ECT
Europe/Andorra
Europe/Bratislava
Europe/Copenhagen
Europe/Luxembourg
Europe/Monaco
Europe/Prague
Europe/Sarajevo
Europe/Tirane
BET
America/Noronha
Etc/GMT+2
Atlantic/Cape_Verde
Africa/Accra
Africa/Bissau
Africa/Dakar
Africa/Lome
Africa/Ouagadougou
America/Danmarkshavn
Atlantic/Madeira
Eire
Etc/GMT-0
Etc/UCT
Etc/Zulu
Europe/Lisbon
GB-Eire
Greenwich
UCT
WET
Africa/Bangui
Africa/Douala
Africa/Libreville
Africa/Ndjamena
Africa/Tunis
Atlantic/Jan_Mayen
Etc/GMT-1
Europe/Belgrade
Europe/Brussels
Europe/Gibraltar
Europe/Madrid
Europe/Oslo
Europe/Rome
Europe/Skopje
Europe/Vaduz

Europe/Vatican
Europe/Vienna
Europe/Warsaw
Europe/Zagreb
Europe/Zurich
MET
Poland
ART
Africa/Blantyre
Africa/Bujumbura
Africa/Cairo
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Kigali
Africa/Lubumbashi
Africa/Lusaka
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Tripoli
Asia/Amman
Asia/Beirut
Asia/Damascus
Asia/Gaza
Asia/Istanbul
Asia/Jerusalem
Asia/Nicosia
Asia/Tel_Aviv
CAT
EET
Egypt
Etc/GMT-2
Europe/Athens
Europe/Bucharest
Europe/Chisinau
Europe/Helsinki
Europe/Istanbul
Europe/Kaliningrad
Europe/Kiev
Europe/Minsk
Europe/Nicosia
Europe/Riga
Europe/Simferopol
Europe/Sofia
Europe/Tallinn
Europe/Tiraspol
Europe/Uzhgorod
Europe/Vilnius
Europe/Zaporozhye
Israel
Libya
Turkey
Africa/Addis_Ababa
Africa/Asmera
Africa/Dar_es_Salaam
Africa/Djibouti
Africa/Kampala Africa/Khartoum
Africa/Mogadishu
Africa/Nairobi
Antarctica/Syowa
Asia/Aden
Asia/Baghdad
Asia/Bahrain
Asia/Kuwait
Asia/Qatar
Asia/Riyadh
EAT
Etc/GMT-3
Europe/Moscow
Indian/Antananarivo
Indian/Comoro
Indian/Mayotte
W-SU
Asia/Riyadh87
Asia/Riyadh88
Asia/Riyadh89
Mideast/Riyadh87
Mideast/Riyadh88
Mideast/Riyadh89
Asia/Tehran
Iran
Asia/Aqtau
Asia/Baku
Asia/Dubai
Asia/Muscat
Asia/Oral
Asia/Tbilisi
Asia/Yerevan
Etc/GMT-4
Europe/Samara
Indian/Mahe
Indian/Mauritius
Indian/Reunion
NET
Asia/Kabul
Asia/Aqtobe
Asia/Ashgabat
Asia/Ashkhabad
Asia/Bishkek
Asia/Dushanbe
Asia/Karachi
Asia/Samarkand
Asia/Tashkent
Asia/Yekaterinburg

Timezone ID Values

Etc/GMT-5
Indian/Kerguelen
Indian/Maldives
PLT
Asia/Calcutta
IST
Asia/Katmandu
Antarctica/Mawson
Antarctica/Vostok
Asia/Almaty
Asia/Colombo
Asia/Dacca
Asia/Dhaka
Asia/Novosibirsk
Asia/Omsk
Asia/Qyzylorda
Asia/Thimbu
Asia/Thimphu
BST
Etc/GMT-6
Indian/Chagos
Asia/Rangoon
Indian/Cocos
Antarctica/Davis
Asia/Bangkok
Asia/Hovd
Asia/Jakarta
Asia/Krasnoyarsk
Asia/Phnom_Penh
Asia/Pontianak
Asia/Saigon
Asia/Vientiane
Etc/GMT-7
Indian/Christmas
VST
Antarctica/Casey
Asia/Brunei
Asia/Chongqing
Asia/Chungking
Asia/Harbin
Asia/Hong_Kong
Asia/Irkutsk
Asia/Kashgar
Asia/Kuala_Lumpur
Asia/Kuching
Asia/Macao
Asia/Macau
Asia/Makassar
Asia/Manila
Asia/Shanghai
Asia/Singapore
Asia/Taipei
Asia/Ujung_Pandang
Asia/Ulaanbaatar
Asia/Ulan_Bator
Asia/Urumqi
Australia/Perth
Australia/West
CTT
Etc/GMT-8
Hongkong
PRC
Singapore
Asia/Choibalsan
Asia/Dili
Asia/Jayapura
Asia/Pyongyang
Asia/Seoul
Asia/Tokyo
Asia/Yakutsk
Etc/GMT-9
JST
Japan
Pacific/Palau
ROK
ACT
Australia/Adelaide
Australia/Broken_Hill
Australia/Darwin
Australia/North
Australia/South
Australia/Yancowinna
AET
Antarctica/DumontDUrville
Asia/Sakhalin
Asia/Vladivostok
Australia/ACT
Australia/Brisbane
Australia/Canberra
Australia/Hobart
Australia/Lindeman
Australia/Melbourne
Australia/NSW
Australia/Queensland
Australia/Sydney
Australia/Tasmania
Australia/Victoria
Etc/GMT-10
Pacific/Guam
Pacific/Port_Moresby
Pacific/Saipan
Pacific/Truk
Pacific/Yap
Australia/LHI
Australia/Lord_Howe

Asia/Magadan	Etc/GMT-11	Pacific/Efate
Pacific/Guadalcanal	Pacific/Kosrae	Pacific/Noumea
Pacific/Ponape	SST	Pacific/Norfolk
Antarctica/McMurdo	Antarctica/South_Pole	Asia/Anadyr
Asia/Kamchatka	Etc/GMT-12	Kwajalein
NST	NZ	Pacific/Auckland
Pacific/Fiji	Pacific/Funafuti	Pacific/Kwajalein
Pacific/Majuro	Pacific/Nauru	Pacific/Tarawa
Pacific/Wake	Pacific/Wallis	NZ-CHAT
Pacific/Chatham	Etc/GMT-13	Pacific/Enderbury
Pacific/Tongatapu	Etc/GMT-14	Pacific/Kiritimati

APPENDIX E Extending the List of Available Fonts

This section contains the following:

Adding Access to Additional Fonts

By default, RTView provides 12 standard fonts for use when creating RTView displays which, for most situations, are sufficient. However, there are certain circumstances where some additional fonts could be useful. For example, due to differences between available fonts on different platforms, visual issues might arise when displays are built on one platform and are run on a different platform. In such a situation, you can provide access to additional (extended) fonts that can be used in the Viewer, in the thin client, or in both.

Before discussing the steps required to add access to additional fonts, let's start with some basic definitions:

Term	Description
System Fonts	System or platform fonts are provided by the local operating system. A web browser can make use of the system fonts from the local (client) machine. The system fonts vary between different operating systems and between versions of the same operating system, which presents a challenge to web developers since a font available to a browser on one system (for example, Tahoma on Windows) might not be available to a browser running on other systems, although similar fonts (for example, Helvetica on Mac OS) might be available. There are three possible solutions (defined below): • Only use web-safe fonts • Use font families or font stacks • Use web (downloaded) fonts
Web-safe Fonts	The serif, sans-serif, and monospace fonts are known as "web-safe" or standard fonts because they are available in all browsers on all platforms, and they are also available in Java on all platforms.

Font-family	A font-family is a comma separated list of font names used in the CSS style applied to text elements on a web page. The browser attempts to use the fonts in the order they are listed until it finds a font that is available. The browser also applies this process on a per-character basis if the text element contains a character not defined in the specified font. The last font in the list should be a web-safe font. This technique is also known as a "font stack" or as "fallback fonts." For example: **font-family: Helvetica, Lucida Sans, Arial, sans-serif**
Web Fonts	A web page can identify a specific font to be downloaded from a web server rather than using a stack or using system fonts. This specific font downloaded from the web server is known as a "web font."
Font Files	A font file is a file that defines a font. For example, a common format of a font file is TrueType (.ttf). In a Java application, additional fonts can be used by providing TrueType font (.ttf) files. On a web page, web fonts can be downloaded as .ttf, .woff, and .woff2 files. Font files are licensed, and many are licensed by the operating system vendor (Microsoft, for example). However, there are a large number of fonts available under open source licenses (for example: Apache 2.0, and SIL). The web font example that is available in the **custom/fonts** directory uses several open source fonts published on *http://www.google.com/fonts*.

Default Fonts in RTView

RTView uses only the serif, sans-serif, and monospace web safe font types, each of which can be defined as plain, bold, italic, or bold-italic for a total of 12 available default font types. Each font type is assigned an index, which is used when configuring a font property (**lableTextFont**) on an object in the RTView Display Builder. The default indexes are:

1: sans-serif, plain

2: monospace, bold

3: monospace, plain

4: serif, plain

5: monospace, italic

6: serif, bold

7: sans-serif, bold

8: sans-serif, italic

9: serif, bold italic

10: serif, italic

11: sans-serif, bold italic

12: monospace, bold italic

Configuring Extended Fonts

As mentioned previously, extended fonts can be used in the Viewer, in the thin client, or in both. To use extended fonts in the Viewer, the font must be contained within a local .ttf or .otf file. Using extended fonts in the thin client requires the use of either the Font-family (stack) or the Web Font options. The font stack option provides better performance and requires no additional font files or licensing since it uses system fonts already available on the local system. However, since different physical fonts are used, a text element that uses the font

stack option might have a different appearance and size when viewed in a browser on, for example, Windows versus the Apple or Android operating systems. The Web Font option provides a consistent appearance and size in all browsers regardless of the local platform, but requires additional time to download the web fonts the first time the page is opened (fonts that contain non-Latin characters -- Cyrillic, Arabic, and especially CJK (Chinese/Japanese/Korean)-- can be large). Web Fonts also need to be properly licensed.

Note: Fonts used in a browser as a Web Font must be able to be downloaded as a .ttf, woff, .woff2, or .otf file. Internet Explorer might display a warning if a .ttf file is not embeddable (most Microsoft-provided .ttf files are not embeddable).

Extended Fonts Table

You can extend the number of available fonts by adding the desired fonts to a table, which should have one row per font and should contain the following columns:

Column Name	Description
Index (integer)	The font's index, which should be an integer value between 13 and 255. The examples provided in the **custom/fonts** directory (**custom/fonts/WinFontExample/fonts.xml**, for example) use index values of 100 and above. This column is required. **Note:** Using index values 1-12 (which are used by the default fonts in RTView) will override the default fonts, which is not recommended.
Name (string)	The name of the font as it should appear in the Builder property sheet and font selection list. This column is required.
Bold (boolean)	Creates a **bold** version of the font when set to **true**. This column is required.
Italic (boolean)	Creates an *italicized* version of the font when set to **true**. This column is required.
CSS Font Family (string)	This string is used by the thin client to assign the CSS font style to a client-side object that uses the corresponding font. If the thin client uses a system font, then this string should specify the font family or "font stack" that should be used, which should include a comma-separated list of fallback font names ending with a web-safe font. For example: **Trebuchet MS, Helvetica, sans-serif** Alternatively, if you want the thin client to download a web font, then you can enter a single name in this column. For example: **Roboto** This column is optional depending the deployment of interest and fonts being used.

URL (string)	Enter the URL that the thin client should use to download a font file. For example (if a subdirectory named **fonts** exists beneath the rtvdisplay servlet's **webapp** directory): **fonts/Roboto-Regular.ttf** This URL can also specify an address on a different web server (for example: **http://someurl.com/fonts/xyz.woff**) provided that the server supports cross-domain requests (CORS). **Note:** If the thin client is using a system font, then this column should be left blank. This column is optional depending the deployment of interest and fonts being used.
TTF Path (string)	The local path to the .ttf file containing the fonts, which will be used by the RTView builder/viewer and display server. This column is optional depending the deployment of interest and fonts being used.

Enabling the Extended Fonts

To enable the extended fonts, you must make the extended font table available to RTView by creating a global function named **RtvExtFontTable**. This global function must return your defined extended font table containing the required column names and types listed above. See the "Extended Fonts Examples" that are provided in the **custom/fonts** directory for an example of how to add/view extended fonts.

Extended Fonts Examples

The following examples provide a way to view how to add extended Windows and Web fonts.

Windows Fonts Example

1. In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"), go to the **custom/fonts/WinFontExample** directory and type:

   run_builder

   The Display Builder displays. You can click the **Functions** tab (or selto view details for the pre-defined **RtvExtFontTable** function.

2. Click the **Object Palette** tab, click the **Labels** tab, select a label, and place a label in the display.

   The selected label is added to the display and the **Object Properties** for that label are populated.

3. Click the **Label**>**labelTextFont** object property drop down list.

   The 12 standard fonts display in the list, as well as the newly added Windows fonts that are defined in the **fonts.xml** file.

Web Fonts Example

1. In an initialized terminal window (see "Initializing a Command Prompt or Terminal Window"), go to the **custom/fonts/WebFontExample** directory.

   run_builder

The Display Builder displays. You can click the **Functions** tab to view details for the pre-defined **RtvExtFontTable** function.

2. Click the **Object Palette** tab, click the **Labels** tab, select a label, and place a label in the display.

   The selected label is added to the display and the **Object Properties** for that label are populated.

3. Click the **Label**>**labelTextFont** object property drop down list.

   The 12 standard fonts display in the list, as well as the newly added Web fonts that are defined in the **fonts.xml** file.

Running the Thin Client

To view the new fonts in the thin client:

1. Create a display containing the extended fonts by running the builder in the **WebFontExample** or the **WinFontExample** directory.

   Note: This step is optional. There are two example displays, **labels.rtv** and **controls.rtv**, that exist in each directory that you can use instead of creating a new display.

2. In an initialized command window (see "Initializing a Command Prompt or Terminal Window") type:

   run_startup_demoserver

   Note: This step is only required if the tomcat demoserver is not already running.

3. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the directory in which you created your display (**custom/fonts/WebFontExample** or **WinFontExample**) and start the Display Server by typing:

   run_displayserver

4. Open a browser and navigate to **http://localhost:8068/rtvdisplay**.

5. Select the previously created display from the list of available **RTView Displays**.

   Your newly created display should open containing your objects with the extended fonts.

Running the Display Viewer

To view the new fonts in the viewer:

1. Create a display containing the extended fonts by running the builder in the **WinFontExample** directory or the **WebFontExample** directory.

   Note: This step is optional. There are two example displays, **labels.rtv** and **controls.rtv**, that exist in each directory that you can use instead of creating a new display.

2. In an initialized command window (see "Initializing a Command Prompt or Terminal Window"), go to the directory in which you created your display (**custom/fonts/WinFontExample** or **WebFontExample**) and start the Display Viewer by typing:

   run_viewer

3. Click **File** > **Open** and select the previously created display from the list of available RTView displays.

   Your newly created display should open containing your objects with the extended fonts.

Missing Characters in Extended Fonts

Web-safe fonts support all unicode characters, but other fonts might not (for example, open source fonts might not support CJK characters). If a text string contains a character that is not found in the specified font, then the following occurs:

- **Browser:** If a text element contains a character that is not found in the specified font, the browser will try to use the fonts listed in the Font-family until it finds a font that supports the character. Hence, the thin client will always display all of the characters in a text string. When using an extended font, however, non-Latin characters might be displayed using a fallback or system font.
- **Java Application**: If a text string contains a character that is not defined in the specified font, the character might appear as a "tofu" character (a box or a question mark), or it might not be drawn at all. So, the builder/viewer will display tofu or empty characters for non-Latin characters that are not found in an extended font. This is also the case for the thin client for text objects that are rendered in the display server and not in the browser.

Consistent Font Sizing in Windows and Linux

The sizing and spacing of the text on displays configured in Windows but deployed in the thin client via the display server running Linux can, at times, be significantly different. The **rtvfonts.jar** file, which contains open source files that are equivalent in sizing and spacing to the 12 default fonts used in RTView, can be used to address this discrepancy.

The open source fonts contained in **rtvfonts.jar** are:

- Arimo (sans-serif, which is equivalent to Arial)
- Cousine (monospace, which is equivalent to Courier)
- Tinos (serif, which is equivalent to Times New Roman)

Note: The open source fonts in rtvfonts.jar are licensed under the Apache 2.0 open source license (http://www.apache.or/licenses/LICENSE-2.0.html).

Each of the fonts is provided in regular (plain), bold, italic, and bold italic in **.ttf** format (for the display server) and **.woff** format (for the thin client) for a total of 24 available font files (12 default fonts and 12 open source fonts). Once configured, the displays running Linux are rendered using the open source fonts, thus mimicking the text sizing and spacing on the display as it was designed in Windows.

Note: The font files in this jar contain only the Latin character set. Chinese/Japanese/Korean characters will only appear as boxes in text strings that are rendered by the display server (for a label object with **webLabelFlag** = false). Characters in text strings rendered by the browser appear in a fallback font, in which case the font style and size might not match the Latin characters.

Configuring rtvfonts.jar

The following steps are required to configure the display server to use **rtvfonts.jar**:

1. Add **rtvfonts.jar** to the display server's classpath. For example:

   set RTV_USERPATH=$RTV_HOME/lib/rtvfonts.jar

2. Add **rtvfonts.xml**, which is contained in **rtvfonts.jar**, as a global file. For example, add the following line to the display server's **OPTIONS.ini** file:

   global rtv_fonts.xml

APPENDIX F Scheduling Functionality

The Scheduler functionality is supported by the SQL data source as well as by the alerts. For SQL queries, you can define the days, the time(s) of day, and the frequency that you want to run your user-defined SQL queries. For alerts, you can define the days and times of day to enable or disable specific alerts. See "SQL Scheduler" and "Alert Definition Files" for more information. To use the scheduler for SQL queries, you must define a "Schedule Table" and a "Rule Table". For alerts, you must define a "Schedule Table", a "Rule Table", and an "Alert Schedule Map Table". These tables can be created via a database, an XML file, or any other table that can be retrieved using an RTView data source, and must contain the following columns:

Rule Table

Column Name	Definition
Name (string)	The unique name for the rule. Rule names can only contain letters, digits, spaces, underscores, and/or dashes.
Day (string)	The list of days of the week (Mon, Wed, Fri, for example) or specific dates (dd-MMM-yyyy) that this rule is in effect.
Start Time (string)	The time of day that this rule goes into effect, in 24 hour format (hh:mm:ss). If left empty, then 00:00:00 is used.
End Time (string)	The time of day when this rule is no longer in effect, in 24 hour format (hh:mm:ss). If left empty, then 23:59:59 is used.
Exception (boolean)	This value is used differently for SQL queries and alerts: **SQL Queries**: If set to **false** then, when the rule is in effect, the query **is** run at the indicated interval. If set to **true** then, when the rule is in effect, the query is **NOT** run. **Alerts**: If set to **false** then, when the rule is in effect, the alert is enabled. If set to **true** then, when the rule is in effect, the alert is disabled.
Interval (string)	This value is not used for alerts. For SQL queries, this value is the time interval between queries when this rule is in effect (1m, 30m, 1h, for example).
Enable (boolean)	Set to **true** to enable the rule. Set to **false** if you do not want the rule to ever be in effect.
Time Zone (string)	The name of the time zone for the rule (US/Eastern). Leaving the column blank results in using the local time zone. Names that include daylight savings (PST or EDT, for example) should not be used.

Note: You can add additional formats for specifying dates in your rules using the **sl.rtview.scheduler.dateFormat** property in your **.properties** file. For example:

```
sl.rtview.scheduler.dateFormat=MMM dd, yyyy
```

Entering **-scheduler.dumpFormats:true** on the command line will print the accepted formats for days of the week, specific dates, and time zones.

Example Rule Table

Name	Exception	Days	Start Time	End Time	Interval	Time Zone	Enabled	Description
workdays_30m	false	Mon,Tue, Wed,Thu, Fri	09:00:00	17:00:00	30m		true	weekdays 9-5 @30 min
weekends_9_12_15	false	Sat,Sun	09:00:00	17:00:00	3h		true	weekends at 9, noon, 3
lunch_hour_ex	true	Mon,Tue, Wed,Thu, Fri	12:00:00	12:59:59			true	no queries during lunch
holidays_2016_ex	true	1-Jan-2016,4-Jul-2016,25-Dec-2016					true	no queries on these days
workdays30m_ central	false	Mon,Tue, Wed,Thu, Fri	09:00:00	17:00:00	30m	US/ Central	true	weekdays 9-5 @30 min (in US/ Central timezone)

Explanation of Rule Examples

workdays_30m
For SQL queries, runs a query every 30 minutes between 9am and 5pm Monday to Friday.
For alerts, enables the alert between 9am and 5pm Monday to Friday.

weekends_9_12_15
For SQL queries, runs a query on weekends at 9am, noon, and 3pm.
For alerts, enables the alert between 9am and 5pm on weekends.

lunch_hour_ex
For SQL queries, this exception rule prevents a query from being run between noon and 12:59:59 on weekdays.
For alerts, this exception rule disables alerts between noon and 12:59:59 on weekdays.

holidays_2016_ex
For SQL queries, this exception rule prevents a query from being run on 3 specific dates in 2016 (Christmas, 4th of July, and January 1).
For alerts, this exception rule disables alerts on 3 specific dates in 2016 (Christmas, 4th of July, and January 1).
Note that this rule does not specify a Start Time or End Time, so the rule is in effect for the entire day.

workdays30m_central
Same as the workdays_30m rule except that it uses the US/Central timezone when checking the time of day against the Start/End time.

Schedule Table

Column Name	Definition
Name (string)	The unique name of the schedule. Schedule names can only contain letters, digits, spaces, underscores, and/or dashes.
Rules (string)	The name(s) of the rule(s) that define the schedules. Multiple rule names must be separated by commas.
Enabled (boolean)	Set to **true** to enable the schedule. Set to **false** to disable all queries that use the schedule.
Description (string)	The description of the schedule.

Example of Schedule Table Using Rules Defined Above

Name	Rules	Enabled	Description
Weekday30 9to5	workdays_30m, lunch_hour_ex, holidays_ex	true	weekdays @30 minutes from 9-5, except lunch & holidays
AllWeek	workdays_30m, weekends_9_12_15	true	weekdays @30minutes from 9-5 & noon & 3pm, weekends @ 9 & noon & 3pm
test_timezone	workdays30m_central	true	weekdays 9-5, in central timezone

Alert Schedule Map Table

This table is only used when scheduling alerts. This table must contain the following columns:

Column Name	Definition
Alert Name (string)	The name of the alert to which the schedule will be applied. This name must match the name of a defined alert.
Alert Index (string)	The alert index to which the schedule will be applied. For scalar alerts, this must be set to **Default** since scalar alerts do not support indexes. For tabular alerts, enter an index to apply the schedule to that alert index only, or select **Default** to apply the schedule to all indexes. If the **Index Type** is **Default**, this value should also be **Default**.
Index Type (string)	The index type of the **Alert Index**. If the value is **All**, the **Alert Index** value is assumed to contain all indexes. If the value is **Default**, this schedule will be applied to all alert indexes on the specified alert that do not match another row in this table. For scalar alerts, this must be set to **Default**. For tabular alerts that do not define the **indexTypes** property, this must be set to **Default** or **All**.

Schedule (string)	The name of the schedule from the **RtvScheduleTable**.
Enabled (boolean)	If set to **true**, use this alert map row. If set to **false** then, then the alert map is not used.

Enabling Schedule, Rule and Alert Schedule Map Tables

To make the Schedule table and Rule table available in RTView:

- Create a global function named **RtvScheduleTable** using **Function Type=Reference** and attach the function's Table argument to the XML file, SQL query, or whatever RTView data attachment that retrieves the selected table.
- Create a global function named **RtvScheduleRuleTable** using **Function Type=Reference** and attach the function's Table argument to the RTView data attachment that retrieves the rule table.
- Create a global function named **RtvAlertScheduleMap** using **Function Type=Reference** and attach the function's Table argument to the RTView data attachment that retrieves the alert schedule map table.

See "Global Functions and Variables" for more information.