Informatica® RulePoint
6.1.2

User Guide

Informatica RulePoint User Guide
6.1.2
April 2015

Publication Date: 2018-07-20

Table of Contents

Preface

The *RulePoint User Guide* provides instructions on how to use RulePoint. This guide describes how to create and manage key RulePoint components, such as topics, connections, sources, responders, responses, and rules. This guide also provides tips and warnings that you need to be aware of while using RulePoint.

This guide is primarily intended for those who use RulePoint or administer the product. It is assumed that users are familiar with the use of Microsoft Internet Explorer or the Mozilla Firefox browser.

Informatica Resources

Informatica My Support Portal

As an Informatica customer, you can access the Informatica My Support Portal at http://mysupport.informatica.com.

The site contains product information, user group information, newsletters, access to the Informatica customer support case management system (ATLAS), the Informatica How-To Library, the Informatica Knowledge Base, Informatica Product Documentation, and access to the Informatica user community.

Informatica Documentation

The Informatica Documentation team makes every effort to create accurate, usable documentation. If you have questions, comments, or ideas about this documentation, contact the Informatica Documentation team through email at infa_documentation@informatica.com. We will use your feedback to improve our documentation. Let us know if we can contact you regarding your comments.

The Documentation team updates documentation as needed. To get the latest documentation for your product, navigate to Product Documentation from http://mysupport.informatica.com.

Informatica Product Availability Matrixes

Product Availability Matrixes (PAMs) indicate the versions of operating systems, databases, and other types of data sources and targets that a product release supports. You can access the PAMs on the Informatica My Support Portal at https://mysupport.informatica.com/community/my-support/product-availability-matrices.

Informatica Web Site

You can access the Informatica corporate web site at http://www.informatica.com. The site contains information about Informatica, its background, upcoming events, and sales offices. You will also find product

and partner information. The services area of the site includes important information about technical support, training and education, and implementation services.

Informatica How-To Library

As an Informatica customer, you can access the Informatica How-To Library at http://mysupport.informatica.com. The How-To Library is a collection of resources to help you learn more about Informatica products and features. It includes articles and interactive demonstrations that provide solutions to common problems, compare features and behaviors, and guide you through performing specific real-world tasks.

Informatica Knowledge Base

As an Informatica customer, you can access the Informatica Knowledge Base at http://mysupport.informatica.com. Use the Knowledge Base to search for documented solutions to known technical issues about Informatica products. You can also find answers to frequently asked questions, technical white papers, and technical tips. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team through email at KB_Feedback@informatica.com.

Informatica Support YouTube Channel

You can access the Informatica Support YouTube channel at http://www.youtube.com/user/INFASupport. The Informatica Support YouTube channel includes videos about solutions that guide you through performing specific tasks. If you have questions, comments, or ideas about the Informatica Support YouTube channel, contact the Support YouTube team through email at supportvideos@informatica.com or send a tweet to @INFASupport.

Informatica Marketplace

The Informatica Marketplace is a forum where developers and partners can share solutions that augment, extend, or enhance data integration implementations. By leveraging any of the hundreds of solutions available on the Marketplace, you can improve your productivity and speed up time to implementation on your projects. You can access Informatica Marketplace at http://www.informaticamarketplace.com.

Informatica Velocity

You can access Informatica Velocity at http://mysupport.informatica.com. Developed from the real-world experience of hundreds of data management projects, Informatica Velocity represents the collective knowledge of our consultants who have worked with organizations from around the world to plan, develop, deploy, and maintain successful data management solutions. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Global Customer Support

You can contact a Customer Support Center by telephone or through the Online Support.

Online Support requires a user name and password. You can request a user name and password at http://mysupport.informatica.com.

The telephone numbers for Informatica Global Customer Support are available from the Informatica web site at http://www.informatica.com/us/services-and-training/support-services/global-support-centers/.

CHAPTER 1

RulePoint

This chapter includes the following topics:

RulePoint Overview

RulePoint is a sophisticated Complex Event Processing (CEP) solution that plugs into the IT infrastructure and delivers real-time operational business intelligence to users within an enterprise.

RulePoint is unique among CEP products in that it is an application for end users, namely the business analysts looking for operational intelligence to maximize the efficiencies of their business processes, to determine hidden patterns in data, and determine real-time event correlation that can negatively impact their business.

RulePoint differs from data-mining software in that its analyses are conducted in real time based on changes in the state of information. For example, a product is sold, an order is submitted, or the current temperature reaches a defined level. These changes in state trigger events that are fed into the RulePoint system. The business user creates rules to analyze and correlate multiple events from multiple data sources to detect and notify the appropriate user of specific conditions.

The RulePoint Advantage

A key RulePoint advantage is that it enables you to glean through relevant operational intelligence data from multiple sources and act on them in real time. It is difficult for people to detect continuously changing data events, even more difficult to detect events that are contextually relevant to their needs, and nearly impossible to detect related past events or those that occur concurrently. As a result, most people are forced to focus on post-mortem analysis of lost opportunities and missed threats. The worst case scenario results in missed opportunities or major crises. RulePoint offers a clear advantage by enabling you to discover these operational changes as they happen and the opportunity to act on them immediately.

RulePoint is designed for the non-technical end user who needs to detect changes across dynamic information sources and correlate disparate data to spot patterns of interest. RulePoint also makes it easy

for these users to initiate automated response activities to provide real-time situational awareness across the enterprise.

RulePoint differs from data-mining software in that its analyses are conducted in real time based on changes in the state of information. For example, a product is sold, an order is submitted, or the current temperature reaches a defined level. These changes in state trigger events that are fed into the RulePoint system. The business user creates rules to analyze and correlate multiple events from multiple data sources to detect and notify the appropriate user of specific conditions.

How RulePoint Works

RulePoint uses services, which are configurable interfaces that link to another software system, to collect information from relevant streams of data flowing from different data sources. Each piece of information is referred to as an event. These events are published into RulePoint and grouped into categories familiar to and defined by users, which are referred to as topic. RulePoint then uses other services to coordinate responses to those events based on user-defined event processing rules, referred to simply as rules. A rule encapsulates the business logic of analyzing event data from multiple sources to detect specific events based on logical conditions, and then responds to the appropriate party with the proper information.

Most notably, RulePoint enables users to create rules themselves, ensuring that an organization's responsiveness to changing conditions is not hindered by the traditional software development cycle.

When RulePoint detects events that match the conditions specified in a rule, it executes the response specified in the rule. These responses can be simple, such as sending an email, instant message, or text message, or complex, such as updating a database, triggering a web service, initiating other processes across the enterprise, or creating new events used by other rules.

Key Benefits of RulePoint

A key RulePoint advantage is that it enables you to glean through relevant operational intelligence data from multiple sources and act on them in real time.

RulePoint delivers a complete set of Event Detection and Response capabilities including Complex Event Processing, real-time alerting, user-driven rule management, subscription services, and automated response triggering. Through this set of capabilities, RulePoint enables enterprises to achieve true operational intelligence across their existing information sources.

Individual and Complex Event Detection

- Monitor or listen in near-real time to data across diverse sources, including the following sources:
  - Data streams
  - Sensors
  - Communications systems
  - Message queues
  - Web services
  - RSS feeds

- Databases
- Flat files

- Cross reference, filter, and correlate events across disparate sources to detect complex events.
- Leverage third-party entity extraction, natural language processing, and categorization engines as a part of complex event detection processes.
- Define personalized rules and subscription criteria for event detection and response using wizards, advanced mode, or templates.

Real-Time Alerting and Response

- Provide users with context appropriate notifications and alerts which may be complemented by relevant data from other sources, through channels such as email, IM, and web browser.
- Respond to data events across disparate systems based on user-defined rules by initiating transactions, triggering existing workflow systems, and interacting with remote web services.
- Intelligently use server based events to launch desktop applications such as GIS mapping environments, analytic tools, and multimedia viewers.
- Display changing data from underlying systems to real-time web browser pages without requiring client download.
- Facilitate rapid response to notifications and alerts with return receipt tracking, time based escalation, and customized one click follow-up actions.

Management and Administration

- Deploy Event Detection and Response capabilities as an enterprise service accessible to diverse legacy systems, enterprise applications, and users.
- Connect to enterprise information sources such as message queues, databases, and web services using out-of-the box connectors and web based setup pages.
- Leverage accounting, role based security, logging, and process monitoring facilities to create robust enterprise class applications.
- Authenticate users through Lightweight Directory Access Protocol (LDAP), Microsoft Active Directory via LDAP and PKI for Public Key Infrastructure (X.509 client certificates).

Rule Processing

The business user creates rules to analyze and correlate multiple events from multiple data sources to detect and notify the appropriate user of specific conditions.

A rule consists of the following main elements: topic **>** condition **>** response.

You configure each of the three elements separately or within a rule itself. The following elements are either required for operation or can be optionally configured to provide additional functionality to standard processing:

- services
- analytics
- watchlists

- rule wizard
- rule templates
- user policy
- access control

You initially configure and associate the following elements with each other to begin rule processing:

1. **Define services**. RulePoint can receive data pushed from sources or it can pull data from sources on a scheduled basis. Supported services can include SQL, text files, web pages, and RSS feeds.
2. **Create topics**. Topics enable you to organize multiple event sources into logical topic such as *Inventory and Ordering*.
3. **Create responses**. Rules that you will create respond to an event by invoking a response that you created and then referenced in that rule.
4. **Create rules**. Rules apply processing logic to incoming events as they are associated with topic and enable you to identify specific events and respond to those events as they occur. Rules can have multiple conditions for evaluation such as same number of occurrences, same value, same location, and can trigger multiple responses such as sending an email to a particular person and writing to a database.

Note: You can also set up other elements, such as watchlists, to further support your rule processing.

RulePoint Process

Sending alerts and notifications depends on a defined set of triggers and flow of data.

Users write rules to respond to patterns within voluminous, complex streams of data. These patterns are nearly impossible for human beings to detect in real time, which has led to traditional requirements for reporting and historical trend analysis to identify past threats and opportunities. RulePoint delivers this kind of intelligence in real time, enabling employees within organizations to respond to important events as they are occurring.

For example, identification of misuse, fraud, and abuse is an ideal application for RulePoint. By tapping into necessary transaction processing systems, RulePoint can monitor account transactions, collect related information from other data sources and apply rules to identify different patterns of misuse as they are taking place. RulePoint rapidly can deliver targeted alerts and notifications across multiple channels, including Real-Time Alert Manager, instant messages, and email. Notifications across these channels enable the responsible individuals and groups to immediately begin an investigation.

Based on these user-defined rules, RulePoint also can respond using other back-end applications to perform more complex or automated actions. Examples of these actions include the following:

- Triggering a process in a third-party workflow engine.
- Creating a case or ticket in an external case management solution.
- Posting an event to an enterprise message bus so that other business services can consume that data.

CHAPTER 2

RulePoint Concepts

This chapter includes the following topics:

RulePoint Concepts Overview

RulePoint requires certain elements to be initially configured and associated with each other for rule processing to begin. As a prerequisite you create users, define users' access to event sources, define topics, and define responses.

Understanding RulePoint Entities

A rule consists of distinct sections, topics (a collection of events), conditions, and responses.

Topics

The topic of a rule describes the types, sources, and properties of events coming into the system, and how to pre-filter these events. Topics are logical groupings of events that enable users to better manage multiple event sources by funneling them into one grouping. You then can reference the topic(s) in your rules.

You create rules to look for events by referencing a topic that enables you to group similar event sources within RulePoint. You can write a rule to look at a single event or multiple events with a single property or multiple properties that relate to a single topic or multiple topics.

RulePoint supports pre-filtering in the topic section, which means that you can provide simple, initial filtering before defining the conditions of the rule. For example, if your business process is to look at three events within five minutes, you can write the topic section to wait for these three events before sending the results to the condition section for further processing. If three or more events do not occur during any five-minute window, RulePoint will not continue to process the rule.

For example, the following rules scan incoming events to identify one stock volume event and two stock share price events:

- Rule 1: If you identify one stock volume that is greater than 30,000 for the day, then add the stock to the stock watchlist.

- Rule 2: If you identify two stock share prices drop by more than 10% in a five minute window and the stock is on the stock watchlist, then notify the Fraud Team.

Conditions

The condition of a rule is where most of the event processing occurs. It is here where you can define the criteria for matching events on their names, values such as price, temperature, volume, amount, their relation to other events for example, if Name from Criminal List equals Name from Employee List, and other conditions.

You can create granular, specific, and complex condition statements that are flexible enough to meet your business requirements for event analysis.

For example, the following rules scan incoming events for a drop in any stock's daily volume and in any watchlisted stock's volume within five minutes:

- Rule 1: If you see one stock volume that is greater than 30,000 for the day, then add the stock to the Stock watchlist.
- Rule 2: If you see two stock share prices drop by more than 10% in a 5 minute window and the stock is on the Stock watchlist, then notify the Fraud Team.

Responses

The response is where you define how you want RulePoint to respond if your rule's event matches your rule condition(s). In addition to simple notification response, such as send an email or text message, you can configure a response to function like an action. For example, RulePoint supports a grouping called a "watchlist," a list that can be checked by a rule condition. A watchlist can be updated manually by a user, or it can be dynamically updated as a result of a matched condition firing a response. RulePoint supports sending responses to a single user or groups of users via email, instant message, RTAM user interface, and other notification methods.

For example, the following two rules execute two different responses, listed in bold text:

- Rule 1: If you see a stock volume that is greater than 30,000 for the day, then add the stock to the Stock watchlist.
- Rule 2: If you see any stock share price drop by more than 10% in a 5 minute window AND the stock is on the Stock watchlist, then notify the Fraud Team.

Additional System Entities

RulePoint has other system elements that are either required for operation or can be optionally configured to provide additional functionality to standard processing.

Events

Generally, the RulePoint administrator configures RulePoint services to connect to your data sources. The data sources then feed events into your topics, against which you would write a rule. Topics, which are collection of these events, apply properties to the incoming events so that RulePoint can process the data appropriately. You can create properties for structured event sources (name=Chris, stock=INFA, temp=82, RSS news feeds, SQL databases) as well as unstructured event sources (web pages, documents, third party connectors). Define topic properties for incoming events so you can easily create rules for existing event sources as your business requirements change.

Services

Services are components that interact with event data by providing access to external event sources, access to external response sources, and internal processing of data. Analytics are used for internal data processing. They provide additional math functions and expression processing outside of rule commands).

CHAPTER 3

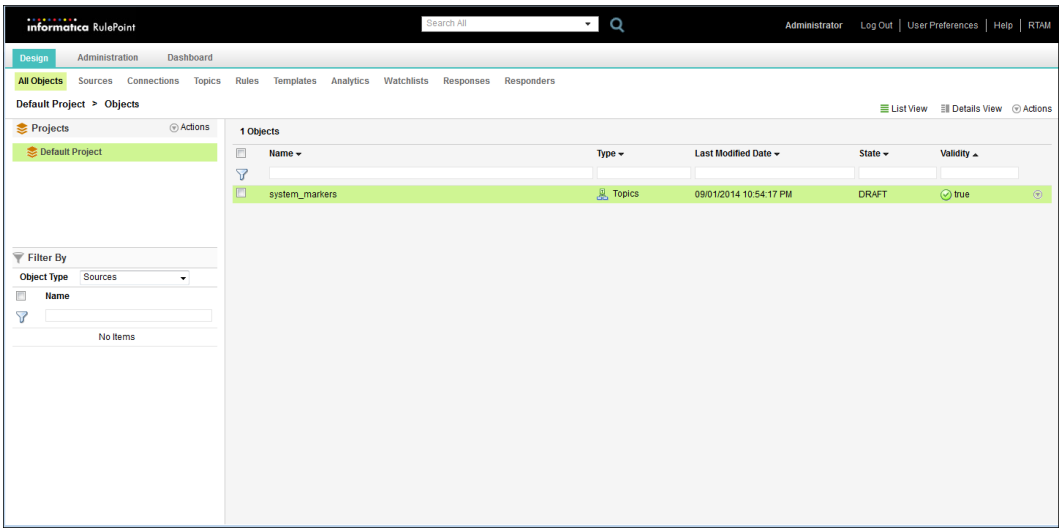
Using RulePoint

This chapter includes the following topics:

Using RulePoint Overview

You use the Informatica RulePoint user interface to administer RulePoint.

The following image shows the RulePoint user interface:



The RulePoint user interface has the following tabs:

- **Design**. Create, manage, and deploy CEP applications. The application is built within a RulePoint project. Each Project contains a series of RulePoint objects that are integrated together using the RulePoint programming model.

- **Administration**. Manage Users within RulePoint, import and export objects, and manage the complete topology of the RulePoint deployment.
- **Dashboard**. View runtime aspects of the entire topology, such as the CPU and memory statistics of each host, and activity data (number of activations, processing time, and so on) for all deployed objects. Also, start and stop deployed objects and trace the flow of events for rules.

The banner of the RulePoint user interface has the following items:

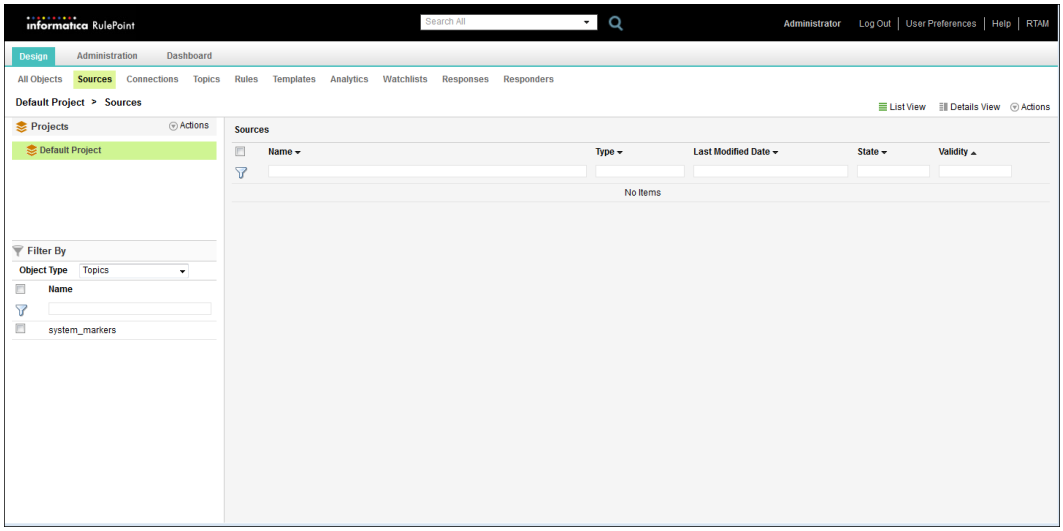
- **Log Out**. Log out of the RulePoint user interface.
- **User Preferences**. Manage your account. The contents of the user preferences dialog box that appears depends on how RulePoint is configured to authenticate users. If local authentication is used, you can edit your password, first name, and last name, and you can enable and disable the RulePoint welcome screen. If remote authentication is used, you can enable and disable the RulePoint welcome screen.
- **Help**. Access RulePoint help.
- **RTAM**. View the RTAM dashboard.
- **Search**. Find objects by name or perform an advanced search.

Design Tab Overview

On the **Design** tab, you create all the entities that you need. You create and manage projects and edit per-user permissions for the project. You also create objects for a project, view information about objects, and deploy objects to the run-time environment. After you have deployed objects, on the **Design** tab, you can undeploy or redeploy those objects, or you can reassign them from one controller to another. You can also define deployment policies for templates.

You create objects within the scope of a project. An object is local to a project and cannot be shared across projects.

The following image shows the **Design** tab:



Each view on the tab consolidates the tasks associated with a RulePoint object. The following views are available on the **Design** tab:

- **All Objects**. View and manage all objects of the project.
- **Sources**. View and manage sources.
- **Connections**. View and manage connections.
- **Topics**. View and manage topics.
- **Rules**. View and manage rules.
- **Templates**. View and manage templates.
- **Analytics**. View and manage analytics.
- **Watchlists**. View and manage watchlists.
- **Responses**. View and manage responses.
- **Responders**. View and manage responders.

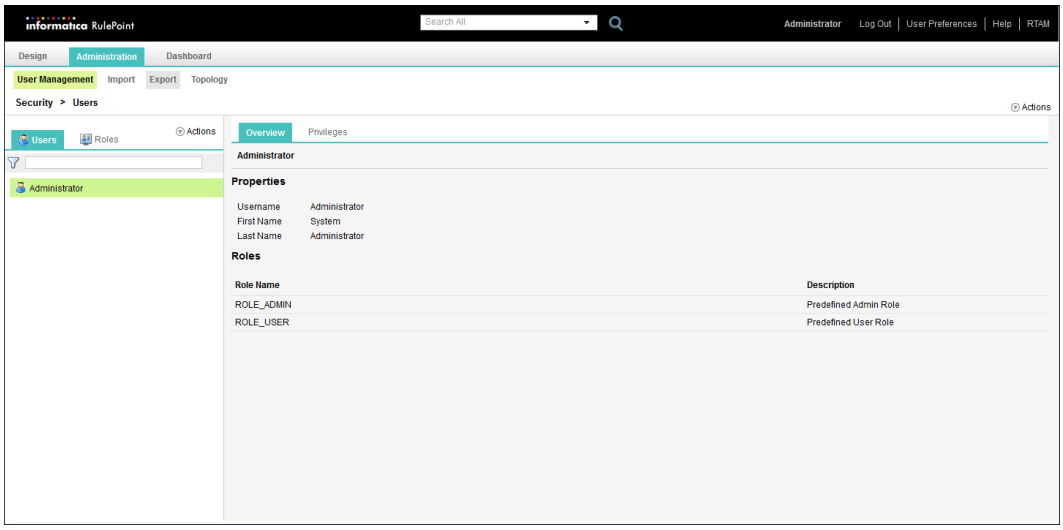
You can configure the appearance of these views. When you click on a project in the navigator, the **List View** displays the configured objects for that project. The **Details View** displays the configured objects. You can additionally select a specific object on the left pane and simultaneously view the details of that object on the right pane.

The navigator pane consists of the **Projects** and **Filter By** sections. You use the **Projects** section to view the projects that have been created and the **Filter By** section to filter objects by type.

Administration Tab Overview

On the **Administration** tab, you perform administrative tasks. You manage users and roles and configure the topology. You also export objects and import objects to and from XML files, respectively.

The following image shows the **Administration** tab:



Each view on the tab consolidates the tasks associated with an administrative function. Following are the views on the tab:

- **User Management**. View and manage roles and users.
- **Export**. Export selected or all objects to an XML file, and to a location of your choice.
- **Import**. Upload XML files to the RulePoint server and then import objects from the XML file.
- **Topology**. Manage the RulePoint topology.

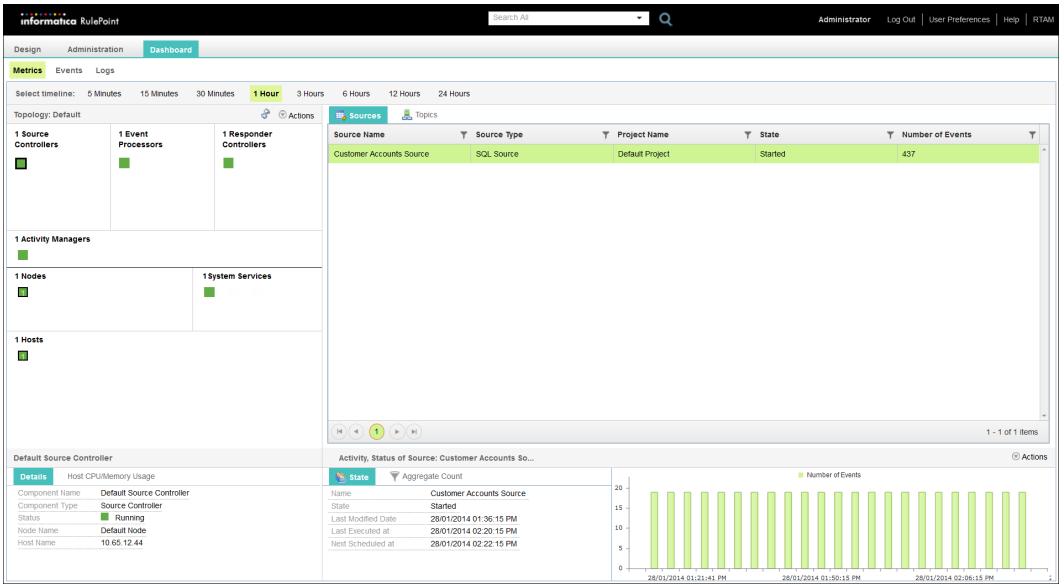
The navigator pane consists of the **Users** and **Roles** tabs. You use the **Users** tab to view and create user accounts and the **Roles** tab to view and create roles. For more information about creating users and roles, see the *Informatica RulePoint Administrator Guide*.

Dashboard Tab Overview

On the **Dashboard** tab, you can review the run-time topology and the performance of deployed objects. From a topology perspective, you can review high availability configurations, the assignment of application services to various hosts, and the objects that are deployed on any given application service. You can also obtain real-time information about CPU and memory usage statistics from various hosts. From an object perspective, you can view the state of a deployed object, start and stop objects, and trace the flow of events for rules, along with event values, over a specified timeline.

The **Dashboard** tab consists of the **Metrics**, **Events**, and **Logs** view.

The following image shows the **Metrics** view on the **Dashboard** tab for a single node topology:



In the top-left pane of the **Metrics** view, the dashboard displays the configured application services, system services, activity manager, nodes, and hosts in the topology. When you select an application service, the dashboard highlights the host and node in which the application service is configured. The contents pane displays the objects that are deployed on the selected host.

The dashboard also consists of timeline-based views that show run-time statistics for a specific period of time in the immediate past. You specify the time period for which you want to view information by selecting a view. Each timeline-based view shows you the following information:

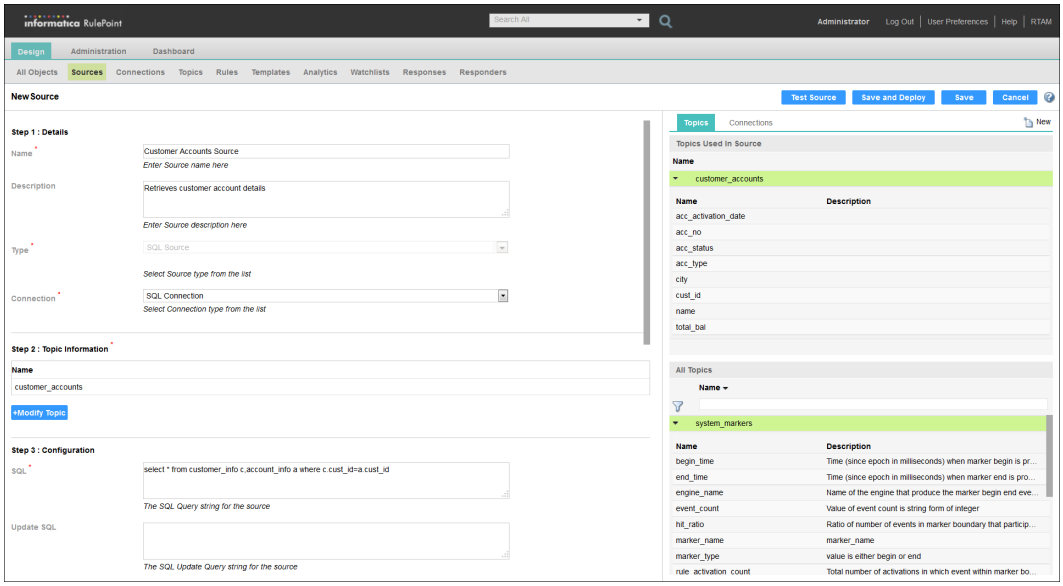
- CPU and memory usage on the selected host, in the form of a graph.
- Name, type, state, and activation count for all deployed objects.
- Average throughput, maximum latency, and average latency of the deployed sources and responders.
- Number of events in topics, number of evaluations and number of activations for rules, and number of alerts for responses.
- Functions that you can administer, for example, stop events from publishing, resume processing, enable rule tracing, purge events, and troubleshoot events.

You can use the **Events** view to view all the generated events or search for specific events. The **Logs** view displays the run-time logs and you can check for errors.

Workspace Overview

In RulePoint, the dialog boxes that you use for creating sources, rules, and responders include a workspace in which you can view and create dependent objects. Workspaces eliminate several steps in user interface navigation and simplify the task of associating secondary objects with the primary object that you are creating. For example, a source requires a topic and a connection to function. Therefore, the **New Source** dialog box includes a workspace that contains tabs for viewing and creating topics and connections. When creating a source, you can use the workspace to view existing topics, connections, and their properties before you select one of those objects for the source. Also, if a secondary object with the desired properties does not exist, you can create the object and then associate it with the primary object without having to leave the dialog box. You cannot, however, edit, delete, or copy secondary objects from the workspace of a primary object.

The following image shows the workspace in the **New Source** dialog box:



Similar views are available when creating rules and responders.

CHAPTER 4

RulePoint Objects

This chapter includes the following topics:

RulePoint Objects Overview

The RulePoint objects are of two types, primary objects and supporting objects. Primary objects include source, responder, and rule. All other objects that supports the primary objects, such as topics, connections, and responses are supporting objects.

The primary objects drive the execution of RulePoint run-time components. The sources and responders get deployed in service controllers, and rules get deployed in engines.

The following table shows the primary objects and their supporting objects:

Primary Objects	Supporting Objects
Source	- Topic - Connection
Responder	- Response - Connection
Rule	- Topic - Analytic - Watchlist - Response - Optional. Template

State of RulePoint Objects

State denotes the current status of the RulePoint objects.

The RulePoint objects can have one of the following states:

Draft

When you create and save an object and do not deploy it, the object is in the Draft state. An object can return to the Draft state if you undeploy it. An object might be valid or not valid only in the design-time environment. You can move only a valid object to Deployed state.

Deployed

After you create and save a primary object, the object is saved as draft in design time. You must deploy the object to run in the run-time environment. When you deploy a primary object, all supporting objects linked to it are deployed automatically. For example, when you deploy a source, the topic and the connection associated with the source are also deployed. You can deploy an object only when the object is in Draft state. An object is valid both in the runtime and design time environment. When a primary object and its supporting objects are deployed successfully, the object is in Deployed state.

You can undeploy an object if you need to change the object properties. For example, for a JDBC connection, if you want to change the database connection information, you can undeploy the associated source. You can undeploy objects only when it is in deployed state. When you undeploy a primary object, all secondary objects associated with it are also undeployed. After you undeploy an object, the state of the object changes to draft. After you complete the changes, you can deploy the objects again.

When you edit a primary object, you can choose to save it first as draft and later deploy it. You can also choose to save and deploy simultaneously by using the **Save and Deploy** option.

Needs_Deployment

You can edit or update the deployed primary or the associated secondary objects without performing an undeploy. When you edit and save the objects, the state of the objects changes to the Needs_Deployment state. You can edit a deployed secondary object. If the secondary object has an associated primary object, you can use the **Save and Update** option so that all the associated primary objects are also deployed.

You can redeploy the objects that you have changed without undeploying them. In this case, the changes are affected only when you redeploy the objects.

You can map deployed objects in one service controller to another set of objects in another service controller. You can map or reassign objects from one service controller to another only when the objects are in the Needs_Deployment state. After you reassign the objects from one service controller to another, the state of the objects changes to the deployed state.

CHAPTER 5

Working with Topics

This chapter includes the following topics:

Topic Overview

The topic of a rule describes the types and properties of events coming into the system, and how to pre-filter these events. Topics are logical groupings of events that enable users to manage multiple event sources by funneling them into one grouping. You can reference the topic(s) in your rules.

You create rules to look for events by referencing a topic that enables you to group similar event sources within RulePoint. You can write a rule to look at a single event or multiple events with a single property or multiple properties that relate to a single topic or multiple topics.

RulePoint supports pre-filtering in the topic section, which means that you can provide simple, initial filtering before defining the conditions of the rule. For example, if your business process is to look at three events within five minutes, you can write the topic section to wait for these three events before sending the results to the condition section for further processing. If three or more events do not occur during any five-minute window, RulePoint will not continue to process the rule.

For example, the following rules scan incoming events to identify one stock volume event and two stock share price events:

- Rule 1: If you identify one stock volume that is greater than 30,000 for the day, then add the stock to the stock watchlist.
- Rule 2: If you identify two stock share prices drop by more than 10% in a five minutes window and the stock is on the stock watchlist, then notify the Fraud Team.

Note: Each topic specifies the default expiration time frame for events categorized into it. Only an administrator or the creator of the topic can change the default expiration time frame of the topic.

The names of the topic and its related property names in rules are not case-sensitive.

Example

Topics categorize events and their properties. When RulePoint receives an event, RulePoint identifies its properties and values and categorizes it under a specific topic.

A world news event will be categorized with the News topic, which can be defined with the heading, URL, full story text, and date and time properties. The topic picks up each of these properties and their values and displays them with the event.

The following table lists examples of topics and their related properties:

Topic	Properties
Stock Quotes	price, symbol, date/time, and financial index
Call Center	location, date/time, representative, activity
Community News	date/time, location, title, full story text, URL
Weather News	date/time, location, title, full story text, URL
Bank Transactions	bank name, date/time, activity
World News	date/time, title, full story text, URL
Business News	date/time, title, full story text, URL
Security Gate	date/time, location, license plate number, license plate state
Network and Server	server name, network IP, date/time, location
Credit Card Transactions	credit card type, balance, credit limit, date/time, limit

Creating a Topic

1. On the **Design** tab, click the **Topics** view.
2. From the **Actions** menu in the top-right corner of the view, select **New**.

   The **New Topic** dialog box appears.
3. Enter values for the topic details.

The following list describes the topic properties that you need to enter:

Property	Description
Name	Name of the topic. The topic name must be unique.
Description	Optional. Description of the topic.
Expires In	Expiration time frame for events coming into the topic. Enter the value in milliseconds. The default value is 600000. The expiration time frame is set for the events in the topic, not for the topic itself. The administrator or the creator of the topic sets this default expiration time frame. If you have the proper permissions, you can override that setting by defining another expiration time frame as long as it is shorter than the default value. You also can override this setting by defining a time frame within a rule. In this case, the value you set will have to be shorter than both the default value and the topic expiration time frames.
Responder access	Select how you want the responder to access the properties of events in the topic. You can select from the following options: - **Named Properties**. When you use Named Properties, only those properties referenced directly by the rule are kept in memory. This is a processing optimization that reduces the amount of memory that active events consume. The system performance is better when the system assesses only named properties. - **All Properties**. Some responses may need access to properties they do not explicitly reference. In this case, you can use the All Properties setting to keep all properties in memory regardless of whether they have been explicitly referenced.

4. In the **Properties** view, add the properties that you want to associate with the topic. You can add multiple properties to the topic.

   Note: You can also create properties in bulk without providing a description for the properties.

5. Click **Save**.

   Note: After saving the topic, you are granted ADMIN permissions to the topic. Only you or an administrator can modify, disable, delete, or grant other users permissions (ADMIN, READ, or WRITE) to the topic.

Copying a Topic

1. On the **Design** tab, click the **Topics** view.
2. Select the topic that you want to copy.

   You can select multiple topics to copy.

3. In the top-right corner of the view, click **Copy**.

   The **Copy Topics** dialog box appears.

4. Enter the name of the topic you want to create as a copy.

   The name of the copied topic must be different from the original topic.

5. Click **Save**.

Viewing All Topics

By default, the List View displays all the configured topics sorted by the last modified date. You can also sort them by name, state, or validity by clicking the appropriate column title. The State column indicates the status of the topic. The Validity column indicates whether or not the source to which the topic is connected is valid.

The following table describes each column displayed on the Topics page:

Property	Description
Name	Name of the topic.
Last Modified Date	The date and time that the topic was last modified.
State	The current status of the topic, such as DRAFT, DEPLOYED, or NEEDS_DEPLOYMENT.
Validity	Validity of the topic. The value is true if valid, false if invalid.

Viewing a Specific Topic

On the **Topics** page, the **Details View** displays all the configured topics on the left panel. Select a specific topic to view the topic information on the right panel.

You can use the topic property information to write rules that have a higher chance of matching by using a topic property that has a higher number of matches. For example, if the Topic title matches 80% and its headline matches only 30%, you would write a rule using title rather than headline.
The following table describes each column displayed for the topic properties:

Property	Description
Details	Details of the property. The fields that are displayed are Name, Description, Validity, State, Deployable, Expires In, Responder Access, Last Modified Date, Created Date, and Created By.
Properties	Name and description of each topic property.
Access Control List	Default permission for the user. The Access Control List view has the following properties: - Name. Name of the user. - Access. Grant user permissions to the connection. - Permissions. Access rights of the user to the connection.

Editing a Topic

You can edit the properties of an existing topic. You can also change the name of the topic that you want to edit. Any change that you make to a topic affects all rules that reference the topic. After you edit a topic that

has associated primary objects, you can choose to save the topic or simultaneously save and deploy the referenced primary objects.

1. On the **Design** tab, click the **Topics** view.
2. In the Topics content panel, select the topic name that you want to edit.
3. From the menu, click **Edit**.

   The **Edit Topic** dialog box appears.
4. Change the details and properties of the topic.
5. Click **Save**.

   The **Details View** displays the updated topic. The state of the topic is in the Draft state.
6. If you edit a deployed topic that has associated primary objects, perform the following tasks to save the topic:
   a. Click **Save**.

      A message appears displaying the name and the type of primary objects that reference the topic.
   b. Click **Continue**.

      A success message appears, stating that the topic is updated successfully.
   c. Click **OK**.

      The state of the topic and the objects that reference it changes from Deployed to Needs_Deployment state.
   d. If you want to redeploy the objects, click the specific topic in the Topics content panel, and then click **Update RunTime** from the menu.

      The **Dependent Objects** dialog box displays the list of primary objects that will be redeployed.
   e. Click **Continue**.

      The preview message displays successful redeployment. If the topic is referenced in a rule, the message prompts you if you want to trace the rules. If the topic is referenced in the source, the message prompts you to preview the events. If there are invalid objects, fix them, and click **Continue**.

      The state of the objects changes from Needs_Deployment state to Deployed state.
   f. Click **OK** to preview the events or trace the rules, or **Cancel** to escape the preview.
7. If you edit a deployed topic that has associated primary objects, perform the following tasks to simultaneously save and deploy the referenced primary objects:
   a. Click **Save and Update**.

      The **Dependent Objects** dialog box displays the objects that will be redeployed.
   b. Click **Continue**.

      The preview message displays successful redeployment. If the topic is referenced in a rule, the message prompts you if you want to trace the rules. If the topic is referenced in the source, the message prompts you to preview the events. If there are invalid objects, fix them, and click **Continue**.

      The state of the objects changes from Needs_Deployment state to Deployed state.

Deleting a Topic

You can delete an existing topic. You cannot delete topics that are referenced in sources or rules or topics that are in the Deployed or NEEDS_DEPLOYMENT state. If you attempt to delete a topic that is currently associated with a rule, RulePoint returns an error listing all of the rules that are associated with this topic. Rulepoint does not delete the topic until you edit or delete the rules associated with the topic.

1. On the **Design** tab, click the **Topics** view.
2. In the contents panel, select the topic that you want to delete.
3. From the menu, select **Delete**.

   A message prompts you to verify if you want to delete the topic.
4. Click **OK** to delete the topic.

Viewing Related Objects

You can view the related objects of an existing topic. The related objects are the primary objects that are associated with the topic.

1. On the **Design** tab, click the **Topics** view.
2. In the contents panel, select the topic that you want to view.
3. Select **View Related Objects** from the menu.

   The **Related Objects** dialog box appears.
   You can view the RulePoint objects that are associated with the topic.

CHAPTER 6

Working with Connections

This chapter includes the following topics:

Connections Overview

You can create different connection types, such as JDBC connection or JMS connection to connect the RulePoint services to the target database.

The connections can be shared across multiple service types. You can create the connections independently and map a single connection to various source objects.

Predefined Connections

RulePoint provides a set of predefined connection types. You can create the connections based on the source or responder types that you want to create.

The following table shows the types of predefined connections and the sources and responders that use the connection type:

Connection Type	Description	Source	Responder	Test Connection
Email Connection	Connection string to an email server that is defined for the organization.	None	Email Responder	Yes
Instant Messaging (Jabber/ XMPP) Connection	Jabber connection string to connect to a Jabber or XMPP server.	Instant Messaging (Jabber or XMPP) Source	Instant Messaging (Jabber or XMPP) Responder	Yes
SQL Connection	JDBC connection to connect to the target database. The SQL connection specifies database connection configuration parameters, such as the JDBC connection string, user name, password, and connection pool size.	SQL Source	SQL Responder	Yes
JMS Connection	Connects to JMS provider. A JMS ConnectionFactory object is used by the client to make connections to the server.	JMS Source	JMS Responder	Yes

Connection Type	Description	Source	Responder	Test Connection
Web Service Connection	Connection for web services.	Web Service Source	Web Service Responder	Yes
Ultra Messaging Connection	Connection for Ultra Messaging source and responder to connect to the UM sending and receiving application.	Ultra Messaging Source	Ultra Messaging Responder	Yes

Email Connection

When you create an email connection, you enter the name and the configuration details for the connection through the RulePoint user interface.

The following table describes the connection and configuration properties that you need to enter:

Property	Description
Name	Name of the connection. The connection name must be unique.
Description	Optional. Description of the connection.
Type	Type of connection. Select **Email Connection**.
Email Server	The fully qualified domain name of the SMTP server that is used to send outbound email from RulePoint. For example, `mail.mycompany.com`.
Port	Specifies the port number to connect to the SMTP server. The default value is 25. Valid values are from 1 to 65535.
Username	Optional. If required by the SMTP server, the SMTP server user name.
Password	Optional. If required by the SMTP server, the password associated with the SMTP server username.
Retry Count	The number of connection retry attempts that RulePoint should make. The default value is 3.
Retry Delay	The amount of time, in milliseconds, to wait between connection attempts. The default value is 1000.

Instant Messaging (Jabber/XMPP) Connection

The Jabber or Instant Messaging source service receives instant messages through a Jabber or XMPP server connections. For example, Jabber users can send messages to a RulePoint Jabber user.

When you create a Jabber connection, you enter the name and the configuration details for the connection through the RulePoint user interface.

The following table describes the connection and configuration properties that you need to enter:

Property	Description
Name	Name of the connection. The connection name must be unique.
Description	Optional. Description of the connection.
Type	Type of connection. Select **Instant Messaging (Jabber/XMPP) Connection**.
Server	The fully-qualified host name or DNS service name of the Jabber or XMPP Server to which this service connects.
Port	This port field is *not* for this DNS service request since the DNS supplies the port.
User Name	The username used by RulePoint to connect to the Jabber or XMPP server.
Password	The password associated with the user name used by RulePoint to connect to the Jabber or XMPP server.
Publish Presence	Option to create an event to a specified topic each time an IM user enters, leaves, or changes the status on the IM server. You can select from the following options: - Select True to create events. - Select False when you do not want to create an event or assert the events as presence packets.

SQL Connection

When you create a JDBC connection, you enter the name and the configuration details for the connection through the RulePoint user interface.

The following table describes the connection and configuration properties that you need to enter:

Property	Description
Name	Name of the connection. The connection name must be unique.
Description	Optional. Description of the connection.
Type	Type of connection. Select **SQL Connection**.
Driver Class	The name of the JDBC driver class used to establish connections to the specified database. The administrator installs the JDBC driver for the target database on the RulePoint server. For example, use the following sample driver class based on the database type: - DB2. `com.informatica.jdbc.db2.DB2Driver` - Oracle. `com.informatica.jdbc.oracle.OracleDriver` **Note:** Copy the JDBC drivers of the target database to the following directories: - `RulePoint_6\lib`, so that the run-time environment can use the drivers. - `RulePoint_6\design\webapps\rulepoint\WEB-INF\lib`, so that the design-time environment can use the drivers.
Connection URL	The JDBC connection string that you can use to connect to the target database. For example, use the following sample connection string based on the database type: - DB2. `jdbc:informatica:db2://hostname:portnumber;databaseName=dbname` - Oracle. `jdbc:informatica:oracle://hostname:portnumber;databaseName=dbname`

Property	Description
User Name	User name to log in to the target database.
Password	Password to log in to the target database.
Initial Pool Size	Optional. Initial Pool Size. The default value is 1.
Minimum Pool Size	Optional. Minimum number of connections to hold in the connection pool. The default value is 1.
Maximum Pool Size	Optional. Maximum number of connections to hold in the connection pool. The default value is 5.
Acquire Increment	Optional. Number of connections to inquire at once when increasing the pool size. The default value is 1.
Login Timeout	Optional. Log in timeout, in seconds. The default value is 3.
Retry Count	Number of connection retry attempts that RulePoint should make. The default value is 3. Setting this field to zero indicates that RulePoint should not retry the connection attempt.
Retry Delay	The amount of time, in milliseconds, to wait between connection attempts. The default value is 1000.
Maximum Idle Time	The maximum idle time that a connection can remain pooled but unused before being discarded. The default value is 1200 seconds. Idle connections are tested for validity within half this interval. If the maximum idle time is 0, the connections never expire.
Acquire Retry Attempts	A limit for the number of times that RulePoint attempts to acquire a connection from the database before giving up. If the value is less than or equal to zero, RulePoint makes an indefinite number of attempts to fetch a connection. The default value is 0.
Acquire Retry Delay	A time interval, in milliseconds, for two successive connection acquisition attempts.
Checkout Timeout	The maximum time period, in milliseconds, to wait for a connection to be checked-in or acquired, if the connection pool is exhausted. Zero specifies that the wait time is indefinite. A positive value causes a time-out and an SQLException to occur after the specified time period.

JMS Connection

The JMS source service produces events by receiving messages from a JMS provider and publishing those messages as events. When a JMS Service object is initialized, it instantiates its own JMS Connection Factory.

When you create a JMS connection, you enter the name and the configuration details for the connection through the RulePoint user interface.

The following table describes the connection and configuration properties that you need to enter:

Property	Description
Name	Name of the connection. The connection name must be unique.
Description	Optional. Description of the connection.
Type	Type of connection. Select **JMS Connection**.
JNDI Context Factory	The JMS provider specific initial JNDI context factory implementation for connecting to the JNDI service. This value is a fully qualified class name of the Initial Context Factory. For example, the class name of the Initial Context Factory for ActiveMQ is `org.apache.activemq.jndi.ActiveMQInitialContextFactory`. Your must add the JMS Client jars to the RulePoint server. For more information, see the documentation of the JMS provider.
JNDI Package Prefixes	Optional. A colon-delimited list of package prefixes to use when loading URL context factories. These are the package prefixes for the name of the factory class that will create a URL context factory. For values, see the documentation of the JMS provider.
Connection URL	The location of the naming service and port on which to connect. For example, `tcp://jndiserverA:61616`
JMS Connection Factory	The name of the object in the JNDI server that enables the JMS Client to create JMS connections. For example, `jms/QCF` or `jmsSalesSystem`. The default value is `ConnectionFactory`. For more information about this configuration, see the documentation of the JMS server administrator.
User Name	Optional. Enter authentication user name if required.
Password	Optional. Enter password for the authentication user name.

Web Services Connection

When you create a Web Services connection, you enter the name and the configuration details for the connection through the RulePoint user interface.

The following table describes the connection and configuration properties that you need to enter:

Property	Description
Name	Name of the connection. The connection name must be unique.
Description	Optional. Description of the connection.
Type	Type of connection. Select **Web Service Connection**.
URL	The URL of the WSDL file that contains the operations that you want to execute.
Login Name	The Web Service user name for Web Service authentication.
Password	The password associated with the Web Service user name.

Property	Description
Cache WSDL	Optional. The amount of time in seconds that the WSDL is retained in a cache. The default is zero which indicates a disabled cache.
User Name Space	Optional. User name space.
Retry Count	The number of connection retry attempts that RulePoint should make. The default value is 3. Setting this field to zero indicates that RulePoint should not retry the connection attempt.
Retry Delay	The amount of time, in milliseconds, to wait between connection attempts. Default is 1000.

Ultra Messaging Connection

The Ultra Messaging (UM) connection in RulePoint needs access to the configuration file in the Ultra Messaging application so that RulePoint communicates with the Ultra Messaging application.

When you create a UM connection, you enter the LBM file name and configuration details for the connection through the RulePoint user interface. UM configuration options defined in the LBM configuration files determine how the UM components and RulePoint read and write messages.

The following table describes the configuration properties for a UM connection:

Property	Description
Name	Name of the connection. The connection name must be unique.
Description	Optional. Description of the connection.
Type	Type of connection. Select **Ultra Messaging Connection**.

Property	Description
LBM Configuration File	The name of the LBM configuration file `lbm.cfg` in the UM sending and receiving application. The source and responder use the configuration file to read and write UM messages. You need to add the LBM configuration file in the following RulePoint installation directory path: `RULEPOINT_HOME/service-conf`. The LBM file contains the UM properties based on the type of UM edition used. The UM configuration file requires the following properties: - UMS. resolver_unicast_daemon config - UMP. ume_store config and resolver_unicast_daemon - UMQ. umq_queue_name config and resolver_unicast_daemon The **lbm.cfg** file also contains the lbm configuration parameters that are bound to the scope, that is, context, source, or receiver. You can override the values in the **lbm.cfg** file by defining your own parameters and options. The following configurations are examples of properties that an LBM file contains: - Host name of the LBMRD server to which the UM service connects. - The LBMRD port number. - The UM multicast interface address that the transmitter uses to send the outbound messages. - Late join configuration, which enables the messaging receiver to get previously sent data from the transmitter's history buffer. With late join, the source stores sent messages according to its Late Join configuration options so that a joining receiver can receive any of these messages that were sent before it joined the group. - Explicit ACK configurations that enable the receiver to send explicit acknowledgments to the transmitter, and acknowledges the receipt of a sent message. For information on configuration file samples for each of the UM source editions, see the Appendix.
LBM Context Parameters	The parameters used for overriding the configuration options provided in the LBM configuration file. You can also add the configuration options in the configuration file without modifying the file. The parameters of the LBMRD context defined in the `lbm.cfg` file. You can override the configuration options that are of the context scope, or define new configuration options in the `lbm.cfg` file. Use the following pattern to configure the LBMRD context parameters: `[context option_name option_value]` where: - context is the scope to which the option applies. - option_name is the predefined name for the option. - option_value is the value that you want to assign to the option.

Creating a Connection

1. On the **Design** tab, click the **Connections** view.
2. From the **Actions** menu in the top-right corner of the view, select **New**.

   The **New Connection** dialog box appears.
3. Enter values for the connection type and related configuration details.
4. Optionally, click **Test Connection** to test the connection.

   You test a connection to validate the connection information provided.
5. Click **OK** on the **Test Connection** dialog box.
6. Click **Save**.

Copying a Connection

1. In the **Design** tab, click the **Connections** view.
2. Select the connection that you want to copy.

   You can select multiple connections to copy.
3. From the **Actions** menu in the top-right corner of the view, select **Copy**.

   The **Copy Connections** dialog box appears.
4. Enter the name of the connection you want to create as a copy.

   The name of the copied connection must be different from the original connection.
5. Click **Save**.

Viewing All Connections

By default, the **List View** displays the connections sorted alphabetically by name. You can also sort them by Type, Last Modified Date, State, or Validity by clicking the appropriate column name. State denotes the current status of the connection. Validity indicates that the source or responder for the connection is valid or invalid.

The following table describes each column displayed on the Connections page:

Property	Description
Name	Name of the connection.
Type	Type of connection.
Last Modified Date	The date and time that the connection was last modified.
State	The current status of the connection, such as draft, deployed, or needs_deployment.
Validity	Validity of the connection. The value is true if valid, false if invalid.

Viewing a Specific Connection

On the Connections page in **Details View**, click the name of a specific connection in the left panel. You can view the connection information on the right panel.

The following table describes each column displayed for the connection properties:

Property	Description
Details	Details of the connection. The details that are displayed are Name, Type, Description, Validity, State, Deployable, Last Modified Date, Created Date, and Created By.
Properties	Properties of the connection.
Access Control List	Default permission for the user. The Access Control List view has the following properties: - Name. Name of the user. - Access. Grant user permissions to the connection. - Permissions. Access rights of the user to the connection.

Editing a Connection

You can edit the properties of an existing connection.

1. In the **Design** tab, click the **Connections** view.
2. In the Connections content panel, select the connection name that you want to edit.
3. From the menu on the right-hand side, click **Edit**.

   The **Edit Connection** dialog box appears.
4. Change the details and properties of the connection.
5. Optionally, click **Test Connection** to test the connection.

   You test a connection to validate the connection information provided.
6. Click **Save**.
7. If you edit a deployed connection that has associated primary objects, you can choose to save the connection. Perform the following tasks:
   a. Click **Save**.

      A message appears displaying the name and the type of primary objects that reference the connection.
   b. Click **Continue**.

      A success message appears, stating that the connection is updated successfully.
   c. Click **OK**.

      The state of the connection and the objects that reference it changes from Deployed to Needs_Deployment state.
   d. If you want to redeploy the objects, click the specific connection in the Connections content panel, and then click **Update RunTime** from the menu.

      The **Dependent Objects** dialog box displays the list of primary objects that will be redeployed.

e. Click **Continue**.

The preview message displays successful redeployment. If the connection is referenced in a rule, the message prompts you if you want to trace the rules. If the connection is referenced in the source, the message prompts you to preview the events. If there are invalid objects, fix them, and click **Continue**.

The state of the objects changes from Needs_Deployment state to Deployed state.

f. Click **OK** to preview the events or trace the rules, or **Cancel** to escape the preview.

8. If you edit a deployed connection that has associated primary objects, you can also choose to simultaneously save and deploy the referenced primary objects. Perform the following tasks:

a. Click **Save and Update**.

The **Dependent Objects** dialog box displays the objects that will be redeployed.

b. Click **Continue**.

The preview message displays successful redeployment. If the connection is referenced in a rule, the message prompts you if you want to trace the rules. If the connection is referenced in the source, the message prompts you to preview the events. If there are invalid objects, fix them, and click **Continue**.

The state of the objects changes from Needs_Deployment state to Deployed state.

Deleting a Connection

You can delete an existing connection.

1. In the **Design** tab, click the **Connections** view.
2. In the contents panel, select the connection that you want to delete.

   You can see the connection properties in **Details View**.
3. From the menu, select **Delete**.

   A message prompts you to verify if you want to delete the connection.
4. Click **OK** to delete the connection.

Viewing Related Objects

You can view the related objects of an existing connection. The related objects are the primary objects that references to the connection.

1. On the **Design** tab, click the **Connections** view.
2. In the contents panel, select the connection name.
3. From the menu, select **View Related Objects**.

   The **Related Objects** dialog box appears.
4. You can see the sources and responders that references the connection.

CHAPTER 7

Working with Sources

This chapter includes the following topics:

Source Overview

Sources connect to external systems to fetch data. The sources convert the fetched data into events. The events are published on topics.

You can configure a source through the RulePoint web user interface. RulePoint can receive data pushed from sources or it can pull data from sources on a scheduled basis. Supported services include SQL, text files, web pages, and RSS feeds.

Schedulable Sources

Schedulable sources enable you to generate events on a schedule that you set, such as one time every 30 minutes. An example of a schedulable source service is an RSS or Atom News Reader service that extracts events from an RSS news feed.

The schedulable sources pull the data into RulePoint from the outside world. In case of a failover to back up, pull sources start from where they left off using the run-time database parameters.

Listener Sources

Listener sources generate events at any time based on particular occurrences, not on a scheduled basis. An example of a listener source service is an Instant Messaging service. An Instant Messaging service listens for instant messages that meet certain criteria and then publishes an event on receiving a matching message.

The listener source service pushes the data into RulePoint through an external source. In case of a failover to the secondary, push sources continue to receive messages that are not explicitly acknowledged.

Prerequisites for Sources

Before you create a source, ensure to create a topic and a connection for the source based on the requirements.

Schedules Overview

Schedules are specified intervals at which the source has to poll data and publish them as events. You can schedule only the pull sources.

The schedules can either be Static or Dynamic.

Static Schedule

A static schedule generates events on a schedule that you set, such as one time every 30 minutes. An example of a static schedule is an RSS or Atom News Reader service that extracts events from an RSS news feed.

Dynamic Schedule

The dynamic schedule waits for a configured period after the previous task has completed before running again. You can configure the time interval to wait between the executions. The dynamic schedule ensures that there is no overlapping of queries while executing the same instance of the source task.

Creating a Schedule

1. In the Source content panel, select the source for which you want to create the schedule.

   You can view the source information in **Details View**. The source information has the schedules column.

2. From the menu, select **Create Schedule**.

   The **Create Schedule for <source_name>** dialog box appears.
   The schedules can either be **Static** or **Dynamic**.

3. To set the **Dynamic** schedule type, enter the repeat interval.

   The interval is measured in milliseconds.

4. To set the **Static** schedule type, enter the values as described in the following table:

Property	Description
Starts	Configure when to start the schedule. - Select **Immediately** to start the schedule immediately. - Select a date and time to start the schedule on a particular date.
Run Every	Configure when to run the schedules. - To have the schedule run at specific intervals, select the first option and specify how long between each scheduled run. - To have the schedule run on specific days, select the second option and then select the check boxes for the day you want to schedule to run. The source will run once per day at the time specified in the **Starts** section.
Ends	Configure when to end the schedule run. - To have the schedule run indefinitely, click **No end date**. - To have the schedule run a specific number of times, click **Run 1 times**. Type the desired number in the text box. Not available with the **Run on Specific Days** option. - To end the schedule on a particular date, click **Run until**. Enter the desired time.

5. Click **Add Schedule**.

Viewing Schedule for a Specific Source

The Schedules view is empty if no schedules are set for a source service. You can view the schedule for a source in **Details View**.

When you set one or more schedules for a source service, the Schedules view displays an entry for each schedule that includes the following data:

Property	Description
Type	Type of schedule, either static or dynamic.
Starts	The start date and time set for this schedule.
Run Every	The interval at which this schedule is set to run.
Ends	The end date and time set for this schedule.
Run State	The current status of the schedule, active or disabled.

Predefined Sources

RulePoint provides a list of predefined sources that are available after you install RulePoint.

The following table describes the predefined sources of RulePoint and their requirement for connection and schedule:

Source Type	Description	Connection	Schedule	Test Source
Event Generator	Reads an XML file from a specified URL, parses events from the file, and then publishes those events to a specific topic.	No	Yes	No
File Input Source	Monitors a file for changes and generates an event when the service detects a change to the contents of the file.	No	Yes	No
URL Monitoring Source	Monitors a URL for changes and generates an event when the service detects a change to the contents of the URL.	No	Yes	No
Instant Messaging Source (Jabber or XMPP)	Receives instant messages by connecting to a Jabber or XMPP server and publishes those messages as events.	Yes (Instant Messaging Connection)	No	No
JMS Source	Produces events by receiving messages from a JMS provider and publishes those messages as events.	Yes (JMS Connection)	No	No
RSS Source	Reads news feeds in RSS format and processes them as RulePoint events.	No	Yes	No

Source Type	Description	Connection	Schedule	Test Source
SQL Source	Connects to a database and executes an SQL query or command to publish a RulePoint event.	Yes (SQL connection)	Yes	Yes
Web Service Source	Sends a formatted SOAP message to run a remote WSDL operation.	No	Yes	No
Ultra Messaging Source	Receives messages from a UM topic, publishes these messages to a RulePoint topic, and processes them as RulePoint events.	Yes	No	No

Event Generator Source

The Event Generator source service reads an XML file from a specified URL, parses events from the file, and then publishes those events to a specific topic.

Event Generator Source Properties

When you create an Event Generator source, you enter the name and the configuration details for the source through the RulePoint user interface.

The following table describes the configuration properties of an Event Generator source:

Property	Description
Name	Name of the source. The source name must be unique.
Description	Optional. The description of the service. Make the description intuitive so that other users can relate to the use of this service.
Type	Type of source. Select **Event Generator**.
Topic	The topic to which RulePoint posts the events. Select the topic name that corresponds to the source. You can attach multiple topics to an event generator source. An event is read from the input event file, and the source publishes the event to only those of the attached topics that are listed in the event.
Date Format	The format used to parse the date and time. You can parse the date format if an event in the generator file specifies that you must include the `CURRENTTIME`, `CURRENTDATE`, or `CURRENTTIMESTAMP` function as a property in a generated event. The default format uses the Java API date format specified in the user preferences. The RulePoint default date format is `EEE, d MMM yyyy HH:mm:ss Z`.

Property	Description
Rate	The number of events posted to the topic as a single block of events. Default is 1.
Iterations	The number of times to publish the file before the service terminates. Default is 1. For example, if the number of iterations is two, RulePoint publishes the set of events twice before terminating the service. The generator loops over the file and feeds events from the beginning to end again for each additional iteration. For example, the file contains three events, A, B, C, and the number of iterations is two. The generator feeds the events into the system as A, B, C, A, B, C.
Delay	Optional. The number of milliseconds to delay between publishing blocks of events to the topic. Default is -1, that is, use the delay specified in the event xml file.
Post Processing Action	Disposition of event files after processing. Select one of the following options: - archive files. Moves the files to the archive directory, `RULEPOINT_HOME\archive`. - delete files. Deletes the files. - do nothing. Leaves the files untouched. The default post processing action is to do nothing.
Event File or Directory	Optional. The name of the XML file that contains the simulation data. The XML data file(s) must conform to the RulePoint EventGenerator XSD schema file. Place this file in the following directory path: `[RulePointHome]\data` If you specify a file name, RulePoint uses the file name as the source of the XML. If you specify a directory, RulePoint uses all the files in that directory. RulePoint processes the files in ascending order by file name.
Marker ID	Optional. The name to assign to the marker. This is the batch name that the sever associates with all published events. Avoid using multiple batch names as the system treats them as a single string. Specify one batch name for each source. When you configure a marker ID in the source, it demarcates a set of events between the marker_begin and the marker_end event. For more information, see "Markers" in the *RulePoint Administration Guide*.
Property Metadata Map	Optional. The unique name-value pairs of static metadata or additional information about this service that becomes part of the event properties. Do not use a null value for any property. Do not include space in the name of the metadata property. The name-value pairs appear in every event published by this source service. Format for Property Metadata: `Name=Jane Doe, Age=10` If you input an entry that contains a delimiter, such as a comma or a semicolon, you must place the entry into a quote. Putting a numeric entry into quotes makes it a string or non numeric. You cannot do math computation with a non numeric entry.
Marshaller Class Name	Class name of the marshaller. The marshaller class is responsible for converting or enriching the source event properties. It must include the full class name including the classpath. You must add the jar containing this class into the RulePoint library.
Marshaller Properties	Properties of the marshaller. You must pass the properties as key value pairs, separated by a comma, to the marshaller class. For example, property1=value1,property2=value2.

Event Generator Example

XML Event File

The following is an example of a properly configured XML event file:

```
<?xml version=”1.0“encoding=”UTF-8”?>
<sim:generator xmlns:sim="http://www.informatica.com/EventGenerator"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.informatica.com/EventGenerator http://
[RULEPOINT_DT_HOST]:[RULEPOINT_DT_PORT]/rulepoint/EventGenerator.xsd">
                http://www.informatica.com/EventGenerator.xsd">
<event topic=”stock”>
        <property>
                <name>price</name>
                <value>10</value>
        </property>
        <property>
                <name>date</name>
                <function>CURRENTDATE</function>
        </property>
        <property>
                <name>time</name>
                <function>CURRENTTIME</function>
        </property>
        <property>
                <name>timestamp</name>
                <function>CURRENTTIMESTAMP</function>
        </property>
</event>
</sim:generator>
```

The file shows the location of the RulePoint XSD schema file (http://[RULEPOINT_DT_HOST]:[RULEPOINT_DT_PORT]/rulepoint/EventGenerator.xsd), a price property, and the CURRENTTIME, CURRENTDATE, and CURRENTTIMESTAMP functions. The parameters CURRENTTIME, CURRENTDATE, and CURRENTTIMESTAMP return the same value.

Changing the XML Schema File Location

Your XML data files must point to the RulePoint XSD schema file. By default, the schema file is located on the Internet at the following URL:

```
http://[RULEPOINT_DT_HOST]:[RULEPOINT_DT_PORT]/rulepoint/EventGenerator.xsd
```

File Input Source

The File Input source service monitors and sends an alert when the file changes.

By default, the File Input source shows the new contents of the monitored file. If you want to see the full content of the file, you can select all contents from the **Mode** drop-down option menu.

File Input Source Properties

When you create a File Input source, you enter the name and the configuration details for the source through the RulePoint user interface.

The following table describes the configuration properties of a File Input source:

Property	Description
Name	Name of the source. The source name must be unique.
Description	Optional. The description of the service. Make the description intuitive so that other users can relate to the use of this service.
Type	Type of source. Select **File Input Source**.
Topic	The topic to which the RulePoint posts the events. Select the topic name that corresponds to the source.
Pathname	The path to a file that the service monitors. This path must be absolute.
Mode	The mode to get the contents of the file. Choose from the following options: - new. Get the new contents since the last change. - all. Get the entire file every time the file changes. The default mode is new.
Marker ID	Optional. The name to assign to the marker. This is the batch name that the sever associates with all published events. Avoid using multiple batch names as the system treats them as a single string. Specify one batch name for each source. When you configure a marker ID in the source, it demarcates a set of events between the marker_begin and the marker_end event. For more information, see "Markers" in the *RulePoint Administration Guide*.
Property Metadata Map	Optional. The unique name-value pairs of static metadata or additional information about this service that becomes part of the event properties. Do not use a null value for any property. Do not include space in the name of the metadata property. The name-value pairs appear in every event published by this source service. Format for Property Metadata: `Name=Jane Doe, Age=10` If you input an entry that contains a delimiter, such as a comma or a semicolon, you must place the entry into a quote. Putting a numeric entry into quotes makes it a string or non numeric. You cannot do math computation with a non numeric entry.
Marshaller Class Name	Class name of the marshaller. The marshaller class is responsible for converting or enriching the source event properties. It must include the full class name including the classpath. You must add the jar containing this class into the RulePoint library.
Marshaller Properties	Properties of the marshaller. You must pass the properties as key value pairs, separated by a comma, to the marshaller class. For example, property1=value1,property2=value2.

File Input Source Example

You can use the File Input source service to monitor and generate an event when a file changes.

You can use the File Input source service to monitor RulePoint logs or Tomcat server logs. When an exception related to error occurs, the File Input source can generate an event. For example, you can use the File Input source service in the healthcare industry where you can monitor patient history stored in files. When any new information is added to the file, the service can generate events for the entries.

URL Monitoring Source

The URL Monitoring source service monitors a web page and generates an event when the service detects a change to the contents of the web page. When the web page content changes, this service creates an event with one or more properties that contains the contents of the entire document.

The service supports monitoring http, https, ftp, and file. If you specify a local file in the URL field, the file that you specify do not need to exist when you create the service. You can regard the error generated in the web application console as informational.

You must configure the machine-to-resource authentication outside RulePoint.

URL Monitoring Source Properties

When you create a URL Monitoring source, you enter the name and the configuration details for the source through the RulePoint user interface.

The following table describes the configuration properties of a URL Monitoring source:

Property	Description
Name	Name of the source. The source name must be unique.
Description	Optional. The description of the service. Make the description intuitive so that other users can relate to the use of this service.
Type	Type of source. Select **URL Monitoring Source**.
Topic	The topic to which the RulePoint posts the events. Select the topic name that corresponds to the source.
URL	The URL of the web page to connect.
Marker ID	Optional. The name to assign to the marker. This is the batch name that the sever associates with all published events. Avoid using multiple batch names as the system treats them as a single string. Specify one batch name for each source. When you configure a marker ID in the source, it demarcates a set of events between the marker_begin and the marker_end event. For more information, see "Markers" in the *RulePoint Administration Guide*.
Property Metadata Map	Optional. The unique name-value pairs of static metadata or additional information about this service that becomes part of the event properties. Do not use a null value for any property. Do not include space in the name of the metadata property. The name-value pairs appear in every event published by this source service. Format for Property Metadata: `Name=Jane Doe, Age=10` If you input an entry that contains a delimiter, such as a comma or a semicolon, you must place the entry into a quote. Putting a numeric entry into quotes makes it a string or non numeric. You cannot do math computation with a non numeric entry.
Marshaller Class Name	Class name of the marshaller. The marshaller class is responsible for converting or enriching the source event properties. It must include the full class name including the classpath. You must add the jar containing this class into the RulePoint library.
Marshaller Properties	Properties of the marshaller. You must pass the properties as key value pairs, separated by a comma, to the marshaller class. For example, property1=value1,property2=value2.

URL Monitoring Source Example

You can use the URL Monitoring source service to monitor and generate an event when a web page changes.

A original URL document contains the following entries:

- ABCD
- EFG
- XYZ

The updated version of the document contains the following entries:

- ABCD
- XYZ
- 123

The changes in the document triggers an event with the following event properties:

```
contents: ABCD
XYZ
123
url: http://<hostname>/<filename>
```

Instant Messaging (Jabber or XMPP) Source

The Instant Messaging source service receives instant messages by connecting to a Jabber or XMPP server and publishing those messages as events.

For example, Jabber users can send messages to a RulePoint Jabber user.

Instant Messaging Source Properties

When you create an Instant Messaging source, you enter the name and the configuration details for the source through the RulePoint user interface.

The following table describes the configuration properties of a Instant Messaging source:

Property	Description
Name	Name of the source. The source name must be unique.
Description	Optional. The description of the service. Make the description intuitive so that other users can relate to the use of this service.
Property Metadata Map	Optional. The unique name-value pairs of static metadata or additional information about this service that becomes part of the event properties. Do not use a null value for any property. Do not include space in the name of the metadata property. The name-value pairs appear in every event published by this source service. Format for Property Metadata: `Name=Jane Doe, Age=10` If you input an entry that contains a delimiter, such as a comma or a semicolon, you must place the entry into a quote. Putting a numeric entry into quotes makes it a string or non numeric. You cannot do math computation with a non numeric entry.
Type	Type of source. Select **Instant Messaging Source (Jabber/XMPP)**.
Connection	Select the connection name to which you want to associate the source. You must create a connection for Jabber service before you create an Instant Messaging source.

Property	Description
Topic	The topic to which the RulePoint posts the events. Select the topic name that corresponds to the source.
Presence Priority	Presence priority for the source. Default priority is 1.
Marshaller Class Name	Class name of the marshaller. The marshaller class is responsible for converting or enriching the source event properties. It must include the full class name including the classpath. You must add the jar containing this class into the RulePoint library.
Marshaller Properties	Properties of the marshaller. You must pass the properties as key value pairs, separated by a comma, to the marshaller class. For example, property1=value1,property2=value2.

Instant Messaging (Jabber or XMPP) Source Example

You want to chat using an instant messenger, such as Facebook Chat. The instant messaging client connects to Facebook Chat through the Jabber or XMPP service. When you send or receive messages, the source service publishes the messages as events and posts the events to the corresponding topics.

JMS Source

The JMS source service produces events by receiving messages from a JMS provider and publishes those messages as events.

When you initialize a JMS service object, it instantiates its own JMS Connection Factory.

If RulePoint cannot connect to the external server, RulePoint might disable the service, disable any rules referencing the service, and write messages to the `rulepoint.log` file.

JMS Source Properties

When you create a JMS source, you enter the name and the configuration details for the source through the RulePoint user interface.

The following table describes the configuration properties of a JMS source:

Property	Description
Name	Name of the source. The source name must be unique.
Description	Optional. The description of the service. Make the description intuitive so that other users can relate to the use of this service.
Type	Type of source. Select **JMS Source**.
Connection	Select the connection name to which you want to associate the source. You must create a JMS connection before you create a JMS source.
JMS Destination	Name of the Queue or Topic on the JMS Provider (as defined in the JMS Connection Factory created by the JMS Administrator) from where RulePoint will receive JMS messages, (e.g. 'QueueA' or 'topic/SalesSystemsTopic'). Use 'topic/' prefix to specify a Topic

Property	Description
Client ID	Client identifier to identify the connection.
Subscription Name	Durable subscription name. Durable subscriptions can receive messages sent while the subscribers are not active. Durable subscriptions provide the flexibility and reliability of queues, but still allow clients to send messages to many recipients.
Acknowledgement Mode	Specifies the acknowledgement mode for non-transacted sessions. Following are the three modes: Auto Acknowledge - The session automatically acknowledges a client's receipt of a message either when the client has successfully returned. Client Acknowledge - A client acknowledges a message by calling the message's acknowledge method (Needs subscription name to be supplied). Duplicates OK, Acknowledge - This option instructs the session to lazily acknowledge the delivery of messages. Note:- When using Client Acknowledge mode source HA needs to enabled.
JMS Message Selector	Optional. Criteria for filtering message header or message properties, to limit which JMS messages RulePoint receives.
Marshaller Class Name	Class name of the marshaller. The marshaller class is responsible for converting or enriching the source event properties. It must include the full class name including the classpath. You must add the jar containing this class into the RulePoint library.
Marshaller Properties	Properties of the marshaller. You must pass the properties as key value pairs, separated by a comma, to the marshaller class. For example, property1=value1,property2=value2.

JMS Source Example

JMS providers are message-oriented middleware systems that send and receive JMS messages. WebSphere MQ JMS is an example of a JMS provider.

The RulePoint JMS Source connects to a JMS provider when it reads data from the configured destination - topic or queue. It then publishes the messages that it receives from the configured destination as events to the corresponding topics.

You can also configure a JMS source to read messages from an ultra messaging topic. For more information, see Appendix B: Creating an Ultra Messaging JMS Source.

RSS Source

The RSS source service reads news feeds in RSS or Atom format and processes them as RulePoint events.

RSS Source Properties

When you create a RSS source, you enter the name and the configuration details for the source through the RulePoint user interface.

The following table describes the configuration properties of a RSS source:

Property	Description
Name	Name of the source. The source name must be unique.
Description	Optional. The description of the service. Make the description intuitive so that other users can relate to the use of this service.
Type	Type of source. Select **RSS/Atom Source**.
Topic	The topic to which the RulePoint posts the events. Select the topic name that corresponds to the source.
URL	The URL for the RSS news feed.
User Name	Optional. The user name to access the specified URL if the URL requires authentication.
Password	Optional. The password associated with the user name required to access the specified URL.
Last Published Date	The date and time of posting the most recent event by this service. The RSS source publishes items with dates after the last published date. The date is updated automatically based on the contents of the RSS or ATOM source. Format for Last Published Date: `yyyy-MM-dd HH:mm:ss z`
Marker ID	Optional. The name to assign to the marker. This is the batch name that the sever associates with all published events. Avoid using multiple batch names as the system treats them as a single string. Specify one batch name for each source. When you configure a marker ID in the source, it demarcates a set of events between the marker_begin and the marker_end event. For more information, see "Markers" in the *RulePoint Administration Guide*.
Property Metadata Map	Optional. The unique name-value pairs of static metadata or additional information about this service that becomes part of the event properties. Do not use a null value for any property. Do not include space in the name of the metadata property. The name-value pairs appear in every event published by this source service. Format for Property Metadata: `Name=Jane Doe, Age=10` If you input an entry that contains a delimiter, such as a comma or a semicolon, you must place the entry into a quote. Putting a numeric entry into quotes makes it a string or non numeric. You cannot do math computation with a non numeric entry.
Marshaller Class Name	Class name of the marshaller. The marshaller class is responsible for converting or enriching the source event properties. It must include the full class name including the classpath. You must add the jar containing this class into the RulePoint library.
Marshaller Properties	Properties of the marshaller. You must pass the properties as key value pairs, separated by a comma, to the marshaller class. For example, property1=value1,property2=value2.

RSS Source Example

You can use the RSS source service to generate a RulePoint event when a new RSS or Atom feed comes in from the subscribed website.

The RSS reader checks the subscribed feeds regularly for new work, downloads any updates that it finds, and reads the feeds. The RSS source then generates a RulePoint event for the updates.

An RSS file is essentially an XML formatted plain text. The RSS file itself is relatively easy to read both by automated processes and by humans alike. This could be placed on any appropriate communication protocol for file retrieval, such as http or ftp, and reading software would use the information to present a neat display to the end user.

SQL Source

The SQL source service connects to a database and executes SQL queries or commands to create RulePoint events. The SQL source publishes an event for each row returned from an SQL query.

You can configure the SQL Source service to pull only the new or updated records from the database based on the last scheduled run of the SQL service. The filter to select only the updated records is built into the SQL query based on information retained and provided by the SQL source.

SQL Source Properties

When you create an SQL source, you enter the name and the configuration details for the source through the RulePoint user interface.

The following table describes the configuration properties of an SQL source:

Property	Description
Name	Name of the source. The source name must be unique.
Description	Optional. The description of the service. Make the description intuitive so that other users can relate to the use of this service.
Type	Type of source. Select **SQL Source**.
Connection	Select the connection name to which you want to associate the source. You must create a JDBC connection before you create an SQL source.
Topic	The topic to which the RulePoint posts the events. Select the topic name that corresponds to the source.
SQL	The SQL query to run against the target database. Use double angle brackets (chevrons) surrounded column names (<<column name>>) to identify the substitution parameters in the SQL statement. Each column name becomes a topic property. The topic property name is the SQL column name and the topic property value is the SQL column value.
Update SQL	Optional. You can run the UPDATE statement for each record extracted from the query and published to RulePoint. You can parameterize the SQL statement with SQL query column names.

Property	Description
Transaction Size	The frequency to update the parameters with data from the most recently processed records. The frequency is measured in numbers. When you input a SQL UPDATE statement, the transaction size is the number of UPDATE statement invocations allotted for each transaction based on the number of records published. An SQL source calls an update query as per the specified transaction size. You can set the following transaction sizes: - 0. Update query is called at the end of the publish cycle. - 'n' (1 <= n < Number of records). Update query is called after every n events published by the source. - 'n' (n >= Number of records). Update query is called at the end of the publish cycle.
Buffer Result	It determines whether to maintain a buffer of the result sets to publish to the topic. Used only when defining a SQL source service. Select from the following options: - Yes to publish each event separately. - No to publish all events from the result set at the same time. The default is No.
Marker ID	Optional. The name to assign to the marker. This is the batch name that the sever associates with all published events. Avoid using multiple batch names as the system treats them as a single string. Specify one batch name for each source. When you configure a marker ID in the source, it demarcates a set of events between the marker_begin and the marker_end event. For more information, see "Markers" in the *RulePoint Administration Guide*.
Parameters	Optional. The parameters to use in the SQL query. Enter the parameters separated by comma. Parameters are a set of name value pairs to substitute as values to the parameters in the SQL query. The field has values if the SQL query has parameters. Parameters sent to the SQL query are empty strings for empty fields.
Property Metadata Map	Optional. The unique name-value pairs of static metadata or additional information about this service that becomes part of the event properties. Do not use a null value for any property. Do not include space in the name of the metadata property. The name-value pairs appear in every event published by this source service. Format for Property Metadata: `Name=Jane Doe, Age=10` If you input an entry that contains a delimiter, such as a comma or a semicolon, you must place the entry into a quote. Putting a numeric entry into quotes makes it a string or non numeric. You cannot do math computation with a non numeric entry.
Marshaller Class Name	Class name of the marshaller. The marshaller class is responsible for converting or enriching the source event properties. It must include the full class name including the classpath. You must add the jar containing this class into the RulePoint library.
Marshaller Properties	Properties of the marshaller. You must pass the properties as key value pairs, separated by a comma, to the marshaller class. For example, property1=value1,property2=value2.

SQL Source Example

A stock broker wants to process stock market data feed. The stock broker wants to retrieve information about all the stocks or some preferred stocks based on the requirement.

You have a Stock table with the Symbol, Price, Volume, and RecordTime information. Consider the Stock table has the following stock information:

Symbol	Price	Volume	RecordTime
ABCD	10	5000	07-NOV-10 10.35.25.000000000 AM
EFG	22	7300	07-NOV-11 10.35.36.000000000 AM
XYZ	19	1200	27-NOV-12 10.38.25.000000000 AM

Simple SQL Query

You can use the following SQL query to retrieve all information from the Stock table and publish events for each row returned from the SQL query:

```
select * from Stock
```

The SQL query in the SQL source service results in three RulePoint events, one event for each row in the Stock table. Each event has four properties each, one for each column name in the Stock table. The SQL query extracts the same data every time it runs unless you constrain the query based on time, date, or some other criteria.

You can constrain the SQL query by defining a volume limit.

```
SELECT * from Stock WHERE Volume > 3000
```

This SQL query results in two RulePoint events, one event for symbol ABCD and the other for symbol EFG. Each event has four properties each, one for each column name in the Stock table.

You can also constrain the SQL query by defining a limit in the table property.

```
select symbol, price from stock where rownum <= 2
```

The SQL query results in two RulePoint events for the first two rows retrieved from the Stock table. Each event has two properties, one each for Symbol and Price.

Parameterized SQL Query

You can parameterize the SQL query and use the columns returned from the SQL query as inputs in the statement.

```
select symbol, price, recordtime from stock where recordtime > TO_DATE (<<recordtime>>, 'DD-MON-YY')
```

The parameter in the query is `recordtime`. You can define the parameter as a name-value pair. For example, `recordtime=06-NOV-12`.

The SQL query results in RulePoint events for all records from the Stock table that are saved after the `recordtime`.

Update SQL Query

The SQL Source service can be configured to pull only new or updated records from the database based on the last scheduled run of the SQL service. The filter to select only the updated records is built into the SQL query based on information retained and provided by the SQL source itself.

You can use the SQL query to select only unprocessed records.

In the following example, a stock table has a `processed_ind` column indicating which records have been processed, `F` for not processed and `T` for processed:

```
SELECT id, processed_ind, to_char(timestamp,'MM/dd/yyyy HH24:MI:SS') as timestamp, symbol,
price from processed_stock WHERE processed_ind='F'
```

You can use the following SQL query to update the `id` if the `processed_ind` column is processed:

```
UPDATE processed_stock set processed_ind='T' WHERE id=<<id>>
```

Note: When you update the SQL query, you do not need to initialize the `id`.

Ultra Messaging Source

You can configure an Ultra Messaging (UM) source to connect to a UM application to fetch data. An Ultra Messaging application can function both as a sending and receiving application.

Before you create a source, you must configure a UM connection and a topic. The Ultra Messaging Source reads messages from the external UM topic and publishes them as events to the RulePoint topic. The Source Topic is the external UM topic from which the UM source reads messages. The default value for the Source Topic is "rulepoint_input_topic." The RulePoint Ultra Messaging Source reads the messages from the UM topic and publishes events to the RulePoint topic that is bound to the source.

When you configure a connection for a UM source, the connection configurations must include the LBM configuration file. The file contains the Ultra Messaging configurations that provides information to RulePoint about where the Ultra Messaging processes are running. The LBM configurations differ for different UM source editions and for different types of functionalities.

You can use one of the following UM source editions, such as Ultra Messaging Streaming (UMS), Ultra Messaging Persistence (UMP) , or Ultra Messaging Queuing (UMQ). You can also connect to the UM related applications, such as PowerExchange for Ultra Messaging (PowerExchange for UM) or Vibe Data Stream for Machine Data.

For information about configuring the UM stream, persistence, queue stores, PowerExchange for UM, and Vibe Data Stream for UM, see Appendix A, “Connecting to an Ultra Messaging Application” on page 238. For more information about the UM functionalities, see the *Ultra Messaging Documentation*.

Ultra Messaging Source Properties

When you create an Ultra Messaging (UM) source, you enter the name and the configuration details of the source through the RulePoint user interface.

The following table describes the configuration properties of a UM source:

Property	Description
Name	Name of the UM source. The source name must be unique.
Description	Optional. The description of the source. Provide a description that is intuitive so that all users can relate to the use of this service .
Type	The type of source. Select **Ultra Messaging Source**.
Connection	Select the connection name to which you want to associate the source. **Note:** You must create a UM connection before you create a UM source.

Property	Description
Topic	The configured topic to which you want RulePoint to publish the events. Select the topic name that corresponds to the source.
Message Type	The message type. It can be Text (String) or Byte Array, or LBM Properties. If you set the message type as Text or Byte Array, the Ultra Messaging Source reads the message on the UM topic and publishes it as the value of the event property "ummsg." If you set the message type as LBM properties, the Ultra Messaging Source reads information from the message headers and publishes them as event properties. The ummsg property is set as an empty byte array of size 1.
Source Topic	The name of the external UM topic configured in the UM sending application.
Session ID	The UM TCP session ID, which is a 64-bit value. You need to set the session ID for the UM source which is the similar to the session ID in the UM sending application. The session ID uniquely identifies the following UM sources and receivers with unique topics: - If you are using a UMP source, the store uses the session ID to identify all the sources and receivers for a particular application. - If you are using a UMQ source, the queue uses the session ID defined in the UM connection to identify each partition uniquely and to ensure that the queue delivers messages only once. - If you are using a UMS source, the data stream uses the session ID of the receiver to receive messages from the UMS source with the same session ID. If you want to disable the session ID, set the value to -1.
Receiver Parameters	The configuration parameters of a UM receiving application. You can configure receiver parameters for a UM source to override the receiver parameters on which the UM runs. You can also define the receiver or context scopes for the configuration parameters in the `lbm.cfg` file. You define an option to assign values to an object's attributes. Use the following pattern to configure the receiver parameters: `[receiver option_name option_value]` where: - receiver is the scope to which the option applies. - option_name is the predefined name for the option. - option_value is the value that you want to assign to that option.
Marshaller Class Name	Class name of the marshaller. The marshaller class is responsible for converting or enriching the source event properties. It must include the full class name including the classpath. You must add the jar containing this class into the RulePoint library.
Marshaller Properties	Properties of the marshaller. You must pass the properties as key value pairs, separated by a comma, to the marshaller class. For example, property1=value1,property2=value2.

Web Page Monitor Source

The Web Page Monitor source service monitors a web page for changes and generates an event when the service detects a change to the contents of the web page.

When the web page content changes, this service creates an event with multiple properties that describe how the content changed. The source service supports HTTP and HTTPS modes.

If you specify a local file in the URL field, the file that you specify does not need to exist when you create the service. You can regard the error generated in the web application console as informational.

You must configure the machine-to-resource authentication outside RulePoint.

Web Page Monitor Source Properties

The following table describes the configuration properties of a Web Page Monitor source:

Property	Description
Name	Name of the source. The source name must be unique.
Description	Optional. The description of the service. Make the description intuitive so that other users can relate to the use of this service.
Property Metadata Map	Optional. The unique name-value pairs of static metadata or additional information about this service that becomes part of the event properties. Do not use a null value for any property. Do not include space in the name of the metadata property. The name-value pairs appear in every event published by this source service. Format for Property Metadata: `Name=Jane Doe, Age=10` If you input an entry that contains a delimiter, such as a comma or a semicolon, you must place the entry into a quote. Putting a numeric entry into quotes makes it a string or non numeric. You cannot do math computation with a non numeric entry.
Type	Type of source. Select **Web Page Monitor**.
Topic	The topic to which the RulePoint posts the events. Select the topic name that corresponds to the source.
URL	The URL of the web page to connect.
Marker ID	Optional. The name to assign to the marker. This is the batch name that the sever associates with all published events. Avoid using multiple batch names as the system treats them as a single string. Specify one batch name for each source. When you configure a marker ID in the source, it demarcates a set of events between the marker_begin and the marker_end event. For more information, see "Markers" in the *RulePoint Administration Guide*.
Marshaller Class Name	Class name of the marshaller. The marshaller class is responsible for converting or enriching the source event properties. It must include the full class name including the classpath. You must add the jar containing this class into the RulePoint library.
Marshaller Properties	Properties of the marshaller. You must pass the properties as key value pairs, separated by a comma, to the marshaller class. For example, property1=value1,property2=value2.

Web Page Monitor Source Example

You can use the Web page Monitor source service to monitor and generate an event when a web page changes.

A original web page contains the following entries:

- ABCD
- EFG
- XYZ

The web page is updated to contain the following entries:

- ABCD
- XYZ

- 123

The changes in the web page triggers an event with the following event properties:

```
count: 1
url: http://PublicWebServer/myfile.txt
text: ABCD
XYZ
123
date: 1/01/08 01:01:01 PM
diff: ABCD
<del class="changed">EFG </del><ins class="diff modified">XYZ </ins>
<del class="changed">XYZ</del><ins class="diff modified">123</ins>
```

The following table describes the contents of the properties in this example:

Property	Description
count	The incremental count identifying the number of times this particular URL has changed.
url	Monitored URL.
text	The entire contents of the updated URL.
date	The date and time of event generation.
diff	The specific changes made to the content of the URL. In the example, the <del class> tag identifies deleted content, and the <ins class> tag identifies added content.

Web Service Source

The Web Service source service sends a formatted SOAP message to execute a remote WSDL operation and use the result to create one or more events.

If Rulepoint cannot connect to the external server, then RulePoint cannot create the Web Service source.

Web Service Source Properties

When you create a Web Service source, you enter the name and the configuration details for the source through the RulePoint user interface.

The following table describes the configuration properties of a Web Service Source:

Property	Description
Name	Name of the source. The source name must be unique.
Description	Optional. The description of the service. Make the description intuitive so that other users can relate to the use of this service.
Type	Type of source. Select **Web Service Source**.
Topic	The topic to which the RulePoint posts the events. Select the topic name that corresponds to the source.

Property	Description
Service	Service to list the operations to run.
Operation To Execute	The WSDL operation that you want to run.
Input Parameters	List of parameters for your WSDL operation.
Retry Count	The number of connection retry attempts by RulePoint. The default value is 1. Setting this field to zero indicates that RulePoint must not retry to connect.
Retry Delay	The amount of time, in milliseconds, to wait between connection attempts. The default value is 5000.
Marker ID	Optional. The name to assign to the marker. This is the batch name that the sever associates with all published events. Avoid using multiple batch names as the system treats them as a single string. Specify one batch name for each source. When you configure a marker ID in the source, it demarcates a set of events between the marker_begin and the marker_end event. For more information, see "Markers" in the *RulePoint Administration Guide*.
Property Metadata Map	Optional. The unique name-value pairs of static metadata or additional information about this service that becomes part of the event properties. Do not use a null value for any property. Do not include space in the name of the metadata property. The name-value pairs appear in every event published by this source service. Format for Property Metadata: `Name=Jane Doe, Age=10` You can also use the semicolon as a separator, such as `Name=Doe ; Age=10`. If you input an entry that contains a delimiter, such as a comma or a semicolon, you must place the entry into a quote. Putting a numeric entry into quotes makes it a string or non numeric. You cannot do math computation with a non numeric entry.

XPath Expression	Optional. If the operation returns a result, the type of output that the Web Service expects. The output types include an XML document or the result of the application of an XPath expression that you provide.
	The XPath expressions that you provide depend on the Web Service. For example, if you use a Web Service that returns an integer representing elevation, you must provide an XPath expression for each elevation element. The expression then creates key-value pairs from the result and publishes an event.
	XPath expressions must start with a slash (/). Do not use functions. For example,
	`//Distance`
	`//Distance | //CityName1 | //CityName2`

Marshaller Class Name	Class name of the marshaller. The marshaller class is responsible for converting or enriching the source event properties. It must include the full class name including the classpath. You must add the jar containing this class into the RulePoint library.
Marshaller Properties	Properties of the marshaller. You must pass the properties as key value pairs, separated by a comma, to the marshaller class. For example, property1=value1,property2=value2.

Web Service Source Example

The Web Services Provider is the provider entity of the PowerCenter web service framework that makes PowerCenter workflows and data integration functionality accessible to external clients through web services.

The Web Services Hub is an application service in the PowerCenter domain that uses the SOAP standard to receive requests and send responses to web service clients. The Web Services Hub interacts with the Integration Service and the Repository Service to process requests and generate responses. When the Web Services Hub sends or receives messages to or from external clients, the source service publishes the messages as events and posts the events to the corresponding topics.

Creating a Source

1. On the **Design** tab, click the **Sources** view.
2. From the **Actions** menu on the top-right corner of the view, select **New**.

   The **New Source** dialog box appears.
3. Enter the source details, topic information, configuration details, and marshaller details.

   Marshaller details are optional information for the sources that require Marshaller information.
4. If you do not have the dependent objects for the source, you can create the objects using the workspace on the right hand side of the **Source** dialog box.
5. If you do not have a related topic for the source, create the topic.

   For information on how to create a topic, see "Creating a Topic" on page 30.
6. If you do not have a related connection for the source, create the connection.

   For information on how to create a connection, see "Creating a Connection" on page 42.
7. Optionally, to test the source, click **Test Source**.

   The **Test Source** option is available only for SQL sources. When you click **Test Source**, it populates the topic properties from the source database.

   The **Edit Topic - Step 1 of 2** dialog box appears.
8. Save the topic properties.
9. Click **Save**.

   The source is now created and is in Draft state. You must now create a schedule for running the source and then deploy the source.
10. If the source has a configured schedule, click **Save and Deploy** to deploy the source.

    A message prompts you if you want to deploy.
11. Click **OK**.

    A message appears stating successful deployment and prompts you if you want to preview the events.
12. Click **OK** to view the preview, or click **Cancel**.

    Note: When you choose to preview the events, if no events are generated, a message appears stating that the source is still executing and no events are generated. Click **OK**.

    The Event Preview page appears displaying the generated events with the details. You can search for specific events using **Event Search**.
13. Click **Close** after you view the preview.

Running a Source Once

After you create a source, you can run the source once to verify the connections and configurations for the source.

After you create and deploy the source, you can run the source. You do not have to schedule the source to run it once. When you run the source once, RulePoint creates the topic schema and redirects you to the related topic. You can edit the topic, if required. Running the source once does not create any event or trigger a rule.

1. Click the **Dashboard** tab.
2. Click the source controller on which you deployed the source.
3. In the contents panel, click the **Sources** tab, and then select the source that you want to run.
4. From the **Actions** menu in the lower panel, click **Run Once**.

   A success message appears.
5. Click **OK**.

Deploying a Source

1. On the **Design** tab, click the **Sources** view.
2. In the Source content panel, select the source you want to deploy.

   To deploy a source, the source must be in draft state, and you must add a schedule to the source.
3. From the menu, select **Deploy**.
4. To confirm that you want to deploy the source, click **OK**.

   The source is deployed. When you deploy a source, the topic and the connection associated with the source also gets deployed automatically.
5. To confirm that you want to preview the events, click **OK**.

   Note: If the events are not generated, a message appears stating that the source is still executing. Click **OK**.

   The events preview page appears which displays the events generated for the source. You can search for events using **Event Search**. You can choose to switch on **Keep Alive** to view events as and when they are generated.
6. Click **Close** after you complete viewing the events.

Viewing Event Preview

You can preview the events generated for a deployed source. You have the option to search for specific events. Use the **Auto Fresh** feature if you want to view events as they are generated.

1. On the **Design** tab, click the **Sources** view.
2. In the Sources contents panel, select the source to preview the events for that source.

3. Select **Preview Events** from the menu on the right-hand side of the selected source.

   The event preview report displays the events generated for the topic associated with the rule.

Copying a Source

Dynamic schedules are copied only if the source has dynamic or static schedules.

1. On the **Design** tab, click the **Sources** view.
2. Select the source that you want to copy.

   You can select multiple sources to copy.
3. From the **Actions** menu on the top-right corner of the view, select **Copy**.

   The **Copy Source** dialog box appears.
4. Enter the name of the source you want to create as a copy.

   The name of the copied source must be different from the original source.
5. Click **Save**.

Editing a Source

You can edit an existing source.

1. On the **Design** tab, click the **Sources** view.
2. Select the source that you want to edit.
3. From the menu in the right-hand side, click **Edit**.

   The **Edit Source** dialog box appears.
4. Change the details and properties of the source you want to edit.
5. Click **Save**.
6. To deploy the source, click **Save and Deploy**.

   Note: If you edit a source that is deployed, click **Save and Redeploy**. The source and the associated secondary objects are now in the Deployed state.

Viewing All Sources

By default, List View displays sources sorted by the last modified date. You can also sort them by Type, Last Modified Date, State, or Validity by clicking the appropriate column name. State denotes the current status of the connection. Validity indicates whether the source is valid or not valid.

The following table describes each column displayed on the sources page:

Property	Description
Name	Name of the source.
Type	Type of the source.
Last Modified Date	The date and time that the source was last modified.
State	The current status of the source, such as draft, deployed, or needs_deployment.
Validity	Validity of the source. The value is true if valid, false if not valid.

Viewing a Specific Source

On the Source page, select **Details View**. Click the name of a specific source to display in the left panel. You can view the source information on the right panel.

The following table describes each column displayed for the source properties:

Property	Description
Details	Details of the source. The details are Name, Source Type, Description, Validity, State, Deployable, Connection, Topic, Last Modified Date, Created Date, and Created By.
Properties	Properties of the source.
Schedules	List of schedules associated with the source.
Access Control List	Default permission for the user. The Access Control List view has the following properties: - Name. Name of the user. - Access. Grant user permissions to the source. - Permissions. Access rights of the user to the source.

Viewing Related Objects

You can view the related objects of an existing source. The related objects are the topic and connections that references to the source.

1. On the **Design** tab, click the **Sources** view.

2. In the Sources content panel, select the source name.
3. From the menu, select **View Related Objects**.

   The **Related Objects** dialog box appears.
4. You can see the topics and connections associated with the source.

Using Marshaller and Unmarshaller

A marshaller can be attached to a source, to enrich the events that are published by the source, before they can be processed by the Event Processor. Similarly, on the responder side, you can attach a marshaller (called as an Unmarshaller) to enrich or to do some bookkeeping on the activation before sending it out to the responder.

A typical use case for a marshaller would involve a source that is configured to listen to an external JMS provider which publishes Map messages. You might want to enrich the entries in the map before they can be processed by the Event Processor. So, you can create a Marshaller by using the standard interfaces, where the MapMessage entries can be processed as required.

Complete the following steps to configure a Marshaller:

1. Enter the fully-qualified class name in the Marshaller class field.
2. Enter the properties (key-value pairs) that you want to pass on to the Marshaller, in the Marshaller Properties field. Use commas as delimiters.
3. Copy the jar file containing the marshaller class to the `RULEPOINT_HOME/lib` directory.
4. Restart the run-time environment, if it is running.

CHAPTER 8

Working with Responders

This chapter includes the following topics:

Responder Overview

Responders handle dispatching of alerts to external systems. When a rule condition matches, the rule processing engine creates a response. RulePoint then sends the response to the specified external system through a conduit called responder.

With a responder service, you can define the interface parameters for a particular type of response, which is the action to be taken when a rule activates. From a single responder service, you can create multiple specific responses.

An example of a responder service is an email service that defines the mail server and account used to send an email when a rule activates. You can define different recipients and the content of the email to be sent when a particular rule activates.

After you create a responder service, you then must create and associate specific response to the responder service. After you save the responder service, you have the ADMIN permissions to the responder service. Only you or an administrator can change, disable, delete, run, or grant other users permissions (ADMIN, READ, WRITE, or EXECUTE) to the responder service.

Predefined Responders

RulePoint provides a list of predefined responders that are available after you install RulePoint.

The following table describes the predefined responders of RulePoint and their requirement for connection:

Responder Type	Description	Connection
Email Responder	Sends an alert through an email message.	Yes (Connection For Email Service)
Event Transformer	Transforms event output and publishes it back into the RulePoint system as a new event.	No
File Responder	Responds to events by writing to a file.	No
HTTP Service Responder	Performs an HTTP GET from or POST to a specified URL and, if required, gets or sends specified parameters.	No
Instant Messaging (Jabber/XMPP) Responder	Sends an instant message by connecting to a Jabber or XMPP server.	Yes (Connection For Jabber Service)
JMS Responder	Connects to JMS Provider and sends responses.	Yes (JMS connection)
RTAM Responder	Sends an alert to a RTAM server.	No
SQL Responder	Connects to a database through a JDBC connection and executes an SQL command.	Yes (JDBC connection)
Ultra Messaging Responder	Connects to a UM receiving application, and sends responses.	Yes (Connection for UM service)
Watchlist Responder	Performs actions against a watchlist.	No
Web Service Responder	Sends a formatted SOAP message to run a remote Web Services (WSDL) operation.	No

Email Responder

The email responder service responds to events by sending an email message.

Email Responder Properties

When you create an Email responder, you enter the name and the configuration details for the responder through the RulePoint user interface.

The following table describes the configuration properties of an email responder:

Property	Description
Name	Name of the responder. The responder name must be unique.
Description	Optional. Description of the responder.
Type	Select Email Responder.
Connection	Select the connection name that you want to associate the responder to. You must create a connection for email service before you create an email responder.
To	A comma-delimited list of email addresses to which RulePoint sends the response.
Subject	The subject of the email response.
Body	The body of the email response.
BCC	Optional. A comma-delimited list of email addresses to send the responses to as a blind copy.
CC	Optional. A comma-delimited list of email addresses to send the responses to as a copy.
Content Type	The content type of the email response. You can specify text or html. These values are not case sensitive. The default value is text.
From	The email address from which the email response must originate.
Reply To	Optional. A comma-delimited list of email addresses to send replies to the email response.
Date	Optional. The date that you want to appear as the email send date. The date must be in the following RFC 2822 format: `[day-of-week], DD MMM YYYY HH:MM ZONE` For example: `Mon, 01 Jan 2008 14:20 EST`
Marshaller Class Name	Class name of the marshaller.
Marshaller Properties	Key-value pairs to pass as properties of the marshaller.

Event Transformer

The Event Transformer responder service transforms event output and publishes it back into the same RulePoint system as a new event.

Event Transformer Properties

When you create an Event Transformer responder, you enter the name and the configuration details for the responder through the RulePoint user interface.

The following table describes the configuration properties of an event transformer responder service:

Property	Description
Name	Name of the responder. The responder name must be unique.
Description	Optional. Description of the responder.
Type	Select **Event Transformer**.
Properties	Optional. The topic properties to copy to the event response. To include all of the properties, use the asterisk sign. If you have configured the topic with **Named Properties**, then the properties referenced in the `WITH` portion of the rule are copied.
Parameters	Optional. A comma-delimited list of name-value pairs provided in the response. For example, `symbol=XYZ, price=10` It is referenced in a rule as `params`.
Topic	The topic on which the event transformer creates an event.
Marshaller Class Name	Class name of the marshaller.
Marshaller Properties	Key-value pairs to pass as properties of the marshaller.

File Output

The File Output Responder service responds to events by writing to a file.

This responder service responds in the following two modes:

- Appends text to an existing text file.
- Overwrites the contents of a text file.

File Output Properties

When you create a File Output responder, you enter the name and the configuration details for the responder through the RulePoint user interface.

The following table describes the configuration properties of a File Output responder:

Property	Description
Name	Name of the responder. The responder name must be unique.
Description	Optional. Description of the responder.
Type	Select **File Output**.

Property	Description
Path Name	Path of the output file. For example, `c:/temp/myfile.txt`. The responder service writes the data into this output file.
Mode	The mode to write the contents to the file. Choose from the following options: - append. Writes the latest addition to the file. - overwrite. Writes the full text to the file each time. The default mode is append.
Contents	The data or text that the responder service writes to the file.
Marshaller Class Name	Class name of the marshaller.
Marshaller Properties	Key-value pairs to pass as properties of the marshaller.

HTTP Service

The HTTP Responder service responds to events by performing an HTTP GET from or POST to a specified URL.

The responder service gets or sends specified parameters in the payload of the HTTP request, if required.

HTTP Service Properties

When you create a HTTP Service responder, you enter the name and the configuration details for the responder through the RulePoint user interface.

The following table describes the configuration properties of a HTTP service responder:

Property	Description
Name	Name of the responder. The responder name must be unique.
Description	Optional. Description of the responder.
Type	Select **HTTP Service**.
Target URL	The URL of the object.
Method	The method to be performed on the object specified in the URL field.
User Name	Optional. If the HTTP server requires authentication, the user name used to access the HTTP server.
Password	Optional. If the HTTP server requires authentication, the password associated with the user name used to access the HTTP server.

Property	Description
Content Type	Optional. The HTTP character set and content type. If you leave this field left blank, the default value is ISO-8859-1 for the character set, and text or plain for the content type. Ensure that the HTTP server to which you are connecting supports and accepts one or both. To set the content type, enter text or html, text or xml into the field. To set the content type and character set, use the following format: `text/xml;charset=ISO-8850-1`
Message Body	Optional. If you use multipart or mime-encoded as content type to post data, the message body is the form data. If you use the body parameter, it ignores any specified key-value parameter pairs.
Parameters	Optional. A comma-delimited list of name-value pairs provided in the response. For example, `symbol=XYZ, price=10`.
Marshaller Class Name	Class name of the marshaller.
Marshaller Properties	Key-value pairs to pass as properties of the marshaller.

Instant Messaging (Jabber or XMPP) Responder

The Jabber or Instant Messaging responder service responds to events by connecting to a Jabber or XMPP server and sending instant messages. For example, RulePoint can masquerade as a user and respond according to the rules you define.

Instant Messaging (Jabber or XMPP) Responder Properties

When you create an Instant Messaging responder, you enter the name and the configuration details for the responder through the RulePoint user interface.

The following table describes the configuration properties of Instant Messaging responder:

Property	Description
Name	Name of the responder. The responder name must be unique.
Description	Optional. Description of the responder.
Type	Select **Instant Messaging Responder (Jabber/XMPP)**.
Connections	Select the connection name that you want to associate the responder to. You must create a connection for Jabber Service before you create an Instant Messaging responder.
To	The user name of the instant messaging account recipient.
Subject	The subject of the instant message.
Message Body	The body of the instant message.

Property	Description
Marshaller Class Name	Class name of the marshaller.
Marshaller Properties	Key-value pairs to pass as properties of the marshaller.

JMS Responder

JMS Responder Properties

When you create a JMS responder, you enter the name and the configuration details for the responder through the RulePoint user interface.

The following table describes the configuration properties of JMS responder:

Property	Description
Name	Name of the responder. The responder name must be unique.
Description	Optional. Description of the responder.
Type	Select **Instant Messaging Responder (Jabber/XMPP)**.
Connection	Select the connection name that you want to associate the responder to. You must create a JMS connection before you create a JMS responder.
Parameters	The parameters to use in the SQL query. Enter the parameters separated by comma. Parameters are a set of name-value pairs to substitute as values to the parameters in the SQL query. The field has values if the SQL query has parameters. Parameters are sent to the SQL query as an empty string for empty fields.
JMS Destination	Name of the Queue or Topic on the JMS Provider (as defined in the JMS Connection Factory created by the JMS Administrator) from where RulePoint will receive JMS messages. For example, 'QueueA' or 'topic/SalesSystemsTopic'. Use 'topic/' prefix to specify a Topic.
Delivery Mode	The delivery mode for publishers which ensures that an application receives all the published messages. You can use the persistent of non-persistent delivery mode for the publishers.
Retry Count	The number of connection retry attempts by RulePoint. The default value is 1. Setting this field to zero indicates that RulePoint must not retry to connect.
Retry Delay	The amount of time, in milliseconds, to wait between connection attempts. The default value is 5000.
Body	The body of the JMS response message.
Priority	Priority of the message.
Expiration Time	Time after which the JMS response expires.

Property	Description
Marshaller Class Name	Class name of the marshaller.
Marshaller Properties	Key-value pairs to pass as properties of the marshaller.

RTAM Responder

The Real-Time Alert Manager responder service responds to events by sending alerts to a Real-Time Alert Manager server.

RTAM Responder Properties

When you create a RTAM responder, you enter the name and the configuration details for the responder through the RulePoint user interface.

The following table describes the configuration properties of a RTAM responder:

Property	Description
Name	Name of the responder. The responder name must be unique.
Description	Optional. Description of the responder.
Type	Select **RTAM Responder**.
To	The user name of the RTAM user who would receive the alert.
Groups	Optional. A group address to receive the Response message as defined by your RTAM administrator.
Subject	The subject of the alert.
Body	The body of the alert. You can format the body as text or using simple HTML tags, such as <B>, <I>, <UL>, .
Actions	Optional. A semicolon-delimited list of comma-delimited name-link pairs for actions to show in the RTAM interface for this alert. For example, ActionName1, ActionLink1; ActionName2, ActionLink2
Channels	Optional. A semicolon-delimited list of channels to deliver the alert to. The default value is blank.
Header	Optional. The header information for the alert, which the user can view in RTAM.
Metadata	Additional information or metadata.
Message Priority	The priority to place on the RTAM alert. Different priorities function differently within RTAM. The range in ascending order is 0 to 5, 5 being the highest priority. The priority 0 indicates that the message is not yet prioritized.

Property	Description
Marshaller Class Name	Class name of the marshaller.
Marshaller Properties	Key-value pairs to pass as properties of the marshaller.

SQL Responder

The SQL responder service responds to events by connecting to a database and executing SQL commands.

In a SQL responder service, RulePoint passes the event properties as SQL parameters to the SQL statement and ignores the result set.

SQL Responder Properties

When you create an SQL responder, you enter the name and the configuration details for the responder through the RulePoint user interface.

The following table describes the configuration properties of an SQL responder:

Property	Description
Name	Name of the responder. The responder name must be unique.
Description	Optional. Description of the responder.
Type	Select SQL Responder.
Connection	Select the connection name that you want to associate the responder to. You must create a JDBC connection before you create an SQL responder.
SQL	The SQL query for the responder. Use double angle brackets (chevrons) surrounded column names (<<column name>>) to identify the substitution parameters in the SQL statement. The data values provided in the response replaces the parameters with the matching parameter names.
Parameters	The parameters to use in the SQL query. Enter the parameters separated by comma. Parameters are a set of name-value pairs to substitute as values to the parameters in the SQL query. The field has values if the SQL query has parameters. Parameters are sent to the SQL query as an empty string for empty fields.
Marshaller Class Name	Class name of the marshaller.
Marshaller Properties	Key-value pairs to pass as properties of the marshaller.

SQL Responder Example

You can create an SQL responder to record details of alerts in the RulePoint database or any other target database.

Consider the example of Stock table from the SQL Source services in which you have a Stock table with the Symbol, Price, Volume, and RecordTime information.

SQL Responder for a source with a parameterized query

You can use the following SQL query to insert information into the Stock table:

```
insert into Stocks(symbol, price) values('TEST','81')
```

This SQL query inserts a row in the Stock table with values for Symbol and Price columns.

SQL Responder for a source with query using limit

You can constrain the SQL query by inserting the values for selected columns of the table.

```
insert into Stocks(symbol, price) values('LIMIT','13')
```

This SQL query inserts values to the Symbol and Price column of the Stock table.

Ultra Messaging Responder

When an event matches the rule condition, the Ultra Messaging (UM) responder writes processed native messages to the destination topic defined in the UM responder configuration. A UM receiving application consumes the processed native messages.

Ultra Messaging Responder Properties

When you create an UM responder, you enter the name and the configuration details for the responder through the RulePoint user interface.

The following table describes the configuration properties of an UM responder:

Property	Description
Name	Name of the responder. The responder name must be unique.
Description	Optional. Description of the responder.
Type	The responder type. Select **Ultra Messaging Responder**.
Connection	The type of connection to connect to the UM responder. Select **umconnection**.
Message Type	The intended message type. It can be Text (String), Byte Array, or LBM Message Properties.
Message Body	The body of the UM message that the responder sends on the configured external UM topic. The message type can be a text (string) of byte array, or the LBM properties: If you set the message type as Text or Byte Array, the Ultra Messaging Responder publishes the message on the destination UM topic and publishes it as the value of the response property "ummsg." If you set the message type as LBM Properties, the Ultra Messaging Responder writes response properties as Ultra Messaging message headers and sends them as message properties. The "ummsg" message property sets as an empty byte array of size 1.

Property	Description
Destination Topic	Name of the topic on the UM receiving application that receives the processed UM native messages.
Session ID	Enter a UM session ID, which is a 64-bit value to identify the UM receiver. The session ID is unique and must be the same for the UM source and the receiver. Default is -1. In this case, any receiver can receive the messages from the configured UM topics.
Source Parameters	The configuration parameters of the source defined in the `lbm.cfg` file. You can override the configuration parameters in the source type by specifying the options. Use the following pattern to enter the source parameters: `[source option_name option_value]` where: - source is the scope to which the option applies. - option_name is the predefined name for the option. - option_value is the value that you want to assign to the option. **Note:** If you have multiple source parameters, place each parameter on a separate line.
Marshaller Class Name	Class name of the marshaller. The marshaller class is responsible for converting or enriching the source event properties. It must include the full class name including the classpath. You must add the jar containing this class into the RulePoint library.
Marshaller Properties	Properties of the marshaller. You must pass the properties as key value pairs, separated by a comma, to the marshaller class. For example, `property1=value1,property2=value2`.

Watchlist Responder Service

The Watchlist responder service responds to events by performing actions against a watchlist.

The responder service can add, remove, change, or replace values in a watchlist.

Watchlist Responder Properties

When you create a Watchlist responder, you enter the name and the configuration details for the responder through the RulePoint user interface.

The following table describes the configuration properties of a Watchlist responder:

Property	Description
Name	Name of the responder. The responder name must be unique.
Description	Optional. Description of the responder.
Type	Select **Watchlist Responder**.
Watchlist Name	The name of the watchlist on which to operate. The DRQL syntax references the watchlist name as objectName.

Property	Description
Operation	The type of operation. Choose from the following options: - add. Adds a value to the watchlist. - remove. Removes a value from the watchlist. - change. Changes an existing value in the watchlist. - replace. Replaces all of the values in the watchlist. If you specify an action other than any action in the preceding list, RulePoint substitutes that action with add.
Old Value	Optional. If the operation is remove or change, the value on which to operate.
New Value	Optional. If the operation is add, change, or replace, the new value replaces the value in the **Old Value** field.
Marshaller Class Name	Class name of the marshaller.
Marshaller Properties	Key-value pairs to pass as properties of the marshaller.

Web Service Responder

The Web Service Responder service responds to events by sending a formatted SOAP message to execute a remote web services service.

You define a Web Service Responder service the same way that you define a Web Service source service.

Web Service Responder Properties

When you create a Web Service responder, you enter the name and the configuration details for the responder through the RulePoint user interface.

The following table describes the configuration properties of a Web Service responder:

Property	Description
Name	Name of the responder. The responder name must be unique.
Description	Optional. Description of the responder.
Type	Select **Webservice**.
Connection	Select the connection name that you want to associate the responder to. You must create a web service connection before you create a web service responder.
Service	Optional. Service to list the operations to run.
Operation To Execute	Operations defined within the WSDL file.
Input Parameters	A comma-delimited list of name-value pairs provided to the Web Service. The operation and parameters that you list must be defined in the WSDL. For example, `GetStock.price=10`

Property	Description
Marshaller Class Name	Class name of the marshaller.
Marshaller Properties	Key-value pairs to pass as properties of the marshaller.

Creating a Responder

1. On the **Design** tab, click the **Responders** view.
2. From the **Actions** menu in the upper-right corner of the view, select **New**.

   The **New Responder** dialog box appears.
3. Enter the responder details and configuration details.
4. If you do not have the dependent objects for the responder, you can create the objects using the workspace on the left hand side of the Responder dialog box.
5. If you do not have a related connection for the responder, create the connection.

   For information on how to create a connection, see "Creating a Connection" on page 42.
6. Click **Save**.

   The responder is created and is in Draft state. If you do not create a response associated with the responder, the responder will be in invalid state. You must create a response and associate the response to the responder before you deploy the responder.
7. To deploy the responder and the associated objects, click **Save and Deploy**.

   The responder and associated objects are in Deployed state.

Deploying a Responder

To deploy a responder, the responder must be in draft state.

1. In the Responder content panel, select the responder you want to deploy.
2. Click **Deploy** from the menu on the right-hand side.

   The **Deploy Responder** dialog box appears.
3. Select the responder name you want to deploy.
4. Click **Save**.

   When you deploy a responder, the response and the connection associated with the responder also gets deployed automatically.

Copying a Responder

1. On the **Design** tab, click the **Responders** view.

2. Select the responder that you want to copy.

   You can select multiple responder to copy.
3. From the **Actions** menu in the upper-right corner of the view, select **Copy**.

   The **Copy Responders** dialog box appears.
4. Enter the name of the responder you want to create as a copy.

   The name of the copied responder must be different from the original topic.
5. Click **Save**.

Editing a Responder

You can edit an existing responder.

1. On the **Design** tab, click the **Responders** view.
2. Select the responder that you want to edit.
3. From the menu on the right-hand corner, select **Edit**.

   The **Edit Responder** dialog box appears.
4. Change the details and properties of the responder you want to edit.
5. Click **Save**.

   The responder is now created and is in Draft state.

   Note: If you edit a deployed responder, click **Save and Redeploy** to deploy the responder and the associated objects. The created source is now in the Deployed state.

Reassigning a Responder

In a high availability configuration, you can reassign responders deployed in different responder engines. For example, you deployed a responder in one responder engine. If you want to switch the responder to another responder engine, you can use the reassign option.

1. On the **Design** tab, click the **Responders** view.
2. In the contents panel, select the responder you want to reassign.

   To reassign a responder, the responder must be in deployed state.
3. From the menu on the right-hand corner, select **Reassign**.
4. Select the Responder Controller to reassign the responder.
5. Click **Save**.

   The responder is reassigned to the selected responder controller.

CHAPTER 9

Working with Responses

This chapter includes the following topics:

Responses Overview

Rules respond to an event condition by invoking a response that you create and then reference in that rule.

For example, you can reference the following types of responses in a rule:

- Email Message. Generic email with default values for the to, from, subject, and body fields.
- RTAM Alert. Generic Real-Time Alert Manager (RTAM) alert with default values for the to, subject, channels, and priority fields.
- IM Message. Generic IM with default values for the screen name and message text fields.

After you create a responder service, you must create a specific response and associate the response with the responder service. Deploying a responder service, automatically deploys the associated response.

When a rule invokes that response, the response implements the service using its specific field values. Additionally, you can override the configuration of the responses by specifying new values in the DRQL of an advanced rule or template. It is not required to have responses configuration in DRQL.

Note: If the response name contains a space, place the response within quotation marks when referencing the response in a rule. If the response name does not contain a space, you do not need the quotation marks. When you reference response parameters in a rule, place the parameter value within quotation marks.

The parameter names in the RulePoint response user interface do not match the parameter names that you define within DRQL syntax.

Email Response

After you have defined the Email responder service, you can create a response that sends an email message.

Email Response Properties

When you create an Email response, you enter the name and the configuration details for the responder through the RulePoint user interface.

The following table describes the configuration properties of an email response:

Property	Description
Name	Name of the response. The response name must be unique.
Description	Optional. Description of the response.
Responder Information	Name and description of the associated Email responder. You must create an Email responder before you create an Email response.
Subject	The subject of the email response.
Reply To	Optional. A comma-delimited list of email addresses to send email response replies.
Body	The body of the email response.
Date	Optional. The date that you want to appear as the email send date. The date must be in the following RFC 2822 format: `yyyy-MM-dd HH:mm:ss Z` For example: `2008-01-01 14:20:30 EST`
CC	Optional. A comma-delimited list of email addresses to send a copy of the response.
Content Type	The content type of the email response. You can specify text or html. These values are not case sensitive. The default value text.
BCC	Optional. A comma-delimited list of email addresses to send a blind copy of the response.
To	A comma-delimited list of email addresses to send the response.
From	The email address from which the email response originates.

Email Response Example

After you define an email responder service, you can create a response that sends an email message.

You can run the response with its default values using the following syntax:

```
WHEN 1 stock s WITH s.symbol="XYZ" THEN EmailStockResponse
```

In this DRQL, *EmailStockResponse* is the name of the response.

If you want to override any of the default values in your response within your rule text, you can specify the changes in individual parameters. The following rule provides the syntax for all of the parameters for this service:

```
WHEN 1 stock s WITH s.symbol="XYZ" THEN EmailStockResponse WITH
to="adoe@address.com,bmith@address.com", subject="stock update for stock ${s.symbol}",
body="${s.symbol} event is available", from="cdoe@address.com",
replyTo="ddoe@address.com", cc="edoe@address.com", bcc="fdoe@address.com",
contentType="html"
```

Event Transformer Response

After you define an Event Transformer responder service, you can create a response that transforms the event output and publishes it back into the RulePoint system as a new event.

Event Transformer Response Properties

When you create an Event Transformer response, you enter the name and the configuration details for the responder through the RulePoint user interface.

The following table describes the configuration properties of an Event Transformer response:

Property	Description
Name	Name of the response. The response name must be unique.
Description	Optional. Description of the response.
Responder Information	Name and description of the associated Event Transformer responder. You must create an Event Transformer responder before you create an Event Transformer response.
Properties	Optional. The topic properties to copy to the event response. To include all of the properties, use the asterisk sign. If you have configured the topic with **Named Properties**, then the properties referenced in the WITH portion of the rule are copied.
Parameters	Optional. A comma-delimited list of name-value pairs provided in the response. For example, `symbol=XYZ, price=10`. It is referenced in a rule as params.

Event Transformer Response Example

You can create a response with its default values using the following syntax:

```
WHEN 1 stock WITH symbol="XYZ" THEN TransformStockResponse
```

In this DRQL, *TransformStockResponse* is the name of the response.

If you want to override any of the default values in the response within the rule text, you can specify the changes in individual parameters. The DRQL in the following rule replaces the parameters for this service:

```
WHEN 1 stock WITH symbol="XYZ" THEN TransformStockResponse WITH params="volume=${volume}"
```

In the following example, an event with topic *matchedstock* is created for the first event of all matched stock events to a news event within a 60 minute window of the most current event (stock or news). The properties of the new event use the symbol and price from the first matched stock event in the window. The new event also contains the total number of stock within the window and the title from the news topic.

```
When  stock, 1 news with stock.symbol in news.title and count(stock.price) as total > 0
slide within 60 minutes then TransformsEvent with topic="stock(1) as matchedstock",
properties="stock.*", params="last_hour_count=${total}, news_title=${news.title}"
```

For another example, this response describes a new event with topic *matchedstock*. It has all stock topic properties in the event. It does not contain news properties. The new event with topic *matchedstock* is created; it has title and reference from the news topic. The property *title* is renamed to *reference_title*. It does not contain stock properties.

```
WHEN sensor group by sensor.id,sensor.status WITH sensor.status ='down'
ANDcount(sensor.status) = 5 slide within 5 minutes then TransformsEvent with
topic="sensor(4:5) as FailedSensor", properties="*", detection_time="$
{drql_timestamp(rel), date hh:mm:ss.SSSSS}"
```

In the following example, the rule gathers all sensor events with the same sensor identifier and status within five minutes. If five sensor events in that time period have a status of 'down', then events with topic Failed Sensor are created for the last two matched sensor events. The detection_time property is set to the timestamp when each sensor event is consumed by RulePoint using the date format to include milliseconds. The property selector relative 'rel' selects the timestamp property relative to transformed sensor event.

```
WHEN sensor WITH match(sensor.id) AND match(sensor.status) AND sensor.status ='down' AND
count(sensor.status) = 5 slide within 5 minutes then TransformsEvent with
topic="sensor(4:5) as Failed Sensor", properties="*", detection_time="$
{drql_timestamp(rel), date hh:mm:ss.SSSSS}"
```

File Output Response

After you define the File Output response, you can create a response that responds to an event by writing to a text file.

File Output Response Properties

The following table describes the configuration properties of a File Output response:

Property	Description
Name	Name of the response. The response name must be unique.
Description	Optional. Description of the response.
Responder Information	Name and description of the associated File Output responder. You must create a File Output responder before you create a File Output response.
Path Name	The path to a file to be written to by the service. This path should be absolute.

Property	Description
Mode	The default mode is append. The options are append or overwrite. The overwrite option writes the full text to the file each time as opposed to adding only the latest addition.
Contents	The data or text to be written to the file.

File Output Response Example

In the following example, assuming Stock stock has a symbol XYZ. The contents *Buy XYZ* is added to the file. You can run the response with its default values using the following syntax:

```
WHEN 2 stock THEN FileOutputDemo with Contents="Buy ${symbol}", mode="append"
```

In this DRQL, *FileOutputDemo* is the name of the response.

The following example shows how to create and add the symbol, price, volume, and the last trading time of a stock to a file:

```
WHEN 1 stock THEN FileOutputDemo with file="/stocks/${symbol}.txt", contents="At closing
time ${lastTradeTime}, the price of Stock ${symbol} was ${price} and volume was $
{volume}", mode="append"
```

HTTP Response

After you define the HTTP responder service, you can create and associate an HTTP response that performs an HTTP GET from or POST to a specified URL.

HTTP Response Properties

When you create an HTTP response, you enter the name and the configuration details for the response through the RulePoint user interface.

The following table describes the configuration properties of a HTTP response:

Property	Description
Name	Name of the response. The response name must be unique.
Description	Optional. Description of the response.
Responder Information	Name and description of the associated HTTP responder. You must create an HTTP responder before you create an HTTP response.
Content Type	Optional. The HTTP character set and content type to be used. If left blank, the default is ISO-8859-1 for the character set, and text or plain for the Content Type. Ensure that the HTTP server to which you are connecting supports and accepts one or both. To set the content type, enter text or html, text or xml into the field. To set the content type and character set, use the following format: `text/xml;charset=ISO-8850-1`

Property	Description
Target URL	The URL of the object.
User Name	Optional. The user name used to access the HTTP server.
Password	Optional. The password associated with the user name used to access the HTTP server.
Method	The method that you want to perform on the object specified in the URL field.
Message Body	Optional. If you post data using the content type multipart or mime-encoded, the form data. If you use the body parameter, it ignores any specified key-value parameter pairs.
Parameters	Optional. A comma-delimited list of key-value pairs provided in the response. For example, `symbol=XYZ, price=10` are key-value pairs.

HTTP Response Example

After you define an HTTP Responder service, you can create a response that performs an HTTP GET from or POST to an URL.

You can run the response with its default values using the following syntax:

```
WHEN 1 stock s WITH s.symbol="XYZ" THEN HTTPStockResponse
```

In this DRQL, *HTTPStockResponse* is the name of the response.

If you want to override any of the default values in your response within your rule text, you can specify those changes in individual parameters. The following rule provides the syntax for all of the parameters for this service:

```
WHEN 1 stock s WITH s.symbol="XYZ" THEN HTTPStockResponse WITH symbol="${s.symbol}",
price="${s.price}", url="http://localhost/submitStockData.html", method="POST",
userName="username", password="password", contentType="text/plain; charset=ISO-8859-1",
body="Hello World"
```

Note: The following keywords are restricted to use as response property name and the named properties for this response type:

- Response property names: target, topic, icon
- Named properties for this service: URL, method, user name, password, content type, body

Instant Messaging (Jabber or XMPP) Response

After you define the Instant Messaging responder service, you can create a response that sends an instant message.

Instant Messaging (Jabber or XMPP) Response Properties

When you create an Instant Messaging response, you enter the name and the configuration details for the response through the RulePoint user interface.

The following table describes the configuration properties of an Instant Messaging response:

Property	Description
Name	Name of the response. The response name must be unique.
Description	Optional. Description of the response.
Responder Information	Name and description of the associated Instant Messaging responder. You must create a Instant Messaging responder before you create a Instant Messaging response.
To	The user name of the instant messaging account recipient.
Message Body	The body of the instant message.
Subject	The subject of the instant message.

Instant Messaging Response Example

You can run the response with its default values using the following syntax:

```
WHEN 1 stock WITH symbol="XYZ" THEN IMStockResponse
```

In this DRQL, *IMStockResponse* is the name of the response.

If you want to override any of the default values in the response within the rule text, you can specify the changes in individual parameters. The following rule provides the syntax for all of the parameters for this service:

```
WHEN 1 stock WITH symbol="XYZ" THEN IMStockResponse WITH to="yourhandle", subject="stock
update", body="${symbol} event is available"
```

JMS Response

After you define a JMSResponder service, you can attach a JMS Response to it. You can create a JMS Response to send a JMS message to the destination configured in the responder.

JMS Response Properties

When you create a JMS response, you enter the name and the configuration details for the response through the RulePoint user interface.

The following table describes the configuration properties of a JMS response:

Property	Description
Name	Name of the response. The response name must be unique.
Description	Optional. Description of the response.
Responder Information	Name and description of the associated JMS responder. You must create a JMS responder before you create a JMS response.
Priority	Priority of the message. Priority values range from 0 to 9. A lower priority number implies lower priority.
Expiration Time	Message expiration time, in milliseconds.
Parameters	Message parameters.
Body	Message body.

JMS Response Example

You can run the response with its default values by using the following syntax:

```
WHEN 1 stock WITH symbol="XYZ" THEN JmsResponse
```

In this DRQL, JmsResponse is the name of the response.

If you want to override any of the default values in the response within the rule text, you can specify the changes in individual parameters. The following rule provides the syntax for all of the parameters for this service:

```
WHEN 1 stock WITH symbol="XYZ" THEN JmsResponse WITH destination="jmsQueue", body="$
{symbol} event is available"
```

RTAM Response

After you define a RTAM responder, you can create a response to send an RTAM alert.

To receive RTAM alerts, the designated recipient users must exist in the RTAM database to store their alerts. Initial user login requires authentication.

RTAM Response Properties

When you create a RTAM response, you enter the name and the configuration details for the response through the RulePoint user interface.

The following table describes the configuration properties of a RTAM response:

Property	Description
Name	Name of the response. The response name must be unique.
Description	Optional. Description of the response.
Responder Information	Name and description of the associated RTAM responder. You must create a RTAM responder before you create a RTAM response.
To	The user name of the RTAM user who would receive the alert.
Groups	Optional. A group address to receive the Response message as defined by your RTAM administrator.
Subject	The subject of the alert.
Body	The body of the alert. You can format the body as text or using simple HTML tags, such as <B>, <I>, <UL>, .
Actions	Optional. A semicolon-delimited list of comma-delimited name-link pairs for actions to show in the RTAM interface for this alert. For example, ActionName1, ActionLink1; ActionName2, ActionLink2
Channels	Optional. A semicolon-delimited list of channels to deliver the alert to. The default value is blank.
Header	Optional. The header information for the alert, which the user can view in RTAM.
Metadata	Additional information or metadata.
Message Priority	The priority to place on the RTAM alert. Different priorities function differently within RTAM. The range in ascending order is 0 to 5, 5 being the highest priority. The priority 0 indicates that the message is not yet prioritized.

RTAM Response Example

You can run the response with its default values using the following syntax:

```
WHEN 1 stock s WITH s.symbol="XYZ" THEN RTAMStockResponse
```

In this DRQL, *RTAMStockResponse* is the name of the response.

If you want to override any of the default values in the response within the rule text, you can specify the changes in individual parameters. The following rule provides the syntax for all of the parameters for this service:

```
WHEN 1 stock s WITH s.symbol="XYZ" THEN RTAMStockResponse WITH to="username",
groups="Yourcompany_HR", subject="Found stock with symbol: ${symbol}", body="The stock
symbol <B>${symbol}</B> has been found in this stock quote.", actions="Stock
information, ${StockHistoryURL};Total volume, http://stockvolume.com/q=${symbol}",
channels="Stock", header="Stock Alert", priority="3"
```

SQL Response

After you define the new SQL responder service, you can create and associate a SQL response that executes an SQL command.

SQL Response Properties

When you create an SQL response, you enter the name and the configuration details for the response through the RulePoint user interface.

The following table describes the configuration properties of an SQL response:

Property	Description
Name	Name of the response. The response name must be unique.
Description	Optional. Description of the response.
Responder Information	Name and description of the associated SQL responder. You must create an SQL responder before you create an Email response.
SQL	The SQL query for the responder. Use double angle brackets (chevrons) surrounded column names (<<column name>>) to identify the substitution parameters in the SQL statement. The data values provided in the response replaces the parameters with the matching parameter names.
Retry Count	The number of connection retry attempts by RulePoint. The default value is 1. Setting this field to zero indicates that RulePoint must not retry to connect.
Retry Delay	The amount of time, to wait between connection attempts. The default value is 5000 and the unit is in milliseconds.

SQL Response Example

After you define an SQL Responder service, you can create a response that processes an SQL command.

You can run the response with its default values using the following syntax:

```
WHEN 1 stock WITH symbol="XYZ" THEN SQLStockResponse
```

In this DRQL, `SQLStockResponse` is the name of the response.

If you want to override all default parameters in your response within the rule text, you can specify the parameters or name value pairs for changes in individual parameters. The following rule provides the syntax for overriding all of the parameters for the SQL responder service:

```
WHEN 1 stock WITH symbol="XYZ" THEN SQLStockResponse WITH params="name=$
{stock.symbol},value=${stock.price}"
```

If you want to override a specific parameter in your response within your rule text, you can specify that parameter as a response property. The following rule provides the syntax for overriding a specific parameter value for this service:

```
WHEN 1 stock WITH symbol="XYZ" THEN SQLStockResponse WITH value="${stock.price}"
```

In another example, create the following SQL statement using name value pair:

```
INSERT into Stock (name, value) values(<<name>>,<<value>>)
```

You can use the following parameters to populate the SQL statement:

```
name =${stock.symbol}, value=${stock.price}
```

Note: All undefined parameters are set to the value of empty string (which is NULL in Oracle). SQL statements must handle the format of the parameter value provided at the substitution point in the statement.

Using Brackets [] to Substitute IN Clause Parameters for an SQL Statement

The brackets around a comma-separated list is a formatting feature that you can use to update the IN clause of the parameters of a responder or response. You can use the feature to update the parameters by a comma-separated list in brackets, [XYZ, INFA, ABC].

1. Create an SQL responder.

   ```
   UPDATE stock set processed_ind='A' WHERE name in <<names>>
   ```

2. Create and associate an SQL response with the responder.

   ```
   updateresult
   ```

3. When the events `event1: symbol=XYZ`, `event2: symbol=INFA`, and `event3: symbol=ABC` come into the Stock table, the following rule triggers:

   ```
   WHEN 3 stock s THEN updateresult  WITH names="[ ${s.symbol(*,COMMA)}]"
   ```

Using a Raw List to Substitute IN Clause Parameters for an SQL Statement

The raw list is a formatting feature that allows you to update the SQL Responder or the SQL Response. You can use the feature to update the parameters by a list of stock symbols.

1. Create an SQL responder.

   ```
   UPDATE stock set processed_ind='A' WHERE name in <<names>>
   ```

2. Create and associate an SQL response with the responder.

   ```
   updateresult
   ```

3. When the events `event1: symbol=XYZ`, `event2: symbol=INFA`, and `event3: symbol=ABC` come into the Stock table, the following rule triggers:

   ```
   WHEN 3 stock s THEN updateresult  WITH names=" ${s.symbol(raw )}"
   ```

Using the rule below, when the following events come into **stock1**: event1: symbol =IBM, event2: symbol=INFA, and event3: symbol=ABC, the rule fires.

For more information about other RulePoint formatting codes, such as NL and COLON, see *Formatting_the_Output_of_your_Response*.

Watchlist Response

After you define a Watchlist responder, you can create a response that performs action against a watchlist.

Watchlist Response Properties

When you create a Watchlist response, you enter the name and the configuration details for the response through the RulePoint user interface.

The following table describes the configuration properties of a Watchlist response:

Property	Description
Name	Name of the response. The response name must be unique.
Description	Optional. Description of the response.
Responder Information	Name and description of the associated Watchlist responder. You must create a Watchlist responder before you create a Watchlist response.
Watchlist Name	The name of the watchlist on which to operate. The DRQL syntax references the watchlist name as objectName.
Operation	The type of operation. Choose from the following options: - add. Adds a value to the watchlist. - remove. Removes a value from the watchlist. - change. Changes an existing value in the watchlist. - replace. Replaces all of the values in the watchlist. If you specify an action other than any action in the preceding list, RulePoint substitutes that action with add.
Old Value	Optional. If the operation is remove or change, the value on which to operate.
New Value	Optional. If the operation is add, change, or replace, the new value replaces the value in the **Old Value** field.

Watchlist Response Example

You can run the response with its default values using the following syntax:

```
WHEN 1 stock with symbol="XYZ" THEN WatchlistStockResponse
```

In this SQL query, *WatchlistStockResponse* is the name of the response.

The following rule provides the syntax for adding a value to the watchlist:

```
WHEN 1 stock s WITH s.symbol="XYZ" THEN WatchlistStockResponse WITH objectName="Stock",
action="add", newValue="${symbol}"
```

The following rule provides the syntax for deleting a value from the watchlist:

```
WHEN 1 stock s WITH s.symbol="XYZ" THEN WatchlistStockResponse WITH objectName="Stock",
action="remove", oldValue="${symbol}"
```

The following rule provides the syntax for changing an existing value in the watchlist:

```
WHEN 1 stock s WITH s.symbol="XYZ" THEN WatchlistStockResponse WITH objectName="Stock",
action="change", oldValue="ABCD", newValue="${symbol}"
```

The following rule provides the syntax for replacing all of the values in the watchlist:

```
WHEN 1 stock s WITH s.symbol="XYZ" THEN WatchlistStockResponse WITH objectName="Stock",
action="replace", newValue="${symbol}"
```

Note: The watchlist must exist before the watchlist response can change it.

Web Service Response

After you have defined the new service, you can create a response that will send a formatted SOAP message to execute a remote WSDL service.

Web Service Response Properties

The following table describes the configuration properties of a Web Service response:

Property	Description
Name	Name of the response. The response name must be unique.
Description	Optional. Description of the response.
Responder Information	Name and description of the associated Web Service responder. You must create an Web Service responder before you create an Web Service response.
Operation To Execute	Operations defined within the WSDL file.
Service	Web service to which the request is to be sent.
Parameters	A comma-delimited list of key=value pairs provided to the Web Service. The operation and parameters that you list must be defined in the WSDL. For example: GetStock.price=10

Web Service Response Example

You can run the response with its default values using the following syntax:

```
WHEN 1 stock WITH symbol="XYZ" THEN WDSLStockResponse
```

In this SQL query, *WDSLStockResponse* is the name of the response.

If you want to override any of the default values in the response within the rule text, you can specify those changes in individual parameters. The following rule provides the syntax for all of the parameters for this service:

```
WHEN 1 stock WITH symbol="XYZ" THEN WDSLStockResponse WITH GetStock.price="${s.price}"
```

Ultra Messaging Response

After you have defined the Ultra Messaging (UM) responder service, you can create a response that sends a UM message. Rules respond to an event condition by invoking a response that you create and then reference in that rule.

Ultra Messaging Response Properties

When you create a UM response, you enter the name and the configuration details for the responder through the RulePoint user interface.

The following table describes the configuration properties of a UM response:

Property	Description
Name	Name of the UM response. The response name must be unique.
Description	Optional. Description of the UM response.
Responder	Name and description of the associated UM responder, which has the configured destination topic, the message body, and the message type. You must create a UM responder before you create an UM response.
Message Body	The intended UM message body.
Destination Topic	The topic name in the UM receiving application that will receive the processed UM native messages.
Session ID	A 64-bit value that uniquely identifies UM receivers with unique topics. Set to -1 if you want to disable the session ID.

Creating a Response

1. On the **Design** tab, click the **Responses** view.
2. From the **Actions** menu in the top-right corner of the view, select **New**.

   The **Create Response** dialog box appears.
3. Enter the name and description for the response.
4. Select the **Responder Information** you want to associate the response to.
5. Enter the response properties.

   The response properties are similar to the selected responder properties.
6. Click **Save**.

Copying a Response

1. On the **Design** tab, click the **Response** view.
2. Select the response that you want to copy.

   You can select multiple response to copy.
3. From the **Actions** menu in the top-right corner of the view, select **Copy**.

   The **Copy Responses** dialog box appears.
4. Enter the name of the response you want to create as a copy.

   The name of the copied response must be different from the original topic.
5. Click **Save**.

Editing a Response

You can edit the properties of an existing response. You can change the name of the response that you want to edit. Any change that you make to a response affects all rules that reference the response.

1. On the **Design** tab, click the **Responses** view.
2. In the Response content panel, select the response name that you want to edit.
3. From the menu in the right-hand side, select **Edit**.

   The **Edit Response** dialog box appears.
4. Change the details and properties of the response.
5. Click **Save**.
6. If you edit a deployed response that has associated primary objects, you can choose to save the response. Perform the following tasks:
   a. Click **Save**.

      A message appears displaying the name and the type of primary objects that reference the response.
   b. Click **Continue**.

      A success message appears, stating that the response is updated successfully.
   c. Click **OK**.

      The state of the response and the objects that reference it changes from Deployed to Needs_Deployment state.
   d. If you want to redeploy the objects, click the specific response in the Responses content panel, and then click **Update RunTime** from the menu.

      The **Dependent Objects** dialog box displays the list of primary objects that will be redeployed.
   e. Click **Continue**.

      The preview message displays successful redeployment. If the response is referenced in a rule, the message prompts you if you want to trace the rules. If there are invalid objects, fix them, and click **Continue**.

      The state of the objects changes from Needs_Deployment state to Deployed state.

f. Click **OK** to trace the rules, or **Cancel**.

7. If you edit a deployed response that has associated primary objects, you can also choose to simultaneously save and deploy the referenced primary objects. Perform the following tasks:

   a. Click **Save and Update**.

      The **Dependent Objects** dialog box displays the objects that will be redeployed.

   b. Click **Continue**.

      The preview message displays successful redeployment. If the response is referenced in a rule, the message prompts you if you want to trace the rules. If there are invalid objects, fix them, and click **Continue**.

      The state of the objects changes from Needs_Deployment state to Deployed state.

Deleting a Response

You can delete an existing response.

1. On the **Design** tab, click the **Responses** view.
2. In the Responses content panel, select the response name that you want to delete.
3. From the menu on the right-hand side, select **Delete**.

   A message prompts you to verify if you want to delete the response.
4. Click **OK** to delete the response.

Viewing Related Objects

You can view the related objects of an existing response. The related objects are the primary objects that references to the response.

1. On the **Design** tab, click the **Responses** view.
2. In the contents panel, select the response.
3. From the menu in the right-hand side, select **View Related Objects**.

   The **Related Objects** dialog box appears. You can view the list of responders and rules that references the response.

CHAPTER 10

Working with Watchlists

This chapter includes the following topics:

Watchlist Overview

Watchlists contain the items that you store as a single object with a unique name that you define. You can reference this name in a rule so that it can use the data stored in the object. You can update a watchlist manually. You can also update a watchlist dynamically when the results of a condition matches a watchlist response.

Watchlists are useful because you can change the items within the watchlist at any time, and any rule referencing that watchlist automatically uses those new items. Start modifying a small watchlist, and progressively add to it after you have a good understanding of the various configuration options.

For example, if you want to create multiple rules regarding the stock portfolio, you can create a watchlist containing all the stock symbols in the current portfolio. While creating rules, you can reference the watchlist instead of specifying each individual stock symbol in multiple rules. In the future, if your stock portfolio changes, you can change the watchlist instead of changing each rule.

Watchlist Types

Watchlists can be of two types, text or list. The type of watchlist that you choose depends on the items in your watchlist and how you want RulePoint to evaluate those items.

Text

A text watchlist contains a single item or a single text string. The following is an example of a text watchlist:

```
ABCD

Informatica, situated in Bangalore, releases Yosmite RulePoint
```

List

A list watchlist contains a group of individual items separated by line breaks. For example:

```
ABCD
EFG
XYZ
```

Creating a Watchlist

1. On the **Design** tab, click the **Watchlists** view.
2. From the **Actions** menu in the top-right corner of the view, select **New**.

   The **New Watchlist** dialog box appears.
3. Enter values for the watchlist details.

   The following list describes the watchlist properties that you need to enter:

Property	Description
Name	Name of the watchlist. The watchlist name must be unique.
Description	Optional. Description of the watchlist.
Type	Select the watchlist type.
Content	The text, for a watchlist of type Text, or the list of items separated by line breaks, for a watchlist of type List.

4. Click **Save**.

Copying a Watchlist

1. On the **Design** tab, click **Watchlists**.
2. Select the watchlist that you want to copy.

   You can select multiple watchlists.
3. From the **Actions** menu in the top-right corner of the view, select **Copy**.

   The **Copy Watchlists** dialog box appears.
4. In the Copied Watchlist column, type the name of the watchlist you want to create as a copy.

   The name of the copied watchlist must be different from the original watchlist.
5. Click **Save**.

Editing a Watchlist

You can edit the properties of an existing watchlist.

1. On the **Design** tab, click the **Watchlists** view.
2. In the contents panel, select the watchlist that you want to edit.
3. From the menu in the right-hand corner, select **Edit**.

   The **Edit Watchlist** dialog box appears.
4. Change the details and properties of the watchlist.
5. Click **Save**.
6. If you edit a deployed watchlist that has associated primary objects, you can choose to save the watchlist. Perform the following tasks:
   a. Click **Save**.

      A message appears displaying the name and the type of primary objects that reference the watchlist.
   b. Click **Continue**.

      A success message appears, stating that the watchlist is updated successfully.
   c. Click **OK**.

      The state of the watchlist and the objects that reference it changes from Deployed to Needs_Deployment state.
   d. If you want to redeploy the objects, click the specific watchlist in the Watchlists content panel, and then click **Update RunTime** from the menu.

      The **Dependent Objects** dialog box displays the list of primary objects that will be redeployed.
   e. Click **Continue**.

      The preview message displays successful redeployment. If the watchlist is referenced in a rule, the message prompts you if you want to trace the rules. If there are invalid objects, fix them, and click **Continue**.

      The state of the objects changes from Needs_Deployment state to Deployed state.
   f. Click **OK** to trace the rules, or **Cancel**.
7. If you edit a deployed watchlist that has associated primary objects, you can also choose to simultaneously save and deploy the referenced primary objects. Perform the following tasks:
   a. Click **Save and Update**.

      The **Dependent Objects** dialog box displays the objects that will be redeployed.
   b. Click **Continue**.

      The preview message displays successful redeployment. If the watchlist is referenced in a rule, the message prompts you if you want to trace the rules. If there are invalid objects, fix them, and click **Continue**.

      The state of the objects changes from Needs_Deployment state to Deployed state.

Deleting a Watchlist

You can delete an existing watchlist. If you attempt to delete a watchlist that is currently referenced in a rule, RulePoint returns an error listing all of the rules that references this watchlist. Rulepoint does not delete the watchlist until you edit or delete the rules listed.

1. On the **Design** tab, click the **Watchlists** view.
2. In the contents panel, select the watchlist that you want to delete.
3. From the menu in the right-hand corner, select **Delete**.

   A message prompts you to verify if you want to delete the watchlist.
4. Click **OK**.

Viewing Related Objects

You can view the related objects of an existing watchlist. The related objects are the primary objects that references to the watchlist.

1. On the **Design** tab, click the **Watchlists** view.
2. In the contents panel, select the watchlist.
3. From the menu in the right-hand corner, select **View Related Objects**.

   The **Related Objects** dialog box appears. You can view the list of rules that references the watchlist.

Referencing Watchlists in a Rule

To reference a watchlist in a rule, you must specify the exact name of the watchlist when defining your rule conditions.

To reference a watchlist in a rule, you specify the watchlist using the following syntax:

```
watchlist:[WatchlistName]
```

If a watchlist name contains a space, you must place the watchlist name inside double quotes. For example, `watchlist:"My Stock List"`.

The following DRQL shows the usage of a watchlist:

```
WHEN 1 stock s WITH watchlist:myStocks contains s.symbol
```

CHAPTER 11

Working with Analytics

This chapter includes the following topics:

Analytics Overview

Analytics analyze data within a system and implements a data processing function.

You can reference an analytic in rules to determine whether to activate a rule or not. Analytics can provide simple functions such as math, text, date functions, or more advanced functions such as connecting to a remote server and performing data processing before returning.

Analytic Types

Analytics are of two types. You can create predefined analytics and configurable analytics. You can create instances of configurable analytics.

The configurable analytics include SQL, Web Service, Hash, AddToDate, and SubtractFromDate analytics.

SQL Analytic

The SQL analytic runs an SQL query or command against a target database.

When you run SQL commands and queries as an analytic, you can enrich the data to provide additional information during rule processing that is not available in the event data. An SQL analytic, when used in a rule, requires an SQL connection, so create an SQL connection before you create the SQL analytic.

The following table defines the properties of an SQL Analytic:

Property	Description
Name	The name of the SQL analytic. This must be a unique name.
Description	Optional. The description of the analytic.
Type	Select **SQL Analytic**.
Connection	Select the connection type from the list of available connections.
SQL Query	The SQL query to run against the target database. Use question marks to identify the substitution parameters in the SQL statement that will be replaced by the data values provided in the rule.
Condition Evaluation Required	Specifies whether the analytic is part of conditional evaluation within a rule. Set to 'false' if the result of the analytic can be used only for enrichment. The default value is True.
Cache Duration	Time period, in seconds, for which the result of the analytic is to be cached.

Example

Consider a transaction based SQL Service that you can use to calculate the total number of transactions for a particular merchant.

The following data represents the information that you would provide when creating an SQL analytic called *TransactionCount* to use as an analytic in the rule:

- Name. *TransactionCount*
- Description. Returns the total number of transactions made with a specific merchant, with the merchant name and purchase date as the input parameters.
- Connection. Choose the created SQL connection.
- SQL Query. `SELECT count(*) FROM creditcard.purchase where Merchant_Name=? and Purchase_Date < ?`
- Condition Evaluation Required: Select True and False.
- Cache Duration: Total number of milliseconds for which the analytic result will be cached.

The following rule uses the *TransactionCount* SQL analytic to determine the total number of transactions for a particular merchant before a specified date. If the *TransactionCount* is greater than 40 transactions, then respond appropriately.

```
WHEN 1 transaction WITH TransactionCount( Merchant_Name , Purchase_Date ) as result > 40
then response with body = "${result}"
```

The order of arguments of the analytic must match the position of the question marks.

Web Services Analytics

The Web Service analytic sends a formatted SOAP message to run a remote Web Services (WSDL) operation and returns the result.

Running a Web Service as an analytic is useful for enriching the data to provide additional information during rule processing that is not available in the event data. To use the Web Service analytic in a rule, you must create a Web Service and then reference that service in the rule.

The following table defines the properties of a Web Services analytic:

Property	Description
Name	The name of the Web Services analytic. This must be a unique name.
Description	Optional. The description of the analytic.
Type	Select **Webservice Analytic**.
Connection	Select the connection name that you want to associate the responder to. You must create a web service connection before you create a web service analytic.
Service	Select the service to list the operations from the list of available services.
Operation to Execute	Select the Web Services operation to run.
Parameter List	Optional. List of parameters for the selected Web Services operation.
XPath Expression	If you specify an XPath expression in the analytic, you get either a single value or a list of many text values in return. If you do not specify any XPath expression, you get the entire XML document in return. For the output xml to change, you must clear the XPath expression and enter it again.
Condition Evaluation Required	Specifies whether the result of the analytic is part of conditional evaluation within a rule. Set to 'false' if the result of the analytic can be used only for enrichment. The default value is True.
Cache Duration	Time period, in seconds, for which the result of the analytic is to be cached.

Example

Consider a subscription-based Web Service that you can use to convert temperature from one unit (degree Celsius) to another unit (degree Fahrenheit). Here, the required values are the temperature value, the source unit, and the target unit . You can define the order of the values in the **Parameter List** field. To use this analytic in a rule, the rule passes the value of temperature to convert, the source unit, and the target unit, and then retrieves the converted temperature in the target unit as the result.

To use the Web Service analytic, complete the following tasks:

1. Create the Web Service connection. The following URL is an example: http://www.webservicex.net/ConvertTemperature.asmx?WSDL

   Note: Openly available webservice URL might not work.
2. Create a Web Service with an appropriate name.
3. Choose the type as Web Service analytic and select the connection. Selecting the connection will provide the operation to execute.

4. Provide the following parameter list:
   ConvertTemp.Temperature,ConvertTemp.FromUnit,ConvertTemp.ToUnit
5. Following is the XPath expression:
   /*/soap:Body/ConvertTempResponse/ConvertTempResult
6. Condition Evaluation Required: Select True and False.
7. Cache Duration: Total number of milliseconds for which this analytic result is to be cached.

The following rule uses the web service analytic:

```
when 1 tempConvert1 tc with tempwsanalytic1(temperature, degreeCelsius, degreeFahrenheit) as result != "" then response with body = "${result}"
```

Hash Analytic

When you run a hash analytic, you use cryptographic algorithms that map each data input to a distinct hash value.

Hashing converts original data of a variable length to return a hash value of a fixed length based on the algorithm you use. The returned hash value maps to the original data. The hash value is used for further processing instead of the actual value. Supported cryptographic algorithms in RulePoint that provide the hashing function are MD2, MD5, SHA-1, SHA-256, SHA-384, and SHA-512.

The following table describes the properties of a hash analytic:

Property	Description
Name	The name of the Hash analytic. This must be a unique name.
Description	Optional. The description of the analytic.
Type	Select **Hash Analytic**.
Hash Type	The algorithm type: - MD2 - MD5 - SHA-1 - SHA-256 - SHA-384 - SHA-512
Salt Value	A random value that you need to add to generate the hashed value.
Conditional Evaluation Required	Select 'true' if the result of this analytic is part of a conditional evaluation within a rule. Select 'false' if the result of this analytic can only be used for enrichment. Default is True.
Cache Duration	Total number of seconds that the result of the hash analytic will be kept cached. Default is 0.

Example

Consider a property value of an event, with string (str1) and integer (int1) as the property values of a source that you want to convert to hash analytic values. Create a hash analytic using the hash algorithm type from the user interface, and then reference the hash analytic in the rule.

To create a hash analytic, complete the following tasks:

1. Choose the hash algorithm type, such as MD2, MD5, SHA-1, SHA-256, SHA-384, or SHA-512.

2. Provide a salt value, if required.
3. Select Condition Evaluation Required to True or False.
4. Provide the Cache Duration for which the analytic result will be cached.

To convert event property values into hash values, you can create the following hash analytic rule using any one of the supported hash analytic algorithms:

```
when 1 hashtopic m with hashanalytic("hello",100) as result !="" and list2vars(result,
stringhashvalue, numberhashvalue) then response with body= "value of hello value is $
{ stringhashvalue } and value of number value is  ${numberhashvalue}"
```

AddToDate Analytic

Use the AddToDate analytic to add years, months, days, hours, minutes, or seconds to a given date.

The following table defines the properties of an AddToDate analytic:

Property	Description
Name	The name of the AddToDate analytic. This must be a unique name.
Description	Optional. The description of the analytic.
Type	Select AddToDate Analytic.
Unit	Select years, months, days, hours, minutes, or seconds.
Condition Evaluation Required	Specifies whether the result of the analytic is part of conditional evaluation within a rule. Set to "false" if the result of the analytic can be used only for enrichment. Default is True.
Cache Duration	Time period, in seconds, for which the result of the analytic will be cached.

Example

Create an AddToDate analytic of name "add2date" and choose the units as "years."

Note: The date format must be in Java SimpleDateFormat.

```
when 1 addDate ad  with add2date(ad.datevalue,ad.dateformat,ad.count)  as  dateadd  =
"12/01/2038" then  response with body = "value of date add ${dateadd}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
ad.datevalue= 12/01/2013 ad.dateformat= MM/dd/yyyy* ad.count=25	Adds 25 years to the date.	2/01/2038

SubtractFromDate Analytic

Use the SubtractFromDate analytic to subtract years, months, days, hours, minutes, or seconds from a given date.

The following table defines the properties of the SubtractFromDate analytic:

Property	Description
Name	The name of the SubtractFromDate analytic. This must be a unique name.
Description	Optional. The description of the analytic.
Type	Select SubtractFromDate Analytic.
Unit	Select years, months, days, hours, minutes, or seconds.
Condition Evaluation Required	Specifies whether the result of the analytic is part of conditional evaluation within a rule. Set to "false" if the result of the analytic can be used only for enrichment. Default is True.
Cache Duration	Time period, in seconds, for which the result of the analytic will be cached.

Example

Create a SubtractFromDate analytic of name "subfrmdate" and choose the units as "years."

Note: The date format must be in Java SimpleDateFormat.

```
when 1 subDate sd with subfrmdate(sd.datevalue,sd.dateformat,sd.count) as datesub =
"12/01/1988" then response with body = "value of date add ${ datesub}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
ad.datevalue= 12 /01/2013 ad.dateformat= MM/dd/yyyy* ad.count=25	Subtracts 25 years from the date.	12/01/1988

Creating an Analytic

You can create an analytic of type SQL or Web Services through the RulePoint user interface.

1. On the **Design** tab, click the **Analytics** view.
2. From the **Actions** menu in the top-right corner of the view, select **New**.

   The **New Analytic** dialog box appears.
3. Enter the analytic name, description, analytic type, connection, and the configuration details.
4. Click **Save**.

Copying an Analytic

1. On the **Design** tab, click the **Analytics** view.
2. Select the analytic that you want to copy.

   You can select multiple analytics to copy.
3. From the **Actions** menu in the top-right corner of the view, select **Copy**.

   The **Copy Analytic** dialog box appears.
4. Enter the name of the analytic you want to create as a copy.

   The name of the copied analytic must be different from the original analytic.
5. Click **Save**.

Editing an Analytic

You can edit the properties of an existing analytic. You can also rename of the analytic that you want to edit. Any change that you make to an analytic affects all rules that reference the analytic.

1. On the **Design** tab, click the **Analytics** view.
2. In the contents panel, select the analytic that you want to edit.
3. From the menu in the right-hand corner, select **Edit**.

   The **Edit Analytic** dialog box appears.
4. Change the details and properties of the analytic.
5. Click **Save**.
6. If you edit a deployed analytic that has associated primary objects, you can choose to save the analytic. Perform the following tasks:
   a. Click **Save**.

      A message appears displaying the name and the type of primary objects that reference the analytic.
   b. Click **Continue**.

      A success message appears, stating that the analytic is updated successfully.
   c. Click **OK**.

      The state of the analytic and the objects that reference it changes from Deployed to Needs_Deployment state.
   d. If you want to redeploy the objects, click the specific analytic in the Analytics content panel, and then click **Update RunTime** from the menu.

      The **Dependent Objects** dialog box displays the list of primary objects that will be redeployed.
   e. Click **Continue**.

      The preview message displays successful redeployment. If the analytic is referenced in a rule, the message prompts you if you want to trace the rules. If there are invalid objects, fix them, and click **Continue**.

      The state of the objects changes from Needs_Deployment state to Deployed state.
   f. Click **OK** to trace the rules, or **Cancel**.

7. If you edit a deployed analytic that has associated primary objects, you can also choose to simultaneously save and deploy the referenced primary objects. Perform the following tasks:
   a. Click **Save and Update**.

      The **Dependent Objects** dialog box displays the objects that will be redeployed.
   b. Click **Continue**.

      The preview message displays successful redeployment. If the analytic is referenced in a rule, the message prompts you if you want to trace the rules. If there are invalid objects, fix them, and click **Continue**.

      The state of the objects changes from Needs_Deployment state to Deployed state.

Deleting an Analytic

You can delete an existing analytic.

1. On the **Design** tab, click the **Analytics** view.
2. In the contents panel, select the analytic that you want to delete.
3. From the menu in the right-hand corner, select **Delete**.

   A message prompts you to verify if you want to delete the analytic.
4. Click **OK** to delete the analytic.

Viewing Related Objects

You can view the related objects of an existing analytic. The related objects are the primary objects that references to the analytic.

1. On the **Design** tab, click the **Analytics** view.
2. In the content panel, select the analytic name.
3. From the menu in the right-hand corner, select **View Related Objects**.

   The **Related Objects** dialog box appears. You can see the list of rules and connections that references the analytic.

Predefined Analytics

RulePoint provides a predefined set of analytics that you can use when writing rules. You cannot create instances of the predefined analytics.

The following table describes the predefined analytics, arranged in alphabetical order:

Analytics	Type and Description	Return Type
abs	Math: Returns the absolute value of a number.	List of real numbers
average	Statistical: Returns the average of its arguments.	Real number
ceil	Math: Rounds a number to the nearest integer or to the nearest multiple of significance.	List of real numbers
charat	String: Returns the character from a string at the specified index.	Character
clean	String: Removes all nonprintable characters from text.	List of string
compareto	String: Compares two given strings.	Integer
concat	String: Joins several items into one text item.	String
confidence	Statistical: Returns the confidence interval for a population mean.	Real number
correlation	Statistical: Returns the correlation coefficient between two data sets.	Real number
count	Statistical: Counts the number of occurrences of event properties.	Integer
current	Lookup: Selects the current event out of an event set.	Any type
date	Date and Time: Converts a date-time string into a format that RulePoint can use to compare with other date-time values.	Date

Analytics	Type and Description	Return Type
datedifference	Date: Returns the difference between two dates or time in years, months, days, hours, months, or seconds.	Real number
endswith	String: Checks if all the text inputs end with the given string.	Boolean value
even	Math: Rounds a number to the nearest even integer.	List of real numbers
floor	Math: Rounds a number down, toward zero, to the nearest multiple of significance.	List of real numbers
getcurrentdate	Date: Returns the current system date and time.	Date
getdatefrommillis	Date: Coverts milliseconds into date.	Date
getdateinmillis	Date: Coverts the date and time into milliseconds.	Long
geodistance	Geo spatial: Returns minimum distance between two geo points.	Real number
geoinbound	Geo spatial: Checks if a line has its points starting outside a geo polygon and ending inside the geo polygon.	Boolean value
geoinsidearea	Geo spatial: Determines whether a geo reference point is inside a defined polygon.	Boolean value
geointersect	Geo spatial: Validates whether two geo polygons intersect.	Boolean value
geooutbound	Geo spatial: Determines whether points of a line start inside a polygon and end outside the polygon.	Boolean value
geooutsidearea	Geo spatial: Determines whether a geo referenced point is outside a defined polygon.	Boolean value
geopassthru	Geo spatial: Verifies if a geo line passes through a specified geo point.	Boolean value

Analytics	Type and Description	Return Type
indexof	Lookup and reference: Accesses events with a specific index. An event set lists events from the newest event to the oldest event.	Any type
integer	Math: Rounds a number down to the nearest integer.	List of integers
isblank	Informational: Verifies whether all the data are blank.	List of Boolean values
isholiday	Informational: Verifies whether a date is in the holiday list.	Boolean value
isnum	Informational: Verifies whether all data contains only numbers.	List of Boolean values
istext	Informational: Verifies whether all data contains only text.	List of Boolean values
isweekday	Informational: Checks if the given date is a weekday.	Boolean value
isweekend	Informational: Verifies if the given date is a weekend.	Boolean value
large	Statistical: Returns the k-th largest value in a data set.	Real number
list2vars	Transform: Transforms result list in to multiple values and assigns them to temporary variables.	Any type
lower	String: Converts the text to lower case.	List of strings
max	Statistical: Evaluates a list of numbers and returns the number with the largest value.	Real number
min	Math: Evaluates a list of numbers and returns the number with the lowest value.	Real number
mod	Math: Returns the remainder after division.	Real number
mode	Statistical: Returns the most common value in a data set.	Real number
mround	Math: Returns a number rounded to the desired multiple	List of real numbers

Analytics	Type and Description	Return Type
odd	Math: Rounds a number up to the nearest odd integer.	List of real numbers
parse	String: Splits the text at the specified delimiter and returns a list of the text.	List of strings
past	Lookup: Selects the past event from an event set.	Any type
percentile	Statistical: Returns the k-th percentile of values in a range.	Real number
power	Math: Returns the result of a number raised to a power.	Real number
proper	String: Capitalizes the first letter in each word of a text value.	List of strings
range	Statistical: Calculates and returns the difference between the minimum and maximum numeric values provided.	Real number
regex	Regular expression: Applies a regular expression to a string and returns all matching substrings.	String
replace	String: Replaces characters within text.	List of strings
round	Math: Rounds a number to a specified number of digits.	List of real numbers
rounddown	Math: The rounddown analytic rounds a number down, towards zero.	List of real numbers
roundup	Math: Rounds a number up, away from zero.	List of real numbers
sign	Math: Returns the sign of a number.	List of real numbers
small	Statistical: Returns the k-th smallest value in a data set.	Real number
startswith	String: Validates if all the text that starts with the given string.	Boolean value
stdev	Statistical: Estimates standard deviation based on a sample.	Real number

Analytics	Type and Description	Return Type
stringequals	String: Validates if two strings are equal.	Boolean value
stringindexof	String: Returns the index within the string of the first occurrence of the specified substring.	Integer
stringlastindexof	String: Returns the index within the string of the rightmost occurrence of the specified substring.	Integer
substring	String: Returns the substring of the string based on the index provided.	String
sum	Math: Adds a list of numbers.	Real number
trim	String: Removes spaces from the text.	List of strings
truncate	Math: Truncates a number to an integer.	List of integers
upper	String: Converts the text to upper case.	List of strings
xpath	XML lookup: Navigates to a specific element and attribute in xml using xpath expression and returns the list of objects that match.	Any type

abs

The ABS analytic returns the absolute value of a number. The absolute value of a number is the number without its sign.

Syntax

```
abs(number,[number2],…)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number	Required	The real number for which you want the absolute value.
number2, ...	Optional	The real numbers for which you want the absolute value.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1 stockTopic s with abs(s.value) as result=10.12 then response with body="${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value = 10.12 or -10.12	Returns the absolute value.	10.12

Error Condition

The analytic does not run when there is a type mismatch, that is, when the values contain nonnumeric data. On type mismatch, the analytic returns an error, stops the rule evaluation, and writes the mismatch error to the log file.

average

The average analytic calculates and returns the average of real numbers.

Syntax

```
average(number1,[number2],…)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number1	Required	The real number value for which you want the average value.
number2, ...	Optional	The real number values for which you want the average value.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 3 stockTopic s with  average(s.value) as  result >10 then response with body=”${result} ”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.values = 8,12, and 14	Average of three event values must exceed 10.	>10

Error Condition

The analytic does not run when there is a type mismatch, that is, when the values contain nonnumeric data. On type mismatch, the analytic returns an error, stops the rule evaluation, and writes the mismatch error to the log file.

ceil

The ceil analytic returns a number rounded up, away from zero, to the nearest multiple of significance.

Syntax

```
ceil(number1,[number2],...,significance)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number1	Required	The real number value that you want to round up.
number2, ...	Optional	The real numbers values that you want to round up.
significance	Required	The multiple to which you want to round the value.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1  stocktopic  s with ceil(s.value,1) as result=4 then response with body="event
set result is ${ result }"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value = 3.5	Rounds 3.5 up to nearest multiple of 1.	4

Error Condition

The analytic does not run when there is a type mismatch, that is, when the values contain nonnumeric data. On type mismatch, the analytic returns an error, stops the rule evaluation, and writes the mismatch error to the log file.

charat

The charat analytic returns the character from a string at the specified index.

Syntax

```
charat(string,indexvalue)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
string	Required	The string whose character you want to retrieve.
indexvalue	Required	The index value that you specify for the string to return the character.

Example

```
when 1 Greetings g with charat(g.wish,4) as result !="" then response with body="$
{result}"
```

The following table describes the result for the event property value that you pass:

Usage	Description	Result
w.wish="hello world"	Character value at position 4.	0

clean

The clean analytic removes all nonprintable characters from text.

Syntax

```
clean(text1,[text2],..)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
text1	Required	Removes all nonprintable characters from the text parameter.
text2, ...	Optional	Removes all nonprintable characters from the text.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	text
Watchlist values	text

Example

```
when 1 stocktopic s with clean(s.text) as result = “Testing”  then response with body=" ${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
=CHAR(27)&"Testing"&CHAR(28)	Removes the nonprintable characters from the text string.	Testing

compareto

The compareto analytic compares two given strings. The value 0 if 2nd string is lexicographically equal to the 1st string; a value less than 0 if 2nd string is lexicographically greater than 1st string; and a value greater than 0 if the argument is a 2nd string lexicographically less than 1st string.

Syntax

```
compareto(string1, string2)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
string1	Required	The first string that you want to compare.
string2	Required	The second string that you want to compare.

Example

```
when 1 compare c with compareto(c.string1, c.string2) as result<0  then response with body=”${result}”
```

The following table describes the result for the event property value that you pass:

Usage	Description	Result
s.string1=hello s.string2=welcome	Since hello comes before welcome	greater than zero

concat

The concat analytic joins several items into one text item. The joined items can be text, numbers, event properties, or a combination of those items.

Syntax

```
concat(text1,[text2],..)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
text1	Required	The text item to be concatenated.
text2, ...	Optional	The text items to be concatenated.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	text
Watchlist values	text

Example

```
WHEN 1 stock s with concat(s.value1,s.value2) as result = “HighValue” then response WITH
body="${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value1 =High s.value2=Value	concat analytic joins High and Value.	HighValue

confidence

The confidence analytic returns the confidence interval for a population mean by using a normal distribution.

Syntax

```
confidence(alpha,standard_dev,size)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
alpha	Required	The significance level used to compute the confidence level. The confidence level equals 100*(1 - alpha)%. It means an alpha of 0.05 indicates a 95 percent confidence level. The alpha is of real number type with value less than or equal to 1.
standard_dev	Required	The population standard deviation for the data range and is assumed to be known. This is of real number type.
size	Required	The sample size. Integer type.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number for alpha and standard_dev. Size is of integer type.

Example

```
when 1 stocktopic s with confidence(s.alpha,s.standard_dev,s.size) as result >10 then
response with body=" ${result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.alpha =0.06 s.standard_dev=45.7 s.size=65	s.alpha and s.standard_dev are decimal numbers. s.size is an integer number.	>10

Error Condition

The analytic does not run when there is a type mismatch, that is, when the values contain nonnumeric data. On type mismatch, the analytic returns an error, stops the rule evaluation, and writes the mismatch error to the log file.

correlation

The correlation analytic returns the correlation coefficient of list1 and list2. Use the correlation coefficient to determine the relationship between two lists of values. For example, you can examine the relationship between a list of an old stock price and a new stock price.

Syntax

```
correlation(list1,list2)
```

The following table describes the required and optional arguments for the correlation function:

Argument	Required Optional	Description
list1	Required	A list of values.
lis2	Required	A second list of values.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Watchlist values	Real number

Example

```
when 1 stocktopic with correlation(watchlist:stockprice,watchlist:stocknewprice) as
result > 0 then response with body=" ${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
Watchlist stockprice: 5,7,15 Watchlist stocknewprice: 9,13,18	Correlates the two concatenated watchlist values and returns the result.	0.96

Error Condition

The analytic does not run when there is a type mismatch, that is, when the values contain nonnumeric data. On type mismatch, the analytic returns an error, stops the rule evaluation, and writes the mismatch error to the log file.

count

The count analytic counts the number of occurrences of event properties. The count analytic does not count the values of the related properties.

Syntax

```
count(event property1, [event property2],...)
```

The following table provides the required and optional arguments:

Argument	Required Optional
event property1	Required
event property2	Optional

Example

```
WHEN votes v with count(v.florida, v.virginia) as result  > 1000 then response with
body=”${result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
Event properties of Florida and Virginia for topic, votes	Sums votes from Florida and Virginia from a single topic, votes, and sends response if it is greater than 1000.	>1000

current

The current analytic selects the current event out of an event set.

Syntax

```
current(event.property)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
event.property	Required	The event property of the latest event.

Example

```
when 3 transactions t with current(t.amount) as result=25 then response with body =”${result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
amount[2]=15 amount[1]=20 amount[0]=25	The current event value should be 25.	25

date

The date analytic converts a date-time string into a format that RulePoint can use to compare with other date-time values. The default format is `MM/dd/yy hh:mm:ss a`.

Syntax

```
date(String)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
string	Required	A date value of string type.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	text

Example

```
WHEN 1 news n with date(n.pub) as datevalue > date("01/01/13 1:00:00 PM")  then response with body=”${datevalue)”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
n.pub =" 02/02/13 1:00:00 AM"	Comparing the value of n.pub that is of string type.	>01/01/13 1:00:00 PM

datedifference

The datedifference analytic returns the difference between two dates in terms of years, months, days, hours, minutes, seconds.

Syntax Option 1

```
datedifference(startdate,enddate,dateformat,units)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
startdate	Required	The start date.
enddate	Required	The end date. The format of the start and end date must be same.
dateformat	Required	The date format followed for the start and end date.
units	Required	The difference unit. This can be years, months, days, hours, minutes, or seconds.

Example

```
when 1 yearsDiff  df with datedifference(df.startdate,df.enddate, df.dateformat,df.unit)
as result = 1 then  response with body= "${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
df.startdate= 2011/02/12 df.enddate= 2013/01/01 df.dateformat= yyyy/MM/dd df.unit=years	Returns the difference between two dates in terms of year.	1

Syntax Option 2

```
datedifference(starttime,endtime,timeformat,units)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
starttime	Required	The start time.
endtime	Required	The end time. The format of the start and end time must be same.
timeformat	Required	The time format followed for the start and end date.
units	Required	The difference unit. This can be hours, minutes, or seconds.

Example

```
when 1 timeDiff df with datedifference(df.starttime,df.endtime, df.timeformat,df.unit)
as result = 841 then  response with body= "${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
df.starttime= 09:14:11 df.endtime= 23:15:35 df.dateformat= HH:mm:ss df.unit=minutes	Returns the difference between two times in terms of minutes.	841

endswith

The endswith analytic checks if all the text inputs end with the given string.

Syntax

```
endswith(text1,[text2],…,string)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
text1	Required	The end text of the first text item that the analytic must verify.
text2, ...	Optional	The end text of other text items that the analytic must verify.
string	Required	The string value with which the text ends.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	text
Watchlist values	text

Example

```
when 1 stocktopic s with endsWith(s.nature,"High") as result  then response with body="$
{ result }"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
n. nature="returnsHigh"	Validates whether the text ends with High.	true

even

The even analytic returns the number rounded up to the nearest even integer.

Syntax

```
even(number1,[number2],..)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number1	Required	The real number value that you want the analytic to round to the nearest even integer.
number2...	Optional	The real numbers that you want the analytic to round to the nearest even integer.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1 stocktopic s with even(s.value) as result=34 then response with body=" ${ result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value =33.78	Rounds the number to the nearest integer value.	34

Error Condition

The analytic does not run when there is a type mismatch, that is, when the values contain nonnumeric data. On type mismatch, the analytic returns an error, stops the rule evaluation, and writes the mismatch error to the log file.

floor

The floor analytic rounds a number down, toward zero, to the nearest multiple of significance.

Syntax

```
floor(number,[number2],significance)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number1	Required	The numeric value that you want to round.
number2	Optional	The numeric values that want to round.
significance	Required	The multiple to which you want to round.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1  stocktopic  s with floor(s.value,1) as result=3 then response with body="event
set result is ${ result }"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value =3.5	Rounds 3.5 down to the nearest multiple of 1.	3

Error Condition

The analytic does not run when there is a type mismatch, that is, when the values contain nonnumeric data. On type mismatch, the analytic returns an error, stops the rule evaluation, and writes the mismatch error to the log file.

getcurrentdate

The getcurrentdate analytic returns the current date and time of the system.

Syntax

```
getcurrentdate()
```

Example

```
when 1 Period p  with getcurrentdate() as result !=””then response with body=”${result}”
```

The following table describes the result for the event property value that you pass:

Usage	Description	Result
Call the analytic to get the date.	Returns the current date and time of the system.	Sat Jan 25 08:05:05 IST 2014

getdatefrommillis

The getdatefrommillis analytic returns the date and time for the provided milliseconds.

Syntax

```
getdatefrommillis(milliseconds,dateformat)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
milliseconds	Required	Date in milliseconds that you want to convert to the date format.
dateformat	Required	The date and time format that you want to convert the date in milliseconds.

Example

```
when 1 dtfromMillis dfm with getdatefrommillis(dfm.millisvalue,dfm.dateformat) as x =
"2013/11/26" then testresponse
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
dfm.millisvalue= " 1385448067337" dfm.dateformat= yyyy/mm/dd	Get date from milliseconds in the specified format.	2013/11/26

getdateinmillis

The getdateinmillis analytic converts date and time into milliseconds.

Syntax

```
getdateinmillis(date,dateformat)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
date	Required	The date and time that the analytic must convert to milliseconds.
dateformat	Required	The date and time format for the mentioned date.

Example

```
when 1 dttoMillis dtm with getdateinmillis(dtm.datevalue, dtm.dateformat)  as x =
"1325399400000" then  testresponse
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
dtm.datevalue= 2012/01/01 12:00:00 dtm.dateformat= yyyy/MM/dd HH:mm:ss dtm.dateformat= yyyy/MM/dd HH:mm:ss	Returns the date in milliseconds.	1325399400000

geodistance

The geodistance analytic returns the minimum distance between two geo points. A geo point represents the longitude and latitude coordinates.

Syntax

```
geodistance(latitude1, longitude1, latittude2, longitude2)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
latitude1	Required	The value of the first latitude point.
longitude1	Required	The value of the first longitude point.

Argument	Required Optional	Description
latitude2	Required	The value of the second latitude point.
longitude2	Required	The value of the second longitude point.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
WHEN 1 boat b WITH geodistance(b.x1,b.y1,b.x2,b.y2) as distance =10 then response with
body= "${distance}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
b.x1=10 b.y1=10 b.x2=10 b.y2=20	Finds the distance between two geo points.	10

geoinbound

The geoinbound analytic checks if a line has its points starting outside a polygon and ending inside the polygon. The first four arguments refer to the points of the line. The remaining arguments refer the points of the polygon.

The polygon can have the following coordinates:

- *n* number of latitude and longitude pairs. For example, it can be polyLat1, polyLong1, polyLat2, polyLong2, ..., ..., polyLatN, polyLongN . The number of latitude arguments must be equal to the number of longitude arguments. Specify the values directly in the analytic or it can be accessed as an event property.
- Access list of latitudes and a list of longitudes from watchlists whose type is list. Both latitude and longitudes coordinate lists must be the same size, and each coordinate in either list must correspond to the coordinate in the same position in the other list. For example, geoinbound (l.lat1,l.long1,l.lat2,l.long2, Watchlist:lats, Watchlist:longs). In this case, Watchlist:lats and Watchlist:longs are watchlists of type list.
- Access a single two-dimensional list of comma-separated latitudes and longitudes as arguments from watchlists whose type is text . For example, geoinbound (l.lat1,l.long1,l.lat2,l.long2, Watchlist:polgygon). In this case, Watchlist:polgygon is of type text.

The following criteria defines how you can create a polygon:

- The polygon must have at least three sides.

- Polygons are always two-dimensional, horizontal shapes.
- All polygons must be closed. If you do not close the polygon in the analytic, the analytic automatically closes the polygon by connecting the last specified point to the first point. To close the polygon in the analytic, set the arguments for the last polygon point to be the same as those of the first polygon point.
- Calculate all lines between polygon points as straight.
- Polygon lines can be crossed to create multiple inside areas.
- The order of the arguments in the analytic define the order in which the points of the polygon connect. For example, if you define the points as A, D, C, B, the polygon is constructed from point A to D, D to C, and C to B.
- Coordinate literal arguments must be strings placed within quotation marks.

Syntax

```
geoinbound(l.lat1,l.long1,l.lat2,l.long2, polyLat1, polyLong1, polyLat2, polyLong2,
polyLat3, polyLong3
,[polyLat4],[ polyLong4],..)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
l.lat1,l.long1,l.lat2,l.long2	Required	The four geo points which form the line.
polyLat1, polyLong1, polyLat2, polyLong2, polyLat3, polyLong3	Required	The minimum points that form a polygon. The number of latitude arguments must be equal to the number of longitude arguments.
[polyLat4],[polyLong4],...	Optional	The other points that form the polygon.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1 line l with
geoinbound(l.lat1,l.long1,l.lat2,l.long2,100,100,100,-100,-100,-100,-100,100) as result
then response with body= "${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
l.lat1 =150 l.long1=150 l.lat2=0 l.long2=0	Verify that the line has its points starting outside a geo polygon and ending inside the geo polygon.	true

geoinsidearea

The geoinsidearea analytic takes latitude-longitude reference points and multiple latitude-longitude points of a polygon, and returns true if the reference point falls inside the polygon. If the reference point falls on the points of the polygon, it returns true.

The criteria for creating a polygon and its coordinates is the same as for the geoinbound() analytic.

Syntax

```
geoinsidearea(pointLat,pointLong,polyLat1,polyLong1,polyLat2,polyLong2,polyLat3,polyLong3
,[polyLat4],[ polyLong4],..)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
pointLat, pointLong	Required	The two geo points that form the point.
polyLat1, polyLong1, polyLat2, polyLong2, polyLat3, polyLong3	Required	The minimum points that form a polygon. The number of latitude arguments must be equal to the number of longitude arguments.
[polyLat4],[polyLong4],...	Optional	The other points that form the polygon.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
WHEN 1 boat b with geoinsidearea(b.lat, b.long, "100.0","100.0", "100", "-100.0",
"-100.0", "-100.0", "100.0","-100.0") as result then response with body= "${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
b.lat=0 b.long=0	Provide latitude-longitude reference points and multiple latitude-longitude points of a polygon to verify the points are within the defined polygon.	true

geointersect

The geointersect analytic validates whether two geo polygons intersect.

The criteria for creating a polygon and its coordinates is the same as for the geoinbound() analytic.

Syntax

```
geointersect(count of coordinates in 1st polygon,count of coordinates of 2nd polygon,1st polygon coordinates,2nd polygon coordinates)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
count of coordinates in 1st polygon	Required	The count of all the geo points of the first polygon. If the polygon points are accessed from a watchlist, the value for the count of the coordinates is 0.
count of coordinates in 2nd polygon	Required	The count of all the geo points of the second polygon. If the polygon points are accessed from a watchlist, the value for the count of the coordinates is 0.
1st polygon coordinates	Required	All the geo points of the first polygon.
2nd polygon coordinates	Required	All the geo points of the second polygon.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1 polygon p1 ,1 polygon p2  with
geointersect(p1.count,p2.count,p1.x1,p1.y1,p1.x2,p1.y2,p1.x3,p1.y3,
p2.x1,p2.y1,p2.x2,p2.y2,p2.x3,p2.y3) as result then response with body="${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
p1.count : 6 p2.count: 6 p1.x1=50,p1.y1=50,p1.x2=0,p1.y2=0,p1.x3=100,p1.y3=0 p2.x1=20,p2.y1=20,p2.x2=0,p2.y2=0,p2.x3=50,p2.y3=0	Verifies whether the two polygons are intersecting.	true

geooutbound

The geooutbound analytic checks if a line has its points starting inside a polygon and ending outside the polygon. The first four arguments refers to the points of the line. The remaining arguments refer the points of the polygon.

The criteria for creating a polygon and its coordinates is the same as for the geoinbound() analytic.

Syntax

```
geooutbound(l.lat1,l.long1,l.lat2,l.long2, polyLat1, polyLong1, polyLat2, polyLong2,
polyLat3, polyLong3
,[polyLat4],[ polyLong4],..)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
l.lat1,l.long1,l.lat2,l.long2	Required	The four geo points that form the line.
polyLat1, polyLong1, polyLat2, polyLong2, polyLat3, polyLong3	Required	The minimum points that form a polygon. The number of latitude arguments must be equal to the number of longitude arguments.
[polyLat4],[polyLong4],...	Optional	The other points that form a polygon.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1 line l with
geooutbound(l.lat1,l.long1,l.lat2,l.long2,100,100,100,-100,-100,-100,-100,100) as result
then response with body= "${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
l.lat1 =0 l.long1=0 l.lat2=150 l.long2=150	Verifies if the line has its points starting inside a geo polygon and ending outside the geo polygon.	true

geooutsidearea

The geooutsidearea analytic takes latitude-longitude reference points and multiple latitude-longitude points of a polygon, and returns true if the reference point falls outside the polygon. If the reference point falls on the points of the polygon, it returns false.

The criteria for creating a polygon and its coordinates is the same as for the geoinbound() analytic.

Syntax

```
geooutsidearea(pointLat,pointLong,polyLat1,polyLong1,polyLat2,polyLong2,polyLat3,polyLong
3,[polyLat4],[ polyLong4],..)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
pointLat, pointLong	Required	The two geo points that form the point.
polyLat1, polyLong1, polyLat2, polyLong2, polyLat3, polyLong3	Required	The minimum points that form a polygon. The number of latitude arguments must be equal to the number longitude arguments.
[polyLat4],[polyLong4],...	Optional	The other points that form a polygon.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
WHEN 1 boat b with geooutsidearea(b.lat, b.long, "100.0","100.0", "100", "-100.0",
"-100.0", "-100.0", "100.0","-100.0") as result then response with body= "${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
b.lat=200 b.long=200	Provide latitude-longitude reference points and multiple latitude-longitude points of a polygon to verify the points are outside the defined polygon.	true

geopassthru

The geopassthru analytic verifies if a geo line passes through a specified point.

The criteria for creating a polygon and its coordinates is the same as for the geoinbound() analytic.

Syntax

```
geopassthru(pointlat,pointlong,linelat1,linelong1,linelat2,linelong2)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
pointLat, pointLong	Required	The coordinates forming the geo point.
linelat1,linelong1,linelat2,linelong2	Required	The minimum points that form a polygon. The number of latitude arguments must be equal to the number longitude arguments.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1 line l with geopassthru(l.lat1,l. long1,l.x1,l.y1,l.x2,l.y2) as result then
response with body="${ result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
l.lat1=50,l. long1=50 l.x1=0,l.y1=0,l.x2=100,l.y2=100	Verifies if the line passes through the specified geo point.	true

indexof

The indexof analytic accesses events with a specific index. An event set lists events from the newest event to the oldest event. For example, when two events are sent, the first older event index is 1 and the latest event index is 0.

Syntax

You have the following options for using an indexof analytic:

- `indexof(event property,event index)`

  The following table describes the required and optional arguments:

Argument	Required Optional	Description
event property	Required	Assesses any one of the event properties.
event index	Required	Assesses a specific event in an event set.

- `indexof(event property,starting index,ending index)`

  The following table describes the required and optional arguments:

Argument	Required Optional	Description
event property	Required	Assesses any one of the event properties.
event index	Required	Specifies the index range, from start to end.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example 1

```
when 2 stock s with indexof(s.prop1,1 ) as  result >4 then response with body= "$
{result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.prop[1]=5 à 1st event s.prop[0]=1 à latest event	Verifies if an event property of a specific index is greater than 4.	5

Example 2

```
when 3 stock s with indexof(s.prop1,0,2 ) as result >4 then response with body= "${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.prop[2]=2 à 1st event s.prop[1]=1 à 2nd event s.prop[0]=5 à latest event	Verifies whether all the values of an event property in the specified index range is greater than 4.	5

Error Condition

For all topic rules, the compiler cannot perform the index bound check. You get a warning message when the index is out of bounds. If you swap the position of the event index with the event property, for example, if you write an indexof analytic as `indexof(event index,event property)`, the DRQL rule compiles, but results in an error during the runtime.

integer

The integer analytic rounds a number down to nearest integer.

Syntax

```
integer(number1,[number2],...)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number	Required	The real number that you want to round down to an integer.
number2	Optional	The real numbers that you want to round down to an integer.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1 stocktopic s with integer(s.stockprice) as result= 9  then response with body="${result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.stockprice = 9.9	Real value is rounded down to an integer value.	9

Error Condition

The analytic does not run when there is a type mismatch, that is, when the values contain nonnumeric data. On type mismatch, the analytic returns an error, stops the rule evaluation, and writes the mismatch error to the log file.

isblank

The isblank analytic verifies if all data are blank.

Syntax

```
isblank(item1,[item2],…)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
item1	Required	Verifies whether the item is blank.
item2,...	Optional	Verifies whether the items are blank.

Example

```
when 1 stocktopic s with isblank(s.value) as result  then response with body= “${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value=""	Validates whether the value sent to topic property is null.	true

isholiday

The isholiday verifies whether a date is in a holiday watchlist.

Syntax

```
isholiday(date1,watchlist:HolidayWatchlist)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
date1	Required	Verifies whether the specified date is in the holiday list. Specify the format of the date value.
[date2],...	Optional	Verifies whether the specified dates are in the holiday list.

Example

```
when 1 hol1 h with isholiday(h.myDate,watchlist:ListOfHols) as result then response with body="${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
h.myDate = 12/25/2010 mm/dd/yyyy watchlist:ListOfHols 12/25/2010 01/01/2011	Verifies the date input with the existing watchlist values.	true

isnum

The isnum analytic checks whether all data contains only numbers.

Syntax

```
isnum(item1,[item2],…)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
item1	Required	Verifies whether the item is a number.
item2	Optional	Verifies whether the items are numbers.

Example

```
when 1 stocktopic s with isnum(s.value) as result  then response with body=" ${ result }"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value=55	Validates whether the data input is a number.	true

istext

The istext analytic checks if all the data contains only text.

Syntax

```
istext(item1,[item2],…)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
item1	Required	Verifies whether this item is a text.
item2, ...	Optional	Verifies whether the items are text.

Example

```
when 1 stocktopic s with istext(s.stock) as result  then response with body=" ${result }"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.stock="INFA"	Validates whether the data input contains only text.	true

isweekday

The isweekday analytic checks if the given date is a weekday.

Syntax

```
isweekday(date1)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
date1	Required	Verifies whether the date is a weekday.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	String

Example

```
when 1 stocktopic s with isweekday(s.purchasedate) as result then response with body=" $
{result }"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.purchasedate= "1/1/14 1:00:00 PM "	Validates whether the input date is a weekday.	true

isweekend

The isweekend analytic checks if the given date is a weekend.

Syntax

```
isweekend(date1)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
date1	Required	Verifies whether the date is a weekend.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	String

Example

```
when 1 stocktopic s with isweekend(s.purchasedate) as result  then response with body=" ${result }"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.purchasedate= "1/5/14 1:00:00 PM "	Validates whether the input date is a weekend.	true

large

The large analytic returns the k-th largest value in a data set. You can use this analytic to select a value based on its relative standing. For example, you can use large to return the highest, runner-up, or third-place score.

Syntax

```
large(list of numbers, k)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
list of numbers	Required	Generally this is a watchlist that has a list of numbers.
k	Optional	The position, from the largest, in the array or cell range of data to return. 1 refers to the largest number in the list of numbers.

Example

```
when 1 stocktopic with large(watchlist:numberlist,3 ) as result = 15  then response with body=" result  is ${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
watchlist: numberlist 50 20 15 10	Returns the third largest number in the list.	true

list2vars

The list2vars() analytic transforms the result list obtained from another analytic or from an event set into multiple temporary variables that holds a list of values.

Syntax

```
list2vars(result,tempvar1,[tempvar2],...,[tempvarn])
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
result	Required	Result list obtained from another analytic or from an event set.
tempvar1	Required	Temporary variable that holds the first value of the list.
tempvar2,...,tempvar*n*	Optional	tempvar2 to tempvar*n* that holds 2nd value to nth value in the list.

Example for Using list2vars from the Event Set

```
when 3 stock s with list2vars(s.price, a, b, c) then response with body =”value of
temporary variable  a is ${a}, b value is  ${b} and c value is ${c} “
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.price[2]=10 s.price[1]=20 s.price[0]=30	Assigns values for temporary variables from the latest to the oldest. The first (oldest) event value is assigned to C and the third (latest) event value is assigned to A.	${a} resolves to 30 ${b} resolves to 20 ${c} resolves to 10

Example for Using list2vars Result List Obtained from Another Analytic

```
when 1 card c WITH sqlgetAccountInfo(c.customerID) as  result != "" AND
list2vars(result, FirstName, LastName, AcctID, PurchaseLimit, CardType) then response
with body =”value of FirstName is ${FirstName} ,
```

The SQL Analytic, sqlgetAccountInfo, pulls a row with five columns, casts the row into a temporary variable result, and then uses list2Vars() analytic in the next condition to create multiple temporary variables.

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
First Name=John Last Name=Watson Account ID=12345 Purchase Limit=1000 CardType=credit	Returns a list and the values are assigned to temporary variables.	FirstName=John LastName= Watson AcctID=12345 PurchaseLimit=1000 CardType= credit

The analytics whose result set can be a list are abs, ceil, clean, endswith, even, floor, integer, isblank, isholiday, isnum, istext, isweekeday, isweekend, lower, mround, odd, parse, proper, round, rounddown, roundup, sign, startswith, trim, truncate, and upper.

lower

The lower analytic converts all uppercase letters in a text string to lowercase.

Syntax

```
lower(text1,[text2],,,)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
text1	Required	The text that you want to convert to lowercase.
text2,...	Optional	The text that you want to convert to lowercase.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	string
Watchlist values	string

Example

```
when 1 stock s  with lower(s.symbol) as result= “infa”  then response with body="$
{result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.symbol= INFA	Converts the text to lowercase.	infa

max

The max analytic returns the largest value in a set of values.

Syntax

```
max(number1,[number2],...)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number1	Required	Numbers for which you want to find the maximum value.
number2,...	Optional	Other numbers for which you want to find the maximum value

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
WHEN stock s WITH max(s.value1,s.value2,value3)  as result =45.0 then response with body
=”${result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value1=22.0 s.value2=37.8 s.value3=45.0	Returns the maximum value in the list.	45.0

min

The min analytic returns the smallest value in a set of values.

Syntax

```
min(number1,[number2],...)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number1	Required	Number for which you want to find the minimum value.
number2	Optional	Other numbers for which you want to find the minimum value.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
WHEN  stock s WITH min(s.value1,s.value2,value3)  as result =22.0 then response with
body =”${result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value1=22.0 s.value2=37.8 s.value3=45.0	Returns the minimum value in the list.	22.0

mod

The mod analytic returns the remainder after the number is divided by the divisor. The result has the same sign as the divisor.

Syntax

```
mod(number, divisor)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number	Required	The number for which you want to find the remainder.
divisor	Required	The number by which you want to divide the number.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Integer

Example

```
mod(17,3)= 2
when stock s with max(s.value1,3)  as result =2  then response with body =”${result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value1=17	Returns the remainder after dividing 17 by 3.	2

mode

The mode analytic returns the most frequently occurring, or repetitive, value in an array or range of data.

Syntax

```
mode(number1,[number2],...])
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number	Required	The first number argument for which you want to calculate the mode.
divisor	Required	The other number arguments for which you want to calculate the mode.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1 stocktopic s with mode(watchlist:modenumbers) as result =5  then response with
body=" ${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
watchlist: modenumbers 15 5 27 32 5	Returns the most common value in the data set.	5

mround

The mround analytic returns a number rounded to the desired multiple.

Syntax

```
mround(number1,[number2],..., multiple)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number1	Required	The value to round.
number2	Optional	Other values you want to round.
multiple	Required	The multiple to which you want to round the number.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1  stockTopic with mround(s.value ,0.2) as result > 1.4 then response with body=" ${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value=1.3	Rounds 1.3 to a nearest multiple of 0.2.	1.4

odd

The odd analytic returns the number rounded up to the nearest odd integer.

Syntax

```
odd(number1,[number2],...)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number1	Required	The real number value that you want to round to the nearest odd integer.
number2	Optional	Other numbers for which you want to round to the nearest odd integer.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1 stocktopic s with odd(s.value) as result=33 then response with body=" ${ result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value =31.52	Rounds the number to the nearest odd number.	33

Error Condition

The analytic does not run when there is a type mismatch, that is, when the values contain nonnumeric data. On type mismatch, the analytic returns an error, stops the rule evaluation, and writes the mismatch error to the log file.

parse

The parse analytic splits the text at the specified delimiter and returns the text.

Syntax

```
parse(text, delimiter)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
text	Required	The text that contains the delimiter.
delimiter	Required	The delimiter that is used to split the text.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number

Example

```
when 1 presenter p  with parse(p.greetings,” ,”) as result !=””  then response with
body="${result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
p.greetings="HELLO,WORLD"	Splits Hello,World into two values using the delimiter.	Hello World

past

The past analytic creates a historical set of events by removing the current event from the set of a rule.

Syntax

```
past(event.property)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
event.property	Required	The event property other than the latest event.

Example

```
when 2 transactions t with past(t.amount) as result= 15 then response with body =”${result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
amount[1]=15 amount[0]=20	The past event value should be 15.	15

percentile

The percentile analytic returns the k-th percentile of values in a range. You can use this analytic to establish a threshold of acceptance. The calculation of percentile is performed using the Apache math library.

Syntax

```
percentile(list,k)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
list	Required	The list of data that defines relative standing. This can be watchlist of data, list, or event properties.
k	Required	The percentile value in the range 0 to 1, inclusive.

Example

```
when 1 students s with percentile(watchlist:scores,) as result =1.9 then response body=”${result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
watchlist:scores 1 2 3 4	30th percentile of the watchlist.	1.9

power

The power analytic returns the result of a number raised to a power.

Syntax

```
power(number, power)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number	Required	The base number. It can be any real number.
power	Required	The exponent to which the base number is raised.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number

Example

```
when stock  s with power(s.raised,3) as result=8 then response with body="${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.raised=2	Raises 2 to the power of 3.	8

proper

The proper analytic capitalizes the first letter in a text string and other letters in text that follow any character other than a letter. Converts all other letters to lowercase letters.

Syntax

```
proper(text1,[text2],...)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
test1	Required	Text for which you need to apply the proper analytic.
test2	Optional	Other text for which you need to apply the proper analytic.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	string
Watchlist values	string

Example

```
when 1 Greeting s with proper(s.value) as result =”How’R U”  then response with
body=”value of greeting is ${result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value=hOw'r u	Capitalizes the first letter in a text string and other letters in the text that follow any character other than a letter. Converts all other letters to lowercase letters.	How'R U

range

The range analytic calculates and returns the difference between the minimum and maximum numeric values provided.

Syntax

```
range(list of numbers)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
list of numbers	Required	The list of numbers for which you want to calculate the range.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1 rangelist rl with range(rl.value1, rl.value2, rl.value3) as result =5 then
response with body =”${result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
rl.value1=6 rl.value2=8 rl.value3=11	Calculates the difference between the minimum and maximum values.	5

regex

The regex analytic applies a regular expression to a string and returns all matching substrings, which are then compared to another value in the condition.

Syntax

```
regex(string,regular expression)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
string	Required	The string for which you need to apply the regular expression.
regular expression	Required	The Java regular expression that you want to apply to the string.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	String

Example

```
when 1 Regt s with regex(s.data,”[0-9]") as  result !=''  then rtamresponse with body="${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.data=123456	Determines whether numerical data exists.	Validates to true.

replace

The replace analytic replaces part of a text string, based on the number of characters you specify, with a different text string.

Syntax

```
replace(old_text1,[old_text2],,, new_text ,start_num, num_chars)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
old_text1	Required	Text in which you want to replace some characters.
Old_text2	Optional	Text in which you want to replace some characters.
new_text	Required	The text that will replace characters in old_text.

Argument	Required Optional	Description
start_num	Required	The position of the character in old_text that you want to replace with new_text.
num_chars	Required	The number of characters in old_text that you want to replace with new_text.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Old_text and new_text can also be event properties.	String

Example

```
when 1 greeting s with replace(s.value1,s.value2,8,5) as result="welcome hello" then
response with body="${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value1= "welcome world" s.value2="welcome hello"	Replaces part of a text string, based on the number of characters you specify, with a different text string.	welcome hello

round

The round analytic rounds a number to a specified number of digits.

Syntax

```
round(number1,[number2],,,num_digits)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number1	Required	The number that you want to round.
number2	Optional	The other numbers that you want to round.
num_digits	Required	The number of digits to which you want to round the number argument.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1 stocktopic s with round(s.stockprice,2) as result = 27.28 then response with
body=" ${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.stockprice=27.2785	R ounds 27.2785 to 2 decimal points.	27.28

rounddown

The rounddown analytic rounds a number down, towards zero.

Syntax

```
rounddown(number1,[number2],,,num_digits)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number	Required	The number that you want to round down.
number2	Optional	The other numbers that you want to round down.
num_digits	Required	The number of digits to which you want to round down the number argument.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1 stocktopic s with rounddown(s.stockprice,2) as result = 27.27 then response with
body=" ${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.stockprice=27.2785	Rounds down 27.2785 to 2 decimal points	27.27

roundup

The roundup analytic rounds a number up, away from zero.

Syntax

```
roundup(number1,[number2],,,num_digits)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number	Required	The number that you want to round up away from zero.
number2	Optional	The other numbers that you want to round up.
num_digits	Required	The number of digits to which you want to round up the number argument.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1 stocktopic s with roundup(s.stockprice,2) as result = 27.28 then response with
body=" ${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.stockprice=27.2785	Rounds up 27.2785 to 2 decimal points.	27.28

sign

The sign analytic determines the sign of a number. Returns 1 if the number is positive, zero (0) if the number is 0, and -1 if the number is negative.

Syntax

```
sign(number1,[number2],,,)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number	Required	Any real number.
number2	Optional	Any real number.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number

Example

```
when 1 numbersign s with sign(s.value) as result =-1 then response with body="${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value=-25	Determines the sign of -25.	-1

small

The small analytic returns the k-th smallest value in a data set. Use this analytic to return values with a particular relative standing in a data set.

Syntax

```
small(list of numbers, k)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
list of numbers	Required	An array or range of numerical data for which you want to determine the k-th smallest value.
k	Required	The position from the smallest in the array or range of data to return.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event properties separated by a comma	Real number

Example

```
when 1 stocktopic with small(watchlist:numberlist,3 ) as result = 20 then response with
body=" result  is ${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
watchlist: numberlist 50 20 15 10	Returns the third smallest number in the list.	20

startswith

The startswith analytic verifies if all the text starts with the given string.

Syntax

```
startswith(text1,[text2],…,string)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
text1	Required	The text that you want to verify whether it starts with the specified string.
text2	Optional	The other text that you want to verify whether it starts with the specified string.
string	Required	The string value with which the text starts.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Text
Watchlist values	Text

Example

```
when 1 stocktopic s with startswith(s.nature,”returns”) as result  then response with
body="${ result }"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
n. nature="returns High"	Validating whether text ends with returns.	true

stdev

The stdev analytic estimates the standard deviation based on a sample. The standard deviation is a measure of how widely values are dispersed from the average value, that is the mean.

Syntax

```
stdev(number1,[number2],...])
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number1	Required	The first number argument corresponding to a sample of a population.
number2	Optional	A single array or a reference to an array instead of arguments separated by commas.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Text
Watchlist values	Text

Example

```
when 1 stockdeviation s with stdev(s.p1,s.p2,s.p3,s.p4) as result>2.73 then response
with body="${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
p1 1.98 p2 2.45 p3 -3.56 p4 0	Returns the standard deviation of the set of numbers.	2.7329638978

Error Condition

The analytic does not run when there is a type mismatch, that is, when the values contain nonnumeric data. On type mismatch, the analytic returns an error, stops the rule evaluation, and writes the mismatch error to the log file.

stringequals

The stringequals analytic validates if two strings are equal.

Syntax

```
stringequals(string1,string2)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
string1	Required	The first string that you want to compare.
string2	Required	The second string that you want to compare.

Example

```
when 1 PropCompare  p with stringequals(p.prop1,p.prop2)  as result  then response with
body="${result}"
```

The following table describes the result for the event property value that you pass:

Usage	Description	Result
p.prop1=INFA p.prop2=INFA	Compares two strings.	true

stringindexof

The stringindexof analytic returns the index within the string of the first occurrence of the specified substring. If it does not occur as a substring, -1 is returned.

Syntax

```
stringindexof(string1,substring)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
string1	Required	The first string.
substring	Required	The second substring from which you want to retrieve the occurrence of the first index value.

Example

```
when 1 greetings g with stringindexof(g.wish,”l”) as result !=”” then response with body
=”$result}”
```

The following table describes the result for the event property value that you pass:

Usage	Description	Result
g.wish="hello world"	Returns the index of l on its first occurrence.	2

stringlastindexof

The stringlastindexof analytic returns the index within the string of the right-most occurrence of the specified substring.

Syntax

```
stringindexof(string1,substring)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
string1	Required	The first string.
substring	Required	The second substring from which you want to retrieve the occurrence of the first index value.

Example

```
when 1 greetings g with stringlastindexof(g.wish,”l”) as result = 9 then response with
body =”$result}”
```

The following table describes the result for the event property value that you pass:

Usage	Description	Result
g.wish="hello world"	Returns the index of l on its last occurrence.	3

stringlength

The stringlength analytic returns the length of given string.

Syntax

```
stringlength(string)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
string1	Required	The string whose length you want to retrieve.

Example

```
when 1 strleng s with stringlength(s.value) as result =6 then response with body="$
{result}"
```

The following table describes the result for the event property value that you pass:

Usage	Description	Result
s.value="Hello"	The length of the string.	5

substring

The substring analytic returns substring of the string based on the index provided.

Syntax

```
substring(string, start index,[end index])
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
string	Required	The string whose substring value you want to retrieve.
start index	Required	The starting index inclusive of the substring. If you do not specify the end index, the entire substring from the index specified will be retrieved.
end index	Optional	The end index exclusive of the substring. The substring retrieved will be within the range of the starting and the ending index.

Example

```
when 1 substr s with substring(s.greetings,6) as result !=””  then response with body=”${result}”
```

The following table describes the result for the event property value that you pass:

Usage	Description	Result
s.string1="Hello World"	Returns the substring after the index 6.	World

sum

The sum analytic adds a list of numbers.

Syntax

```
sum(number1,[number2],...])
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number	Required	The first number argument that you want to add
number2	Optional	The other number arguments that you want to add.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Text
Watchlist values	Text

Example

```
WHEN 2 stock WITH sum(s.price) as result =100 then response with body=”${result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.price[1]=25 s.price[0]=75	Returns the sum of the two events.	100

trim

The trim analytic removes all spaces from the text except for single spaces between words.

Syntax

```
trim(text1,[text2],...)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
text1	Required	The text from which you want spaces removed.
text2	Optional	The other list of text from which you want spaces removed.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Text
Watchlist values	Text

Example

```
when 1 stock s with trim(s.period) as result =”Second Quarter” then response with body=”${result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.period=" Second Quarter "	Removes all spaces from text except for single spaces between words.	Second Quarter

truncate

The truncate analytic truncates a number to an integer by removing the fractional part of the number.

Syntax

```
truncate(number1,[number2],[num_digits])
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
number	Required	The number you want to truncate.
number2	Optional	The number you want to truncate.
num_digits	Optional	The number specifying the precision of the truncation. The default value for num_digits is 0 (zero).

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	Real number
Watchlist values	Real number

Example

```
when 1 stock s with truncate(s.value) as result=10 then response with body="${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.value=10.9	Truncates 10.9 by removing the fractional part of the number.	10

upper

The upper analytic converts the text to upper case.

Syntax

```
upper(text1,[text2],...)
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
text1	Required	The text you want to convert to uppercase.
text2	Optional	The other text you want to convert to uppercase.

The following table provides the datatype for the parameters you can pass:

Parameter Input Options	Datatype
Event property	String
Watchlist values	String

Example

```
when 1 stock s  with upper(s.symbol) as result= “INFA”  then response with body="$
{result}”
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
s.symbol= infa	Returning the upper case of the text	INFA

Error Condition

The analytic does not run when there is a type mismatch, that is, when the values contain nonnumeric data. On type mismatch, the analytic returns an error, stops the rule evaluation, and writes the mismatch error to the log file.

xpath

The xpath analytic navigates to a specific element and attribute in the XML using the xpath expression and returns the list of objects that match.

Syntax

```
xpath(xml content,"xpath expression")
```

The following table describes the required and optional arguments:

Argument	Required Optional	Description
xml content	Required	Any element that returns the xml content. The element can be a web service analytic that returns an xml document.
xpath expression	Optional	An xpath expression.

Example

```
WHEN 1 rssnews rn WITH xpath(rn.contents,"/rss/channel/headline") as result contains
"World Cup Finalists" then response with body="${result}"
```

The following table describes the result for the event property value that you pass:

Event Property Value	Description	Result
World Cup Finalists Return Home	Navigates to the specific element and attribute.	World Cup Finalists Return Home

This rule activates when an rssnews event with a headline of "World Cup Finalists Return Home" occurs.

Predefined Operators

RulePoint provides a predefined set of operators that you can use when writing rules. You cannot create instances of the predefined operators.

The following table describes the predefined operators, arranged in alphabetical order:

Filters/Operators	Description
contains	Performs a search comparison on event properties.
equalsSet	Verifies that event sets have equal members with a comparison set, regardless of their order.
has	The has construct provides the same functionality as the contains analytic.
in	Performs word and substring searches on event properties.
match	Verifies that all events have the same value for the specified property.

Filters/Operators	Description
unmatched	Verifies that all of the events do not have the same value for the specified property, but the values are not required to be unique.
search	Enables Boolean search logic and Lucene search expressions.
unique	Verifies that all events have a different value for the specified property.
distinct	Used before the `with` clause in DRQL, and verifies that all events have a different value for the specified property.
group by	Used before the `with` clause in DRQL, and verifies that all events have the same value for the specified property.

contains

The contains operator performs word and substring search on event properties, but functions differently based on operations of the rule.

The contains analytic computes the following types of object:

- Item. An item is a property of an event, a text watchlist, or a single value returned from an analytic function.
- List. A list is a watchlist, an array returned from an analytic function, or a temporary variable list.
- Set. A set is an event set that is typically pooled from a WHEN n set rule.

Item

When the item object *contains an item*, the contains analytic returns true if the item on the right exists in the item on the left. When a rule includes two items, the contains analytic uses a substring match to check whether the item contains all elements of the set. For example, in the following rule, the analytic checks whether the news tagline contains the phrase, *days*:

```
WHEN 1 news n WITH n.tagline contains "days" then response
```

When the item object *contains a list*, the contains analytic returns true if any element in the list exists in the item on the left. When the rule includes an item and then a list, the analytic uses a word boundary match to check whether the single item contains any of the items in the list. For example, in the following rule, the analytic verifies whether the tagline contains any of the elements in the following watchlist:

```
WHEN 1 news n WITH n.tagline contains watchlist:myList then response
```

When the item object *contains a set*, the contains analytic returns true if all elements in the set exist in the item on the left. When the rule includes an item and then a set, the analytic uses a word boundary match to check whether the item contains all elements of the set on the right. For example, in the following rule, the analytic verifies whether the tagline contains all of the keywords in the following set:

```
WHEN 1 news n WITH n.tagline contains mySet then response
```

List

When the list object *contains an item*, the contains analytic returns true if any element in the left list is an exact match to the item. When the rule includes a list and then an item, the analytic checks whether any of the elements in the list is an exact match to the item on the right. For example, in the following rule, the analytic verifies whether the watchlist, *myList*, contains the tagline:

```
WHEN 1 news n WITH watchlist:myList contains n.tagline then response
```

When the list object *contains a list*, the contains analytic returns true if any element in the left list is an exact match for any item in the right list. When the rule includes a list and then a list, the analytic checks whether any of the elements in the list on the left is an exact match to any item on the list on the right hand side. For example, in the following rule, the analytic verifies whether the tagline contains any of the elements in the *myList* watchlist:

```
WHEN 1 news n with sqlQuery(n.id) contains watchlist:myList then response
```

When the list object *contains a set*, the contains analytic returns true if any element of the list is an exact match for all items in the set. The elements do not have to be the same element in the list. When the rule includes a list and then a set, the analytic checks whether any of the elements in the list on the left is an exact match to all items in the set on the right hand side. For example, in the following rule, the analytic verifies whether the *myList* watchlist contains all of the elements in the tagline set:

```
WHEN 1 news n WITH watchlist:myKeywords contains n.tagline then response
```

Set

When the set object *contains an item*, the contains analytic returns true if the item on the right exists in all elements in the set. When the rule includes a set and then an item, the analytic uses a word boundary match to check whether the item on the right is a match to all items in the set. For example, in the following rule, the analytic verifies whether the tagline set contains the keyword in the *myItem* watchlist:

```
WHEN 3 news n WITH n.tagline contains Watchlist:myItem then response
```

Note: The behavior is different when you use the past() analytic with the contains() analytic. The past() analytic computes events as a set, but in this instance it considers the events as a list. Therefore, RulePoint checks whether any of the elements in the list is an exact match to the item on the right as opposed to word boundary matching.

When the set object *contains a list*, the contains analytic returns true if any element of the list exists in all elements in the set. The elements do not have to be the same element in the list. When the rule includes a set and then a list, the analytic uses a word boundary match to check whether any element of the list exists in all items in the set. For example, in the following rule, the analytic verifies whether the *myList* watchlist set contains all of the elements in the tagline:

```
WHEN 3 news n WITH n.tagline contains Watchlist:myList then response
```

When the set object *contains a set*, the contains analytic returns true if all elements in the right set exist, using a word boundary match, in all elements in the left set . When the rule includes a set and then a set, the analytic uses a word boundary match to check whether all elements in the right hand set exist in all elements of the left hand set. The elements do not have to be the same element in the left set. For example, in the following rule, the analytic verifies whether the tagline set contains all of the elements in the *mySet* watchlist:

```
WHEN 3 news n, 3 Stock s WITH n.tagline contains s.symbol then response
```

equalsSet

The equalsSet operator verifies that event sets have the same members as a comparison set, regardless of their order.

The equalsSet operator returns true if the set on the left side contains all the elements in the set on the right-hand side. The two sides also must be equal in size.

For example, the following rule checks if you can find all the elements of the stock symbol set in the comma delimited string list (the right operand):

```
WHEN 3 stock s WITH s.symbol  equalsSet "XYZ, ABCD, EFG" then response
```

The following rule checks if you can find all of the elements of the stock symbol set in the comma delimited string list (the left operand):

```
WHEN 3 stock s WITH "XYZ, ABCD, EFG"  equalsSet s.symbol then response
```

The following rule checks whether all of the elements in the p.first_name set contains all of the items in the list returned by the SQL analytic:

```
WHEN 2 person p WITH p.first_name  equalsSet getPersonSql("EmployeeList") then response
```

The equalsSet operator returns true if the size of the list returned by the SQL analytic is equal to *p.first_name set size (size=2)*, and if the set and list contain the same elements.

The equalsSet operator is similar to the contains operator except that you can place a comma delimited string list on the right-hand side.

has

The has operator provides the same functionality as the contains operator.

See "contains" on page 178.

in

The in operator performs both word and substring searches on event properties, but functions differently based on the operations that the rule requires. The in operator provides the same functionality as the contains operator and the has operator except that the arguments are swapped.

The in operator computes the following types of object:

- Item. An item is a property of an event, a text watchlist, or a single value returned from an operator function.
- List. A list is a watchlist, an array returned from an operator function, or a temporary variable list.
- Set. A set is an event set that is typically pooled from a WHEN n set rule.

Item

An item in an item returns true if the item on the left exists in the item on the right. When a rule includes two items, the in operator uses a substring match to check whether the item on the right contains the item on the left. For example, in the following rule, the operator checks whether the phrase, *days* is in the news tagline:

```
WHEN 1 news n WITH "days" in n.tagline then response
```

A list in an item returns true if any element in the list exists in the item on the right. When the rule includes a list and then an item, the operator uses a word boundary match to check whether the single item contains any of the items in the list. For example, in the following rule, the operator verifies whether any of the elements in the following watchlist is in the tagline:

```
WHEN 1 news n WITH watchlist:myList in n.tagline then response
```

A set in an item returns true if all elements in the set exist in the item on the right. When the rule includes a set and then an item, the operator uses a word boundary match to check whether all of the elements in the set are in the item. For example, in the following rule, the operator verifies whether all of the keywords of *mySet* is in the tagline:

```
WHEN 3 news n WITH mySet in n.tagline then response
```

List

An item in a list returns true if any element in the right list is an exact match to the item in the left list. When the rule includes and item and then a list, the operator checks whether any of the elements in the list is an exact match to the item on the left. For example, in the following rule, the operator verifies whether the watchlist, *myList*, is in the tagline:

```
WHEN 1 news n WITH n.tagline in watchlist:myList then response
```

A list in a list returns true if any element in the right list is an exact match for any item in the left list. When the rule includes a list and then a list, the operator checks whether any of the elements in the list on the right is an exact match to any item on the list on the left. For example, in the following rule, the operator verifies whether the tagline contains any of the elements in the *myList* watchlist:

```
WHEN 1 news n WITH n.tagline in watchlist:myList then response
```

A set in a list returns true if any element of the list is an exact match for all items in the set. The elements do not have to be the same element in the list. When the rule includes a set and then a list, the operator checks whether any of the elements in the list on the right is an exact match to all items in the set on the left. For example, in the following rule, the operator verifies whether the *myList* watchlist contains all of the elements in the tagline set:

```
WHEN 1 news n WITH n.tagline in watchlist:myKeywords then response
```

Set

An item in a set returns true if the item on the left exists in all elements in the set. When the rule includes an item and then a set, the operator uses a word boundary match to check whether the item on the left is an exact match to all items in the set. For example, in the following rule, the operator verifies whether the tagline set contains the keyword in the *myItem* watchlist:

```
WHEN 3 news n WITH myItem in n.tagline then response
```

A list in a set returns true if any element of the list exists in all elements in the set. The elements do not have to be the same element in the list. When the rule includes a set and then a list, the operator uses a word boundary match to check whether any element of the list exists in all items in the set. For example, in the following rule, the operator verifies whether the *myList* watchlist set contains all of the elements in the tagline:

```
WHEN 3 news n WITH n.tagline contains Watchlist:myList then response
```

When the set object *contains a set*, the contains operator returns true if all elements in the right set exist, using a word boundary match, in all elements in the left set . When the rule includes a set and then a set, the operator uses a word boundary match to check whether all elements in the right hand set exist in all elements of the left hand set. The elements do not have to be the same element in the left set. For example, in the following rule, the operator verifies whether the tagline set contains all of the elements in the *mySet* watchlist:

```
WHEN 3 news n, 3 Stock s WITH n.tagline contains s.symbol then response
```

match

The match operator is a unary operator that verifies that all events have the same value for the specified property.

The match operator is useful when dealing with an event set with multiple events where you want to group events by a common property value.

For example, the following rule verifies that three credit card transactions is processed by the same merchant:

```
WHEN 3 transaction t  WITH match(t.Merchant_ID)  then response
```

The following rule verifies that three credit card transactions were processed by the same merchant and the total transaction value is greater than $500:

```
WHEN 3 transaction t WITH match(t.Merchant_ID) AND sum(t.Purchase_Amount)> 500 then
response
```

consider that two stock events have the same symbol, and two bond events have the same symbol. The following rule verifies that the stock symbols do not require to be the same as the bond symbols:

```
WHEN 2 stock s, 2 bond b WITH match(s.symbol) AND match(b.symbol) then response
```

The following rule provides the same result using a concatenated syntax:

```
WHEN 2 stock s, 2 bond b WITH match(s.symbol, b.symbol) then response
```

Note: The "match" operator is deprecated. The "group by" operator replaces the "match" operator.

See: "Using the distinct, group by, and ummatched Filters" on page 184.

notMatch

The notMatch operator is a unary operator that verifies that all events do not have the same value for the specified property, but the values do not require to be unique.

For example, the following rule verifies that the merchants in three credit card transactions are not the same:

```
WHEN 3 transaction  t WITH notMatch(t.Merchant_ID) then response
```

Consider that two stock events have the same symbol, and two bond events have the same symbol. The following rule verifies that the stock symbols do not require to be the different as the bond symbols:

```
WHEN 3 stock s, 3 bond b WITH notMatch(s.symbol) AND notMatch(b.symbol) then response
```

The following rule provides the same result using a concatenated syntax:

```
WHEN 3 stock s, 3 bond b WITH notMatch(s.symbol, b.symbol)  then response
```

search

The search operator enables Boolean search logic, such as AND, OR, NOT, and Lucene search expressions, such as * and ?.

For example, the following rule uses Boolean search logic to find a person's name in a news story:

```
WHEN 1 news  n WITH story search "(Joe OR Chris) AND Mike" then response
```

The above rule evaluates true when a news story contains the name Joe and Mike or Chris and Mike. In the example, the use of the parentheses is required to enforce precedence.

For example, the following rule uses Lucene search expressions to find variations of the word "test":

```
WHEN 1 news  n WITH story search "test*"  then response
```

The rule evaluates true when a news story contains any variation of the word test, such as test, tests, tester, testing.

The search operator does not support the escape character '\".

unique

The unique operator is a unary operator that verifies that all events have a different value for the specified property.

You cannot reference more than one property for a topic of a rule. The unique operator works with one property for a topic of a rule.

For example, the following rule verifies that three credit card transactions were processed by different merchants:

```
WHEN 3 transaction t WITH unique(t.Merch_ID) then response
```

The following rule verifies that three credit card transactions were processed by different merchants and the total transaction value is greater than $500:

```
WHEN 3 transaction  t WITH unique(t.Merch_ID) AND sum(t.Purchase_Amt) > 500 then response
```

Consider that two stock topic events have the different symbols, and two bond events have different symbols. The following rule verifies that the stock symbols do not require to be different from the bond symbols:

```
WHEN 2 stock s, 2 bond b WITH unique(s.symbol) AND unique(b.symbol) then response
```

The following rule provides the same result using a concatenated syntax:

```
WHEN 2 stock s, 2 bond b WITH unique(s.symbol, b.symbol)  then response
```

Note: The unique operator is deprecated. The "distinct" operator replaces the "unique" operator.

See: "Using the distinct, group by, and ummatched Filters" on page 184

group by

The group by filter verifies that all events have the same value for the specified property, and is used before the `with` clause in DRQL.

The group by filter is useful when you are dealing with an event set that contains multiple events and you want to group the events by a common property value.

For example, the following rule verifies that three credit card transactions were processed by the same merchant:

```
WHEN 3 transaction t group by t.Merchant_ID then response
```

The following rule verifies that three credit card transactions were processed by the same merchant and the total transaction value is greater than $500:

```
WHEN 3 transaction t group by t.Merchant_ID WITH sum(t.Purchase_Amount)> 500 then
response
```

Consider that two stock events have the same symbol, and two bond events have the same symbol. The following rule verifies that the stock symbols are not the same as the bond symbols:

```
WHEN 3 transaction t group by t.Merchant_ID WITH sum(t.Purchase_Amount)> 500 then
response
```

Note: The "group by" operator replaces the "match" operator.

distinct

The distinct filter verifies that all events have a different value for the specified property, and is used before the `with` clause in DRQL.

For example, the following rule verifies that three credit card transactions were processed by different merchants:

```
WHEN 3 transaction  t distinct t.Merch_ID WITH then response
```

The following rule verifies that three credit card transactions were processed by different merchants and the total transaction value is greater than $500:

```
WHEN 3 transaction t distinct t.Merch_ID  with sum(t.Purchase_Amt) > 500 then response
```

Consider that two stock topic events have different symbols, and two bond events have different symbols. The following rule verifies that the stock symbols do not have to be different from the bond symbols:

```
WHEN 2 stock s, 2 bond b distinct s.symbol,b.symbol then response
```

Note: The "distinct" operator replaces the "unique" operator.

unmatched

The unmatched filter verifies that all events have the same value for the specified property, but the values do not require to be unique.

For example, the following rule verifies that the merchants in the three credit card transactions are not the same:

```
WHEN 3 transaction t unmatched merchant_id then response
```

In this case, if the merchant_id values across the three events are 123, 124, and 123, the filter considers the events as unmatched events.

The two topics have the following rule syntax:

```
WHEN 3 stock s, 3 bond b unmatched s.symbol, b.symbol then response
```

Using the distinct, group by, and ummatched Filters

The distinct and group by operators filter sets of events before they are processed by a rule. Each of these filtering analytic operators take an argument that specify a single property, which is used as the basis for the filter.

You can specify any number or combination of these operators in a rule. For example, all of the following rules are valid:

```
when 2 stock s group by s.symbol with ..
when 2 stock s group by s.symbol,s.price with ..
```

Understanding Filtering Behavior

The filtering behavior for these operators is as follows:

- group by. All of the events must have the same value for the specified property.
- unmatched. All of events do not have the same value for the specified property, but the values are not required to be unique.
- distinct. All of the events must have a different value for the specified property.

For example, consider the following table:

Values	Passes Match	Passes Unique	Passes Unmatched
A	Y	Y	Y
A, B	N	Y	Y
A, A, A	Y	N	N
A, A, B	N	N	Y
A, B, C	N	Y	Y

CHAPTER 12

DRQL

This chapter includes the following topics:

DRQL Overview

In the RulePoint user interface, you can create rules using the advanced mode, rule wizard, and templates. You can create an advanced rule by writing the rule in the Detect and Respond Query Language (DRQL) syntax of RulePoint. The DRQL of RulePoint has its own syntax and grammar.

A DRQL rule statement consists of WHEN, WITH, and THEN clauses that define the rule. The WHEN clause specifies the topics and the number of events required for each topic. The WITH clause specifies the conditions that must be met. The THEN clause specifies the responses to invoke when the rule activates.

For example, in the following rule when two stock events with a stock symbol of XYZ come into the system, you want to send an email to the address, broker@company.com:

```
WHEN 2 stock WITH symbol = "XYZ" THEN EmailBrokerResponse WITH to="broker@company.com"
```

Note:

- The comparison of string literals and topic property string values are case sensitive. For example, when evaluating a rule, xyz does not equal XYZ.
- To improve readability, the DRQL examples show all keywords and operators, such as WHEN, THEN, and OR in all capital letters. These elements are not case sensitive.

- The examples in this section assumes the existence of Stock and News topics with the appropriate properties and a well defined email responder service and a response named EmailBrokerResponse.

Example

Consider the statement, *when milk is low, go to grocery store*. In DRQL syntax, the statement is broken down as shown in the following table:

WHEN	WITH	THEN
milk	is low	go to grocery store

If required, you can increase the complexity of each clause, as shown in the following table:

WHEN	WITH	THEN
2% milk	is more than 1/2 empty	go to grocery store tonight
2% milk	is low and past sell-by date	go to grocery store now

You can include multiple topics with different conditions as shown in the following table:

WHEN	WITH	THEN
2% milk	is more than 1/2 empty	go to grocery store tomorrow
eggs	less than a dozen	go to grocery store tomorrow

Consider the following example of a stock rule written using DRQL:

```
WHEN stock s WITH s.symbol="XYZ" AND s.price > 10 THEN EmailResponse
```

The following table describes the meaning of each clause in the example:

WHEN	WITH	THEN
WHEN stock s	WITH s.symbol="XYZ" AND s.price > 10	THEN EmailResponse
When a Stock topic event, s, occurs at any time...	When Stock topic event s has a symbol of XYZ and a price greater than $10...	When Stock topic event s has a symbol of XYZ and a price greater than $10, send my defined Email response.

Note:

- When creating rules in Advanced Mode, you must assign an alias to the topic. In this case, s references the topic stock as its alias. This is to differentiate which property to derive from which topic if you have more than one topic.
- By default, the AND operator takes precedence over the OR operator, but you can change the order of precedence.
- You must separate consecutive topics and response properties with commas.

Topic Aliases in DRQL

Topic aliases in DRQL helps you differentiate and reference multiple topics in a rule. You do not require an alias when you use a single topic. You define aliases in a WHEN statement so that you can easily reference multiple topics in other parts of the rule.

In the following example, an alias for stock is not required:

```
WHEN 1 stock
```

If a rule uses multiple topics, you must use an alias. You must specify an alias for the first topic, such as `s` for stock which is a stock event. Specify a second unique alias for the second topic, such as n for news which is a news event.

```
WHEN 1 stock s, 1 news n
```

Note: If you use an alias as reference within the condition of a rule, you must also use the alias in the responder parameters as opposed to using the dotted topic name as reference. Avoid mixing the use of one reference type with the other in the same rule.

You can then use these aliases to identify the topic for which you want to evaluate a particular property. For example, in the following rule, `s.symbol` indicates that you want to evaluate the symbol property of the stock event. Also, `n.headline` indicates that you want to evaluate the headline property of the News event.

```
WHEN 1 stock s, 1 news n WITH s.symbol = "XYZ" OR n.headline contains "XYZ"
```

WHEN Clause

Use the WHEN clause to specify the topics and the number of events required for each topic.

For example, the following clause looks for one stock event:

```
WHEN 1 stock
```

The following clause looks for five stock events `s` and one news event `n`:

```
WHEN 5 stock s, 1 news n
```

While creating a rule, if you do not specify the number for the topic in the WHEN clause , you receive an alert to specify the number of topics. If you proceed without specifying a number, the rule will look for all events for the specified topic.

WITH Clause

Use the WITH clause to specify the conditions that you want to apply to the event properties in your WHEN clause.

Within a condition, you can include two operands and an operator, or a single special case unary operator. Within a rule, you can combine multiple conditions.

Specifying Topic Properties

In a single topic rule, when referencing a topic property in a condition, you can include the full topic name or the topic alias followed by dot (.) and the topic property. However, in a multiple topic rule, you must reference the topic.

For example, the following clause looks for stock event s with a stock symbol of XYZ:

```
WITH symbol = "XYZ"
```

You can also define the same condition using the alias s:

```
WITH s.symbol = "XYZ"
```

The use of an alias becomes important when you are specifying multiple topics in a rule and need to identify the topic for which you are evaluating a particular property.

You can define the same condition using the full topic name:

```
WITH stock.symbol = "XYZ"
```

Using Operators

Between two conditions, you can use the AND, OR, AND NOT, and OR NOT operators.

For example, the following clause looks for stock event s with a stock symbol of XYZ and a price greater than $10:

```
WITH symbol = "XYZ" AND price > 10
```

The same rule with alias:

```
WITH s.symbol = "XYZ" AND s.price > 10
```

The following clause looks for stock event s with any stock symbols, if they match:

```
WITH match(symbol)
```

THEN Clause

Use the THEN clause to specify the responses that need to be invoked when the rule activates.

You do so by referencing a previously defined response and any necessary parameters. Your defined responses provide default values to execute your response, but you can override these default values directly in the rule.

In the following example, when the conditions of your rule are met, the THEN clause sends an email to broker@company.com using the parameters defined in the response named EmailBrokerResponse:

```
THEN EmailBrokerResponse WITH to="broker@company.com"
```

To invoke multiple Responses, separate the Responses with “AND”:

```
THEN EmailBrokerResponse WITH to="broker@company.com" AND IMBrokerResponse WITH
to="broker@company.com"
```

Using Quotation Marks

When you write rules, you must place quotation marks around string constant values and any item that contains a space.

For example, in the following rule, you must place the values for the symbol, to, and body parameters in quotation marks because they are string constant values. You must also place the name of the response in quotation marks because it contains a space.

```
WHEN 1 stock WITH symbol = "XYZ" AND price > 10 THEN "Email Response" WITH
to="broker@company.com", body="stock quote update"
```

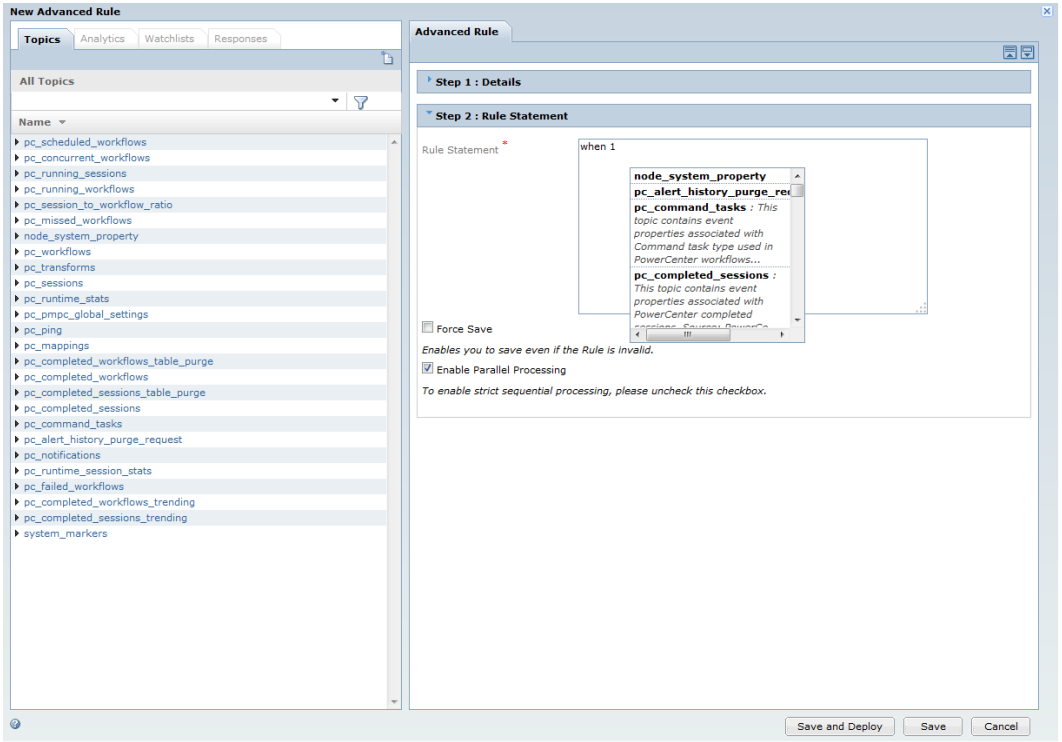
Do not place numeric values within quotation marks because RulePoint treats it as a string instead of a numeric value. In the example, notice that the value 10 is not in quotation marks.

Note: Do not use quotation marks when specifying values in the rule wizard because RulePoint assumes them to be present in the appropriate locations.

Using Autocomplete

To help you write DRQL syntax, autocomplete keeps track of information that you have recently typed and tries to anticipate what you want to type next. Autocomplete displays possible options for each statement identifier and topic, topic property, response, and response parameter names.

For example, when you start writing a rule, WHEN displays in an autocomplete list. After you select WHEN from the list, a list of possible topics displays in an autocomplete list, as shown in the following image:



Select an item, and that item is placed in the rule syntax. Autocomplete continues to list possible selections for each item.

System Property Keywords

The following table lists the system property keywords that you can use in rules or responses:

Keyword	Description
drql_rulename	Gets the name of the rule when referenced within a rule. For example, ${drql_rulename}.
drql_ruleid	Gets the ID of the rule when referenced within a rule. This keyword is used for deep linking back to the RulePoint user interface. For example, ${drql_ruleid}.
drql_response	Gets the name of the response. For example, ${drql_response}.
drql_eventformattedtime	Gets the time in milliseconds when the event was published in the system. The date is pre-formatted based on the default date and time format of the system. For example, ${drql_eventformattedtime(dd/MM/yyyy HH:mm:ss)}.
drql_eventtime	Gets the time in milliseconds when the event was published in the system. For example, ${drql_eventtime}.

Date and Number Formatter

You can format numbers in responses.

Use the date formatter to format an event property that contains the time in milliseconds. You can use the Java's SimpleDateFormat syntax to format an event from January 1, 1970:

```
${Mytopic.drql_evenformattedttime(1,date m/d/y)}
```

Use the number formatter to format an event property that contains a numeric value. You can use the Java's NumberFormat syntax to format an event that contains a numeric value. The format always appears last in the parenthesized format expression as shown in the following code:

```
${Stock.price(*,COMMA, number #,##0.0)}
```

Referencing Watchlists in a Rule

To reference a watchlist in a rule, you specify the watchlist using the following syntax:

```
watchlist:[WatchlistName]
```

The following rule shows the example of watchlist reference usage:

```
WHEN 1 stock WITH watchlist:myStocks contains symbol
```

Note: If a watchlist name contains a space, you must place the watchlist name inside double quotes as shown in the following syntax:

```
watchlist:"My Stock List"
```

Referencing Events in Responses

You can use the ${} function to display the values of a specified property or temporary variable in the response.

To display the values of that property, you must specify an event property within the braces of the function.

Referencing a Single Event with a Single Property Value

You can reference a property value of a single event in a rule using one of the following syntax:

```
${[topicProperty]}
```

or

```
${[topicAlias].[topicProperty]}
```

or

```
${[topicName].[topicProperty]}
```

Here, *topicAlias* is the name of the alias that you assign to a topic, such as s for stock. *topicName* is the name of the topic and *topicProperty* is the name of the event property.

Note: Use of topic name or topic alias is optional while referencing single topic rules.

For example, the following rule specifies that the body of the response will display the price of the one XYZ stock event matched by the rule in place of ${price}:

```
WHEN 1 stock WITH symbol = "XYZ" AND price < 90 THEN EmailBrokerResponse WITH
to="broker@company.com", body="XYZ is trading at ${price}."
```

The result of the rule is an email containing “XYZ is trading at 72.”

If you reference a topic property within a response that does not contain a value, the rule evaluates, but the response output displays null where the value appears normally.

Referencing One Property Value in an Event Set

When using event sets with multiple values for a single property, you can display the property value for one of the events in the set using the following syntax:

```
${[topicProperty]([n])}
```

or

```
${[topicAlias].[topicProperty]([n])}
```

or

```
${[topicName].[topicProperty]([n])}
```

Here, *topicAlias* is the name of the alias that you assign to a topic, such as s for stock. *topicName* is the name of the topic, *topicProperty* is the name of the event property, and *n* is the number of the event in the set for which you want output.

Note: Rules maintain the order of events from the latest to the oldest.

For example, the following rule specifies that the body of the response which will display the three prices of the XYZ stock event s matched by this rule:

```
WHEN 3 stock s WITH symbol = "XYZ" AND price < 90 THEN EmailBrokerResponse WITH
to="broker@company.com", body="XYZ is trading at the following prices. One price is $
{s.price(1)}, another price is ${s.price(2)}, and the final price is ${s.price(3)}."
```

Here, ${s.price(1)},${s.symbol(1)} is the value of latest event and ${s.price(3)},${s.symbol(3)} has the value of the oldest event.

Referencing a Range of Properties from an Event Set

When using event sets with multiple values for a single property, you can display more than one property value for each event in the set.

You can define the data in a rule using the following syntax:

```
${[TopicName].[TopicProperty](n:m, [format])}
```

Here, *n* represents the offset within the range of events, and *m* represents the count of events starting from the offset. For example, five events from the latest to the oldest are e5,e4,e3,e2,e1 and the notation is 2:4. The formatting starts from 2 (e4) and the count of the four events (e4,e3 ,e2 and e1) are considered for the formatting.

The following rule emails a broker with the last two out of three prices, separated by a comma:

```
WHEN 3 stock s WITH symbol = "XYZ" AND price < 90 THEN EmailBrokerResponse WITH
to="broker@company.com", body="The last two prices of XYZ are ${s.price(2:3,COMMA)}."
```

Referencing the Oldest Property Value in an Event Set

When using event sets with multiple values for a single property, you can display the oldest property value for each event in the set.

You can define the oldest property value in a rule using the following syntax:

```
${[TopicName].[TopicProperty](last)}
```

Referencing the Property Value Relative to Each Event in an Event Set

When using the responders, Event Transformer, Event Publisher, and Event Recorder, each event in the event sets is processed one at a time.

To access the property of the processed event, use the *rel* property selection keyword:

```
${[TopicName].[TopicProperty](rel)}
```

Referencing All Property Values in an Event Map

When using event sets with multiple values for a single property, you can display all the property values for each event in the set. You can also specify the format in which you want to display the values.

You can define this information in your rule using one of the following syntax:

```
${[topicProperty](*, [Format])}
```

Note: Use this option only if there is a single topic.

or

```
${[topicAlias].[topicProperty](*, [Format])}
```

or

```
${[topicName].[topicProperty](*, [Format])}
```

Here, *topicAlias* is the name of the alias that you assign to the topic, such as s for stock. *topicName* is the name of the topic, *topicProperty* is the name of the event property, and *Format* is the format to use when displaying the values in the response. The default output format is a space separated list of string values.

If you want to change the default output format, specify one of the formats using its format code as shown in the following table:

Format Code	Display Format
TAB	Tab separated list of string values, where <TAB> is the tab character. For example: `ABCD<TAB>EFG<TAB>XYZ`
COMMA	Comma separated list of string values. For example: `ABCD,EFG,XYZ`
NL	New line separated list of values. For example: - ABCD - EFG
COLON	Colon separated list of string values. For example: `ABCD:EFG:XYZ`

XML	`<propertyname id=nnnn>value</propertyname>`
	Here, *nnnn* is the array index of this element, starting from one and incrementing by one.
	For example, if the response contains three events, the first event into the system is numbered 1. The XML format for these events would be as follows:
	`<symbol id=1>ABCD</symbol>`
	`<symbol id=2>EFG</symbol>`
	`<symbol id=3>XYZ</symbol>`

BR	HTML break separated list of string values, where is the HTML break tag. For example: `ABCD EFG XYZ`

For example, the following rule specifies that the body of the response that displays all the prices of the XYZ stock event s matched by this rule:

```
WHEN 1 stock WITH symbol = "XYZ" AND price < 90 THEN EmailBrokerResponse WITH
to="broker@company.com", body="XYZ is trading at ${price(*,COMMA)}"
```

Consider that the six stock events received from the source had price values of 72, 75, 71, 72, 74, and 73. The result of this rule is an email containing “XYZ is trading at 73,74,72,71,75,72”.

Referencing Temporary Variables in Responses

You can specify temporary variables previously defined by the AS clause in responses.

To specify the temporary variables, you can set the analytic result as a variable, and then define a condition for the result as shown in the following response:

```
WHEN 3 stock s, dailystockinfo d group by s.symbol with average(s.price) AS AvgPrice > 0
AND AvgPrice > d.threedayaverage THEN EmailBrokerResponse WITH to="broker@company.com",
body="XYZ is trading at the average price of ${AvgPrice}."
```

Note:

- The group by analytic forces the response to evaluate matching symbols.

- Setting the analytic result as a temporary variable enables you to reference the result in the responses. Using the previous example, you can include the AvgPrice temporary variable in the response DRQL to display its value in the response.
- The assignment of a particular result to a temporary variable is valid only during the evaluation of that rule. After the evaluation, the value of that temporary variable is removed.

Selecting Topics

RulePoint requires that all rules must reference at least one topic.

You can also reference multiple topics. For each topic, you can define no conditions, one condition, or multiple conditions.

Note: If you do not define any conditions for a topic, RulePoint evaluates all events for that rule and invokes the specified responses. This might impact system performance when receiving a high volume of events that activate the rule.

Specifying the Number of Topic Occurrences

Usually, you might want to specify the number of event occurrences from that topic required to activate a rule.

If you specify an event count for the topic, the rule activates when the specified number of events meet the rule condition. For example, the following rule activates when any one stock event has a price property that is greater than $10:

```
WHEN 1 stock WITH price > 10 then response
```

If 100 stock events occur with a stock price greater than $10, the rule activates 100 times.

The following rule activates when five stock events have a price greater than $10:

```
WHEN 5 stock WITH price > 10 then response
```

If you do not specify the number of event occurrences required to activate a rule, the rule activates for all events that meet the condition.

In the following example, the one news event must contain all of the symbols for every stock event that occurs:

```
WHEN stock s, 1 news n WITH n.headline contains s.symbol then response
```

You can specify the number of occurrences for the stock event. The following rule activates when the headline property of one News event contains the same stock symbol found in one stock event.

```
WHEN 1 stock s, 1 news n WITH n.headline contains s.symbol then response
```

Consider that you want to see whether the total amount of transactions for the day exceeds $10,000. You do not require to specify the number of events that meet the rule condition, because you want to evaluate all of the transaction events that occur that day. The following rule activates when the sum of all of the transactions for the day exceeds 10,000:

```
WHEN BankTransaction t WITH sum(t.amount) > 10,000 WITHIN 1 day then response
```

Specifying the Time Frame

Every topic has a default expiration time frame specified when defining the topic. You can override that time frame by specifying a time frame within an individual rule.

You can set the expiration time frame of events at multiple places in RulePoint. The lowest expiration setting always takes precedence. The expiration time frame of events are shown in the following list:

- System default expiration
- Topic expiration
- Event specific timestamp
- Rule time frame

For example, if you set the system default expiration to one minute and set the topic specific expiration to seven days, those topics still expire after one minute. Conversely, if you set the topic specific expiration to 30 seconds, a value smaller than the system default expiration, then that expiration setting takes precedence.

You can use the within statement to specify a time frame to evaluate a rule. You can specify days, hours, minutes, and seconds. For example, the following rule activates when five stock events with symbol equal to XYZ occur within 10 minutes of receiving the first stock event with symbol equal to XYZ:

```
When 5 stock with symbol="XYZ" slide within 10 minutes then response
```

The value you set must be shorter than both the default value and the topic expiration time frames.

RulePoint evaluates all of the topics and conditions in the rule using the expiration time frame. You cannot specify different time frames for different topics within the same rule. Specifying a time frame in one rule does not affect the time frame for the topic in any other rule.

Specifying Event Set Behavior

When selecting topics, you can specify an event set behavior to describe how you want RulePoint to dispose of events after they have participated in the activation of a rule.

The following event set behaviors are available and are listed in priority order:

1. removeAll. Evaluate each event once and then remove all participating events from future evaluation. This is the default value. You can use the removeAll behavior if you do not want to specify another behavior in the rule.
2. removeOldest. Remove the oldest participating event from future evaluation.
3. removeNone. Keep all participating events and use them in future evaluation.
4. removePrevious. Remove all previous events and keep only the latest event in future evaluation

You would change the default behavior if you want to reevaluate the prior events along with new events.

For example, the events E1, E2, and E3 activate the following rule:

```
WHEN 3 stock WITH symbol = "XYZ"
```

Event ID	**Time stamp**
E1	10:14am
E2	10:20am
E3	10:30am

Based on which behavior that you specify, the events are either removed or kept for future rule evaluation as shown in the following table:

Event Set Behavior	Events Removed after Activation	Event Remaining after Activation
remove all	E1, E2, E3	-
remove oldest	E1	E2, E3
remove none	-	E1, E2, E3
remove previous	E1, E2	E3

You can specify a different behavior for each topic in a rule. To do so, place the topic name or topic alias in parentheses after the appropriate behavior. For example, in the following rule, the DailyStockInfo topic generates events with historical statistics for stocks:

```
WHEN 1 stock, 1 DailyStockInfo with sum(stock.price) as x >10 then response remove
all(stock), none(DailyStockInfo)
```

If you create a rule that compares current stock events with historical stock events, you might want to remove all stock events but keep all DailyStockInfo events.

Note: If you use more than one behavior with one topic, the behavior with the higher priority takes precedence.

You can define the same event set behavior using topic aliases:

```
WHEN 1 stock s, 1 DailyStockInfo d WITH sum(s.price) as x >10 then response remove
all(s) and none(d)
```

You also can specify one behavior for multiple topics at the same time. To do so, place the comma separated topic aliases in parentheses after the behavior as shown in the following rule:

```
WHEN 1 stock s, 1 dailystockinfo d WITH sum(s.price)>10 then response remove none(s,d)
```

Defining Conditions

In RulePoint, every condition must reference at least one topic, but not every topic must have a condition.

For example, the following rule activates when a stock event has a price property greater than 10 and when any news event is generated:

```
WHEN 1 stock s, 1 news n WITH s.price > 10 then response
```

RulePoint treats standalone topics, such as News in the previous rule, as though they have no conditions.

Note: For RulePoint to evaluate a rule, all of the properties referenced in the conditions of the rule must exist in the events present in the system.

Using Operators Within Conditions

Within a single condition, you must select an operator to compare the left operand and the right operand.

For example, the following rules uses the greater than and equal operators:

```
WHEN 1 stock s WITH s.price > 10
```

```
WHEN 1 stock s WITH s.symbol = "XYZ"
```

You can use the following operators while writing rules:

= (equal)

!= (not equal)

> (greater than)

>= (greater than or equal)

< (less than)

<= (less than or equal)

contains

equalsSet

has

in

match

notmatch

unmatched

search

unique

group by

distinct

The unmatched operators function differently based on whether the rule includes two items, an item and then a list, a list and then an item, or two lists. An item can be a single event, a text watchlist, or a single value returned from an analytic function. A list can be a list watchlist, an event set, or an array returned from an analytic function.

Avoid using unique, match, and not match operators. These operators exist only for backward compatibility. Instead, use the distinct, group by, and unmatched filters.

Equal and Not Equal Operators

You can use the equal and not equal operators to compare both text and numeric values.

You can use the equal and not equal operators for the following sets of conditions:

- When the rule includes two items, the equal operator verifies that the left hand item is equal to the right hand item.

  For example, the rule `WHEN 1 stock WITH price = 10` evaluates true after each of the following events arrives:

  ```
  Topic: Stock
  Event 1, Symbol ABCD, Price 10

  Event 2, Symbol XYZ, Price 10
  ```

The rule, `WHEN 1 stock WITH symbol = "XYZ"` evaluates true after each of the following events arrives:

```
Topic: Stock
Event 1, Symbol XYZ, Price 9

Event 2, Symbol XYZ, Price 10
```

- When the rule includes an item and a list, the equal operator verifies that the item must be equal to every item in the list. Consider an example of a set of three stock events. The equal operator verifies that all three stock event properties are equal to the item.

  The rule `WHEN 3 stock WITH price = 10` evaluates true after each of the following events arrives:

  ```
  Topic: Stock
  Event 1, Symbol ABCD, Price 10

  Event 2, Symbol XYZ, Price 10

  Event 2, Symbol EFG, Price 10
  ```

  The rule `WHEN 3 stock WITH symbol = "XYZ"` evaluates true after each of the following events arrives:

  ```
  Topic: Stock
  Event 1, Symbol ABCD, Price 11

  Event 2, Symbol XYZ, Price 10

  Event 2, Symbol EFG, Price 9
  ```

- When the rule includes two lists, the equal operator verifies that both lists have the same items in the same order. If order of the items in the list is not relevant, use the "contains" on page 178 analytic.

  For example, consider two watchlists, MyPrices and MyStocks. The Watchlist type is list. Watchlist values for MyPrices are 10, 15, and 12. Watchlist values for MyStocks are XYZ, ABCD, and EFG. The rules in this example contains the equal operator that verifies that all three stock event properties are equal to the items in the specified watchlist.

  The rule, `WHEN 3 stock s WITH s.price contains watchlist:MyPrices` evaluates true after the third of the following events arrives:

  ```
  Topic: Stock
  Event 1, Symbol EFG, Price 10

  Event 2, Symbol XYZ, Price 15

  Event 2, Symbol ABCD, Price 12
  ```

  The rule, `WHEN 3 stock s WITH s.symbol contains watchlist:MyStocks` evaluates true after the third of the following events arrives:

  ```
  Topic: Stock
  Event 1, Symbol XYZ, Price 14

  Event 2, Symbol ABCD, Price 11

  Event 2, Symbol EFG, Price 13
  ```

Numeric Operators

You use the numeric operators to compare numeric values.

You can use the following numeric operators while writing rules:

= (equal)

!= (not equal)

> (greater than)

>= (greater than or equal to)

< (less than)

<= (less than or equal to)

Note: If you use numeric operators to compare text, the rule evaluates as false.

You can use the numeric operators for the following sets of conditions:

- When the rule includes two items, the greater than operator verifies that the left hand item is greater than the right hand item.
  For example, in the following rule, the greater than operator verifies that all three stock event prices are greater than the specified stock price:

  ```
  WHEN 1 stock WITH price > 10
  ```

  The rule evaluates true after each of the following events arrives:

  ```
  Topic: Stock
  Event 1, Symbol EFG, Price 11

  Event 2, Symbol ABCD Price 12
  ```

- When the rule includes an item and a list, the greater than operator verifies that every item in the list must be greater than the value.

  For example, in the following rule, the greater than operator verifies that all three stock event prices are greater than the specified stock price:

  ```
  WHEN 3 stock WITH price > 10
  ```

  The rule evaluates true after each of the following events arrives:

  ```
  Topic: Stock
  Event 1, Symbol EFG, Price 12

  Event 2, Symbol ABCD Price 11

  Event 3, Symbol XYZ Price 12
  ```

- When the rule includes two lists, the greater than operator verifies that all of the items in one list are greater than all of the items in the other list.

  Consider a watchlist, MyPrices with values 10, 15, and 12. In the following rule, the greater than operator verifies that all three stock event prices are greater than all of the prices in the MyPrices watchlist:

  ```
  WHEN 3 stock WITH price > watchlist:MyPrices
  ```

  The rule evaluates true after the third of the following events arrives:

  ```
  Topic: Stock
  Event 1, Symbol EFG, Price 19

  Event 2, Symbol ABCD, Price 17

  Event 3, Symbol XYZ, Price 25
  ```

Analytic Operators

RulePoint provides the following predefined analytic operators:

- contains
- equalsSet
- has
- in
- search

For more information, see "Predefined Analytics" on page 114.

For information on the usage of match, notMatch, and unique operators, see "Using the distinct, group by, and ummatched Filters" on page 184.

Windows Clause in DRQL

You can configure sliding or tumbling windows in rules to set a boundary for selecting events from event streams. The windows determine what events the rule must process. The events activate or expire as determined by the window type.

Sliding Windows Clause

You can classify sliding windows based on time or count. The clause specifies a window size for the events either in time or in events.

Events expire due to the following reasons:

- In a time window, when an event exceeds the time limit, only those events expire. For an events window, when a new event arrives, only the oldest event expires if the number of events has exceeded the window count.
- Event sets expire when they match a rule condition. In sliding windows, only event sets that match the rule conditions expire. For example, for a "when 2" rule, when a third event arrives and forms an event set combination of (e3,e2) (e3,e1), only those event sets that match the rule conditions expire.

Sliding window based on time

You can configure the time frame in seconds, minutes, and days. The window takes a time interval relative to the time of the current event. If the time stamp of the current event is t, all the events between *t-window time* and *t* are considered in forming the event sets.

The `slide within n seconds` DRQL 1.1 syntax replaces `within n seconds of DRQL` 1.0.

The following single topic rule shows the usage of `slide within n seconds` over stock events:

```
WHEN 2  stocks WITH average(s.price) > 14  slide within 5 seconds  then testresponse
```

The following table shows how the rule maintains a sliding window for a size of 5 seconds and when the condition is 2:

Events - Price values	Time window	Incoming events	Combination of events set that can form when a new event arrives	Cache condition evaluation
10	0 seconds	e1	-	No activation
12	2 seconds	e2	(e2,e1)	No activation
14	4 seconds	e3	(e3,e1), (e3,e2)	No activation as the event set combination does not match the condition.

Events - Price values	Time window	Incoming events	Combination of events set that can form when a new event arrives	Cache condition evaluation
16	6 seconds	e4	(e4,e2), (e4,e3)	One activation seen for (e4,e3) as the condition matches the rule. e1 expires in time. e2 is still active. **Note:** When you use sliding windows in a rule, all the events which do not match the rule or those events that exist within the specified time continue to survive.
38	8 seconds	e5	e5	No activation. e2 expires with time and will not form an event set combination.

Note: Event set refers to a set of events that are formed according to the conditions of a rule. For example, if the condition is "when 2", the event set formed is (e1,e2). If the condition is "when 3", the set combination formed is (e1,e2,e3). Event set combination refers to the number of set combinations of events formed with the arrival of every new event. For example, if it is a "when 2" rule, and if the third event arrives, then the set combination formed is (e3,e2) (e3,e1).

Sliding window based on count

The window considers the count of events. With the arrival of the latest event, if the count exceeds the specified count limit, the oldest event expires. If there is an activation, only the matching event sets within the event set combination expire.

The following rule shows the usage of `slide within n events` over stock events:

```
WHEN 2  stocks WITH average(s.price) > 14  slide within 3 events  then testresponse
```

The following table shows how the rule maintains a sliding window for a size of 3 events and when condition is 2:

Events - Price value	Events count in window	Incoming events	Combination of events set that can form when a new event arrives	Cache condition evaluation
10	1	e1		No activation
12	2	e2	(e2,e1)	No activation
14	3	e3	(e3,e1), (e3,e2), (e2,e1)	No activation as none of the set combinations do not match the condition.

Events - Price value	Events count in window	Incoming events	Combination of events set that can form when a new event arrives	Cache condition evaluation
16	3	e4	(e4,e2), (e4,e3)	One activation seen for (e4,e3). e1 expires with count (e4, e3, and e2 are only considered) and hence will not form an event set combination. **Note:** In sliding windows, all the events which do not match the rule or exists within the specified count will continue to be active.
18	2	e5	(e5,e2)	One activation. e2 does not expire, but count since (e4,e3) expires.

Tumbling Windows Clause

The tumble window clause specifies a window size for the events either in time or in events. Events expire due to the following reasons:

- In a time window, events expire if they exceed the specified time limit. In an events window, when a new event arrives, the oldest event expires if the number of events exceeds the window count.
- Event sets expire when they match the rule condition. In tumbling windows, the entire event set combination that matches the rule conditions expires. For example, if it is a "when 2" rule, and if the third event arrives, for the set combination of (e3,e2) (e3,e1) formed, even if one event set matches the rule condition, all the events in the set combination expire.

Tumbling window based on time

When the time difference between the last event and the current event is equal or just greater than the window time, the rule is activated and all the events are dropped after the evaluation irrespective of their participation in the activations.

The following rule shows the usage of `tumble within n seconds` over stock events:

```
WHEN 2 stocks WITH average(s.price) > 14 tumble within 5 seconds then testresponse
```

The following table shows how a single topic rule that uses a window clause tumbles for a size of 5 seconds and when condition is 2:

Event price value	Time window	Incoming events	Possible event set combination when new event arrives	Cache condition evaluation
10	0 seconds	e1		No activation.
12	2 seconds	e2	(e2,e1)	No activation.
14	4 seconds	e3	(e3,e1), (e3,e2)	No activation. Since none of the set combination does not match the condition.

Event price value	Time window	Incoming events	Possible event set combination when new event arrives	Cache condition evaluation
16	6 seconds	e4	(e4,e2), (e4,e3)	One activation. e1 expires with time and hence will not form an event set combination. Since one event combination matches the rule condition, all other events will expire in the tumbling condition. e2 also expires.
18	8 seconds	e5	Only e5	No activation. e2, e3, and e4 have expired earlier because of the specified tumbling condition.

Tumbling window based on count

The window considers the count of events. With the arrival of the latest event, if the count exceeds the specified count limit, the oldest event expires. If there is activation, all the event sets within the event set combination expires.

The following rule shows the usage of `tumble within n events` over stock events:

```
WHEN 2 stocks WITH average(s.price) > 14 tumble within 3 events then testresponse
```

The following table shows how a single topic rule that uses a tumbling window clause tumbles for a size of 3 events and when condition is 2:

Event price value	Events count in window	Incoming events	Possible event set combination when new event arrives	Cache condition evaluation
10	1	e1	-	No activation.
12	2	e2	(e2,e1)	No activation.
14	3	e3	(e3,e1), (e3,e2)	No activation as none of the set combinations do not match the condition.
16	3	e4	(e4,e2), (e4,e3)	One activation. e1 expires with count and hence will not form a set combination. Since one event combination matches the rule condition, all other events will expire in a tumbling condition. e2 also expires.
18	1	e5	Only e5	N o activation. e2 has expired earlier because of the tumbling condition.

Using Operators Between Conditions

When using multiple conditions, you can apply the following operators to specify whether all, some, or one of the conditions must apply:

- AND
- AND NOT
- OR

- OR NOT
- NOT

For example, the following rules show the usage of AND and OR conditions:

```
WHEN 1 stock WITH price > 10 AND symbol = "XYZ"

WHEN 1 stock WITH symbol = "ABCD" OR symbol = "XYZ"
```

AND and OR Operators

RulePoint evaluates AND and OR conditions from left to right.

The AND operator takes precedence over the OR operator. You should always use parentheses to group the conditions appropriately.

Note: If the first part of an AND clause is false, RulePoint does not evaluate the remaining clause.

Consider the following rule:

```
WHEN 1 stock WITH s.price > 10 AND (symbol = "XYZ" OR symbol="ABCD")
```

To activate this rule, either both of the conditions around the AND operator (s.price > 10 and s.symbol = “XYZ”) must be met, or only the condition after the OR operator. The price of the stock topic event must be greater than 10 and its symbol must be XYZ, or the stock symbol of the stock topic event must be ABCD.

Use the OR operator ONLY between conditions for the same topic. For example, in the following rule, the OR operator applies only to the stock topic:

```
WHEN 1 stock s, 1 news n WITH s.symbol in n.title AND (s.symbol = "XYZ" OR
s.symbol="ABCD")
```

Note: When using the OR operator, all of the properties referenced in the conditions must exist in the events present in the system for RulePoint to evaluate a rule.

NOT Operator

You can use the NOT operator to negate or find the opposite of a stated condition.

The NOT operator is most commonly used in conditions with the in, has, contains, and equalsSet Analytics.

For example, the following rule activates when the symbol of the stock event does not appear on the specified watchlist:

```
WHEN 1 stock WITH NOT symbol in watchlist:StockWatchlist
```

Using Analytics

You can apply an analytic to each condition in a rule.

For example, the following rule uses the average analytic to find the average price of three stock events and activates when that average is greater than 10:

```
WHEN 3 stock WITH average(price) > 10
```

Lists as Arguments for Infix Math Operators

You can use list arguments for infix notations in RulePoint.

The infix math operators take two arguments, left hand side and right hand side. You cannot use a nested list as left hand side argument in a math infix operation. The left hand size is a single list that determines the

number of items in each result list. The right hand side determines the structure of the result in terms of number of lists.

For example, `[1,2,3] * [3,4] = [3,6,9],[4,8,12]` means that each result list has three items, as determined by the left hand size, and there are two three-item list as determined by the right hand side.

Lists as math arguments affect the following math functions:

- addition (a+b)
- subtraction (a-b)
- division (a/b)
- multiplication (a*b)
- power(a,b)
- Modulo, mod(a,b)

For example, let us calculate (a*b) where a=[1,2,3] and b=[[3,4],[4,5]]. You can write the arguments as shown in the following table:

a	**b**
1	3, 4
2	4, 5
3	-

You can do the following basic computation of (a*b):

```
[[1*3,2*3, 3*3],[1*4, 2*4, 3*4]],[[1*4,2*4,3*4], [1*5,2*5, 3*5]]]
```

The result of the computation is as follows:

```
[[3,6,9],[4,8,12]],[[4,8,12],[5,10,15]]]
```

To compute power and modulo, power(a,b) or mod(a,b), see the following examples.

The following example describes how to use the modulo operator:

```
mod(2,[2,3])=[0,1])
```

The following example describes how to use the power operator:

```
power(2,[2,3])=[4,8])
```

Analytics with Temporary Variables

When applying an analytic, you can specify temporary variables using the AS clause. Set the analytic result as a variable, and then define a condition for that result.

For example, the following rule takes the average of three stock events and compares that average to a three-day average using a separate DailyStockInfo topic with a property of threedayaverage:

```
WHEN 3 stock s, dailystockinfo d  group by  s.symbol WITH  average(s.price) AS AvgPrice
> 0 AND AvgPrice > d.threedayaverage then response
```

Note: The group by filter forces the rule to evaluate only matching symbols.

If you set the analytic result as a temporary variable, you can reference that result in the responses. Using the previous example, you would include the AvgPrice temporary variable in the response DRQL to display its value in the response.

The assignment of a particular result to a temporary variable is valid only during the evaluation of that rule. After the evaluation, the value of that temporary variable is removed.

CHAPTER 13

Working with Rules

This chapter includes the following topics:

Rules Overview

You can use the RulePoint rules to identify specific events and respond to those events as they occur. For example, you could write a rule that listens to a news feed, and when the headline of a news story contains London, RulePoint then sends an email to a specified user.

Rules process events on topics. You can use the rule processing language, DRQL to create rules. The rule definition includes information about the topics and the number of events to process, the conditions to check, and the response to generate when there is a deviation. As part of overall rule processing, the rule might call out functions called analytics. Rules also rely on watchlists that act as reference data sets during rule processing.

RulePoint also provides an easy way to parameterize rules using templates. Templates are abstract rules that could become a rule when you provide all its parameters. You can create template rules from a template.

Rule Profile Metrics

You can use the rule profile metrics to ensure that each rule is working as intended and is doing so efficiently. Profile metrics give you a better understanding of how the rule is processing incoming events.

You can also use the profile metrics to collect aggregate data on how a rule is processing incoming events. By reviewing the aggregate data you can tune a rule to perform better.

If you are not sure that the rule is getting the results you want or if you think that you are getting the expected result but the rules are not as efficient as you want them to be, use the profile metrics for better insight into the behavior of the rule.

This task is defined by your user policy and the ACL of the rule. To access the profile metrics you must have both write ACL access for the rule and user policy permission for profile rules enabled by the system administrator. The system administrator can give you the access to this tool.

You can access rule profile metrics from the **Dashboard** tab. To view rule profile metrics, select the desired event processor, and then view the **Rules**, **Responses**, **Analytics**, **Topics**, and **Watchlists** tabs. On the **Rules** tab, choose the rule that you want to profile, and then enable tracing for that rule.

Types of Rules

RulePoint provides three rule types, namely Wizard rule, Template rule, and Advanced rule.

- Wizard rule guides you through each step of defining a rule to select topics, conditions, and responses.
- Template rules are created by the RulePoint administrator and provide you with a framework into which you simply input key values to build a rule.
- Advanced rule enables you to create a rule manually using RulePoint's Detect and Respond Query Language (DRQL). This method is for advanced users only.

Use rule names that are less than 255 characters. If a rule is longer than 255 characters, RulePoint displays an error.

Advanced Rule

The advanced rule enables users to create powerful rules using RulePoint's Detect and Respond Query Language (DRQL).

Only users experienced with DRQL should manually create rules in the advanced mode. All other users should use the rule wizard or a template. You can use the DRQL samples provided with the installation package.

It is recommended that you always use white space before and after keywords in Advanced rules. While not having white space(s) might not cause a syntax error on save; during runtime, rules with keywords that do not have leading and trailing white spaces can cause display errors in the user interface. Rules created in advanced mode cannot be edited using the rule wizard.

Creating an Advanced Rule

1. On the **Design** tab, click the **Rules** view.
2. From the **Actions** menu in the top-right corner of the view, select **New Advanced Rule**.

   The **New Advanced Rule** dialog box appears.
3. Under **Step 1: Details**, enter a name and description for the rule.
4. Under **Step 2: Rule Statement**, type the rule in the DRQL syntax. You can use the DRQL samples for reference. To use the DRQL samples, perform the following steps:
   a. Click **View Samples**.

      The DRQL samples are displayed on the right panel. You can search for sample DRQLs.
   b. Select a DRQL sample to view the rule statement. You can copy a rule statement, and modify it to suit your requirements.
5. If you do not have the dependent objects for the rule, you can create the objects using the workspace on the left hand side of the **Advanced Rule** dialog box.
   - If you do not have the related responses for the rule, create the responses. For information on how to create a response, see "Creating a Response" on page 99.

- If you do not have the related topic for the rule, create the topic. For information on how to create a topic, see "Creating a Topic" on page 30.
- If you do not have the related analytics for the rule, create the analytics. For information on how to create an analytic, see "Creating an Analytic" on page 111.
- If you do not have the related watchlists for the rule, create the watchlists. For information on how to create a watchlist, see "Creating a Watchlist" on page 103.

6. To save the rule even if it is not valid, select **Force Save**.

   If the rule is invalid, you can use **Force Save** to save the rule, but you cannot deploy the rule until you correct the rule.
7. To enable parallel processing for the rule, select **Enable Parallel Processing**.
8. To save the rule, click **Save**.

   The rule is in the Draft state. You must deploy the rule.
9. To save and deploy the rule, click **Save and Deploy**.

   The rule is saved and deployed to the run-time environment.

Template Rules

Templates provide an easy way to create new rules. Templates reduce the risk for errors by non experienced rule writers. Templates represent common scenarios, such as news event alerts, stock quote price checks, and credit card transaction notifications. You can set a range of options, including a datatype rule writers can input or the range of acceptable values for a rule parameter, that users are allowed to use. By narrowing the choices for users, the risk of user errors is significantly reduced. You can also provide valuable instructional text for each template parameter they add to make the use of the template not only easier but efficient.

Rules created from templates are affected by some changes you make to the template when the template rule is being used. You need some basic understanding of how RulePoint updates templates that are in use. Not all changes you make to the templates warrant updates to the associated rules. However, changes made to the DRQL of the template generally affect the associated rules. That is because changes to the DRQL directly affect the behavior of the rules that reference the template. The changes you make to the information about the template, such as description or instructional text, will not affect the associated rules. It is important to know the difference.

Creating a Template Rule

1. On the **Design** tab, click the **Rules** view.
2. From the **Actions** menu in the top-right corner of the view, select **New Template Rule**.

   The **New Template Rule** dialog box appears.
3. In the **Select Template** view, click the template, and then click **Next**.
4. In the **Details** view, in **Step 1**, enter a name and description for the template rule.
5. In **Step 2**, enter parameter information.
6. Optionally, click **Test Template Rule** to test the rule.

   The **Template Rule Validation** dialog box appears.
7. Click **OK**.
8. To save the rule, click **Save**.
9. To save and deploy the rule, click **Save and Deploy**.

Wizard Rule

The Rule Wizard provides an easy-to-use interface for creating a rule, including selecting topics, conditions, and responses.

Rules created using the Rule Wizard cannot be edited in Advanced Mode and vice versa. However, a wizard rule can be converted to an advanced rule or template.

Wizard rules do not support nested analytics. An example of a nested analytic is where you use the sum() analytic inside the geoInsideArea() analytic.

To use nested analytics, you can select one of the following options:

- Wizard mode. Create two conditions. The first condition performs the inner analytic (for example, sum()) and assigns to a temporary variable using the AS clause. The second condition is the outer analytic that uses the AS temp variable name.
- Advanced Mode. Use this mode if the outer analytic itself supports an inner analytic.

The Rule Wizard consists of the following steps that correspond to the WHEN, WITH, and THEN statements in DRQL:

1. Select the event topics, how many events to evaluate for each topic, and a time frame in which to evaluate all of the events for the topics you select.
2. Define the conditions for your rule. You can define one, multiple, or no conditions.
3. Define your responses. At least one active response must be available in RulePoint before using the Rule Wizard.
4. Select if you want parallel processing for the rule. Parallel processing is enabled by default.
5. Specify execution settings.

The comparison of string literals and topic property string values are case sensitive. For example, when evaluating a rule, "xyz" does not equal "XYZ."

Creating a Wizard Rule

1. On the **Design** tab, click the **Rules** view.
2. From the **Actions** menu in the top-right corner of the view, select **New Wizard Rule**.

   The **New Wizard Rule** dialog box appears.
3. In the **Details** section, enter the name and description for the rule.

   Enter a unique name for the rule that clearly distinguishes it from all other rules. Do not name the rule using leading hyphens or numbers. Optionally, enter a description of the rule that clearly states the purpose of the rule.
4. In the **Topics and Events** section, click **Add Topics**, and then select the topic or data source that you want to process.

   Here, you can add the event topics on which to base the rule and the number of occurrences required to activate the rule.
5. In the **Events Processed At a Time** column, specify how many events you want processed at a time.

   If the count is greater than 1, additional options appear, and you must specify how you want the events processed.
6. Choose from the following options about how you would want to process the event:
   - **Process Events arrived anytime**.

- **Process Events within**. Specify the time frame in which events must occur for the rule to activate. For example, you can specify whether a specific symbol must be identified in a Stock topic event within a specific number of hours.

7. In the **Conditions** section, enter the conditions for the rule.

   Conditions are the parts of a rule that define the business logic used to process events to find meaningful patterns and information. Each condition is made up of a left-side operand and a right-side operand.

8. Click **Add Conditions**.

   The **Add Conditions** dialog box appears.

9. Enter the condition name.

10. Select the **Condition Type** from the available options.

    - For the option **Compare Topic Property to Null**, select the `Topic Property` and `Operator`. Select the operator. For more information about the operators, see "Equal and Not Equal Operators" on page 197.
    - For the option **Compare Condition Result with Literal**, select the `Variable`, `Operator`, and `Value`. For more information about the operators, see "Using Operators Within Conditions" on page 197.
    - For the option **Compare Condition Result with Topic Property**, select the `Variable`, `Operator`, and `Topic Property`.
    - For the option **Call an Analytic**, select the `Analytic Name`, `Arguments`, and `Output Names`. For more information about the analytics, see "Predefined Analytics" on page 114.
    - For the option **Compare Topic Properties**, select the `Compare`, `Operator`, and `Compare with`. For more information about the operators, see "Using Operators Within Conditions" on page 197.
    - For the option **Compare to a Literal**, select the `Topic Property`, `Operator`, and `Value`. For more information about the operators, see "Using Operators Within Conditions" on page 197.
    - For the option **Compare a Topic Property to Watchlist**, select the `Watchlist`, `Operator`, and `Topic Property`.
      For more information about the operators, see "Using Operators Within Conditions" on page 197.

11. Click **Add**.

12. Select the operator.

    For the added condition, if you want it to include all results except the one you specify, select **NOT**. Otherwise, leave this field blank. For additional conditions, the default operator is **AND**.

13. In the **Responses** section, enter the response for the rule.

    You select the responses to apply to the rule. More than one response can be referenced in a single rule. You can also reference specific properties of an event in your responses.
    When you select a Response to use in a Rule, you can accept the default parameters and values contained in the response or specify new ones here. For example, you can specify a new email address for an email response. Values defined here will override the default values in the original response.

14. If you do not have the dependent objects for the rule, you can create the objects using the workspace on the left hand side of the Wizard Rule dialog box.

    - If you do not have the related responses for the wizard rule, create the responses. For information on how to create a response, see "Creating a Response" on page 99.
    - If you do not have the related topic for the wizard rule, create the topic. For information on how to create a topic, see "Creating a Topic" on page 30.

- If you do not have the related analytics for the wizard rule, create the analytics. For information on how to create an analytic, see "Creating an Analytic" on page 111.
- If you do not have the related watchlists for the wizard rule, create the watchlists. For information on how to create a watchlist, see "Creating a Watchlist" on page 103.

15. To enable parallel processing for the rule, expand **Execution Settings**, and then select **Enable Parallel Processing**.
16. To save the rule, click **Save**.

    The wizard rule is now created and is in Draft state. You must deploy the wizard rule.
17. To save and deploy the rule, click **Save and Deploy**.

    The wizard rule is saved and deployed, if the rule is valid.

Deploying a Rule

1. On the **Design** tab, click **Rules**.
2. In the content panel, select the rule you want to deploy.

   To deploy a rule, the rule must be in draft state.
3. From the menu in the right-hand side, select **Deploy**.

   A message appears that indicates that the deployment is successful and prompts you if you want to view the rule trace.
4. Click **OK** to preview the rule trace, or click **Cancel**.
5. If you view the rule trace, close the window after you complete viewing the preview, and click **OK** to disable the rule tracing.

Viewing Event Trace Report

You can view the event trace report for a rule. The event trace report provides information about the incoming events.

Before you begin, you must deploy the rule to view the event trace.

1. On the **Design** tab, click the **Rules** view.
2. Select the rule to view the event trace.
3. From the menu in the right-hand side, select **View Trace**.

   The event trace report displays the properties of the configured rule and the associated topic and response. The report also provides a summary of the rule activations.
4. When you complete viewing the event trace, close the window.
5. To confirm that you want to disable the view trace, click **OK**.

Creating an Advanced Rule

You can create an advanced rule by using a wizard rule or a template.

1. On the **Design** tab, click the **Rules** view.
2. Select the wizard rule or the template rule that you want to convert.
3. From the menu on the right-hand side, select **Create Advanced Rule**.

   The **New Advanced Rule** dialog box appears.

4. Enter a name for the advanced rule.

   When you convert the rule, `_advanced` is automatically appended to the existing rule.
5. Review the Rule Statement.
6. Click **Save**.

   The rule is now created.

Copying a Rule

1. On the **Design** tab, click the **Rules** view.
2. Select the rule that you want to copy.

   You can select multiple rules to copy.
3. From the **Actions** menu in the top-right corner of the view, select **Copy**.

   The **Copy Rule** dialog box appears.
4. Enter the name of the rule you want to create as a copy.

   The name of the copied rule must be different from the original rule.
5. Click **Save**.

Viewing All Rules

By default, rules are sorted by the last modified date. You can also sort them by name, type, state, or validity by clicking the appropriate column name.

The following table describes each column displayed on the Rules page:

Property	Description
Name	Name of the template.
Type	Type of rules, such as wizard, template, or advanced rule.
Last Modified Date	The date and time that the rule was last modified.
State	The current status of the rule, such as draft, deployed, or needs_deployment.
Validity	Validity of the template. The value is true if valid, false if invalid.

Viewing a Specific Rule

On the Rules page, click **Details View**. Select the name of a specific rule to display. You can view the rule information on the right panel.

The following table describes each column displayed for the rule properties:

Property	Description
Details	Details of the property that includes name, rule type, description, created date, created by, last modified date, deployable, validity, state, and DRQL statement.
Access Control List	Default permission for the user. The Access Control List view has the following properties: - Name. Name of the user. - Access. Grant user permissions to the connection. - Permissions. Access rights of the user to the connection.

Viewing Rule Samples

You can view DRQL samples provided with the installation package for use in rules or to create your own rules using these as reference.

1. On the **Design** tab, click the **Rules** view.
2. From the **Actions** menu on the upper-right panel, select **View Rule Samples**.

   You can search for specific DRQL samples.
3. Select a DRQL sample to view the rule statement.

   You can view the WHEN, WITH, and THEN conditions used in the rule.

Templates Overview

Templates enable users to easily create new rules without doing so from the beginning each time. You can restrict input from users by adding simple validations to the template. To make templates easy to use, you can add specific user assistance where necessary to make the use of the template easier.

Experienced Advanced Mode (DRQL) rule writers, who are usually administrators, can create several templates that represent various scenarios. A template includes a rule statement that contains substitution parameters and instructional text to define those parameters. A substitution parameter is defined by double angle brackets placed around a text or numeric String (for example, <<PRICE>>).

Note: Only users experienced with rule syntax should create templates using DRQL.

You can create templates in RulePoint either from an existing rule or from the beginning. When creating a template from an existing rule, you will have the rule text available to edit and do not have to write the rule itself. The rest of the steps for each method are identical.

After you create a template, you are granted ADMIN permissions to that template. Only you or an administrator can modify, disable, delete, or grant other users permissions (ADMIN, READ, WRITE, or EXECUTE) to the template. Administrative users are granted ADMIN permissions to all templates.

Creating a Template

1. On the **Design** tab, click the **Templates** view.
2. From the **Actions** menu in the top-right corner of the view, select **New**.

   The **New Template** dialog box appears.
3. In the **Template Information** section, specify a name, description, instructions, and a comment.
4. Under **Rule Statement**, in the **Template** field, specify the rule statement by using parameters.
5. To save the rule even if the rule statement is not valid, click **Force Save**.

   If you select **Force Save**, you can save the rule, but you cannot deploy the rule until you correct the rule statement.
6. To enable parallel processing of incoming events by rules, click **Enable Parallel Processing**.
7. Under **Template Parameters**, provide values to the template parameters.
8. If you do not have the dependent objects for the template, you can create the objects using the workspace on the right-hand side of the **New Template** dialog box.
   - If you do not have the related responses for the template, create the responses. For information on how to create a response, see "Creating a Response" on page 99.
   - If you do not have the related topic for the template, create the topic. For information on how to create a topic, see "Creating a Topic" on page 30.
   - If you do not have the related analytics for the template, create the analytics. For information on how to create an analytic, see "Creating an Analytic" on page 111.
   - If you do not have the related watchlists for the template, create the watchlists. For information on how to create a watchlist, see "Creating a Watchlist" on page 103.
9. Optionally, click **Test Template Parameters** to validate the template parameters.

   The **Template Rule Validation** dialog box appears.
10. Click **OK**.
11. Click **Save**.

Template Properties

You can define the template properties and the template parameters when you create or edit a template.

Template Information

Properties	Description
Name	The name of the template. Do not name a template using a leading hyphen or a number.
Description	Optional. The description of the template.
Instructions	Optional. The instructions for using the template to create rules.
Comment	The description to be used for tracking changes.
Template	DRQL syntax for the template.

Template Parameters

Properties	Description
Field Name	Name of the template parameter as it will appear in the final template. When you create a rule from this template, this is the parameter name for which you must provide or select a value.
Description	The description of the parameter as it will appear in the final template.
Field Type	Choose from the following options to specify the data type you want user to enter for this parameter: - Single Line Text. User can enter a single line of text of no more than 2048 characters. - Paragraph Text. User can multiple lines of text of no more than 2048 characters. - Integer Number. User can enter positive or negative whole numbers negative whole numbers and zero. - Real Number. User can enter any numbers both rational and irrational. - Pick Lists of Watchlists. User can select a watchlist from a list of watchlists. - Picklist (User Defined). A user defined list from which the user may choose a single item. - Pick Lists of Topics. User can select a topic from a list of topics that is displayed for the user. - Pick Lists of Keywords. User can select a keyword from a list of keywords. - Picklist (Watchlist Backed). User can select an item displayed from the a watchlist.
Min Value	Field is displayed when Integer Number or Real Number is selected. Use this field to specify the minimum value that a user can enter for this parameter.
Max Value	Field is displayed when Integer Number or Real Number is selected. Use this field to specify the maximum value that a user can enter for this parameter.
Max Length	Field is displayed when Single Line Text or Paragraph Text is selected. Use this field to specify the maximum value that a user can enter for this parameter.
Test Value	A test value for the parameter is based on the data type that you select. This is required to validate your template. For example, if you are using an integer or a real number you can set a range of numbers that the user can enter by specifying a Minimum (Min) and Maximum (Max) value for the parameter. For Lists of Topics or Lists of Watchlists, you can simply set a Test Value.
Set as default value	Optional. Use the Test Value as default value for the parameter. If you specify a default value, this value pre-populates the parameter field when a user creates a new rule from this template.
Custom Validation	Custom validation that you want to perform. The various options provide appropriate patterns and error messages that are displayed if the pattern is not matched.
Instructions for User	Any instructions that you want to provide to the user. Specify one or more of the following options: - Inline help - Popup help - Left Side Label - Right Side Label

Copying a Template

1. On the **Design** tab, click the **Templates** view.
2. Select the template that you want to copy.

   You can select multiple templates to copy.
3. From the **Actions** menu in the top-right corner of the view, select **Copy**.

The **Copy Templates** dialog box appears.

4. Enter the name of the template you want to create as a copy.

   The name of the copied template must be different from the original template.

5. Click **Save**.

Editing a Template

You can edit and upgrade an existing template.

Upgrading templates is done by an administrative user or a user with WRITE ACL access not only to the template that is being upgraded but also WRITE ACL access to all of the child rules that are associated with the template.

By upgrading the template, you can edit the Rule Template definition, that is the DRQL syntax, and all other fields in the **Parameters** section of a template to modify the template format, which will also upgrade all of the child rules associated with the template.

However, a non-administrator user can upgrade or revert changes to a template if all of the following conditions are true:

- The user has WRITE ACL permissions on the template.
- The user has WRITE ACL permissions on all of the template's children.

Complete the following tasks to edit a template through the RulePoint user interface:

1. On the **Design** tab, click the **Templates** view.
2. Select the template that you want to edit.
3. From the menu in the right-hand side, select **Edit**.

   The **Edit Template** dialog box appears.

4. Edit the template information, template rules, and template parameters.
5. If you want to edit the DRQL statement, click **Upgrade Template**.

   The **Dependent Objects** dialog box appears displaying all the template rules created from this template.

6. If you want to modify all the template rules, click **Continue**.

   Note: If you do not want to upgrade a template rule, you can use the option **Create Advanced Rule**.

7. Click **Save** to save the changes to the template.

   Note: If you edit a template rule that is deployed, click **Save and Redeploy Rules** to deploy the associated template rule. The associated template rule changes from Needs_Deployment to the Deployed state.

Deleting a Template

You can delete an existing template.

1. In the **Design** tab, click the **Templates** view.
2. In the Templates content panel, select the template name that you want to delete.
3. From the menu in the right-hand corner, click **Delete**.

   A message prompts you to verify if you want to delete the template.

4. Click **OK** to delete the template.

Creating Deployment Policies

1. On the **Design** tab, click the **Templates** view.
2. Select the template for which you want to create a deployment policy.
3. From the menu on the right-hand side, select **Create Deployment Policy**.

   The **Create Deployment Policy** dialog box appears.
4. Select the template name, and then assign the template to an event processor.
5. Click **Save**.

   Note: You can update and delete the deployment policies. Use the **Update Deployment Policy** and **Delete Deployment Policy** from the menu to update and delete the deployment policies.

Viewing All Templates

By default, templates are sorted by last modified date. You can also sort them by name, number of rules, or validity by clicking the appropriate column name.

The following table describes each column displayed on the Templates page:

Property	Description
Name	Name of the template.
Last Modified Date	The date and time that the template was last modified.
Number of Rules	The number of rules associated with the template.
Validity	Validity of the template. The value is true if valid, false if invalid.

Viewing a Specific Template

On the Templates page, click the name of a specific template to display. In **Details View**, you can view the template information on the right panel.

The following table describes each column displayed for the template properties:

Property	Description
Details	Details of the template. The details are Name, Description, Created Date, Last Modified Date, Created By, Validity, Deployable, and DRQL Statement.
Rules	List the rules created from the template.
History	If you have upgraded the template before, you can review the change history of the template. Administrative users can sequentially review the history of upgrades to a template. **Revert** is associated with the most recent change. To revert to the most recent change, click the corresponding **Revert** icon next to the history item.

Property	Description
Parameters	Details about the template parameters.
Access Control List	Default permission for the user. The Access Control List view has the following properties: - Name. Name of the user. - Access. Grant user permissions to the connection. - Permissions. Access rights of the user to the connection.

CHAPTER 14

Working with Alerts

This chapter includes the following topics:

Alerts Overview

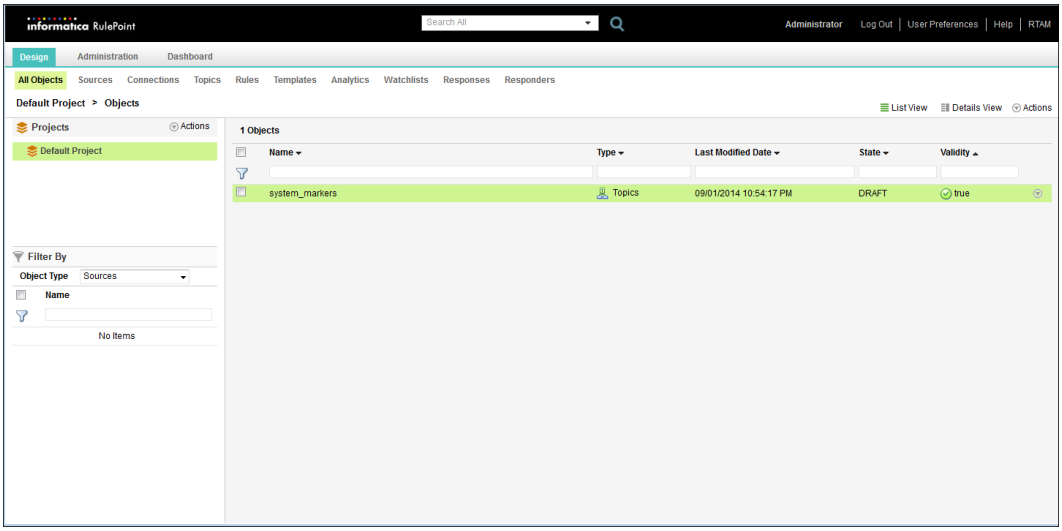
Informatica Real-Time Alert Manager (RTAM) is a web-based dashboard to receive alerts from RulePoint.

RTAM is a web-based platform that is fully integrated with RulePoint to manage incoming alerts. RulePoint generates an alert and then sends it to RTAM.

You can group the RTAM alerts as channels. Each alert has a priority, subject, and body. RulePoint includes a standard RTAM responder to send alerts to RTAM.

You can view, acknowledge, search, send, forward, filter, prioritize, or delete alerts through the RTAM dashboard.

You can go to the RTAM dashboard through a link on the RulePoint user interface as shown in the following image:



Logging In to RTAM

Before logging in to RTAM, you must obtain a valid username and password from the system administrator.

1. Launch a web browser and enter the following URL:

   `http://<host>:<port>/RTAM`

   host is the host name, the fully qualified domain name, or the IP address of the server where you install RulePoint. The *port* is the HTTP port number of the Tomcat server. The default is 8080.

   Based on the configuration, you might have to type *https*. RTAM in the URL must be in capital letters.

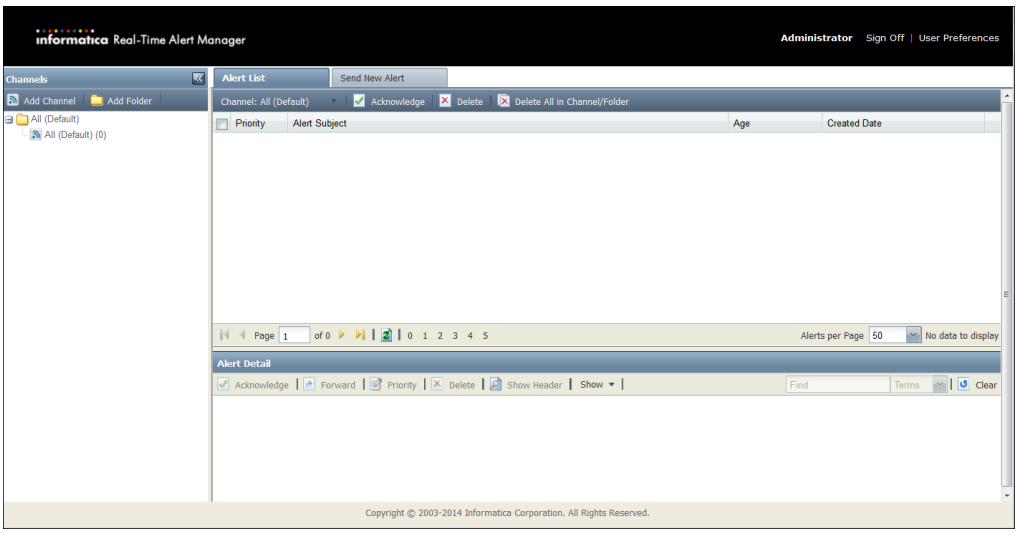
   The RTAM login page appears.

2. Enter the RTAM administrator's credentials.

   By default, the user name and password combination is *Administrator/Administrator1*.

3. Click **Log In**.

   The RTAM home page appears as shown in the following image:



Navigating RTAM

The RTAM user interface is divided into the following three panes:

- **Channels** explorer on the left.
- **Alert List** on the upper right.
- **Alert Detail** on the lower right.

Channels

The Channels explorer is located in the left frame of the RTAM application and displays all of the channels and folders. You can use the Channels explorer to create, move, and delete channels and folders. For more information, see "Organizing the Alerts" on page 222.

Navigation Tabs and Panels

The navigation tabs and panels provide access to major tasks that includes viewing, searching, sending alerts, and setting user preferences.

Alert List

The **Alert List** is located in the upper-right panel of the RTAM main page and displays alerts based on the folders and channels selected in the **Channels Explorer**. In the **Alert List**, you can acknowledge and delete multiple alerts at one time.

Send New Alert

In the **Send New Alert** tab, you can manually create alerts to send to other RTAM users.

User Preferences

In the **User Preferences** section, you can specify the preferences including the colors for alert priority numbers and changing the language.

Alert Detail Panel

The **Alert Detail** panel is located in the lower-right panel of the RTAM main page and displays the details of the alert selected in the Alert List. In the **Alert Detail** panel, you can acknowledge, forward, change priority, and delete alerts.

In the **Alert Detail** panel, you also can search within the alert for specific information. This is useful if the alert is large and cannot be viewed in a single page.

Organizing the Alerts

You can use the Channels explorer to organize the folders, subfolders, and channels.

The **All (Default)** folder already exists and contains all of the channels, folders, and subfolders. You cannot rename, move, or delete the All (Default) folder.

You can associate incoming alerts with a specific channel. RTAM creates channels that have been specified in any RulePoint response that sends alerts to RTAM. When RTAM receives an alert from RulePoint, it places the alert in the specified channel. If you do not specify any channel, RTAM places the alert in the All (Default) channel. If you delete a channel that has an alert associated with it, the alert is moved to the All (Default) channel. If you create the channel again, RTAM moves the alert back to the channel.

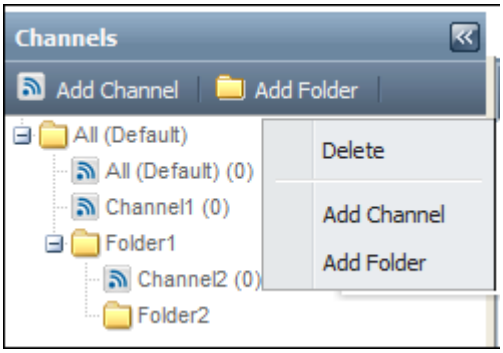
Creating Channels and Folders

When creating new channels, the following rules apply:

- Channel and folder names are case sensitive. For example, RTAM considers “news” to be different from “News.”
- Semi-colons are not allowed in channel and folder names.

You can create as many folders as necessary to organize the channels.

The following image shows how to create a channel and a folder:



Creating Channels

The count next to a channel displays the number of alerts in that channel. This count decreases when you delete an alert. A count is not displayed for folders. Folders are only for organizing your channels.

1. In the Channels explorer, click **Add Channel**, or right-click a folder name and select **Add Channel**.

   The **Channel** dialog box displays.
2. Type the name of the new channel.
3. Click **OK**.

Creating Folders

1. In the Channels explorer, click **Add Folder** or right-click a folder name and select **Add Folder**.

   The **Folder** dialog box appears.
2. Type the name of the new folder.
3. Click **OK**.

Moving Channels and Folders

You can drag and drop channels and folders to rearrange them as well as to move a channel or folder into a folder or subfolder.

You cannot move folders into channels. The All (Default) folder and All (Default) channel cannot be moved.

Deleting Channels and Folders

If you delete a channel, RTAM moves the alerts associated with the deleted channel into the All (Default) channel.

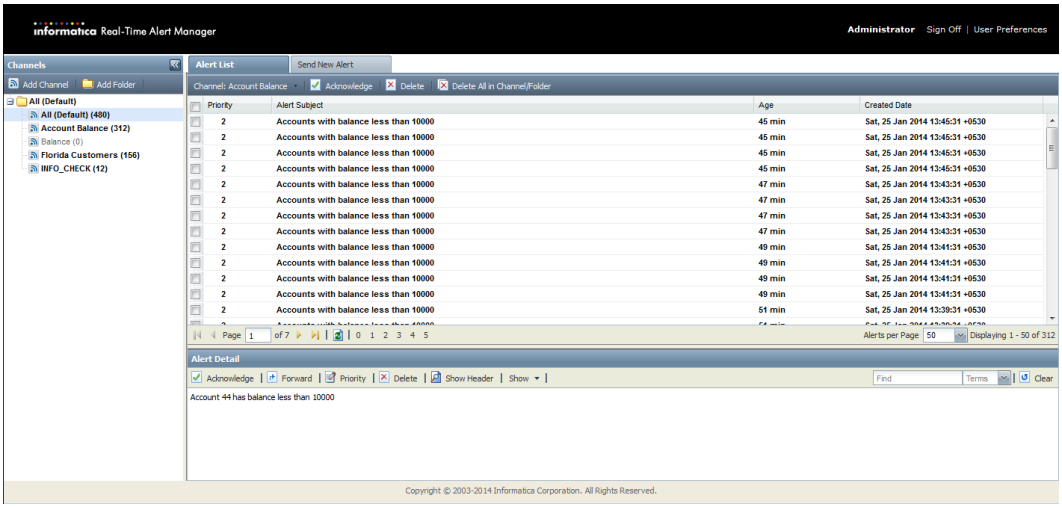
When RTAM receives a subsequent alert associated with the deleted channel, it places that alert into the All (Default) channel or creates a new channel if a specific channel was specified by the RulePoint response. For more information, see the RulePoint online help.
Note: You cannot delete the All (Default) folder or the All (Default) channel.

To delete a channel or a folder, right-click the channel or the folder and select **Delete**.

Viewing Alerts

The **Alert List** tab displays the alerts. By default, all alerts are displayed on the **Alert Inbox**.

To display the alerts in a specific channel, click the name of the channel or folder in the dashboard. You can use the folder or channel drop-down menu option to filter and see the related alerts. For example, in the following image, you can see all the alerts:



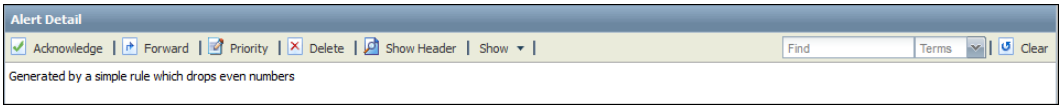
The alert inbox displays the following information for each alert:

- Priority. The alert priority, where 0 is the lowest and 5 is the highest.
- Alert Subject. The subject of the alert. This is what the alert is about.
- Age. How old the alert is. The current time minus the date the alert was created.
- Created Date. The date and time RTAM received the alert.

When you select an alert, the **Alert Detail** panel is populated with details of the selected alert. By default, the first alert in your inbox is selected. For more information about the Alert Detail panel, see "Viewing a Specific Alert" on page 224.

Viewing a Specific Alert

To view a specific alert, click the title of the alert in the **Alert List**. The alert appears in the **Alert Detail** panel as shown in the following image:



The **Alert Detail** panel displays all available information about the selected alert. You can acknowledge, forward, change priority, or delete the alert in the alert detail panel. You can launch other applications and trigger business processes through the alert detail panel. In addition, you can click **Show Header** on the **Alert Detail** toolbar to display the header for the alert, which contains secondary user-specified information associated with the alert.

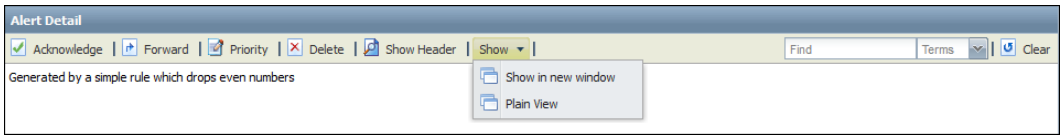
Viewing an Alert in a Separate Window

You can display an alert in a separate window by double-clicking the alert.

Alternatively, click **Show** > **Show in new window** to display an alert in a separate window.

To display the alert in simple ASCII text, click **Show** > **Plain View**.

The following image shows how to view an alert in a separate window:



Managing Alerts

In RTAM, you can have the flexibility to manage the alerts you receive including changing the priority number, filtering, and forwarding alerts to other users.

Filtering the Alert List by Priority

You can filter the list of alerts based on their priority numbers.

Toggle the priority numbers at the bottom of the **Alert List** panel. The **Alert List** will display the selected priority number.

Acknowledging Alerts

Acknowledging alerts is useful in a large scale deployment where multiple users receive the same alert, potentially resulting in duplicated responses or actions.

For example, in a call center, a representative can acknowledge a specific alert and assume responsibility for that alert.

When you acknowledge an alert, the alert title in the Alert List changes from bold to unbold. The acknowledge feature works as it does in your email inbox. You can acknowledge that an alert has been reviewed and it no longer appears bold in your inbox.

Acknowledge an Individual Alert

- To acknowledge an individual alert, click the name of the alert in the **Alert Inbox**. The **Alert List** panel displays the alert.
- Click **Alert Detail** > **Acknowledge** to acknowledge the alert.

Acknowledge More Than One Alert At a Time

Select the check box next to each alert in the **Alert Inbox**, and then click **Actions** > **Acknowledge**.

Note: You cannot unacknowledge an acknowledged alert.

Selecting the Number of Alerts to Display in the Alert List Panel

1. At the bottom of the **Alert List** panel, you can select the number of alerts to display.

2. Select one of the options listed in the **Alerts per Page** list.

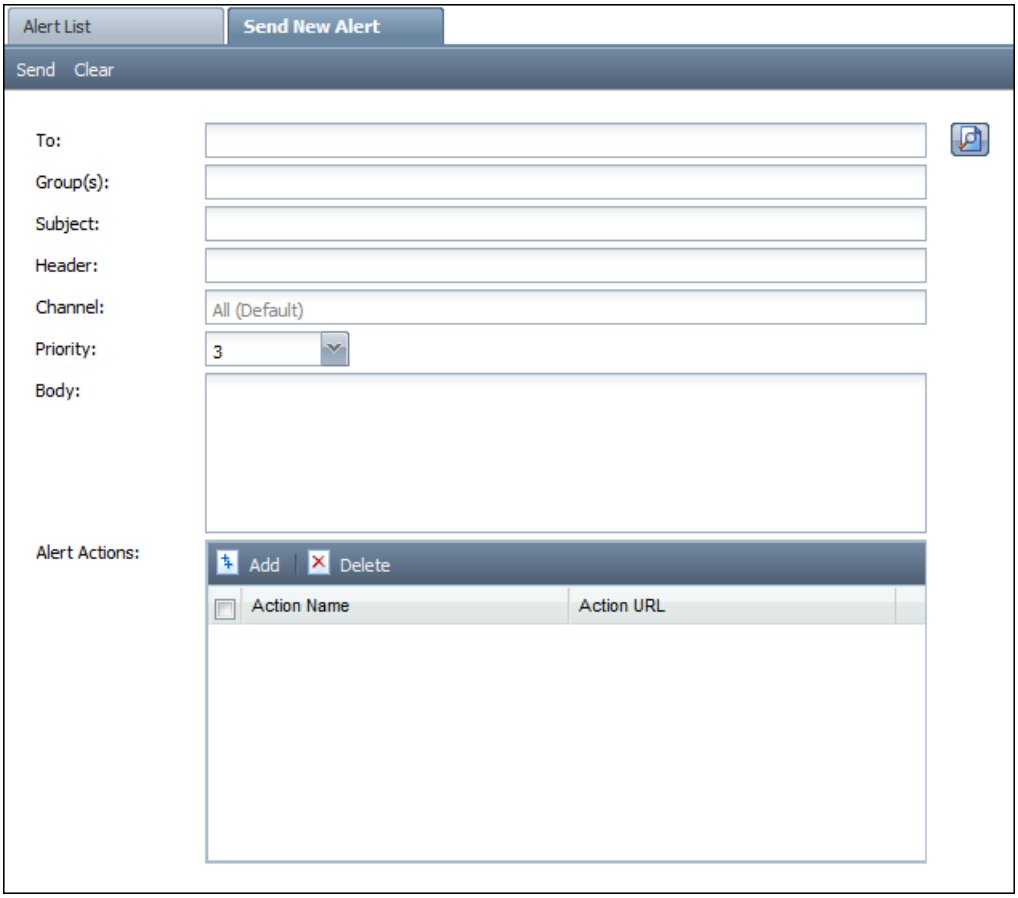
   The Alert List panel is adjusted accordingly.

Sending Alerts

You can manually create alerts to send to other RTAM users, including yourself. Within the alert, you can add an action, which is a hyperlink to a specified URL. More than one action can be added to an alert.

1. Click the **Send New Alerts** tab.

   The following window displays:



2. Create the alert.

   The following table describes the parameter information that you need to create an alert:

Parameter	Description
To	The RTAM user to receive the alert. To select the user, click **To**.
Group(s)	Enter a group name to receive the alert.
Subject	The subject of the alert.

Parameter	Description
Header	Optional. The header of the alert. You can embed HTML in the header. The header contains secondary information associated with the alert.
Channel	Optional. The name of the channel to which RTAM sends the alert. If channel does not exist, RTAM automatically creates one to store the alert. If you do not specify a channel, the alert is sent to the All (Default) channel.
Priority	The priority level of the alert.
Body	The body of the alert. You can embed HTML in the body.
Alert Actions	Optional. Action for the alert.

3. To create an action for the alert, complete the following steps:
   a. In the **Alert Actions** panel, click **Add**.
   b. In the **Action Name** field, type a unique name for the action.
   c. In the **Action URL**, type the link of the action. This will be the hyperlink in the alert. It can be a URL, file pathname, or a mapped network drive.

Deleting Alerts

When you delete an alert, the alert is removed from the inbox. After you delete an alert, you cannot perform actions on it, such as forwarding it or changing its priority.

Delete an Individual Alert

1. Select the channel with the alert that you want to delete.
2. Click the name of the alert in the **Alert List**, and then click **Delete** in the **Alert Detail** toolbar.

Delete More Than One Alert at a Time

1. Select the channel with the alert that you want to delete.
2. Select the check box next to each alert, and then click **Delete** in the **Actions** toolbar.

Delete All Alerts From the Alert Inbox

1. Select the channel with the alert that you want to delete.
2. Click **Delete All** in the **Actions** toolbar.

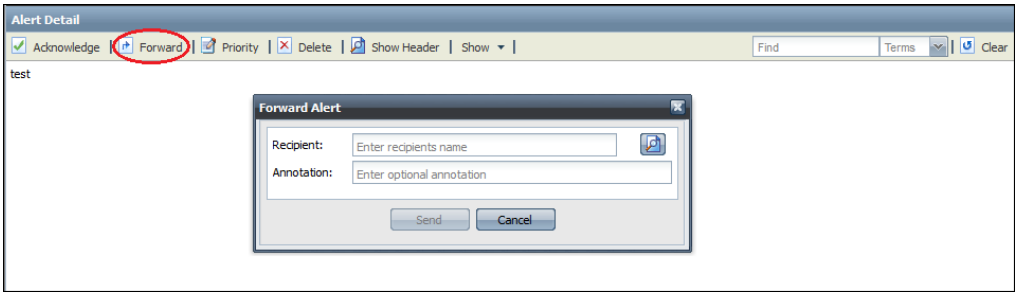
Note: As the alerts are stored individually for each user, deleted alerts are not removed from the inbox of other users who received the alert.

Forwarding an Alert

You can forward alerts to other RTAM users. Even if you previously acknowledged the alert, it will appear unacknowledged to the user that you forward it to. This feature is useful when a representative needs to transfer responsibility for a particular alert to another user.

1. In the **Alert List**, click the name of the alert to display it in the **Alert Detail** panel.

2. In the **Alert** toolbar, click **Forward**.



The **Forward Alert** window appears.

3. In the **Recipient** panel, click **To** and select one or more existing RTAM users.
4. Click **OK**.
5. (Optional) In the **Annotation** field, type a note to send to the user.
6. Click **Send** at the bottom of the **Forward Alert** dialog box.

Changing the Priority

If you change the priority for an alert, the priority is changed for all users who receives the alert.

1. Click the name of the alert to display it in the **Alert Detail** panel.
2. In the **Alert Detail** toolbar, click **Priority**.

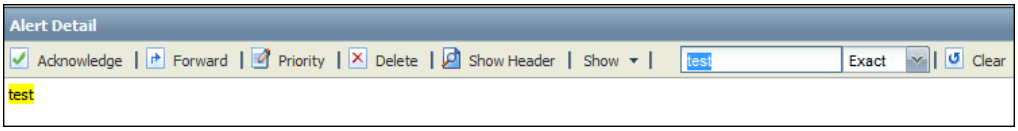
   The **Set Priority** dialog box appears.
3. In the **Priority** list, enter the new priority level.
4. Click **Set**.

Searching for Terms in Alerts

You can search for specific terms within an individual alert.

1. Click the name of the alert to display it in the **Alert Detail** panel.
2. In the drop-down menu next to the **Find** field, select one of the following options:
   - Terms to search for any occurrence of the text, such as alphanumeric characters or words.
   - Exact to search for an exact match of the text.
3. In the **Find** field, type the text you want to find, and then press `Enter`.

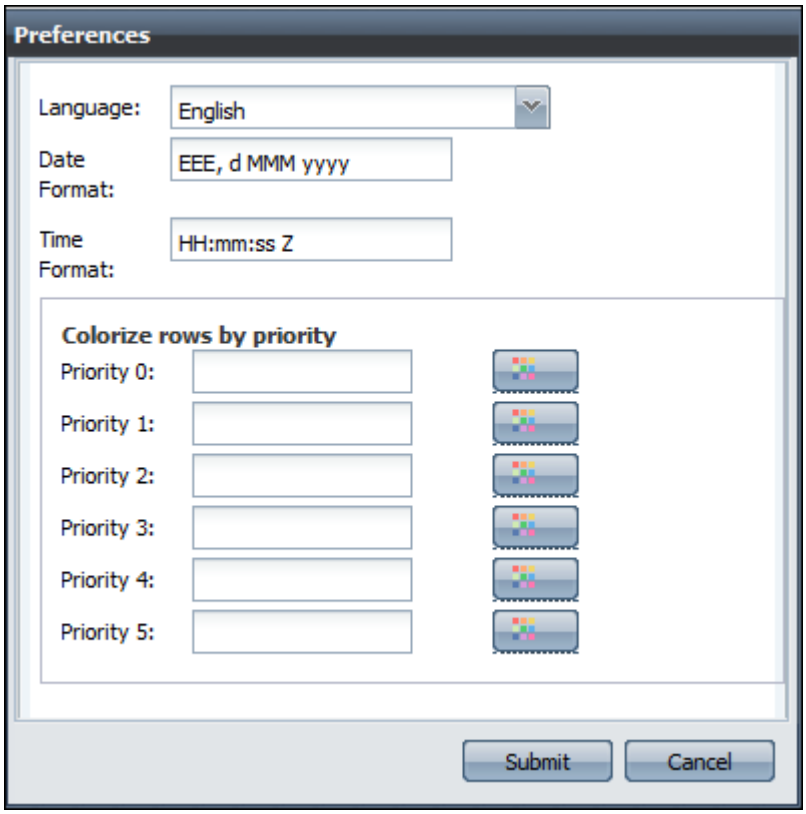
   The search results are highlighted in the **Alert Detail** panel as shown in the following image:



4. Click **Clear** to clear the text from the search field.

Setting User Preferences

In the **Preferences** dialog box, you can change the language preference, date and time formats, and assign a color for each of the alert priority numbers. The following image shows the **Preferences** dialog box:



Setting the Date and Time Format

1. On the RTAM dashboard, click **User Preferences**.

   The **Preferences** dialog box appears.
2. Change the **Date Format** and the **Time Format** as required.
3. Click **Submit**.

The examples in the following table show how date and time patterns are interpreted in the U.S. locale. The given date and time are 2001-07-04 12:08:56 local time in the U.S. Pacific Time time zone.

Date and Time Pattern	Expected Result
"yyyy.MM.dd G 'at' HH:mm:ss z"	2001.07.04 AD at 12:08:56 PDT
2001.07.04 AD at 12:08:56 PDT	Wed, Jul 4, '01
"h:mm a"	12:08 PM
"hh 'o''clock' a, zzzz"	12 o'clock PM, Pacific Daylight Time

Date and Time Pattern	Expected Result
"K:mm a, z"	0:08 PM, PDT
"EEE, d MMM yyyy HH:mm:ss Z"	Wed, 4 Jul 2001 12:08:56 -0700

Assigning a Color for each Alert Priority Number

1. On the RTAM dashboard, click **User Preferences**.

   The **Preferences** dialog box appears.
2. In the **Colorize rows by priority** section, click the corresponding color button to select a color for each of the priority numbers.
3. Click **Submit**.

CHAPTER 15

Setting Access Controls

This chapter includes the following topics:

Access Control Lists in RulePoint

Each instance of the object or project has an associated ACL object.

Each access control entry in an ACL object consists of the following information:

- User name
- Permission, such as read, write, execute, and admin
- Action, such as grant or deny access

You can edit existing permissions or add permissions to a local or LDAP user or role.

Permission Types

Permissions in RulePoint are independent. If you have a privilege to create a specific object, you do not inherit the privilege to view that object. If you need all the available permissions, you need to include all those permissions explicitly. You can grant or deny permissions for user access to objects.

The following table lists the actions for each type of permission:

Permission Type	Action
Read	View an instance of the object. When you have read permission, you can perform tasks in which you view details associated with the object.
Write	Edit and delete an object.

Permission Type	Action
Execute	Deploy, undeploy, redeploy, and reassign a primary object. Use an execute permission to perform the deploy-related actions on the object.
Admin	Edit the ACL of an object, for example, add or remove entries.

ACL Permission Rules

You can control RulePoint user or role access through ACL rules. The ACL rules require you to pass a set of requirements to gain access to a particular object.

In RulePoint, consider the following ACL rules:

- If a user has ACL permissions for a project, that user inherits the ACL permissions for all the objects within that project, unless there is a more specific ACL permission entry at the object level that overrides it.
- If a role has ACL permissions for a project, the user associated with that role inherits the ACL permissions for all the objects within that project, unless there is a more specific ACL permission entry at the object level that overrides it.
- When a user creates an instance of an object, the ACL rule created for that object includes all the four permissions for that user, such as read, write, execute, and admin.
- If a user creates an instance of the object, that user cannot modify the ACL settings in it.
- Users with the super user privilege can access all objects irrespective of the ACL entry.

Evaluating ACL Rule Permission Requirements

ACL rules determine user access to an object if the user meets all permissions required by the matching ACL rules. When a user attempts to access an object, RulePoint evaluates permissions of the matching ACL rules.

Based on the evaluation, RulePoint either grants or denies user access based on the following matching ACL rules:

- If the ACL has a matching entry, where the user ID is the logged in user, has the required permissions, and the grant access for the permissions, the user can access the RulePoint object. For example, if a user has write permission for a specific object with grant access, that user can create, edit, or delete that object.
- If the ACL has a matching entry, where the user ID is the logged in user, has the required permissions, but the deny access for the permissions, RulePoint denies user access to the object. For example, if a user has write permission for a specific object with deny access, that user cannot edit or delete that object.
- If the ACL does not have a matching entry according to the first two rules, RulePoint invokes the parent ACL and runs the rules again until it finds a matching entry. When RulePoint reaches the end of the ACL hierarchy, and there are no more parent ACLs, it denies user access to the object. For example, if a user has read and write permission for a specific project, but no grant or deny access specified at the object level, that user will have the read and write access on all the objects in that project.

Setting Access Controls for an Object

You can set access controls for an object through the RulePoint user interface.

1. On the **Design** tab, select the **Details** view on the right panel.
2. Select an object for which you want to set the access control, and click the **Access Control List** view on the right panel.

   You can select either a topic, connection, source, responder, response, rule, template, or a watchlist.
3. From the menu on the right-hand corner for the selected object, select **Edit Direct Permissions** from the menu.

   The **Direct Permissions** dialog box appears.
4. To add a new user to the ACL of the selected entity, click the **Add** icon.
5. Click the **Search** icon next to the newly added field.
   a. In the **Name** column, select the user or role for whom you want to set the access control.
   b. In the **Access** column, select **GRANT** or **DENY**.
   c. In the **Permissions** column, select the permission that you want to assign to the user or role.

      You can assign ADMIN, READ, WRITE, or EXECUTE permission to users or roles.
   d. Click **OK**.
6. Click **Save**.

CHAPTER 16

Troubleshooting RulePoint Issues

This chapter includes the following topics:

Troubleshooting Overview

The troubleshooting procedures described in this chapter aim to help you resolve commonly known issues. Contact Informatica Global Customer Support if the troubleshooting procedures do not help you resolve a problem.

Troubleshooting Connectivity Issues

A blank screen appears after you log out of the RulePoint user interface.

This issue occurs if the RulePoint session has timed out due to inactivity.

Refresh the browser to return to the login page.

Troubleshooting Issues with Source Controllers

The Dashboard displays deployed SQL sources, but the default source controller is highlighted in yellow, indicating an error condition.

This error can occur when one or more SQL sources contain errors.

- If the SQL sources use the same SQL connection, ensure that the SQL queries are valid.
- If the SQL sources use different SQL connections, test the connection to ensure that the connection is valid and the database is accessible.

Troubleshooting Topology Host Issues

You are unable to update the IP address of the topology host.

You cannot delete the default host entry if you have not created a new host.

Perform the following actions:

1. Create a new host and point all the nodes to the new host.
2. Delete the default host entry.

Troubleshooting Issues with Sources, Responders, and Responses

The RulePoint user interface displays the status of the sources and responders as "Invalid".

The status of the sources and responders appears as invalid on the RulePoint user interface, though the connections and other objects are working as expected. This error can occur in the following situations:

- A source is flagged as invalid if no schedules are associated with it.
- A responder is flagged as invalid if no responses are associated with it.

Perform one of the following actions:

- Associate one or more schedules with the source.
- Associate one or more responses with the responder.

The properties of a rule do not appear in a response.

After you evaluate and activate a rule, the properties of the rule do not appear in the response.

This issue occurs if the properties of the topic that you include in the rules are not qualified with a topic alias or a topic name.

Qualify the topic properties in the rules with either a topic alias or a topic name. The following example shows a topic property qualified with a topic alias:

```
WHEN 1 stock t0 WITH symbol="XYZ" THEN IMStockResponse WITH to="yourhandle",
subject="stock update", body="${t0.symbol} event is available"
```

Where, `t0` is the topic alias.

Troubleshooting Issues with Exporting Objects

You are unable to export objects and artifacts.

The export function fails if one of the following conditions exist:

- Templates without rules.
- Topics that are not associated with any sources or rules.
- Artifacts such as analytics or watchlists that are not used with any rule.

Perform one of the following actions:

- Create one or more rules for every template.
- Create one or more rules that include analytics, topics, and watchlists.

Troubleshooting Deployment and Undeployment Issues

You are not able to undeploy a rule.

The following error message appears when you undeploy a rule:

```
Error YDR-3007. Grid Manager unreachable.
```

You might not be able to undeploy a rule if there is an issue with the topology server and the controllers.

On the Dashboard, verify whether the topology server and the controllers are running. In addition, verify whether the node agent is running.

The Dashboard does not display events for a deployed source.

This error can occur in the following situations:

- The database table to which the source tries to connect does not have data.
- There might be a time lag from when a topic is updated until the time the events status appears on the Dashboard.

Perform one of the following actions:

- Run a query against the database to which you want the source to connect and ensure that the table or view that you are trying to connect contains data.

- Click the **Refresh** button on the Dashboard to update the status of events.

In a high availability setup, the Dashboard does not display the digit 1 in the red or green boxes under the Activity Managers, Nodes, or Hosts sections.

This issue occurs if the topology is not in the running state.

To resolve this issue, terminate all the stale topology processes and restart the topology.

You are not able to deploy a sample project.

The following error message appears when you deploy a sample project:

```
Error YDR-3005. A runtime error occurred.
```

This issue occurs if you start the host agent by using the host name instead of the IP address.

Start the host agent by using the IP address. The following is an example of using the IP address to start the host agent:

```
startHostAgent –h 10.10.10.10 -p19000
```

APPENDIX A

Connecting to an Ultra Messaging Application

This appendix includes the following topics:

Connecting to an Ultra Messaging Application Overview

You can configure an Ultra Messaging Source and Responder in RulePoint to connect to an external Ultra Messaging application.

- You can configure RulePoint to connect to the following source Ultra Messaging editions:

  Ultra Messaging Streaming (UMS). The source and receiver applications send and receive data streams as topics based on a publish/subscribe model. UMS uses topic resolution to listen to all messages in a topic. UMS minimizes message latency and maximizes throughput.
- Ultra Messaging Persistence (UMP). In addition to the functionality of the UMS edition, UMP has a persistent store which provides storage to message streams. UMP delivers a persisted message stream to receiving applications with no additional latency.
- Ultra Messaging Queuing (UMQ). Contains both UMS and UMP edition functionalities. UMQ can read messages from multiple sending applications and write messages to multiple receiving applications concurrently. UMQ ensures decoupling sending and receiving applications and ensures that all messages are delivered only once.

You can also configure RulePoint to connect to the following external Ultra Messaging related applications:

- PowerExchange for Ultra Messaging. PowerExchange for UM integrates PowerCenter with Informatica Ultra Messaging (UM) to read and write UM native messages and UM JMS messages.
- Vibe Data Stream for Machine Data. Uses Informatica Ultra Messaging (UM) for data transport across VDS nodes.

Configure UM Store or UM Queue

You can install UM Store or UM Queue on a remote box and use it with RulePoint.

Perform the following steps to configure the UM store or UM queue:

1. Install the 64-bit Informatica UMQ 6.5 bundle.
2. Set the environment variable LBM_LICENSE_FILENAME to the path of the Ultra Messaging License file.
3. Set *<UMQ_INSTALLATION_DIR>*/UMQ_6.5/bin to the path of the environment variable.
4. Start the lbmrd process. Run the following command:

   `lbmrd -i <IP_OF_THE_LBMRD_HOST> -p <LBMRD_PORT>`
5. Create the logs directory under the `<UMQ_INSTALLATION_DIR>` directory, create the `cache` and `state` sub-directories within the logs directory.
6. Copy the sample `lbmstore.xml` file to `<UMQ_INSTALLATION_DIR>`.

   Note: You require the `lbmstore.xml` file to start the UM Store or Queue. The `lbmstore.xml` file contains a reference to the `lbmstore.cfg` file, which contains the UM configurations.

   The following snippet is a sample of the `lbmstore.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<ume-store version="1.2">
    <daemon>
        <lbm-config>C:\workspace\Informatica\um_config\lbmstore.cfg</lbm-config>
        <log>C:\workspace\Informatica\logs\stored.log</log>
        <pidfile>stored.pid</pidfile>
        <web-monitor>*:15704</web-monitor>
    </daemon>
    <stores>
        <store interface="127.0.0.1" name="my-store" port="44000">
            <ume-attributes>
                <option name="disk-cache-directory" type="store" value="C:\workspace
\Informatica\logs\cache"/>
                <option name="disk-state-directory" type="store" value="C:\workspace
\Informatica\logs\state"/>
                <option name="context-name" type="store" value="remote-store"/>
                <option name="allow-proxy-source" type="store" value="1"/>
            </ume-attributes>

         <topics>
                <topic pattern="." type="PCRE">
                    <ume-attributes>
                        <option name="repository-type" type="store" value="disk"/>
                        <!-- size threshold is 100Mb and size limit is 200 Mb, can
be modified if required -->
                        <option name="repository-size-threshold"
                            type="store" value="104857600"/>
                        <option name="repository-size-limit"
                            type="store" value="209715200"/>
                        <option name="repository-disk-file-size-limit"
                            type="store" value="1073741824"/>
                        <!--
                    10 seconds after source became inactive, proxy source is started
on store.
                    Any new receivers during this 10 seconds inactivity will not be
able to
                    register with stores and request retransmissions.
                -->
                        <option name="source-activity-timeout"
                            type="store" value="10000"/>
                        <option name="receiver-activity-timeout"
                            type="store" value="604800000"/>
                        <option name="source-state-lifetime"
```

```
                            type="store" value="10000000"/>
                        <!--
                      Stores generate keep alive traffic to sources and receivers
every 0.5 second. This is default and can be removed.
                -->
                        <option name="keepalive-interval" type="store" value="500"/>
                        <!--
                      Stores randomly wait 0.5 to 1.5 times proxy-election-interval
before starting a proxy source.
                -->
                        <option name="proxy-election-interval"
                            type="store" value="5000"/>
                        <!--
                      LateJoin: Maximum number of messages newly registered receiver
can rollback (request retransmission).
                -->
                        <option
                            name="receiver-new-registration-rollback"
                            type="store" value="2147483600"/>
                    </ume-attributes>
                </topic>
            </topics>

        </store>
    </stores>
    <queues>
    <queue name="UMQueue" port="14567" group-index="0">
        <ume-attributes>
            <option type="queue" name="dissemination-model" value="SQD"/>
        </ume-attributes>

        <!-- Define the Application Set(s) -->
        <application-sets>
            <application-set name="Set 1">
                <ume-attributes>
                    <option type="queue" name="log-audit-trail" value="1"/>
                </ume-attributes>
                <receiver-types>
                    <receiver-type id="100"/>
                </receiver-types>
            </application-set>
        </application-sets>

        <topics>
            <!-- Define the topic(s) and the Application Sets for them -->
            <topic pattern="." type="PCRE">
                <application-sets>
                    <application-set name="Set 1"/>
                </application-sets>
            </topic>
        </topics>
    </queue>
    </queues>
</ume-store>
```

7. Copy the `lbmstore.cfg` file in the <*UMQ_INSTALLATION_DIR*> directory, and update the path in the `lbmstore.xml` file.

   Note: The `lbmstore.xml` file consists of a configured store, my-store, which runs on port 44000.

   The following snippet is a sample of the `lbmstore.cfg` file:

   ```
   context fd_management_type wincompport
   context resolver_unicast_daemon 127.0.0.1:22000
   ## Uncomment the following if multicast is enabled in the environment
   #context resolver_multicast_interface [MULTICAST_INTERFACE]
   ```

8. To use a port of your choice, edit the port number.
9. Update the `disk-cache-directory` and `disk-state-directory` in the `lbmstore.xml` file with the path of the `cache` and `state` directories.

Note: The `lbmstore.xml` contains a configured queue, UMQueue, that runs on port 14567. The receiver-type ID is set as 100.

10. To use a port of your choice, edit the port number.
11. Start the umestored process. Run using the following command:

    `umestored <PATH_OF_THE_LBMSTORE_CONFIG_XML>`

    The UM store my-store and queue UMQueue starts on the remote box.

Sample LBM Configuration Files for UM Store and Queue

RulePoint uses the lbm.cfg files to communicate with the configured UM Store or Queue.

Use the sample files as reference to create the lbm.cfg files. Edit the .cfg files to update the lbmrd IP address and port according to your configuration.

Sample ums_lbm.cfg file

```
## Comment the following configuration property if you are using Linux
context fd_management_type wincompport
context request_tcp_port_low 37000
context request_tcp_port_high 37400
context resolver_unicast_port_low 20000
context resolver_unicast_port_high 21000
context transport_tcp_maximum_ports 499
context transport_tcp_port_low 37500
context transport_tcp_port_high 37999
## The lbmrd process host and IP address
context resolver_unicast_daemon [IP_OF_HOST_RUNNING_LBMRD]:[LBMRD_PORT]
## Uncomment the following property if you enable multicast in the environment
#context resolver_multicast_interface [MULTICAST_IP]
```

Sample ump_lbm.cfg file

```
## Comment the following configuration property if you are using Linux
context fd_management_type wincompport
context request_tcp_port_low 37000
context request_tcp_port_high 37400
context resolver_unicast_port_low 20000
context resolver_unicast_port_high 21000
context transport_tcp_maximum_ports 499
context transport_tcp_port_low 37500
context transport_tcp_port_high 37999
## The umestored process host and port details
source ume_store [IP_OF_HOST_RUNNING_UMESTORED]:[UMESTORED_PORT]
## The lbmrd process host and ip address
context resolver_unicast_daemon [IP_OF_HOST_RUNNING_LBMRD]:[LBMRD_PORT]
## Uncomment the following property if you enable multicast in the environment
#context resolver_multicast_interface [MULTICAST_IP]
```

Sample umq_lbm.cfg file

```
## Comment the following configuration property if you are using Linux
context fd_management_type wincompport
context request_tcp_port_low 37000
context request_tcp_port_high 37400
context resolver_unicast_port_low 20000
context resolver_unicast_port_high 21000
context transport_tcp_maximum_ports 499
context transport_tcp_port_low 37500
context transport_tcp_port_high 37999
## The UM Queue to which the UM Responder writes
source umq_queue_name UMQueue
## The receiver identifier for the UM Queue from which the UM Source reads messages
receiver umq_receiver_type_id 100
## The lbmrd process host and IP address
context resolver_unicast_daemon [IP_OF_HOST_RUNNING_LBMRD]:[LBMRD_PORT]
```

```
## Uncomment the following property if you enable multicast in the environment
#context resolver_multicast_interface [MULTICAST_IP]
```

Configure RulePoint to Communicate with Vibe Data Stream for Machine Data

You can install Vibe Data Stream for Machine Data in a remote machine and connect with RulePoint. The Vibe Data Stream allows an external UM application to push messages to a UM topic. Configure an Ultra Messaging source in RulePoint to pick messages from the UM topic and publish them as events on a RulePoint topic that is bound to the source.

1. Create a `lbm.cfg` file with the following contents:

```
## Comment out the following configuration property if you use Linux
    context fd_management_type wincompport
    ## The receiver identifier for the UM Queue from which the UM source reads
messages
    receiver umq_receiver_type_id [RECEIVER_TYPE_ID]
    ## The lbmrd process host and ip address
    context resolver_unicast_daemon [IP_OF_HOST_RUNNING_LBMRD]:[LBMRD_PORT]
    ## Uncomment the following property if you enable multicast in the environment
    #context resolver_multicast_interface [MULTICAST_IP]
```

2. Edit the `lbm.cfg` file to set the receiver_type_id to match the configuration in Vibe Data Stream.
3. Edit IP_OF_HOST_RUNNING_LBMRD and LBMRD_PORT to match the configuration of Vibe Data Stream.
4. If you enable multicast in the Vibe Data Stream environment, set the MULTICAST_IP accordingly, and uncomment the line that contains the resolver_multicast_interface.
5. Copy the `lbm.cfg` file to `RULEPOINT_HOME/service-conf`.
6. Create an Ultra Messaging connection that refers to the `lbm.cfg` file in `RULEPOINT_HOME/service-conf` directory.
7. Create an Ultra Messaging source that uses this connection. Set the source name configuration field with the name of the UM topic on which the Vibe Data Stream publishes messages.
8. Set the source message type as Text, Byte Array, or LBM properties according to the type published by Vibe Data Stream.
9. Deploy the Ultra Messaging source.

Sample LBM Configuration File for Vibe Data Stream for Machine Data

RulePoint uses the lbm.cfg file to communicate with the configured Vibe Data Stream for Machine Data.

Use the sample files as reference to create the lbm.cfg files. Edit the .cfg files to update the lbmrd IP address and port according to your configuration.

Sample cfg file

```
## Comment out the following configuration property if you are using Linux
context fd_management_type wincompport
## The receiver identifier for the UM Queue from which UM Source reads messages
receiver umq_receiver_type_id 100
## The lbmrd process host and IP address
context resolver_unicast_daemon 10.25.32.164:22000
```

Configure RulePoint to Communicate with PowerExchange for Ultra Messaging

Before you perform the tasks, you must install the PowerExchange plugin for Ultra Messaging on the PC instance and PC client. For information to install the plugin, create the PC mappings, transformations, UM connection, and other configurations, see the *PowerCentre* and *PowerExchange for Ultra Messaging* documentation.
Perform the following steps to configure RulePoint to connect with PowerExchange for Ultra Messaging:

1. Fetch the following UM configuration options of PowerExchange for UM connection from the UM `config.xml` file:
   - Fetch the IP and port number of the LBMRD process from resolver_unicast_daemon.
   - For UMP, fetch the ume_store config value.
   - For UMQ, fetch the umq_queue_name and umq_receiver_type_id configuration values.
2. Based on the UM edition (UMS, UMP, or UMQ) that you use, create a `lbm.cfg` file. See sample .cfg templates.
3. Update the file with the values obtained for resolver_unicast_daemon, ume_store, umq_queue_name, and umq_receiver_type_id.
4. Create a RulePoint Ultra Messaging connection that refers to this file.
5. Create a RulePoint Ultra Messaging source or responder to use the Ultra Messaging connection.

   The source or responder reads and writes messages from an Ultra Messaging topic. The PowerCentre Exchange for UM reads messages and writes messages from an Ultra Messaging topic.

APPENDIX B

Creating an Ultra Messaging JMS Source

This appendix includes the following topics:

Creating an Ultra Messaging JMS Source Overview

You can implement Ultra Messaging JMS to configure a JMS Source in RulePoint to listen to an Ultra Messaging topic.

You can use one of the following configuration files to configure Ultra Messaging JMS:

- UM XML configuration file. You require the com.latencybusters.jms.LBMXmlContextFactory context factory to use this file.
- JMSConfig XML configuration file. You require the com.sun.jndi.fscontext.RefFSContextFactory context factory to use this file.

For more information on configuring the Ultra Messaging JMS source to listen to an ultra messaging topic, see the chapter "Ultra Messaging JMS Configuration" in the Ultra Messaging Queuing Edition 6.5 Guide.

Before You Begin

Complete the following tasks before you configure the JMS source.

1. Copy the jar files from `<RULEPOINT_HOME/system/UM/jmsclient/lib` to the `<RULEPOINT_HOME>/lib` directory.
2. Restart the RulePoint topology and design time instances , if they are already running.

Configuring JMS Source Using UM XML Configuration File

1. Copy the following sample `umjms.xml` file to `RULEPOINT_HOME`:

```
<?xml version="1.0" encoding="UTF-8"?>
<um-configuration version="1.0">
    <applications>
        <application name="uJMS">
            <contexts>
                <context name="uJMSConnectionFactory" template="">
                    <sources/>
                    <receivers/>
                    <wildcard-receivers/>
                    <options type="context">
                        <option default-value="wincompport"
name="fd_management_type" />
                        <!-- if operational_mode set to embedded, UM JMS overrides
back to sequential -->
                        <option default-value="sequential" name="operational_mode"/>
                        <option default-value="10.65.43.252:22000"
name="resolver_unicast_daemon"/>
                    </options>
                </context>
                <context name="TopicConnectionFactory" template="">
                    <sources/>
                    <receivers/>
                    <wildcard-receivers/>
                    <options type="context">
                        <option default-value="sequential" name="operational_mode"/>
                        <option default-value="10.65.43.252:22000"
name="resolver_unicast_daemon"/>
                    </options>
                </context>
                <context name="QueueConnectionFactory" template="">
                    <sources/>
                    <receivers/>
                    <wildcard-receivers/>
                    <options type="context">
                        <option default-value="sequential" name="operational_mode"/>
                        <option default-value="10.65.43.252:22000"
name="resolver_unicast_daemon"/>
                    </options>
                </context>
                <context name="uJMSConnectionFactory-UMS">
                    <sources/>
                    <options type="source">
                        <option default-value="2048"
name="implicit_batching_minimum_length"/>
                        <!-- For unicast, set transport to lbtru or tcp -->
                        <option default-value="tcp" name="transport"/>
                    </options>
                    <receivers/>
                    <!-- For ULB-->
                    <!-- <options type="receiver">
       <option default-value="100" name="umq_receiver_type_id"/>
       <option default-value="10" name="umq_delayed_consumption_report_interval"/>
     </options> -->
                    <wildcard-receivers/>
                </context>
                <context name="uJMSConnectionFactory-UMP">
                    <sources/>
                    <options type="source">
                        <option default-value="2048"
name="implicit_batching_minimum_length"/>
                        <option default-value="1" name="late_join"/>
```

```
                        <!-- For unicast, set transport to lbtru or tcp -->
                        <option default-value="tcp" name="transport"/>
                        <option default-value="1" name="ume_proxy_source"/>
                        <option default-value="my-store" name="ume_store_name"/>
                        <option default-value="qc" name="ume_store_behavior"/
>
                    </options>
                    <receivers/>
                    <!-- For ULB-->
                    <!--
      <options type="receiver">
        <option default-value="100" name="umq_receiver_type_id"/>
        <option default-value="10" name="umq_delayed_consumption_report_interval"/>
      </options>
       -->
                    <wildcard-receivers/>
                </context>
                <context name="uJMSConnectionFactory-UMQ">
                    <sources/>
                    <options type="source">
                        <option default-value="2048"
name="implicit_batching_minimum_length"/>
                        <option default-value="1" name="late_join"/>
                        <!-- For unicast, set transport to lbtru or tcp -->
                        <option default-value="tcp" name="transport"/>
                        <option default-value="UMQueue" name="umq_queue_name"/>
                    </options>
                    <receivers/>
                    <options type="receiver">
                        <!-- <option default-value="100"
name="umq_receiver_type_id"/> -->
                        <option default-value="10"
name="umq_delayed_consumption_report_interval"/>
                    </options>
                    <wildcard-receivers/>
                </context>

                <context name="TopicConnectionFactory-UMS">
                    <sources/>
                    <options type="source">
                        <option default-value="2048"
name="implicit_batching_minimum_length"/>
                        <!-- For unicast, set transport to lbtru or tcp -->
                        <option default-value="tcp" name="transport"/>
                        <!-- <option default-value="0:100"
name="umq_ulb_application_set"/> -->
                    </options>
                    <receivers/>
                    <!-- For ULB-->
                    <options type="receiver">
                        <!--   <option default-value="100"
name="umq_receiver_type_id"/> -->
                        <option default-value="10"
name="umq_delayed_consumption_report_interval"/>
                    </options>
                    <wildcard-receivers/>
                </context>
                <context name="TopicConnectionFactory-UMP">
                    <sources/>
                    <options type="source">
                        <option default-value="2048"
name="implicit_batching_minimum_length"/>
                        <option default-value="1" name="late_join"/>
                        <!-- For unicast, set transport to lbtru or tcp -->
                        <option default-value="tcp" name="transport"/>
                        <option default-value="1" name="ume_proxy_source"/>
                        <option default-value="my-store" name="ume_store_name"/>
                        <option default-value="qc" name="ume_store_behavior"/>
                        <!-- <option default-value="0:100"
```

```
name="umq_ulb_application_set"/> -->
                    </options>
                    <receivers/>
                    <!-- For ULB-->
                    <!-- <options type="receiver">
        <option default-value="100" name="umq_receiver_type_id"/>
        <option default-value="10" name="umq_delayed_consumption_report_interval"/>
      </options> -->
                    <wildcard-receivers/>
                </context>
                <context name="TopicConnectionFactory-UMQ">
                    <sources/>
                    <options type="source">
                        <option default-value="2048"
name="implicit_batching_minimum_length"/>
                        <option default-value="1" name="late_join"/>
                        <!-- For unicast, set transport to lbtru or tcp -->
                        <option default-value="tcp" name="transport"/>
                        <option default-value="UMQueue" name="umq_queue_name"/>
                    </options>
                    <receivers/>
                    <!-- <options type="receiver">
            <option default-value="100" name="umq_receiver_type_id"/>
            <option default-value="10"
name="umq_delayed_consumption_report_interval"/>
        </options> -->
                    <wildcard-receivers/>
                </context>

                <context name="QueueConnectionFactory-UMS">
                    <sources/>
                    <options type="source">
                        <option default-value="2048"
name="implicit_batching_minimum_length"/>
                        <!-- For unicast, set transport to lbtru or tcp -->
                        <option default-value="tcp" name="transport"/>
                        <!-- <option default-value="0:100"
name="umq_ulb_application_set"/> -->
                    </options>
                    <receivers/>
                    <!-- For ULB-->
                    <!-- <options type="receiver">
        <option default-value="100" name="umq_receiver_type_id"/>
        <option default-value="10" name="umq_delayed_consumption_report_interval"/>
      </options> -->
                    <wildcard-receivers/>
                </context>
                <context name="QueueConnectionFactory-UMP">
                    <sources/>
                    <options type="source">
                        <option default-value="2048"
name="implicit_batching_minimum_length"/>
                        <option default-value="1" name="late_join"/>
                        <!-- For unicast, set transport to lbtru or tcp -->
                        <option default-value="tcp" name="transport"/>
                        <option default-value="1" name="ume_proxy_source"/>
                        <option default-value="my-store" name="ume_store_name"/>
                        <option default-value="qc" name="ume_store_behavior"/>
                        <!-- <option default-value="0:100"
name="umq_ulb_application_set"/> -->
                    </options>
                    <receivers/>
                    <!-- For ULB-->
                    <!-- <options type="receiver">
        <option default-value="100" name="umq_receiver_type_id"/>
        <option default-value="10" name="umq_delayed_consumption_report_interval"/>
      </options> -->
                    <wildcard-receivers/>
                </context>
```

```
                <context name="QueueConnectionFactory-UMQ">
                    <sources/>
                    <options type="source">
                        <option default-value="2048"
name="implicit_batching_minimum_length"/>
                        <option default-value="1" name="late_join"/>
                        <!-- For unicast, set transport to lbtru or tcp -->
                        <option default-value="tcp" name="transport"/>
                        <option default-value="UMQueue" name="umq_queue_name"/>
                    </options>
                    <receivers/>
                    <!-- <options type="receiver">
       <option default-value="100" name="umq_receiver_type_id"/>
       <option default-value="10" name="umq_delayed_consumption_report_interval"/>
     </options> -->
                    <wildcard-receivers/>
                </context>
            </contexts>
            <event-queues>
                <event-queue/>
            </event-queues>
            <application-data>
                <ConnectionFactory name="uJMSConnectionFactory">
                    <options type="ConnectionFactory">
                        <option default-value="UME" name="DEFAULT_TOPIC_TYPE"/>
                        <option default-value="UME" name="DEFAULT_TEMP_TOPIC_TYPE"/>
                        <option default-value="true" name="USE_APP_HEADER"/>
                        <!-- <option default-value="UME1" name="CLIENT_ID"/>
       <option default-value="false" name="DEBUG"/> -->
                        <option default-value="TextMessage"
name="DEFAULT_MESSAGE_TYPE"/>
                    </options>
                </ConnectionFactory>
                <ConnectionFactory name="TopicConnectionFactory">
                    <options type="ConnectionFactory">
                        <option default-value="UME" name="DEFAULT_TOPIC_TYPE"/>
                        <option default-value="UME" name="DEFAULT_TEMP_TOPIC_TYPE"/>
                        <option default-value="true" name="USE_APP_HEADER"/>
                        <!-- <option default-value="UME1" name="CLIENT_ID"/>
       <option default-value="false" name="DEBUG"/> -->
                        <option default-value="TextMessage"
name="DEFAULT_MESSAGE_TYPE"/>
                    </options>
                </ConnectionFactory>
                <ConnectionFactory name="QueueConnectionFactory">
                    <options type="ConnectionFactory">
                        <option default-value="UME" name="DEFAULT_TOPIC_TYPE"/>
                        <option default-value="UME" name="DEFAULT_TEMP_TOPIC_TYPE"/>
                        <option default-value="true" name="USE_APP_HEADER"/>
                        <!-- <option default-value="UME1" name="CLIENT_ID"/>
       <option default-value="false" name="DEBUG"/> -->
                        <option default-value="TextMessage"
name="DEFAULT_MESSAGE_TYPE"/>
                    </options>
                </ConnectionFactory>
                <Destination name="TempQueue">
                    <options type="Destination">
                        <option default-value="TempQueue" name="NAME"/>
                        <option default-value="UMQ" name="TYPE"/>
                        <option default-value="queue" name="DESTTYPE"/>
                        <option default-value="false" name="WILDCARD"/>
                    </options>
                </Destination>
                <Destination name="TempQueue1">
                    <options type="Destination">
                        <option default-value="TempQueue1" name="NAME"/>
                        <option default-value="UMQ" name="TYPE"/>
                        <option default-value="queue" name="DESTTYPE"/>
                        <option default-value="false" name="WILDCARD"/>
```

```
    </options>
</Destination>
<Destination name="UMETopic">
    <options type="Destination">
        <option default-value="UMETopic" name="NAME"/>
        <option default-value="false" name="WILDCARD"/>
        <option default-value="2000" name="REGID"/>
    </options>
</Destination>
<Destination name="DurableTopic">
    <options type="Destination">
        <option default-value="DurableTopic" name="NAME"/>
        <option default-value="false" name="WILDCARD"/>
        <option default-value="61000" name="REGID"/>
        <option default-value="topic" name="DESTTYPE"/>
    </options>
</Destination>
<Destination name="DestTopic">
    <options type="Destination">
        <option default-value="DestTopic" name="NAME"/>
        <option default-value="false" name="WILDCARD"/>
        <option default-value="3000" name="REGID"/>
        <option default-value="topic" name="DESTTYPE"/>
    </options>
</Destination>
<Destination name="BytesTopic">
    <options type="Destination">
        <option default-value="BytesTopic" name="NAME"/>
        <option default-value="false" name="WILDCARD"/>
        <option default-value="7000" name="REGID"/>
        <option default-value="topic" name="DESTTYPE"/>
    </options>
</Destination>
<Destination name="MapTopic">
    <options type="Destination">
        <option default-value="MapTopic" name="NAME"/>
        <option default-value="false" name="WILDCARD"/>
        <option default-value="6500" name="REGID"/>
        <option default-value="topic" name="DESTTYPE"/>
    </options>
</Destination>
<Destination name="ObjectTopic">
    <options type="Destination">
        <option default-value="ObjectTopic" name="NAME"/>
        <option default-value="false" name="WILDCARD"/>
        <option default-value="6000" name="REGID"/>
        <option default-value="topic" name="DESTTYPE"/>
    </options>
</Destination>
<Destination name="TempTopic">
    <options type="Destination">
        <option default-value="TempTopic" name="NAME"/>
        <option default-value="false" name="WILDCARD"/>
        <option default-value="4000" name="REGID"/>
        <option default-value="topic" name="DESTTYPE"/>
    </options>
</Destination>
<Destination name="ReplyTopic">
    <options type="Destination">
        <option default-value="REPLY" name="NAME"/>
        <option default-value="4400" name="REGID"/>
        <option default-value="LBM" name="TYPE"/>
        <option default-value="false" name="WILDCARD"/>
    </options>
</Destination>
<Destination name="RequestTopic">
    <options type="Destination">
        <option default-value="RequestTopic" name="NAME"/>
        <option default-value="5000" name="REGID"/>
```

```
                        <option default-value="LBM" name="TYPE"/>
                        <option default-value="false" name="WILDCARD"/>
                    </options>
                </Destination>
            </application-data>
        </application>
    </applications>
</um-configuration>
```

2. Change the lbmrd daemon, ume store, umq queue name according to your UM store, queue or lbmrd configurations.
3. Configure the following properties to create a JMS connection:
   - JNDI Context Factory. com.latencybusters.jms.LBMXmlContextFactory
   - Connection URL. Absolute path of the umjms.xml. For example: file:/E:/workspace/yosemite/RulePoint_61/um/umjms.xml.
   - JMS Connection Factory. The connection factory to use. For example, uJMSConnectionFactory.
4. Create a JMS source using the configured connection.
5. In the source configuration, set the JMS destination as the name of the UM topic from which the JMS source reads messages.

Configuring JMS Source Using JMSConfig XML File

1. Copy the following sample `config.xml` file to `RULEPOINT_HOME\system\UM\jmsclient\config`:

```
<?xml version="1.0" encoding="UTF-8"?>
<JMSConfig>
    <ConnectionFactories>
        <ConnectionFactory name="uJMSConnectionFactory">
            <FactoryAttributes>
                <Attribute name="DEBUG" value="false"/>
                <Attribute name="DEFAULT_TOPIC_TYPE" value="UME"/>
                <!-- UME, LBM -->
                <Attribute name="DEFAULT_TEMP_TOPIC_TYPE" value="UME"/>
                <!-- UME, LBM -->
                <Attribute name="USE_APP_HEADER" value="true"/>
                <Attribute name="DEFAULT_MESSAGE_TYPE" value="TextMessage"/>
                <!-- BytesMessage, TextMessage, MapMessage, ObjectMessage,
StreamMessage -->
                <Attribute name="CLIENT_ID" value="Client1"/>
                <!-- For multiple connections, instead use Connection setClientID
function to provide a unique client id -->
                <Attribute name="USE_UMP_SESSION_IDS" value="false"/>
            </FactoryAttributes>
            <ContextAttributes>
                <!-- if operational_mode set to embedded, UM JMS overrides back to
sequential -->
                <Attribute name="operational_mode" value="sequential"/>
                <!--
                <Attribute name="resolver_multicast_ttl" value="16"/>
                <Attribute name="resolver_multicast_address" value="224.9.10.11"/>
                <Attribute name="mim_address" value="224.10.10.21"/>
                <Attribute name="transport_lbtrm_multicast_address_low"
value="224.10.10.10"/>
                <Attribute name="transport_lbtrm_multicast_address_high"
value="224.10.10.14"/>
                <Attribute name="request_tcp_port_low" value="16000"/>
                <Attribute name="request_tcp_port_high" value="16050"/>
                <Attribute name="transport_lbtrm_source_port_low" value="14390"/>
```

```
                <Attribute name="transport_lbtrm_source_port_high" value="14399"/>
                <Attribute name="transport_tcp_maximum_ports" value="10"/>
                <Attribute name="transport_tcp_port_low" value="14371"/>
                <Attribute name="transport_tcp_port_high" value="14390"/>
        -->
                <!-- For multicast topic resolution, comment out the next 3 lines -->
                <Attribute name="resolver_unicast_daemon"
value="10.65.43.252:22000"/>
                <!-- <Attribute name="resolver_unicast_daemon"
value="0.0.0.0:15380"/> -->
                <!-- <Attribute name="resolver_unicast_port_low" value="16051"/> -->
                <!-- <Attribute name="resolver_unicast_port_high" value="16099"/> -->
                <Attribute name="transport_lbtrm_data_rate_limit" value="5000000"/>
                <Attribute name="transport_lbtrm_retransmit_rate_limit"
value="1000000"/>
                <Attribute name="transport_lbtrm_receiver_socket_buffer"
value="8000000"/>
                <Attribute name="transport_lbtru_maximum_ports" value="5"/>
                <Attribute name="transport_lbtru_receiver_socket_buffer"
value="524288"/>
                <Attribute name="transport_lbtru_source_socket_buffer" value="0"/>
                <Attribute name="transport_lbtru_data_rate_limit" value="10000000"/>
                <Attribute name="transport_lbtru_datagram_max_size" value="8192"/>
                <Attribute name="transport_lbtru_rate_interval" value="100"/>
                <Attribute name="request_tcp_reuseaddr" value="1"/>
                <Attribute name="response_tcp_nodelay" value="0"/>
            </ContextAttributes>
            <SourceAttributes>
                <!-- For unicast, set transport to lbtru or tcp -->
                <Attribute name="transport" value="lbtrm"/>
                <Attribute name="late_join" value="1"/>
                <Attribute name="ume_store_name" value="my-store"/>
                <!--
                <Attribute name="ume_store_name" value="JMSStore1"/>
                <Attribute name="ume_store_name" value="JMSStore2"/>
                <Attribute name="ume_store_name" value="JMSStore3"/>
                -->
                <Attribute name="ume_store_behavior" value="qc"/>
                <Attribute name="ume_proxy_source" value="1"/>
                <Attribute name="umq_queue_name" value="UMQueue"/>
                <Attribute name="implicit_batching_minimum_length" value="2048"/>
            </SourceAttributes>
            <ReceiverAttributes>
                <Attribute name="umq_receiver_type_id" value="100"/>
                <Attribute name="umq_delayed_consumption_report_interval"
value="10"/>
            </ReceiverAttributes>
        </ConnectionFactory>
        <ConnectionFactory name="TopicConnectionFactory">
            <FactoryAttributes>
                <Attribute name="CLIENT_ID" value="TopicClient1"/>
                <!-- For multiple connections, instead use Connection setClientID
function to provide a unique client id -->
                <Attribute name="USE_UMP_SESSION_IDS" value="false"/>
            </FactoryAttributes>
            <ContextAttributes>
                <!-- if operational_mode set to embedded, UM JMS overrides back to
sequential -->
                <Attribute name="operational_mode" value="sequential"/>
                <!--
                <Attribute name="resolver_multicast_ttl" value="16"/>
                <Attribute name="resolver_multicast_address" value="224.9.10.11"/>
                <Attribute name="mim_address" value="224.10.10.21"/>
                <Attribute name="transport_lbtrm_multicast_address_low"
value="224.10.10.10"/>
                <Attribute name="transport_lbtrm_multicast_address_high"
value="224.10.10.14"/>
                <Attribute name="request_tcp_port_low" value="16000"/>
                <Attribute name="request_tcp_port_high" value="16050"/>
```

```
                <Attribute name="transport_lbtrm_source_port_low" value="14390"/>
                <Attribute name="transport_lbtrm_source_port_high" value="14399"/>
                <Attribute name="transport_tcp_maximum_ports" value="10"/>
                <Attribute name="transport_tcp_port_low" value="14371"/>
                <Attribute name="transport_tcp_port_high" value="14390"/>
        -->
                <!-- For multicast topic resolution, comment out the next 3 lines -->
                <Attribute name="resolver_unicast_daemon"
value="10.65.43.252:22000"/>
                <!-- <Attribute name="resolver_unicast_daemon"
value="0.0.0.0:15380"/> -->
                <!-- <Attribute name="resolver_unicast_port_low" value="16051"/> -->
                <!-- <Attribute name="resolver_unicast_port_high" value="16099"/> -->
                <Attribute name="transport_lbtrm_data_rate_limit" value="5000000"/>
                <Attribute name="transport_lbtrm_retransmit_rate_limit"
value="1000000"/>
                <Attribute name="transport_lbtrm_receiver_socket_buffer"
value="8000000"/>
                <Attribute name="transport_lbtru_maximum_ports" value="5"/>
                <Attribute name="transport_lbtru_receiver_socket_buffer"
value="524288"/>
                <Attribute name="transport_lbtru_source_socket_buffer" value="0"/>
                <Attribute name="transport_lbtru_data_rate_limit" value="10000000"/>
                <Attribute name="transport_lbtru_datagram_max_size" value="8192"/>
                <Attribute name="transport_lbtru_rate_interval" value="100"/>
                <Attribute name="request_tcp_reuseaddr" value="1"/>
            </ContextAttributes>
            <SourceAttributes>
                <!-- For unicast, set transport to lbtru or tcp -->
                <Attribute name="transport" value="lbtrm"/>
                <Attribute name="ume_store_name" value="my-store"/>
                <!--
                <Attribute name="ume_store_name" value="JMSStore1"/>
                <Attribute name="ume_store_name" value="JMSStore2"/>
                <Attribute name="ume_store_name" value="JMSStore3"/>
                -->
                <Attribute name="ume_store_behavior" value="qc"/>
                <Attribute name="ume_proxy_source" value="1"/>
            </SourceAttributes>
        </ConnectionFactory>
        <ConnectionFactory name="QueueConnectionFactory">
            <FactoryAttributes>
                <Attribute name="CLIENT_ID" value="QueueClient1"/>
                <!-- For multiple connections, instead use Connection setClientID
function to provide a unique client id -->
                <Attribute name="USE_UMP_SESSION_IDS" value="false"/>
            </FactoryAttributes>
            <ContextAttributes>
                <!-- if operational_mode set to embedded, UM JMS overrides back to
sequential -->
                <Attribute name="operational_mode" value="sequential"/>
                <!--
                <Attribute name="resolver_multicast_ttl" value="16"/>
                <Attribute name="resolver_multicast_address" value="224.9.10.11"/>
                <Attribute name="mim_address" value="224.10.10.21"/>
                <Attribute name="transport_lbtrm_multicast_address_low"
value="224.10.10.10"/>
                <Attribute name="transport_lbtrm_multicast_address_high"
value="224.10.10.14"/>
                <Attribute name="request_tcp_port_low" value="16000"/>
                <Attribute name="request_tcp_port_high" value="16050"/>
                <Attribute name="transport_lbtrm_source_port_low" value="14390"/>
                <Attribute name="transport_lbtrm_source_port_high" value="14399"/>
                <Attribute name="transport_tcp_maximum_ports" value="10"/>
                <Attribute name="transport_tcp_port_low" value="14371"/>
                <Attribute name="transport_tcp_port_high" value="14390"/>
        -->
                <!-- For multicast topic resolution, comment out the next 3 lines -->
                <Attribute name="resolver_unicast_daemon"
```

```
value="10.65.43.252:22000"/>
                <!-- <Attribute name="resolver_unicast_daemon"
value="0.0.0.0:15380"/> -->
                <!-- <Attribute name="resolver_unicast_port_low" value="16051"/> -->
                <!-- <Attribute name="resolver_unicast_port_high" value="16099"/> -->
                <Attribute name="transport_lbtrm_data_rate_limit" value="5000000"/>
                <Attribute name="transport_lbtrm_retransmit_rate_limit"
value="1000000"/>
                <Attribute name="transport_lbtrm_receiver_socket_buffer"
value="8000000"/>
                <Attribute name="transport_lbtru_maximum_ports" value="5"/>
                <Attribute name="transport_lbtru_receiver_socket_buffer"
value="524288"/>
                <Attribute name="transport_lbtru_source_socket_buffer" value="0"/>
                <Attribute name="transport_lbtru_data_rate_limit" value="10000000"/>
                <Attribute name="transport_lbtru_datagram_max_size" value="8192"/>
                <Attribute name="transport_lbtru_rate_interval" value="100"/>
                <Attribute name="request_tcp_reuseaddr" value="1"/>
            </ContextAttributes>
            <SourceAttributes>
                <!-- For unicast, set transport to lbtru or tcp -->
                <Attribute name="transport" value="lbtru"/>
                <Attribute name="late_join" value="1"/>
                <Attribute name="umq_queue_name" value="UMQueue"/>
            </SourceAttributes>
            <ReceiverAttributes>
                <Attribute name="umq_receiver_type_id" value="100"/>
            </ReceiverAttributes>
        </ConnectionFactory>
    </ConnectionFactories>
    <Destinations>
        <Destination name="TempQueue" type="Queue">
            <DestinationAttributes>
                <Attribute name="Name" value="TempQueue"/>
                <Attribute name="WILDCARD" value="false"/>
                <Attribute name="TYPE" value="UMQ"/>
            </DestinationAttributes>
        </Destination>
        <Destination name="UMETopic" type="Topic">
            <DestinationAttributes>
                <Attribute name="Name" value="UMETopic"/>
                <Attribute name="REGID" value="2000"/>
                <Attribute name="WILDCARD" value="false"/>
            </DestinationAttributes>
            <ReceiverAttributes>
            </ReceiverAttributes>
        </Destination>
        <Destination name="DurableTopic" type="Topic">
            <DestinationAttributes>
                <Attribute name="Name" value="DurableTopic"/>
                <Attribute name="REGID" value="6000"/>
                <Attribute name="WILDCARD" value="false"/>
            </DestinationAttributes>
            <ReceiverAttributes>
            </ReceiverAttributes>
        </Destination>
        <Destination name="DestTopic" type="Topic">
            <DestinationAttributes>
                <Attribute name="Name" value="DestTopic"/>
                <Attribute name="REGID" value="3000"/>
                <Attribute name="WILDCARD" value="false"/>
            </DestinationAttributes>
            <ReceiverAttributes>
            </ReceiverAttributes>
        </Destination>
        <Destination name="BytesTopic" type="Topic">
            <DestinationAttributes>
                <Attribute name="Name" value="BytesTopic"/>
                <Attribute name="REGID" value="7000"/>
```

```
                <Attribute name="WILDCARD" value="false"/>
            </DestinationAttributes>
            <ReceiverAttributes>
            </ReceiverAttributes>
        </Destination>
        <Destination name="MapTopic" type="Topic">
            <DestinationAttributes>
                <Attribute name="Name" value="MapTopic"/>
                <Attribute name="REGID" value="6500"/>
                <Attribute name="WILDCARD" value="false"/>
            </DestinationAttributes>
            <ReceiverAttributes>
            </ReceiverAttributes>
        </Destination>
        <Destination name="ObjectTopic" type="Topic">
            <DestinationAttributes>
                <Attribute name="Name" value="ObjectTopic"/>
                <Attribute name="REGID" value="6000"/>
                <Attribute name="WILDCARD" value="false"/>
            </DestinationAttributes>
            <ReceiverAttributes>
            </ReceiverAttributes>
        </Destination>
        <Destination name="TempTopic" type="Topic">
            <DestinationAttributes>
                <Attribute name="Name" value="TempTopic"/>
                <Attribute name="REGID" value="4000"/>
                <Attribute name="WILDCARD" value="false"/>
            </DestinationAttributes>
            <ReceiverAttributes>
            </ReceiverAttributes>
        </Destination>

        <Destination name="ReplyTopic" type="Topic">
            <DestinationAttributes>
                <Attribute name="Name" value="REPLY"/>
                <Attribute name="REGID" value="4400"/>
                <Attribute name="WILDCARD" value="false"/>
                <Attribute name="TYPE" value="LBM"/>

            </DestinationAttributes>
        </Destination>
        <Destination name="RequestTopic" type="Topic">
            <DestinationAttributes>
                <Attribute name="Name" value="RequestTopic"/>
                <Attribute name="REGID" value="5000"/>
                <Attribute name="WILDCARD" value="false"/>
                <Attribute name="TYPE" value="LBM"/>

            </DestinationAttributes>
        </Destination>
    </Destinations>
</JMSConfig>
```

2. Change the lbmrd daemon, ume store, umq queue name according to your UM store, queue or lbmrd configurations.
3. Generate the `.bindings` file in the `RULEPOINT_HOME\system\UM\jmsclient\bin` directory. Run the following script based on the operating system:
   - For Windows, run `config.bat`.
   - For Linux, run `config.sh`.
4. Configure the following properties to create a JMS connection:
   - JNDI Context Factory. com.sun.jndi.fscontext.RefFSContextFactory

- Connection URL. The absolute path of the directory containing the generated .bindings file. For example, file:/E:/workspace/yosemite/RulePoint_61/system/UM/jmsclient/bin/.
- JMS Connection Factory. The connection factory to use. For example, uJMSConnectionFactory.

5. Create a JMS source using the configured connection.
6. In the source configuration, set the JMS destination as the name of the UM topic from which the JMS source reads messages.

INDEX

A

B

P

Q

R

S

T

U

V

W