



Informatica™

Informatica® RulePoint

6.2

Banking Use Case Tutorial

© Copyright Informatica LLC 1998, 2018

This software and documentation contain proprietary information of Informatica Corporation and are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright law. Reverse engineering of the software is prohibited. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica Corporation. This Software may be protected by U.S. and/or international Patents and other Patents Pending.

Use, duplication, or disclosure of the Software by the U.S. Government is subject to the restrictions set forth in the applicable software license agreement and as provided in DFARS 227.7202-1(a) and 227.7702-3(a) (1995), DFARS 252.227-7013(1)(ii) (OCT 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14 (ALT III), as applicable.

The information in this product or documentation is subject to change without notice. If you find any problems in this product or documentation, please report them to us in writing.

Informatica, Informatica Platform, Informatica Data Services, PowerCenter, PowerCenterRT, PowerCenter Connect, PowerCenter Data Analyzer, PowerExchange, PowerMart, Metadata Manager, Informatica Data Quality, Informatica Data Explorer, Informatica B2B Data Transformation, Informatica B2B Data Exchange Informatica On Demand, Informatica Identity Resolution, Informatica Application Information Lifecycle Management, Informatica Complex Event Processing, Ultra Messaging and Informatica Master Data Management are trademarks or registered trademarks of Informatica Corporation in the United States and in jurisdictions throughout the world. All other company and product names may be trade names or trademarks of their respective owners.

Portions of this software and/or documentation are subject to copyright held by third parties, including without limitation: Copyright DataDirect Technologies. All rights reserved. Copyright © Sun Microsystems. All rights reserved. Copyright © RSA Security Inc. All Rights Reserved. Copyright © Ordinal Technology Corp. All rights reserved. Copyright © Aandacht c.v. All rights reserved. Copyright Genivia, Inc. All rights reserved. Copyright Isomorphic Software. All rights reserved. Copyright © Meta Integration Technology, Inc. All rights reserved. Copyright © Intalio. All rights reserved. Copyright © Oracle. All rights reserved. Copyright © Adobe Systems Incorporated. All rights reserved. Copyright © DataArt, Inc. All rights reserved. Copyright © ComponentSource. All rights reserved. Copyright © Microsoft Corporation. All rights reserved. Copyright © Rogue Wave Software, Inc. All rights reserved. Copyright © Teradata Corporation. All rights reserved. Copyright © Yahoo! Inc. All rights reserved. Copyright © Glyph & Cog, LLC. All rights reserved. Copyright © Thinkmap, Inc. All rights reserved. Copyright © Clearpace Software Limited. All rights reserved. Copyright © Information Builders, Inc. All rights reserved. Copyright © OSS Nokalva, Inc. All rights reserved. Copyright Edifecs, Inc. All rights reserved. Copyright Cleo Communications, Inc. All rights reserved. Copyright © International Organization for Standardization 1986. All rights reserved. Copyright © ej-technologies GmbH. All rights reserved. Copyright © Jaspersoft Corporation. All rights reserved. Copyright © is International Business Machines Corporation. All rights reserved. Copyright © yWorks GmbH. All rights reserved. Copyright © Lucent Technologies. All rights reserved. Copyright (c) University of Toronto. All rights reserved. Copyright © Daniel Veillard. All rights reserved. Copyright © Unicode, Inc. Copyright IBM Corp. All rights reserved. Copyright © MicroQuill Software Publishing, Inc. All rights reserved. Copyright © PassMark Software Pty Ltd. All rights reserved. Copyright © LogiXML, Inc. All rights reserved. Copyright © 2003-2010 Lorenzi Davide, All rights reserved. Copyright © Red Hat, Inc. All rights reserved. Copyright © The Board of Trustees of the Leland Stanford Junior University. All rights reserved. Copyright © EMC Corporation. All rights reserved. Copyright © Flexera Software. All rights reserved. Copyright © Jinfonet Software. All rights reserved. Copyright © Apple Inc. All rights reserved. Copyright © Telerik Inc. All rights reserved. Copyright © BEA Systems. All rights reserved. Copyright © PDFlib GmbH. All rights reserved. Copyright © Orientation in Objects GmbH. All rights reserved. Copyright © Tanuki Software, Ltd. All rights reserved. Copyright © Ricebridge. All rights reserved. Copyright © Sencha, Inc. All rights reserved. Copyright © Scalable Systems, Inc. All rights reserved.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>), and/or other software which is licensed under various versions of the Apache License (the "License"). You may obtain a copy of these Licenses at <http://www.apache.org/licenses/>. Unless required by applicable law or agreed to in writing, software distributed under these Licenses is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the Licenses for the specific language governing permissions and limitations under the Licenses.

This product includes software which was developed by Mozilla (<http://www.mozilla.org/>), software copyright The JBoss Group, LLC, all rights reserved; software copyright © 1999-2006 by Bruno Lowagie and Paulo Soares and other software which is licensed under various versions of the GNU Lesser General Public License Agreement, which may be found at <http://www.gnu.org/licenses/lgpl.html>. The materials are provided free of charge by Informatica, "as-is", without warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

The product includes ACE(TM) and TAO(TM) software copyrighted by Douglas C. Schmidt and his research group at Washington University, University of California, Irvine, and Vanderbilt University, Copyright (©) 1993-2006, all rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (copyright The OpenSSL Project. All Rights Reserved) and redistribution of this software is subject to terms available at <http://www.openssl.org> and <http://www.openssl.org/source/license.html>.

This product includes Curl software which is Copyright 1996-2013, Daniel Stenberg, <daniel@haxx.se>. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at <http://curl.haxx.se/docs/copyright.html>. Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

The product includes software copyright 2001-2005 (©) MetaStuff, Ltd. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at <http://www.dom4j.org/license.html>.

The product includes software copyright © 2004-2007, The Dojo Foundation. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at <http://dojotoolkit.org/license>.

This product includes ICU software which is copyright International Business Machines Corporation and others. All rights reserved. Permissions and limitations regarding this software are subject to terms available at <http://source.icu-project.org/repos/icu/icu/trunk/license.html>.

This product includes software copyright © 1996-2006 Per Bothner. All rights reserved. Your right to use such materials is set forth in the license which may be found at <http://www.gnu.org/software/kawa/Software-License.html>.

This product includes OSSP UUID software which is Copyright © 2002 Ralf S. Engelschall, Copyright © 2002 The OSSP Project Copyright © 2002 Cable & Wireless Deutschland. Permissions and limitations regarding this software are subject to terms available at <http://www.opensource.org/licenses/mit-license.php>.

This product includes software developed by Boost (<http://www.boost.org/>) or under the Boost software license. Permissions and limitations regarding this software are subject to terms available at http://www.boost.org/LICENSE_1_0.txt.

This product includes software copyright © 1997-2007 University of Cambridge. Permissions and limitations regarding this software are subject to terms available at <http://www.pcre.org/license.txt>.

This product includes software copyright © 2007 The Eclipse Foundation. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at <http://www.eclipse.org/org/documents/epl-v10.php> and at <http://www.eclipse.org/org/documents/edl-v10.php>.

This product includes software licensed under the terms at <http://www.tcl.tk/software/tcltk/license.html>, <http://www.bosrup.com/web/overlib/?License>, <http://www.stlport.org/doc/license.html>, <http://asm.ow2.org/license.html>, <http://www.cryptix.org/LICENSE.TXT>, http://hsqldb.org/web/hsqldb_license.html, <http://httpunit.sourceforge.net/doc/license.html>, <http://jung.sourceforge.net/license.txt>, http://www.gzip.org/zlib/zlib_license.html, <http://www.openldap.org/software/>

release/license.html, <http://www.libssh2.org>, <http://slf4j.org/license.html>, <http://www.sente.ch/software/OpenSourceLicense.html>, <http://fusesource.com/downloads/license-agreements/fuse-message-broker-v-5-3-license-agreement>; <http://antlr.org/license.html>; <http://aopalliance.sourceforge.net/>; <http://www.bouncycastle.org/licence.html>; <http://www.jgraph.com/jgraphdownload.html>; <http://www.jcraft.com/jsch/LICENSE.txt>; http://jotm.objectweb.org/bsd_license.html; <http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231>; <http://www.slf4j.org/license.html>; <http://nanoxml.sourceforge.net/orig/copyright.html>; <http://www.json.org/license.html>; <http://forge.ow2.org/projects/javaservice/>; <http://www.postgresql.org/about/licence.html>; <http://www.sqlite.org/copyright.html>; <http://www.tcl.tk/software/tcltk/license.html>; <http://www.jaxen.org/faq.html>; <http://www.jdom.org/docs/faq.html>; <http://www.slf4j.org/license.html>; <http://www.iodbc.org/dataspace/iodbc/wiki/IODBC/License>; <http://www.keplerproject.org/md5/license.html>; <http://www.toedter.com/en/jcalendar/license.html>; <http://www.edankert.com/bounce/index.html>; <http://www.net-snmp.org/about/license.html>; <http://www.openmdx.org/#FAQ>; http://www.php.net/license/3_01.txt; <http://srp.stanford.edu/license.txt>; <http://www.schneier.com/blowfish.html>; <http://www.jmock.org/license.html>; <http://xsom.java.net>; <http://benalman.com/about/license/>; <https://github.com/CreateJS/EaselJS/blob/master/src/easeljs/display/Bitmap.js>; <http://www.h2database.com/html/license.html#summary>; <http://jsoncpp.sourceforge.net/LICENSE>; <http://jdbc.postgresql.org/license.html>; <http://protobuf.googlecode.com/svn/trunk/src/google/protobuf/descriptor.proto>; <https://github.com/rantav/hector/blob/master/LICENSE>; <http://web.mit.edu/Kerberos/krb5-current/doc/mitK5license.html>; <http://jibx.sourceforge.net/jibx-license.html>; <https://github.com/lyokato/libgeohash/blob/master/LICENSE>; <https://github.com/hjiang/jsonxx/blob/master/LICENSE>; and <https://code.google.com/p/lz4/>.

This product includes software licensed under the Academic Free License (<http://www.opensource.org/licenses/afl-3.0.php>), the Common Development and Distribution License (<http://www.opensource.org/licenses/cddl1.php>) the Common Public License (<http://www.opensource.org/licenses/cpl1.0.php>), the Sun Binary Code License Agreement Supplemental License Terms, the BSD License (<http://www.opensource.org/licenses/bsd-license.php>), the new BSD License (<http://opensource.org/licenses/BSD-3-Clause>), the MIT License (<http://www.opensource.org/licenses/mit-license.php>), the Artistic License (<http://www.opensource.org/licenses/artistic-license-1.0>) and the Initial Developer's Public License Version 1.0 (<http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/>).

This product includes software copyright © 2003-2006 Joe Walnes, 2006-2007 XStream Committers. All rights reserved. Permissions and limitations regarding this software are subject to terms available at <http://xstream.codehaus.org/license.html>. This product includes software developed by the Indiana University Extreme! Lab. For further information please visit <http://www.extreme.indiana.edu/>.

This product includes software Copyright (c) 2013 Frank Balluffi and Markus Moeller. All rights reserved. Permissions and limitations regarding this software are subject to terms of the MIT license.

This Software is protected by U.S. Patent Numbers 5,794,246; 6,014,670; 6,016,501; 6,029,178; 6,032,158; 6,035,307; 6,044,374; 6,092,086; 6,208,990; 6,339,775; 6,640,226; 6,789,096; 6,823,373; 6,850,947; 6,895,471; 7,117,215; 7,162,643; 7,243,110; 7,254,590; 7,281,001; 7,421,458; 7,496,588; 7,523,121; 7,584,422; 7,676,516; 7,720,842; 7,721,270; 7,774,791; 8,065,266; 8,150,803; 8,166,048; 8,166,071; 8,200,622; 8,224,873; 8,271,477; 8,327,419; 8,386,435; 8,392,460; 8,453,159; 8,458,230; and RE44,478, International Patents and other Patents Pending.

DISCLAIMER: Informatica Corporation provides this documentation "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of noninfringement, merchantability, or use for a particular purpose. Informatica Corporation does not warrant that this software or documentation is error free. The information provided in this software or documentation may include technical inaccuracies or typographical errors. The information in this software and documentation is subject to change at any time without notice.

NOTICES

This Informatica product (the "Software") includes certain drivers (the "DataDirect Drivers") from DataDirect Technologies, an operating company of Progress Software Corporation ("DataDirect") which are subject to the following terms and conditions:

1. THE DATADIRECT DRIVERS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.
2. IN NO EVENT WILL DATADIRECT OR ITS THIRD PARTY SUPPLIERS BE LIABLE TO THE END-USER CUSTOMER FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR OTHER DAMAGES ARISING OUT OF THE USE OF THE ODBC DRIVERS, WHETHER OR NOT INFORMED OF THE POSSIBILITIES OF DAMAGES IN ADVANCE. THESE LIMITATIONS APPLY TO ALL CAUSES OF ACTION, INCLUDING, WITHOUT LIMITATION, BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY, MISREPRESENTATION AND OTHER TORTS.

Publication Date: 2018-07-19

Table of Contents

Preface	9
Informatica Resources.	9
Informatica My Support Portal.	9
Informatica Documentation.	9
Informatica Web Site.	9
Informatica How-To Library.	9
Informatica Knowledge Base.	10
Informatica Support YouTube Channel.	10
Informatica Marketplace.	10
Informatica Velocity.	10
Informatica Global Customer Support.	10
Chapter 1: Introduction to Banking Use Case.....	11
Banking Use Case Overview.	11
Banking Scenario.	11
Problem.	12
RulePoint Solution.	12
High-Level Overview of Banking Use Cases Covered.	12
Proposed Data Model.	13
Chapter 2: Before You Begin.....	14
Before You Begin Overview.	14
Task 1. Create Database Tables.	15
Table Definition Types in the Banking Database.	15
Task 2: Complete Prerequisites.	17
Task 3. Import Existing Samples.	17
Chapter 3: Alerting Customer Account Balance Details.....	19
Alerting Customer Account Balance Details Overview.	19
Step 1. Create a Project.	20
Step 2. Create a Topic.	21
Step 3. Create an SQL Connection.	22
Step 4. Create an SQL Source.	23
Step 5. Create a Schedule.	24
Step 6. Create an RTAM Responder.	25
Step 7. Create an RTAM Response.	26
Step 8. Create a Wizard Rule.	27
Step 9. Deploy the Wizard Rule, Source, and Responder.	28
Step 10. View the RTAM Alert.	31

Chapter 4: Notifying Stock Price Changes.....	32
Notifying Stock Price Changes Overview.	32
Step 1. Create a Topic	34
Step 2. Create Web Service Connection.	34
Step 3. Create a Web Service Source	35
Step 4. Create a Schedule	35
Step 5. Create Email Connection.	36
Step 6. Create Email Responder	36
Step 7. Create Email Response	37
Step 8. Create Wizard Rule.	38
Step 9. Deploy the Rule, Source, and Responder.	40
Step 10. View the Email Response.	40
Chapter 5: Notifying Credit Card Transactions.....	41
Notifying Credit Card Transactions Overview.	41
Step 1. Create a Topic.	42
Step 2. Create a JMS Connection.	43
Step 3. Create a JMS Source	44
Step 4. Create a JMS Responder	45
Step 5. Create a JMS Response	46
Step 6. Create an Advanced Rule.	46
Step 7. Deploy the Rule, Source, and Responder.	47
Step 8. View the JMS Response.	48
Chapter 6: Notifying Free Health Checkups for Account Holders.....	49
Notifying Free Health Checkups for Account Holders Overview.	49
Step 1. Create a Topic.	51
Step 2. Create a Source	51
Step 3. Create a Schedule.	52
Step 4. Create a Watchlist	52
Step 5. Create an Advanced Rule Using the Watchlist.	53
Step 6. Deploy the Rule, Source, and Responder.	54
Step 7. View the Email Response.	54
Chapter 7: Alerting the Bank Manager about Customers with a Particular Account Type in a City.....	55
Alerting the Bank Manager about Customers with a Particular Account Type in a City Overview.	55
Step 1. Create an Analytic	57
Step 2. Create an Advanced Rule	58
Step 3. Deploy the Rule, Source, and Responder.	58
Step 4. View the RTAM Alert.	58

Chapter 8: Alerting Customer Cash Withdrawal Details.....	60
Alerting Customer Cash Withdrawal Details Overview.	60
Step 1. Create a Topic.	61
Step 2. Create an SQL Source.	62
Step 3. Create a Schedule.	63
Step 4. Create an Advanced Rule.	63
Step 5. Deploy the Rule, Source, and Responder.	63
Step 6. View the RTAM Alert.	64
Chapter 9: Alerting Stock Prices to DMAT Account Holders.....	65
Alerting Stock Prices to DMAT Account Holders Overview.	65
Step 1. Create an Advanced Rule.	67
Step 2. Deploy the Rule, Source, and Responder.	67
Step 3. View the Email Alert.	68
Chapter 10: Checking Customer Cash Withdrawal Details for a Specified Duration.....	69
Checking Customer Cash Withdrawal Details for a Specified Duration Overview.	69
Step 1. Create an Advanced Rule.	70
Step 2. Deploy the Rule, Source, and Responder.	71
Step 3. View RTAM Alert.	71
Chapter 11: Alerting Consequent Credit Card Transaction Rejections.....	72
Alerting Consequent Credit Card Transaction Rejections Overview.	72
Step 1. Create an SQL Responder.	74
Step 2. Create an Advanced Rule.	74
Step 3. Deploy the Rule, Source, and Responder.	75
Step 4. View JMS Messages.	75
Step 5. View the RTAM Alert.	76
Chapter 12: Monitoring Balance Threshold of Customers.....	77
Monitoring Balance Threshold of Customers Overview.	77
Step 1. Create a Template.	79
Step 2. Create a Template Rule.	81
Step 3. Create a Deployment Policy.	82
Step 4. Deploy the Rule, Source, and Responder.	82
Step 5. View the RTAM Alert.	83
Step 6. Upgrade a Template.	83
Step 7. Upgrade a Template to Include an Additional Parameter.	85
Step 8. Update the Rule.	86
Step 9. View the RTAM Alert.	87
Step 10. Revert the Template History.	87

Step 11. Create a Template from the Wizard Rule.	88
Step 12. Create a Template Rule.	90
Step 13. Deploy the Rule, Source, and Responder.	91
Step 14. View the RTAM Alert.	91
Chapter 13: Using the Dashboard Functions.	92
Using the Dashboard Functions Overview.	92
Step 1. Perform Tasks for Deployed Sources.	93
Step 2. Perform Tasks for Deployed Topics.	94
Step 3. Perform Tasks for Deployed Rules.	96
Step 4. Perform Tasks for Deployed Responses.	97
Step 5. Set the Time Line Filter.	98
Step 6. Purge an Application Service.	98
Step 7. Purge the Topology.	99
Step 8. View Errors.	99
Chapter 14: Managing Banking Users and Roles.	101
Managing Banking Users and Roles Overview.	101
Step 1. Create a Banking Role.	102
Step 2. Create a User.	102
Step 3. Log In to RulePoint with New User.	103
Step 4. Provide Project-Level Permissions.	103
Step 5. Edit the Role.	103
Chapter 15: Importing and Exporting Objects.	105
Importing and Exporting Objects Overview.	105
Step 1. Export Selected Objects.	105
Step 2. Import Objects.	107
Chapter 16: Setting Up High Availability.	108
Setting Up High Availability Overview.	108
Step 1. Add a Node.	109
Step 2. Create an Event Processor for High Availability.	110
Step 3. Verify the High Availability Settings.	111
Step 4. Update the Deployment Policy.	112
Step 5. Reassign a Rule.	113
Chapter 17: Using Custom Services.	115
Using Custom Services Overview.	115
Step 1. Create the JAR File.	116
Alert Server Downtime Using Custom Source Overview.	116
Step 1. Create a Topic.	117
Step 2. Create a Source.	117

Step 3. Create a Schedule.	118
Step 4. Create a Responder.	118
Step 5. Create a Response.	119
Step 6. Create an Advanced Rule.	119
Step 7. Deploy Objects.	119
Step 8. View Events on the Dashboard.	120
Step 9. Shut Down the Server.	120
Step 10. View the RTAM Alert.	120
Notify Unexpected Bank Shutdown Using Custom Analytics Overview.	120
Step 1. Create a Topic.	122
Step 2. Create an SQL Connection.	122
Step 3. Create a Source.	123
Step 4. Create a Schedule.	123
Step 5. Create an Analytic.	124
Step 6. Create an Email Connection.	124
Step 7. Create an Email Responder.	125
Step 8. Create an Email Response.	126
Step 9. Create an Advanced Rule.	126
Step 10. Deploy Objects	126
Step 11. View Email Alerts.	127
Chapter 18: Using the REST APIs.....	128
Using the REST APIs Overview.	128
Step 1. Log In to RulePoint Using the REST Call.	129
Step 2. Get Project ID to Create Objects in a Project.	130
Step 3. Create a Topic.	131
Step 4. Create a JDBC Connection.	132
Step 5. Create an SQL Source.	132
Step 6. Add Dynamic Schedule to the Source.	133
Step 7. Create an RTAM Responder.	133
Step 9. Create an RTAM Response.	134
Step 10. Create an Advanced Rule.	135
Step 11. Deploy the Responder.	135
Step 12. Deploy the Rule.	136
Step 13. Deploy the Source.	136
Step 14. View the Objects in the Dashboard.	136
Chapter 19: Using Java Adapter for REST API.....	137
Using Java Adapter For REST API Overview.	137
Step 1. Configure the BankingAdapaterDemo.java File.	138
Step 2. Run the Code.	138
Index.....	140

Preface

This guide focuses on how banks can use RulePoint for creating alerting solutions for various banking requirements.

This guide describes the tasks you need to perform to configure RulePoint objects for various banking use cases, and guides you through the deployment process using working example configurations. Select the appropriate use case for your business environment.

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 takes 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 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

Introduction to Banking Use Case

This chapter includes the following topics:

- [Banking Use Case Overview, 11](#)
- [Banking Scenario, 11](#)
- [Problem, 12](#)
- [RulePoint Solution, 12](#)
- [High-Level Overview of Banking Use Cases Covered, 12](#)
- [Proposed Data Model, 13](#)

Banking Use Case Overview

In this case study, you understand how to use RulePoint to discover important situations from events and help you take timely action for business opportunities in the banking industry.

You understand how to configure different types of models in RulePoint to collect data from disparate sources and effectively respond to a pattern of events for business opportunities as well as to stop problems while or before they happen. The banking scenarios covered in this tutorial help you understand how you can provide personalized information to customers on banking activities.

Banking Scenario

The Banking industry has a user base with billions of customers. Banks manage increasing volumes of data from customer information to account balances and transaction history details. Banks require an end-to-end and timely visibility into the state of data and transactional processes, and provide optimized and reliable processes to cater to growing customer needs.

With a rapid rise in crime rates, it is critical for banks to employ the best technology to monitor customer transactions and harness interactions to help you stay connected to any activity in your accounts. Banks need to roll out smart alerts for cash transactions and other fundamental banking processes, which fundamentally changes how they attract and retain customers.

Problem

The requirement of the bank is to monitor and detect events that might need action, to assist decisions regarding risk and fraud mitigation, and to improve operations effectiveness.

Banks need to be able to collect data from multiple sources, detect key events, and use enhanced alert techniques for various banking transactions. For example, customers need to receive alerts for potential fraudulent transactions for amounts withdrawn that exceed a specified threshold, or when large amounts are withdrawn multiple times in a short span. DMAT account holders must receive alerts when a specific stock price reaches a profitable price. Additionally, the Bank Manager must also receive alerts to ensure added security, who can in turn alert the customer who has no access to mails at the time. The manager can also temporarily block malicious card usage until a verification process is complete.

Given the complexity of data, the immediate requirement of the bank is to be able to scale up, and turn the volume, velocity, and variety of data into actionable and real-time intelligence. A bank needs the entire system infrastructure to run at optimum performance to monitor incoming transactions against individual history, and approve or reject transactions and alert the customer on a real-time basis.

RulePoint Solution

The banking industry requires a scalable and reliable complex event processing product as RulePoint to collect and analyze events that occur from all event-enabled system objects, process the available data, and send in the required alerts.

The RulePoint solution gives the banking industry better insights into customer needs, helps design differentiated and enhanced services based on easy-to-use templates and rules, and optimizes the banking process.

To illustrate the problem and demonstrate the approach, this use case covers an analysis of the data type in the banking scene and proposes the data model that you need to build. The lessons cover the basic usage and the type of RulePoint objects that you need to configure to collect and process events from multiple sources, and deliver corresponding alerts to the configured user or system. Based on the type of alerts you require, you can use simple to complex rules. The use cases depict and examine creating objects required for processing data, and the interacting relationships of events and information throughout the imaginary banking scenario.

High-Level Overview of Banking Use Cases Covered

The lessons in this tutorial represent the RulePoint processing flow using data that simulates a banking database and provides solutions for banking requirements.

Each of the use cases depicted here showcase an end-to-end approach to the RulePoint functionality.

The lessons in this tutorial cover tasks for creating objects to collect, assess, process, and send an alert based on the configured rules. The scenario is simplistic and the types of objects used in the lessons are representative to only the use cases here. You can create custom rules and objects based on your requirement.

This tutorial will cover the following banking use cases:

- Alert customer account balance details
- Notify stock price changes
- Notify credit card transactions
- Notify free health checkups
- Retrieve customer details
- Notify customer cash withdrawal details
- Alert stock prices to DMAT account holders
- Check customer cash withdrawal details for a specified duration
- Alert consequent credit card transaction rejections
- Monitor balance threshold of customers

This tutorial also helps you understand the following functionalities in RulePoint:

- Use the dashboard functions
- Manage banking users and roles
- Import and exporting objects
- Set up high availability

This tutorial will also give you an understanding of custom services available in RulePoint and how to use the REST APIs taking the banking use case as an example. The following lessons guide you through the required tasks:

- Use custom services
 - Use custom sources to alert server downtime
 - Use custom analytics to notify unexpected bank shutdown
- Use the REST API
- Use Java Adapter for REST API

Proposed Data Model

The proposal of the data model for the use case consists of creating five basic tables in an Oracle environment, which simulates the type of customer data collected in the banking database.

You can also choose to use H2, IBM DB2, or Microsoft SQL server. The data includes personal and account information of the customer. For example, data consists of the name, date of birth, email ID, phone number, account number, type of account, total balance in the account, and information of the Relationship Manager assigned to the privileged customer.

The proposed model describes how the bank can use RulePoint to retrieve this information from the database and provide relevant services to the customer. The values acquired from the database are used in rule processing and sending out appropriate alerts.

CHAPTER 2

Before You Begin

This chapter includes the following topics:

- [Before You Begin Overview, 14](#)
- [Task 1. Create Database Tables, 15](#)
- [Task 2: Complete Prerequisites, 17](#)
- [Task 3. Import Existing Samples, 17](#)

Before You Begin Overview

You need to consider whether you want to create the RulePoint objects using the tutorial as a guide, or you want to import existing samples that are available in the installation package.

Lesson Concepts

This lesson walks you through the prerequisites for using the available samples. The tasks require you to create the database tables, complete the prerequisites, and then import the samples. You can then follow the lessons in this tutorial to explore each object.

If you plan to create the objects, create the database tables and complete the prerequisites before you start the other lessons in this tutorial.

The tables that you create must simulate a banking database and represent customer information. The lesson provides you an understanding of the table definition types that you must use. This tutorial is based on samples created in the Oracle database.

Lesson Objectives

In this lesson, you learn to perform the following tasks:

- Create database tables.
- Complete system prerequisites.
- Import objects if you want to use the available samples in the installation folder.

Task 1. Create Database Tables

You can create tables in IBM DB2, Microsoft SQL Server, or Oracle. In this lesson, you create tables in the Oracle database under the user rulepoint. You can make changes as and when the implementation progresses.

1. Run the `ddl <database>_cust_acc.sql` script available in the following directory:
for example, for Oracle, run `ddl oracle_cust_acc.sql` from `<RULEPOINT_HOME>/samples/Banking/db/Oracle`
2. Open command prompt, navigate to `<RULEPOINT_HOME>/samples/Banking/db/<database>`, and log in to the database as RulePoint user.
3. Open the `ddl` file, and update the email IDs with the email IDs of your organisation.
The `ddl`s contain the sample email ID.
4. To see the email response in the banking use cases, edit the email IDs in the script, and then run the following script:

```
cd <RULEPOINT_HOME>/samples/Banking/db/<database>
sqlplus rulepoint/rulepoint@infaorcl
@oracle_cust_acc.sql
commit;
```
5. Verify if the following proposed tables are created, so that you can implement the banking use case:
 - CUSTOMER_INFO
 - ACCOUNT_INFO
 - TRANSACTION_INFO
 - PREFERRED_INFO
 - RELATIONSHIP_MANAGER_INFO

Table Definition Types in the Banking Database

You need to create the following property columns for the database tables.

The following table describes the properties for `customer_info`:

Column	Description
<code>cust_id</code>	A unique ID that represents the customer of the bank. This can be a primary key and used to access other tables.
<code>name</code>	Name of the customer.
<code>dob</code>	Date of birth of the customer.
<code>street</code>	Residential street address of the customer for postal communications by the bank.
<code>city</code>	City that the customer lives in.
<code>state</code>	State that the customer lives in
<code>pin_code</code>	The pin code of the customer home address.

Column	Description
email_id	Email ID of the customer used for email communications by the bank
phone_no	Phone number of the customer used for message and call alerts by the bank.

The following table describes the properties for create account_info:

Column	Description
acc_no	Account number, which is a unique ID, to identify the account of the customer.
acc_type	The type of the account. For example, Current, Savings, or DMAT.
cust_id	Used as a referential key to associate a customer to the account.
acc_status	Status of the account. For example, Active, Inactive, or Suspended.
acc_activation_date	The date when the account was active for use.
total_bal	The total available balance in the account.

The following table describes the properties for transaction_info:

Column	Description
acc_no	Account number, which is a unique ID, to identify the account of the customer.
cust_id	Used as a referential key to associate a customer to the account.
amt_withdrawn	The amount withdrawn at a time.
withdraw_time	The time when the amount was withdrawn.

The following table describes the properties for relationship_manager_info:

Column	Description
rel_manager_id	A unique ID that represents the relationship manager in the bank. This can be a primary key and used to access other tables.
rel_manager_name	The name of the Relationship Manager.
bank_branch	The branch of the bank the manager works in.
pin_code	The pin code of the bank branch address.
email_id	The email ID of the manager used for email communication with the customers.
phone_no	The phone number of the manager used for phone communication with the customers.

The following table describes the properties for preferred_customer_info :

Column	Description
rel_manager_id	A referential key to associate a customer to the account.
cust_id	A unique ID that represents the relationship manager in the bank. This can be a primary key and used to access other tables.

Task 2: Complete Prerequisites

Complete the prerequisites before you create or import the RulePoint objects.

1. Install and configure RulePoint.
2. Set the system variable, RULEPOINT_HOME, to the path of the RulePoint 6.2 installation directory.
3. Start the RulePoint instances. To do this, perform the following tasks from the command prompt:
 - a. To start the topology instance, go to <RULEPOINT_HOME>/bin where topology.bat is located, and enter `topology start <TopologyName>`.
 - b. To start the design-time instance, go to <RULEPOINT_HOME>/design/bin where design.bat is located, and enter `design start`.

For information to set the system variable and the RulePoint instances, see the *RulePoint Installation Guide*.

4. To log in to RulePoint user interface, perform the following steps:
 - a. Launch a web browser, and enter the following URL:
`http://host:port/rulepoint`
 - b. Enter the RulePoint login credentials, and click **Log In**.
5. Consider whether you want to import already created objects, or you want to create the objects. Perform the following tasks:
 - To use objects from the available banking sample XML file, perform the steps under Task 3, and then follow the rest of the lessons to explore the objects.
 - If you want to create the objects, you can continue with the rest of the lessons.

Task 3. Import Existing Samples

Perform these tasks if you want to import existing samples.

1. Run the script `cust_acc.sql` to create the required tables and populate the data for the use case located at the following directory:
`<RULEPOINT_HOME>\samples\Banking\db<database>`
Use the required script from the database that you want to use.
2. Copy the banking custom jars, `portmonitorservice.jar` and `startswithanalytic.jar` from `<RULEPOINT_HOME>/samples/Banking/build` and place it in the following directory:

<RULEPOINT_HOME>/custom

3. Create a project **Banking** in RulePoint.
4. Import the `Banking_Sample.xml` file into the project from `<RULEPOINT_HOME>/samples/Banking/db/<database>`.

The XML file contains the configured Rulepoint objects that you require for the banking use case. For information to import samples, see the chapter, "Importing and Exporting Objects" in this tutorial.

Note: The web service related source and connection is not available in the XML. You need to create your own web service source and connection.

5. Edit the connection properties to point to the database where you run the database scripts.

CHAPTER 3

Alerting Customer Account Balance Details

This chapter includes the following topics:

- [Alerting Customer Account Balance Details Overview, 19](#)
- [Step 1. Create a Project, 20](#)
- [Step 2. Create a Topic, 21](#)
- [Step 3. Create an SQL Connection, 22](#)
- [Step 4. Create an SQL Source, 23](#)
- [Step 5. Create a Schedule, 24](#)
- [Step 6. Create an RTAM Responder, 25](#)
- [Step 7. Create an RTAM Response, 26](#)
- [Step 8. Create a Wizard Rule, 27](#)
- [Step 9. Deploy the Wizard Rule, Source, and Responder, 28](#)
- [Step 10. View the RTAM Alert, 31](#)

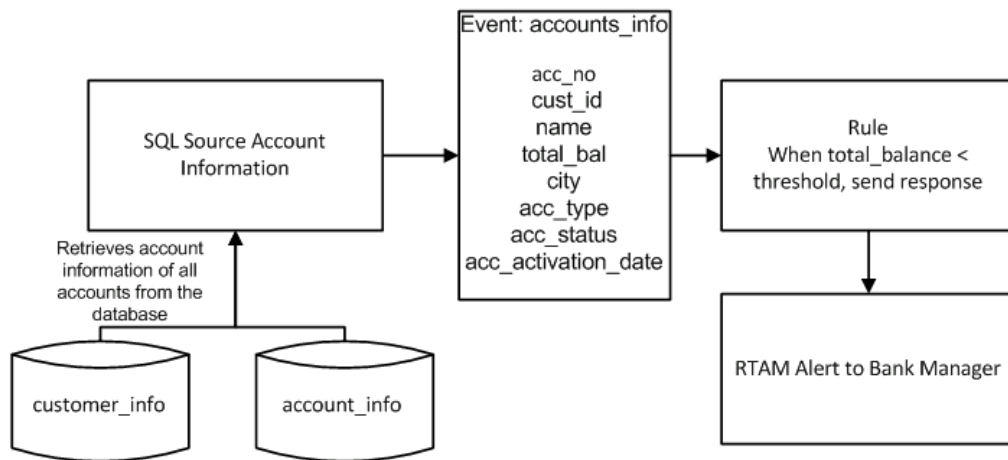
Alerting Customer Account Balance Details Overview

In this lesson, you check for customer accounts with a total balance that is less than 10,000 and send an alert to the manager with the list of these account numbers.

Lesson Concepts

This lesson provides you information to create objects necessary to retrieve specific customer details from the database, deploy the objects in the run-time environment, and see the corresponding rule activation in the dashboard when an event meets the configured condition. In this lesson, you learn to create a topic, SQL connection, SQL source, RTAM responder, RTAM response, and a wizard rule.

The following figure represents the model structure for alerting when the total balance goes below a defined threshold:



The model representation uses an SQL source that connects to the database using an SQL connection and retrieves customer and account information. The source identifies the event properties and publishes all related events on the accounts_info topic. The wizard rule states that when RulePoint receives an event where the total balance is below 10000, create an RTAM response to that event. The RTAM responder delivers the response and you can view the alert on the RTAM.

Lesson Objectives

In this lesson, you complete the following tasks:

- Create a banking use case project.
- Create a topic with customer account information.
- Create an SQL connection to connect to the Oracle database.
- Create an SQL source to retrieve customer account details.
- Create a dynamic schedule.
- Create an RTAM responder to send notifications to the Real-Time Alert Manager.
- Create an RTAM response.
- Create a wizard rule to check for customers with balance below 10000.
- Deploy wizard rule, source, and responder and view objects in the dashboard.
- View an RTAM alert.

Lesson Prerequisites

Before you start this lesson, you must start the RulePoint instances.

Step 1. Create a Project

Create a RulePoint project where you create the RulePoint design objects.

1. Log in to RulePoint.

2. Provide the following user credentials:
 - a. Username: Administrator
 - b. Password: Administrator1
3. On the **Design** tab, select the **Actions** menu on the left pane, and click **New**.
4. Under **Create Project**, configure the following properties:

Field	Value
Name	Banking
Description	Banking Usecase Project

5. Click **Save**.

Step 2. Create a Topic

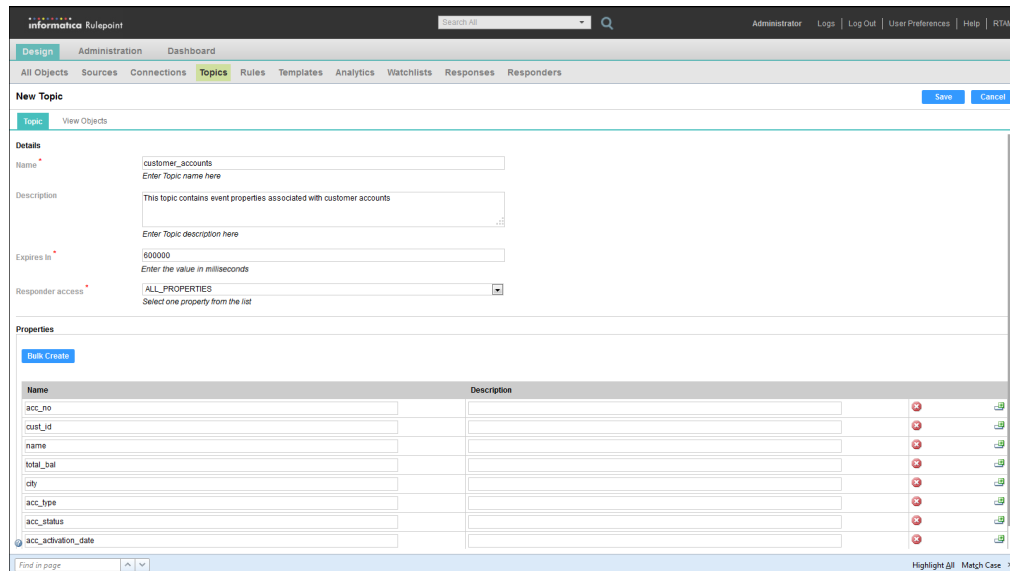
Create a topic to logically group events associated with customer accounts.

1. Select the project **Banking** under which you want to create all the objects related to banking.
2. On the **Design** tab, select **Topics**.
3. In the right pane, click **Actions > New**.
4. Under **Details**, configure the following properties:

Field	Value
Name	customer_accounts
Description	This topic contains event properties associated with customer accounts
Expires in	600000
Responder access	ALL_PROPERTIES

5. Under **Properties**, click **Bulk Create**.
6. Add the following properties under **Topic Properties Names**, and click **Add**.
 - acc_no
 - cust_id
 - name
 - total_bal
 - city
 - acc_type
 - acc_status
 - acc_activation_date

The following figure shows the configured properties for a customer_accounts topic:



7. Click **Save**.

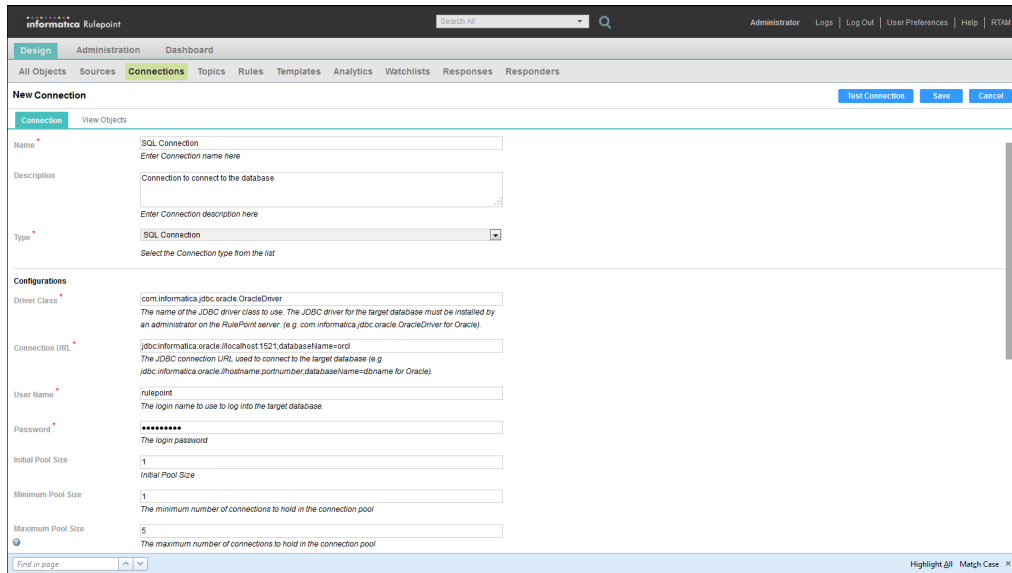
Step 3. Create an SQL Connection

Create an SQL connection to connect the RulePoint service to the target location.

1. On the **Design** tab, select **Connections**.
2. In the right pane, click **Actions > New**.
3. Under **Connection**, enter the following details:
 - a. Name: SQL Connection
 - b. Description: Connection to connect to the database
 - c. Type: SQL Connection
4. Under **Configurations**, configure the following properties:

Field	Value
Driver Class	com.informatica.jdbc.oracle.OracleDriver
Connection URL	jdbc:informatica:oracle://hostname:portnumber;databaseName=dbname
User Name	rulepoint
Password	rulepoint

The following figure shows the configurations for an SQL connection:



5. Click **Test Connection** to verify the success or failure of the connection.
6. Click **Save**.

Step 4. Create an SQL Source

Create an SQL source to fetch customer account details and publish events on the configured topic `customer_accounts`.

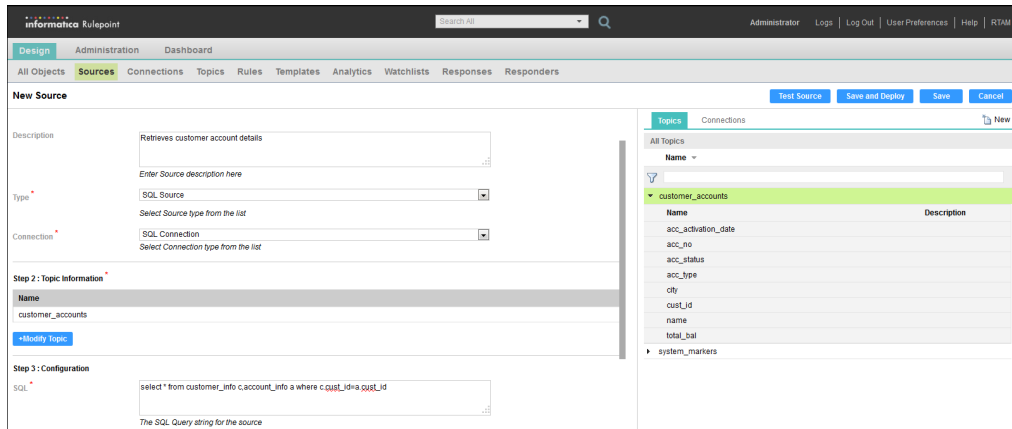
1. On the **Design** tab, select **Sources**.
2. In the right pane, click **Actions > New**.
3. Under **Details**, configure the following properties:

Field	Value
Driver Class	Customer Accounts Source
Description	Retrieves customer account details
Type	SQL Source
Password	rulepoint
Connection	SQL Connection

4. Under **Topic Information**, select the topic name, `customer_accounts`, and click **OK**.
5. Under **Configuration**, type the following information:

```
select * from customer_info c,account_info a where c.cust_id=a.cust_id
```

The following figure shows the configurations for an SQL source:



Note: Test source is available only for an SQL source that runs the SQL query on the database that you specify in the connection. The result of the query is fetched as the topic properties and the **Edit Topic** window appears. For more information, see the *RulePoint User Guide*.

6. Click **Save**.

Step 5. Create a Schedule

Create a dynamic schedule so that the source waits for a configured period after completing the previous task and before it begins publishing events again. You can create a static schedule if you want the source to publish events on a schedule that you set, such as once every 30 minutes.

1. On the **Design** tab, click **Sources**.
2. Select **Customer Accounts Source**, and then click **Create Schedule** from the menu on the right.
3. Configure the following properties:

Field	Value
Schedule Type	Dynamic
Repeat Interval	120000

The following figure shows the **Create Schedule for Customer Accounts Source** dialog box:

Create Schedule for Customer Accounts Source

Schedule Type:

Repeat Interval:

Enter the value in milliseconds

Add Schedule Cancel

4. Click **Add Schedule**.

Step 6. Create an RTAM Responder

Create an RTAM responder to send RTAM alerts when a rule is activated.

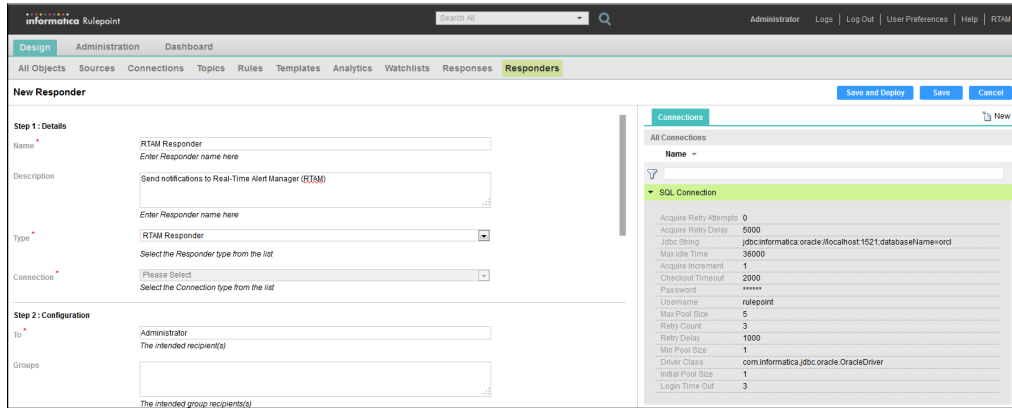
1. On the **Design** tab, select **Responders**.
2. In the right pane, click **Actions > New**.
3. Under **Responder**, configure the following properties:

Field	Value
Name	RTAM Responder
Description	Send notifications to Real-Time Alert Manager (RTAM)
Type	RTAM Responder

4. Under **Configuration**, configure the following properties:

Field	Value
To	RTAM Responder
Subject	rtam alert
Body	rtam alert
Message Priority	3

The following figure shows the configurations for creating a new responder:



Note: The validity of a responder will be false if it does not have a response associated with it.

5. Click **Save**.

Step 7. Create an RTAM Response

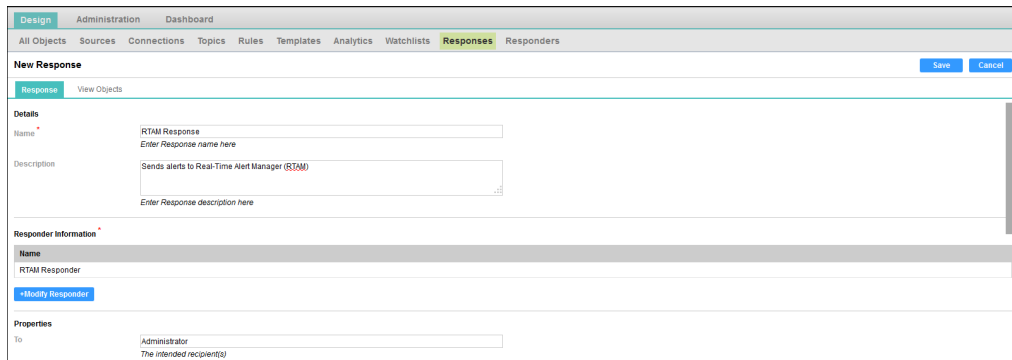
Create a response so that when a rule invokes that response, the response implements the service with specific field values.

1. On the **Design** tab, select **Responses**.
2. In the right pane, click **Actions > New**.
3. Configure the following properties:

Field	Value
Name	RTAM Response
Description	Sends alerts to Real-Time Alert Manager (RTAM)

4. Under **Responder Information**, select **RTAM Responder** from the list, and click **OK**.

The following figure shows the configurations for creating an RTAM response:



Note: All the properties are picked from the responder and are displayed under **Properties**.

5. Click **Save**.

Step 8. Create a Wizard Rule

Create a wizard rule to check for customer accounts with a total balance that is less than 10,000 and send an alert to the manager with the list of these account numbers.

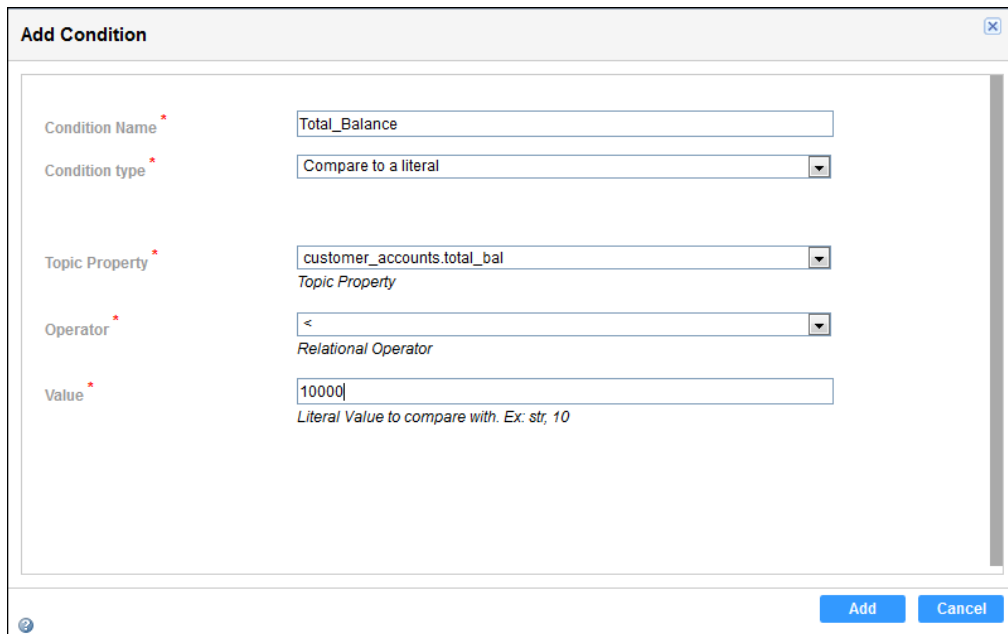
1. On the **Design** tab, select **Rules**.
2. In the right pane, click **Actions > New > New Wizard Rule**.
3. Under **Details**, configure the following properties:

Field	Value
Name	Rule to check for customers with balance below 10000
Description	Checks for customers with balance below 10000

4. Under **Topics & Events**, select the topic **customer_accounts**, and click **OK**.
5. Under **Conditions**, click **Add Conditions**, and configure the following properties:

Field	Value
Condition Name	Total_Balance0
Condition Type	Compare to a literal0
Topic Property	customer_accounts.total_bal
Operator	<
Value	10000

The following figure shows the required configurations for the condition:



The screenshot shows the 'Add Condition' dialog box with the following configuration:

- Condition Name: Total_Balance
- Condition type: Compare to a literal
- Topic Property: customer_accounts.total_bal
- Operator: <
- Value: 10000

Buttons: Add, Cancel

- Click **Add**.
- Under **Responses**, select the response **RTAM Response**, click the **Edit** icon, and configure the following properties:

Field	Value
Subject	Accounts with balance less than 10000
Body	Account \${ customer_accounts.acc_no} has balance less than 10000
Channel	Account Balance
Priority	2

The following figure shows the **Details** view of the configurations for **Add Response**:

The screenshot shows a configuration window titled "Add Response" with a "Details" tab. The fields and their values are as follows:

- Groups:** (Empty field)
- Subject:** Accounts with balance less than 10000
- Body:** Account \${ customer_accounts.acc_no} has balance less than 10000
- Actions:** (Empty field)
- Channels:** Account Balance
- Header:** (Empty field)
- Metadata:** (Empty field)
- Message Priority:** 2

At the bottom right of the window, there are "Save" and "Cancel" buttons.

- Click **Save** to save the response configurations.
- Click **Save**.

Step 9. Deploy the Wizard Rule, Source, and Responder

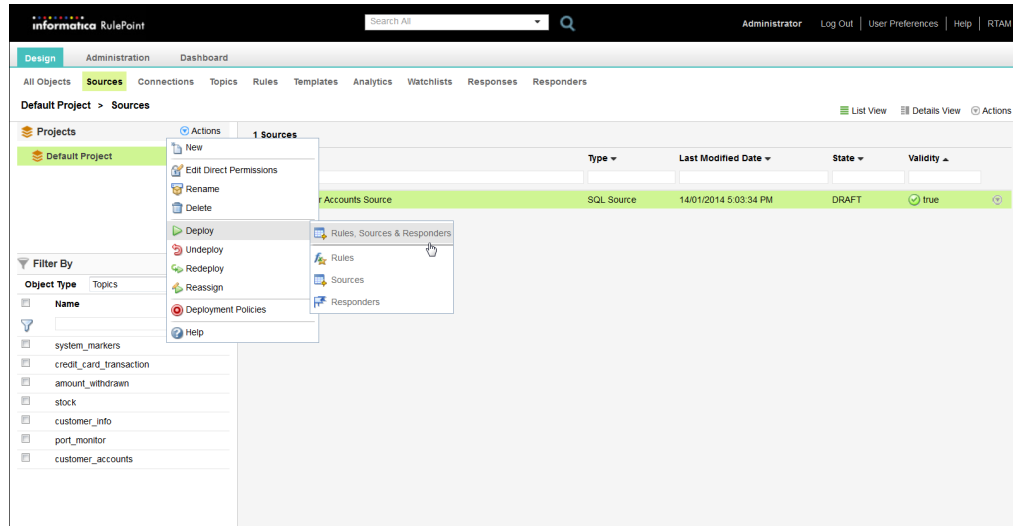
When you create the objects, the objects are in the Draft state and are available only in the design-time environment. When you deploy the objects, the objects will be available in the run-time environment and the

state of the objects changes from Draft to Deployed. If you reference secondary objects in any of the primary objects, the secondary objects are also deployed along with the primary objects.

Before you begin, make sure that the design-time and run-time instances of RulePoint are running.

1. On the **Design** tab, click **Actions > Deploy > Rules, Sources & Responders**.

The following figure displays the **Deploy** action for rules, sources, and responders in the navigator:



2. Select the wizard rule **Rule to check for customers with balance below 10000**, and click **Next**.
3. Select the source **Customer Accounts Source**, and click **Next**.

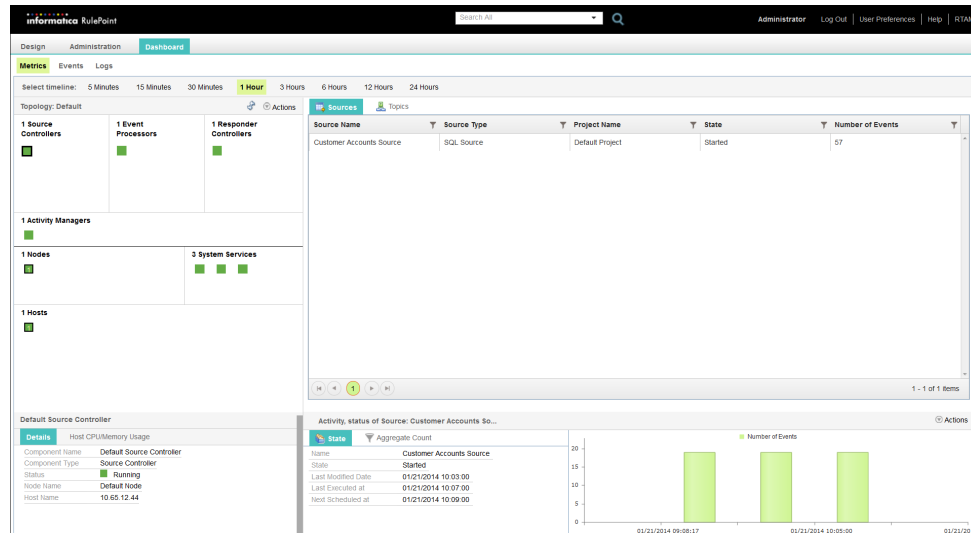
Note: You cannot deploy an object that is not valid. If a source requires a schedule, you cannot deploy that source without configuring a schedule for the source.

4. Select the responder **RTAM Responder**, and click **Deploy**.

Note: You cannot deploy an object that is not valid. You can deploy a responder only if that responder has a configured response.

5. On the **Dashboard** tab, verify if you can view the deployed objects. Perform the following steps:
 - a. Select the default source controller in the left pane to view the deployed sources and supporting objects in the right pane.

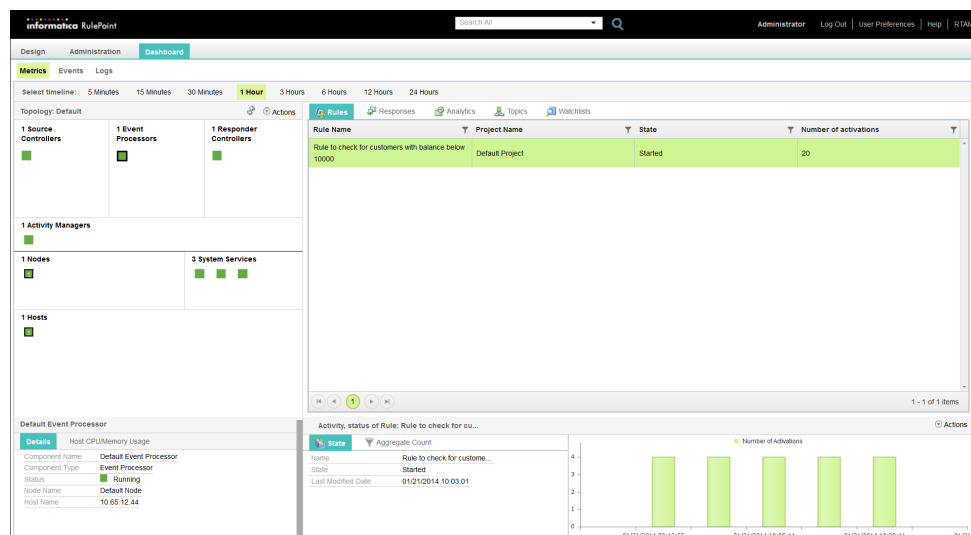
The following figure shows the deployed **Customer Accounts Source** and the number of events generated for the source in the **Sources** view:



You can click the **Topics** view to view the topic associated with the source, the property and the number of events fetched by the source. The graph indicates the time during which the events are fetched by the source.

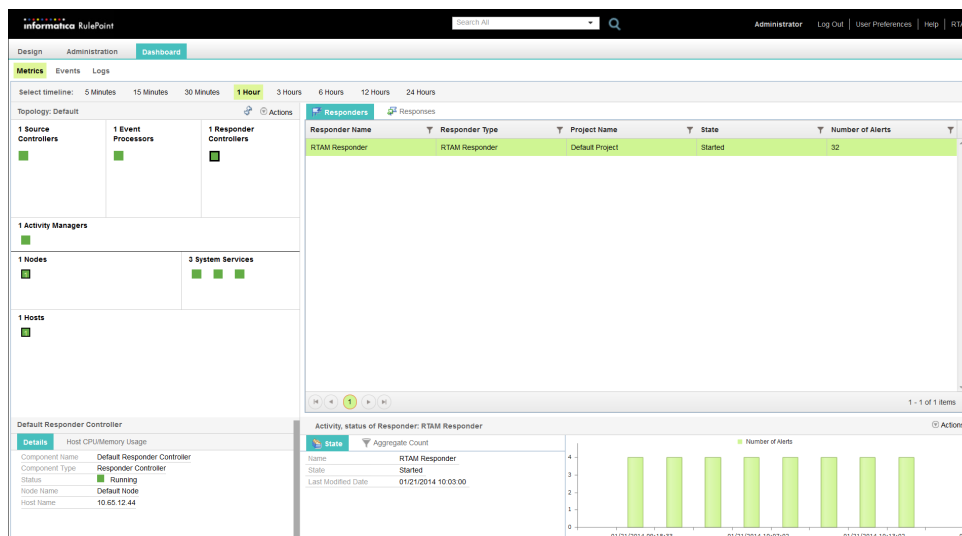
- b. Select the default event processor in the left pane to view the deployed rules and its associated objects, and the number of times the rule is activated.

The following figure shows the deployed **Rule to check for customers with balance below 10000**:



- c. Select the default responder controller in the left pane to view the deployed responder, the associated response, and the number of alerts generated when a rule was activated.

The following figure shows the deployed **RTAM Responder** in the **Responder** view:

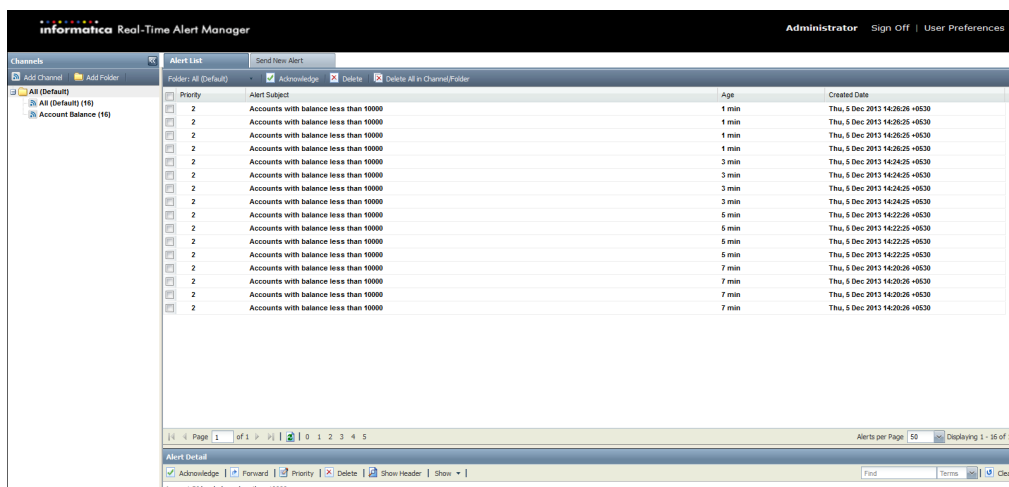


Step 10. View the RTAM Alert

When you configure an RTAM response, RulePoint generates an alert that appears on the RTAM. The response defines the RTAM alert details, including the channel, header, priority, and actions.

1. Log in to RTAM.
2. Under **Account Balance**, verify if the alert displays **Accounts with balance less than 1000**.

The following figure shows the RTAM displaying the alert "Accounts with balance less than 1000."



CHAPTER 4

Notifying Stock Price Changes

This chapter includes the following topics:

- [Notifying Stock Price Changes Overview, 32](#)
- [Step 1. Create a Topic , 34](#)
- [Step 2. Create Web Service Connection, 34](#)
- [Step 3. Create a Web Service Source , 35](#)
- [Step 4. Create a Schedule , 35](#)
- [Step 5. Create Email Connection, 36](#)
- [Step 6. Create Email Responder , 36](#)
- [Step 7. Create Email Response , 37](#)
- [Step 8. Create Wizard Rule, 38](#)
- [Step 9. Deploy the Rule, Source, and Responder, 40](#)
- [Step 10. View the Email Response, 40](#)

Notifying Stock Price Changes Overview

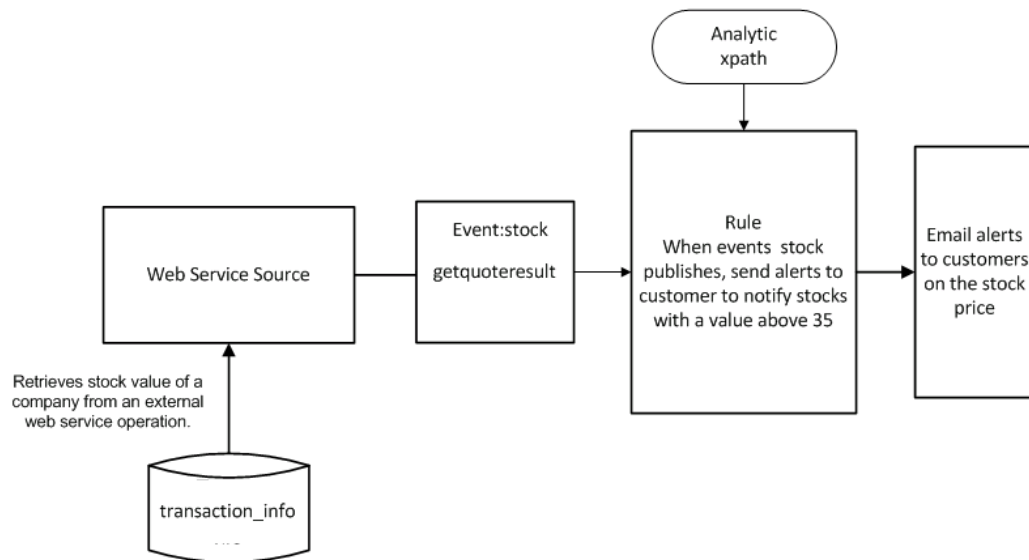
When the stock price of the required company goes above a specified price, the customer must receive an email alert that indicates the stock price.

Lesson Concepts

In this lesson, the tasks involve creating a web service source in RulePoint to retrieve stock information from the web service. The web service source service sends a formatted SOAP message to execute a remote WSDL operation and use the result to create events. When you add the company name as a parameter in the source, the source picks up the stock prices from that company. Rules respond to the event by invoking an email response. The email responder delivers the response to the customer.

The wizard rule is an easy-to-use interface where you can select the topics, conditions, and response for configuring the rule. In this lesson, you learn how to configure a wizard rule to check stock prices with a value that is greater than 35 and send an alert to the customer with the value of the stock.

The following figure represents the data model for sending an email alert to the customer when a specific stock price of a company reaches a desired amount:



The model representation uses a web service source that connects to the web services using a web services connection to retrieve transaction information. The source identifies the event property getquoteresult and publishes the events on the stock topic. The wizard rule states that when RulePoint receives a stock event with a value more than 35, create an email alert to the customer. The rule uses an xpath analytic, which uses an XPath expression "GetQuoteResult" to find information for a specific stock attribute and returns the list of objects that match.

Lesson Objectives

In this lesson, you complete the following tasks:

- Create a topic that contains event properties associated with the stock.
- Create a web service connection.
- Create a web service source to retrieve stock details from the web service.
- Create a schedule for the web service source.
- Create an email connection to connect to the web services.
- Create an email responder to send the email response.
- Create an email response.
- Create a wizard rule to send an email alert of the stock price to specific customers..
- Deploy the rule, source, and responder.
- View the email response.

Step 1. Create a Topic

Create a topic named stock with the property getquoteresult. Events are published on the topic with the values for the property getquoteresult.

1. Select the project **Banking** under which you want to create all the objects.
2. In the contents pane, select **Actions > New**.
3. On the **Design** tab, select **Topics**.
4. In the contents pane, click **Actions > New**.
5. Under **Details**, configure the following properties:

Field	Value
Name	stock
Description	This topic contains event properties associated with stock
Expires in	600000
Responder access	ALL_PROPERTIES

6. Under **Properties**, create the topic property, getquoteresult.
7. Click **Save**.

Step 2. Create Web Service Connection

Create a web service connection to connect the RulePoint service to the web service.

1. On the **Design** tab, select **Connections**.
2. In the contents pane, click **Actions > New**.
3. Under **Connection**, configure the following properties:

Field	Value
Name	Web Service Connection
Description	Connection to connect to the Web Service
Type	Web Service Connection

4. Under **Configurations**, configure the following properties:

Field	Value
URL	http://www.websvcex.net/stockquote.asmx?WSDL

Note: If the web service URL is down, use a different web service connection.

5. Click **Test Connection** to verify the success or failure of the connection.
6. Click **Save**.

Step 3. Create a Web Service Source

Create a web service source to fetch stock events from the web service.

1. On the **Design** tab, select **Sources**.
2. In the contents pane, click **Actions > New**.
3. Under **Details**, configure the following properties:

Field	Value
Name	Stock Source
Description	Retrieves stock details from the web service
Type	Web Service Source
Connection	Web Service Connection

4. Expand **Topic Information**, select the topic name **stock**, and click **OK**.
5. Expand **Configuration**, configure the following properties:

Field	Value
Service	StockQuote
Operation To Execute	GetQuote
Input Parameters	INFA
XPATH Expression	//GetQuoteResult

6. Click **Save**.

Step 4. Create a Schedule

Create a dynamic schedule for the web service source.

1. Under **Stock Source > Schedules**, click **Create New Schedule**.

2. Configure the following properties:

Field	Value
Schedule Type	Dynamic
Repeat Interval	120000

3. Click **Add Schedule**.

The source runs at every two-minute intervals and publishes the stock price of various companies.

Step 5. Create Email Connection

Create an email connection so that RulePoint connects to the mail server.

1. On the **Design** tab, select **Connections**.
2. In the contents pane, click **Actions > New**.
3. Under **Connection**, configure the following properties:

Field	Value
Name	Email Connection
Description	Connection to connect to the mail server
Type	Email Connection

4. Under **Configurations**, configure the following properties:

Field	Value
Email Server	Provide your email server address, for example, mail.company.com

5. Click **Test Connection** to verify the success or failure of the connection.
6. Click **Save**.

Step 6. Create Email Responder

Create an email responder to send email responses to alert the customer of the stock price.

1. On the **Design** tab, select **Responders**.
2. In the contents pane, click **Actions > New**.

- Under **Responder**, configure the following properties:

Field	Value
Name	Email Responder
Description	Sends Email Response
Type	Email Responder

- Under **Configuration**, configure the following properties:

Field	Value
To	Provide a email ID to receive the alerts.
Subject:	Stock Alert
Body	Stock price
Content Type	html
From	admin@mail.company.com

- Click **Save**.

Step 7. Create Email Response

Create an email response to send the email response.

- On the **Design** tab, select **Responses**.
- In the contents pane, click **Actions > New**.
- Configure the following properties:

Field	Value
Name	Email Response
Description	Sends Email Response

- Under **Responder Information**, select **Email Responder**.
- Click **Save**.

Step 8. Create Wizard Rule

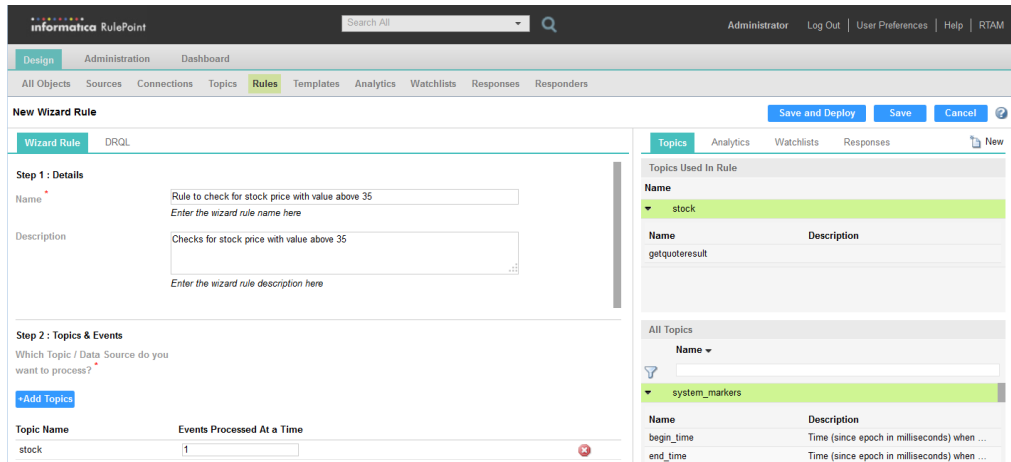
Create a wizard rule to send an email alert of the stock price to specific customers. It checks for stock prices that have a value that is greater than 35 and sends an alert to the customer with the value of the stock.

1. On the **Design** tab, select **Rules**.
2. In the contents pane, click **Actions > New > New Wizard Rule**.
3. Under **Details**, configure the following properties:

Field	Value
Name	Rule to check for stock price with value above 35
Description	Checks for stock price with value above 35

4. Expand **Topics & Events**, select the topic stock, and click **OK**.

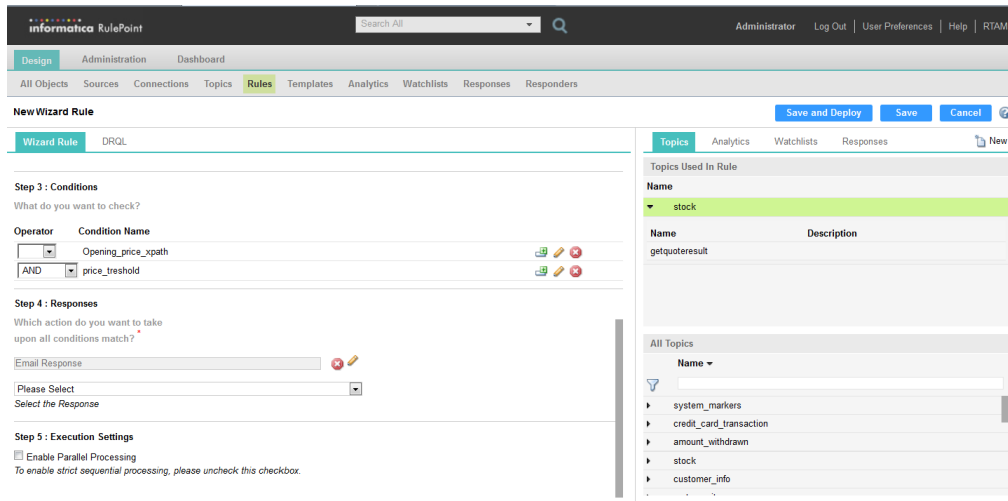
The following figure shows the Wizard Rule configurations for stock alerts:



5. Expand **Conditions**, click the **Add** conditions icon, configure the following properties for the first condition, and then click **Add**:

Field	Value
Condition Name	Opening_price_xpath
Condition Type	Call an analytic
Analytic Name	xpath
Arguments	stock.getquoteresult,'/StockQuotes/Stock/Open'
Output Names	price

The following figure shows the configurations for applying conditions in a wizard rule:



- Configure the following properties for the second condition, and click **Add**:

Field	Value
Condition Name	price_threshold
Condition Type	Compare condition name with Literal
Variable	Analytic Output Name: price
Operator	>
Value	35

- Expand **Responses**, select **Email Response**, click **Edit** icon, and configure the following properties:

Field	Value
To	student01@mail.company.com (the mail address where you want to receive the alerts)
Subject	Stock with price greater than 35
Body	The stock price of the company INFA is \${price}
Content Type	text
From	admin@mail.company.com

- Click **Save** to save the response.
- Click **Save**.

Step 9. Deploy the Rule, Source, and Responder

Deploy the configured rule, source, and responder to the run-time environment.

1. In the navigator, click **Actions > Deploy > Rules, Sources & Responders**.
2. Select the rule **Rule to check for stock price with value above 35**, and click **Next**.
3. Select the source **Stock Source**, and click **Next**.
4. Select the responder **Email Responder**, and click **Deploy**.
5. Navigate to the **Dashboard** and verify that the source, rule, responder, and their related objects are displayed.
6. Verify if events are generated and the rule is activated.

Step 10. View the Email Response

When you configure an email alert, RulePoint sends you an alert by email.

1. Open your email application.
2. Verify that you receive the alert by email.

CHAPTER 5

Notifying Credit Card Transactions

This chapter includes the following topics:

- [Notifying Credit Card Transactions Overview, 41](#)
- [Step 1. Create a Topic, 42](#)
- [Step 2. Create a JMS Connection, 43](#)
- [Step 3. Create a JMS Source , 44](#)
- [Step 4. Create a JMS Responder , 45](#)
- [Step 5. Create a JMS Response , 46](#)
- [Step 6. Create an Advanced Rule, 46](#)
- [Step 7. Deploy the Rule, Source, and Responder, 47](#)
- [Step 8. View the JMS Response, 48](#)

Notifying Credit Card Transactions Overview

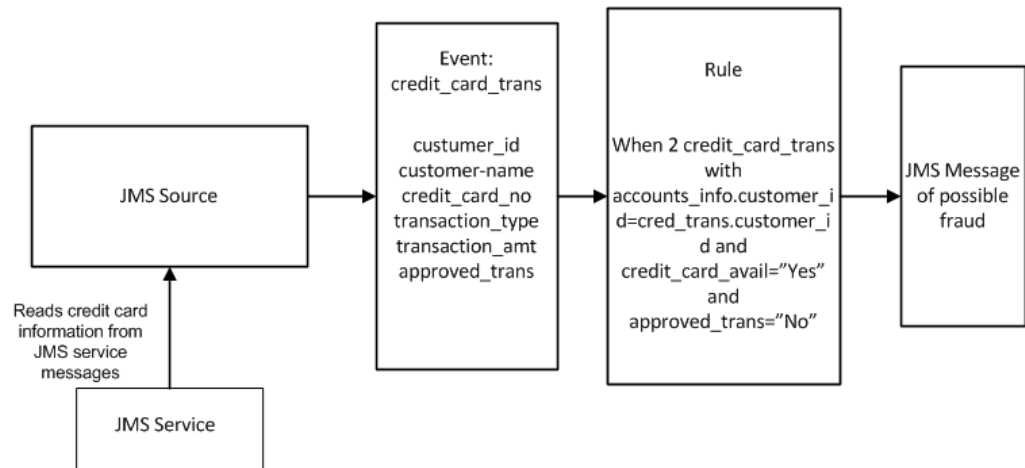
Notify credit card transactions to a customer. If a transaction is rejected consecutively, send an alert stating that a possible fraud is detected in the transaction.

Lesson Concepts

This tutorial provides you information about creating RulePoint objects to read from JMS services that send messages containing the credit card transactions of the customer. The JMS source service produces events by receiving messages from a JMS provider and publishes those messages as events. The messages from a JMS service are in XML format, which contains properties, such as credit card number, customer name, transaction types, transaction amount, and the approved transaction.

This tutorial uses a JMS source and a JMS responder to explain the transactions. The rule is of advanced type where you use DRQL to create the rule.

The following figure represents the model structure for sending a JMS alert for a possible fraud detected in the transaction:



The model representation uses a JMS source that reads the JMS message and publishes it to a topic named `credit_card_info`. The rule states that if the transaction read specifies the `approved_transaction` as false two times in a row, consider the transaction as malicious and send a JMS message to the bank as an alert.

Lesson Objectives

In this lesson, you will learn how to perform the following tasks:

- Create a topic that contains event properties associated with credit card transaction.
- Create a JMS connection to connect to the JMS server.
- Create a JMS source to monitor the credit card transactions.
- Create JMS responder.
- Create JMS response.
- Create advanced rule to send a JMS response when a fraud credit card transaction is detected.
- Deploy the rule, source, and responder.
- View the JMS alert.

Lesson Prerequisites

Before you start this lesson, you must perform the following tasks:

- Start the ActiveMQ server.
- Create the required queues, `credit_card` and `fraud_detection`.

Step 1. Create a Topic

Create a topic `credit_card_transaction` so that events are published on this topic.

1. Select the project **Banking** under which you want to create all the objects related to banking.
2. On the **Design** tab, select **Topics**.
3. In the right pane, click **Actions > New**.

- Under **Details**, configure the following values:

Field	Value
Name	credit_card_transaction
Description	This topic contains event properties associated with credit card transaction
Expires in	600000
Responder access	ALL_PROPERTIES

- Under **Properties**, create the topic property, jmstext.
- Click **Save**.

Step 2. Create a JMS Connection

Create a JMS connection to connect RulePoint to the JMS source.

- On the **Design** tab, select **Connections**.
- In the right pane, click **Actions > New**.
- Under **Connection**, configure the following values:

Field	Value
Name	JMS Connection
Description	JMS Connection
Type	JMS Connection

- Under **Configurations**, configure the following values:

Field	Value
JNDI Context Factory	org.apache.activemq.jndi.ActiveMQInitialContextFactory
Connection URL	tcp://<IP address of the JMS server on the network>

- Click **Test Connection** to verify if the connection is successful.
- Click **Save**.

Step 3. Create a JMS Source

Create a JMS source to monitor credit card transactions.

1. On the **Design** tab, select **Sources**.
2. In the right pane, click **Actions > New**.
3. In the **Details** section, configure the following values:

Field	Value
Name	JMS Source to monitor Credit card Transactions
Description	Monitors Credit card Transactions
Type	JMS Source
Connection	JMS Connection

4. In the **Topic Information** section, select the topic name, **credit_card_transaction**, and click **OK**.
5. In the **Configuration** section, type `credit_card` for **JMS Destination**.

The following figure shows the topic, connection, and other configurations for a JMS source:

The screenshot displays the 'New Source' configuration window in Informatica Rulepoint. The window is organized into three main sections:

- Step 1 : Details:** Contains input fields for 'Name' (JMS Source to monitor Credit card Transactions), 'Description' (Monitors Credit card Transactions), 'Type' (JMS Source), and 'Connection' (JMS Connection).
- Step 2 : Topic Information:** Contains a 'Name' field with the value 'credit_card_transaction'.
- Step 3 : Configuration:** Contains a 'JMS Destination' field with the value 'credit_card'.

On the right side, a 'Topics' pane is visible, showing a list of topics. The topic 'credit_card_transaction' is selected and highlighted in green. Below the list, there is a table with columns 'Name' and 'Description'.

6. Click **Save**.

Step 4. Create a JMS Responder

Configure the values for the JMS responder so that RulePoint sends the alert to the configured destination.

1. On the **Design** tab, select **Responders**.
2. In the right pane, click **Actions > New**.
3. Under **Responder**, configure the following values:

Field	Value
Name	JMS Responder
Description	Sends JMS Response
Type	JMS Responder
Connection	JMS Connection

4. Under **Configuration**, configure the following values:

Field	Value
JMS Destination	fraud_detection
Delivery Mode	Non-Persistent

The following figure shows the responder configurations:

The screenshot shows the 'New Responder' configuration interface in Informatica Rulepoint. The main configuration area includes the following fields:

- JMS Destination:** fraud_detection
- Priority:** 4
- Expiration Time:** 0
- Parameters:** (empty)
- Body:** (empty)
- Delivery Mode:** Non-Persistent
- Retry Count:** 3

The right-hand pane shows the 'Connections' list with a 'JMS Connection' selected. Its details are as follows:

- Name:** JMS Connection
- Package:** Jms
- Prefixes:** ConnectionFactory
- Location:** tcp://10.10.10.10:61616
- Context:** org.apache.activemq.jndi.ActiveMQInitialContext
- Factory:** (empty)
- User Name:** (empty)
- Password:** *****

5. Click **Save**.

Step 5. Create a JMS Response

Configure the mode of response to send the alert.

1. On the **Design** tab, select **Responses**.
2. In the right pane, click **Actions > New**.
3. Under **New Response**, configure the following values:

Field	Value
Name	JMS Response
Description	Sends JMS Response

4. Under **Responder Information**, select JMS Responder.

The following figure shows the JMS response configurations:

The screenshot shows the 'New Response' configuration form in the Informatica Rulepoint interface. The form is divided into several sections:

- Response**: Includes a text input for 'Enter Response name here' and a text area for 'Description' containing 'Sends JMS Response'.
- Responder Information**: Includes a text input for 'Name' containing 'JMS Responder' and a '+Modify Responder' button.
- Properties**: Includes a text input for 'JMS Destination' containing 'fraud_detection', a text input for 'Priority' containing '4', and a text input for 'Expiration Time' containing '0'. Below the 'JMS Destination' input is a detailed instruction: 'Name of the Queue or Topic on the JMS Provider (as defined in the JMS Connection Factory created by the JMS Administrator) to where RulePoint will send JMS messages, (e.g. 'QueueA' or 'topic/SalesSystemsTopic'). Use 'topic' prefix to specify a Topic.'

5. Click **Save**.

Step 6. Create an Advanced Rule

Create an advanced rule to check for fraud in credit card transactions and send a JMS alert.

1. On the **Design** tab, select **Rules**.
2. In the right pane, click **Actions > New > New Advanced Rule**.

- Under **Details**, configure the following values:

Field	Value
Name	Rule to check for fraud in credit card transactions
Description	Checks for fraud in credit card transactions

- Under **Rule Statement**, and provide the following rule:

when 1 credit_card_transaction ccard with ccard.jmstext contains "NO" then "JMS Response" with body="Fraud detected in credit card transaction"

The following figure shows the advanced rule configurations:

The screenshot displays the 'New Advanced Rule' configuration page in Informatica Rulepoint. The top navigation bar includes 'Design', 'Administration', and 'Dashboard'. The main navigation menu has 'All Objects', 'Sources', 'Connections', 'Topics', 'Rules', 'Templates', 'Analytics', 'Watchlists', 'Responses', and 'Responders'. The 'Rules' tab is active. The page title is 'New Advanced Rule' with buttons for 'Save and Deploy', 'Save', and 'Cancel'. The configuration is divided into two steps:

- Step 1: Details**:
 - Name**: Rule to check for fraud in credit card transactions
 - Description**: Checks for fraud in credit card transactions
- Step 2: Rule Statement**:
 - Rule Statement**: when 1 credit_card_transaction ccard with ccard.jmstext contains "NO" then "JMS Response" with body="Fraud detected in credit card transaction"

The right sidebar shows the 'Topics' panel with a search filter and a 'No Items' message. At the bottom, there are instructions: 'Press F11 for full screen' and 'Press Esc to exit full screen'.

- Click **Save**.

Step 7. Deploy the Rule, Source, and Responder

Deploy the objects to the corresponding services in the run-time environment for rule processing to begin.

- In the navigator, click **Actions > Rules, Sources & Responders > Deploy**.
- Select the rule **Rule to check for fraud in credit card transactions**, and click **Next**.
- Select the source **JMS Source to monitor Credit Card Transactions**, and click **Next**.
- Select the responder **JMS Responder**, and click **Deploy**.
- Navigate to the **Dashboard** tab, and verify that the source, rule, responder and the related objects are displayed.

Step 8. View the JMS Response

After you configure and deploy the objects, view the responses in the ActiveMQ server.

1. Go to the URL: `http://<hostname>:<port>/admin/queues.jsp`
2. In the queue list, verify that the queue `credit_card` is present under which the credit card transaction details will be published.
3. Click **Send To** under **Operations** tab of `credit_card`, and enter the following data in the message body, and click **Send**:

```
<event topic="CC_9678535853279753">
  <property>
    <name>Credit Card No</name>
    <value>9678535853279753</value>
  </property>
  <property>
    <name>Customer Name</name>
    <value>John Smith</value>
  </property>
  <property>
    <name>Transaction Type</name>
    <value>sale</value>
  </property>
  <property>
    <name>Merchant Account</name>
    <value>SPINT65280378312</value>
  </property>
  <property>
    <name>Amount</name>
    <value>400</value>
  </property>
  <property>
    <name>Approval</name>
    <value>NO</value>
  </property>
  <property>
    <name>acc_no</name>
    <value>2</value>
  </property>
</event>
```

4. Verify that the counts of enqueued and dequeued messages are updated.
5. Navigate to the **Dashboard** tab in RulePoint, and verify that the counts for the sources, rules, and responders are updated, and the rule is activated.
6. Go to ActiveMQ server URL, and click **fraud_detection** queue, and select the Message ID to view the JMS response.

CHAPTER 6

Notifying Free Health Checkups for Account Holders

This chapter includes the following topics:

- [Notifying Free Health Checkups for Account Holders Overview, 49](#)
- [Step 1. Create a Topic, 51](#)
- [Step 2. Create a Source , 51](#)
- [Step 3. Create a Schedule, 52](#)
- [Step 4. Create a Watchlist , 52](#)
- [Step 5. Create an Advanced Rule Using the Watchlist, 53](#)
- [Step 6. Deploy the Rule, Source, and Responder, 54](#)
- [Step 7. View the Email Response, 54](#)

Notifying Free Health Checkups for Account Holders Overview

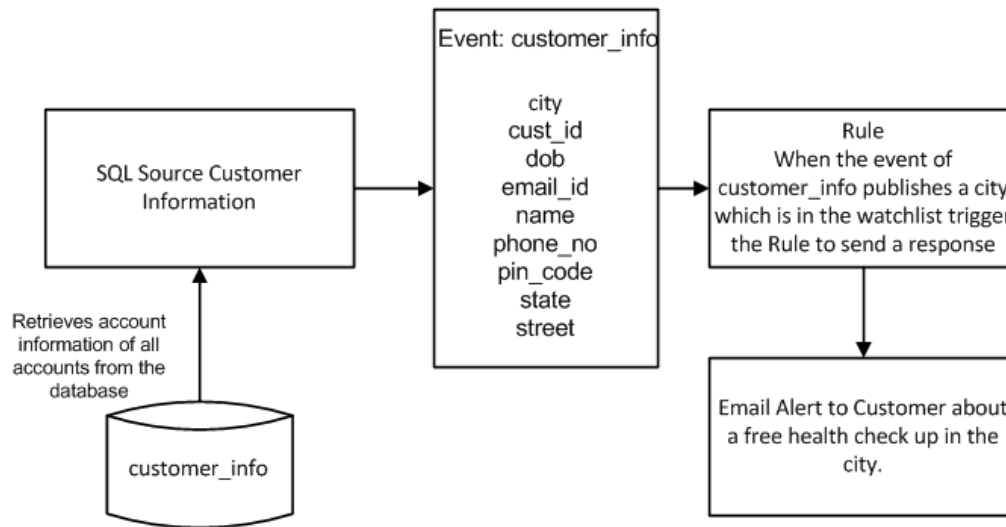
Send an alert to account holders of specific cities notifying a free health checkup.

Lesson Concepts

In this lesson, you learn to create watchlists among other objects to create a free health checkup notification. The watchlist stores items as a single object with a unique name that you define. When you reference the watchlist name in a rule, the rule uses the data stored in the object. You can change the items in a watchlist at any time, and any rule that references that watchlist automatically uses those new items.

In this tutorial, you create a watchlist for Florida cities, and use the watchlist in a wizard rule to send notifications for customers living in the listed cities informing them of a free health checkup.

The following figure represents the model structure for notifying customers living in Florida about a free health checkup:



The model representation uses an SQL source that connects to the database using an SQL connection and retrieves customer information. The source identifies the event properties and publishes all related events on the customer_info topic. The advanced rule states that when the customer lives in the city that is listed in the watchlist, send an email notification about a free health checkup organized in that city.

Lesson Objectives

In this lesson, you will learn how to perform the following tasks:

- Create a topic within a new project.
- Create an SQL source that uses an SQL connection.
- Create a dynamic schedule.
- Create a watchlist listing the cities of Florida.
- Create an advanced rule that uses the watchlist to check for customers living in Florida and send an alert about a free health checkup.
- Deploy the rule, source, and responder.
- View the email response.

Lesson Prerequisites

You must have completed the following tasks from previous lessons:

- Create an SQL connection to connect RulePoint to the source data.
- Create an email connection to connect to your mail server.
- Create an email responder to send the email response.
- Create an email response to receive the response.

Step 1. Create a Topic

Create a topic for customer information and provide the necessary properties for the topic.

1. Select the project **Custom Objects**, and then select **Topics**.
2. In the right pane, select **Actions > New**.
3. Under **Details**, configure the following properties:

Field	Value
Name	customer_info
Expires in	600000
Responder access	ALL_PROPERTIES

4. Under **Properties**, click **Bulk Create**.
5. Configure the following properties in the editor, and then click **Add**:
 - city
 - cust_id
 - dob
 - email_id
 - name
 - phone_no
 - pin_code
 - state1
 - street
6. Click **Save**.

Step 2. Create a Source

Create a source to retrieve customer information and publish events on specific topics.

1. On the **Design** tab, select **Sources**.
2. In the right pane, click **Actions > New**.
3. Under **Details**, configure the following properties:

Field	Value
Name	Customer Information Source
Description	Retrieves customer Information details

Field	Value
Type	SQL Source
Connection	SQL Connection

4. Expand **Topic Information**, select the topic name, customer_info, and click **OK**.
5. Expand **Configuration**, and type `select * from customer_info` under **SQL**.
6. Click **Save**.

Step 3. Create a Schedule

Create a schedule for the SQL source.

1. Under **Customer Accounts Source > Schedules**, click **Create New Schedule**.
2. Configure the following properties:

Field	Value
Schedule Type	Dynamic
Repeat Interval	120000 (in milliseconds)

3. Click **Add Schedule**.

Step 4. Create a Watchlist

Create a watchlist that contains the major cities of Florida. You store this list 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.

1. On the **Design** tab, select **Watchlists**.
2. In the contents pane, select new from the **Actions** menu.
3. Under **Watchlists**, configure the following properties:

Field	Value
Name	Cities in Florida
Description	List of major cities of Florida
Type	List
Content	Melbourne Cooper City Dade City Gulfport Venice Palm Bay Seattle Sunrise

Note: Place each city one below the other.

4. Click **Save**.

The following figure shows the watchlist configurations in the **Watchlist** view of the **Design** tab:

The screenshot shows the 'New Watchlist' configuration page in the Informatica Rulepoint interface. The page has a dark header with the Informatica logo and a search bar. Below the header, there are navigation tabs for 'Design', 'Administration', and 'Dashboard'. Under the 'Design' tab, there are sub-tabs for 'All Objects', 'Sources', 'Connections', 'Topics', 'Rules', 'Templates', 'Analytics', 'Watchlists', 'Responses', and 'Responders'. The 'Watchlists' sub-tab is active. The main content area is titled 'New Watchlist' and includes a 'View Objects' link. The form has four main sections: 'Details', 'Name', 'Description', and 'Type'. The 'Name' field contains 'Cities in Florida'. The 'Description' field contains 'List of major cities of Florida'. The 'Type' field is set to 'List'. The 'Content' field contains a list of city names: Melbourne, Cooper City, Dade City, Gulfport, Venice, Palm Bay, Seattle, and Sunrise. There are 'Save' and 'Cancel' buttons at the top right of the form.

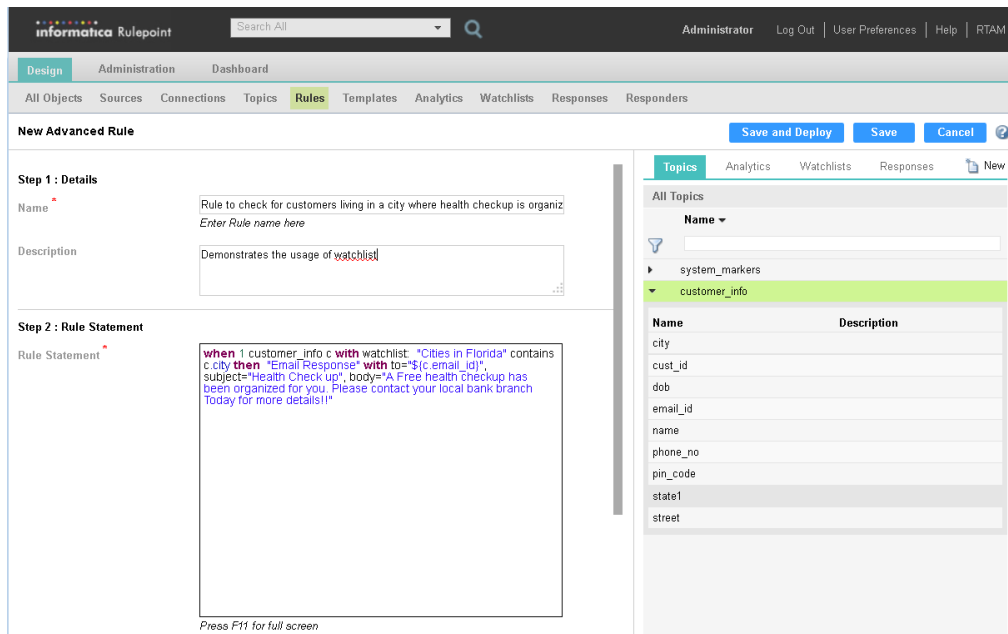
Step 5. Create an Advanced Rule Using the Watchlist

Create a rule that uses a watchlist to check for customers living in a particular city and send an alert about a free health checkup.

1. On the **Design** tab, select **Rules**.
2. In the right pane, click **Actions > Advanced Rule**.
3. Configure the following properties:

Field	Value
Name	Rule to check for customers living in a city where health checkup is organized
Description	Demonstrates the usage of watchlist
Rule Statement	when 1 customer_info c with watchlist: "Cities in Florida" contains c.city then "Email Response" with to="{c.email_id}", subject="Health Check up", body="A Free health checkup has been organized for you. Please contact your local bank branch Today for more details!!"

The following figure shows the advanced rule configurations in the **Rules** view of the **Design** tab:



4. Click **Save**.

Step 6. Deploy the Rule, Source, and Responder

Deploy the configured objects into the run-time environment for rule processing to begin.

1. In the navigator, click **Actions > Rules, Sources & Responders > Deploy**.
2. Select the rule **Rule to check for customers living in a city where health checkup is organized**, and click **Next**.
3. Select the source **Customer Accounts Source**, and click **Next**.
4. Select **Email Responder**, and click **Deploy**.
5. Click the **Dashboard** tab, and verify if the rule is activated and the counts are updated accordingly.

Step 7. View the Email Response

Verify if you receive an email response.

1. Go to Windows Live Mail.
2. Verify if you have received an email.

CHAPTER 7

Alerting the Bank Manager about Customers with a Particular Account Type in a City

This chapter includes the following topics:

- [Alerting the Bank Manager about Customers with a Particular Account Type in a City Overview, 55](#)
- [Step 1. Create an Analytic , 57](#)
- [Step 2. Create an Advanced Rule , 58](#)
- [Step 3. Deploy the Rule, Source, and Responder, 58](#)
- [Step 4. View the RTAM Alert, 58](#)

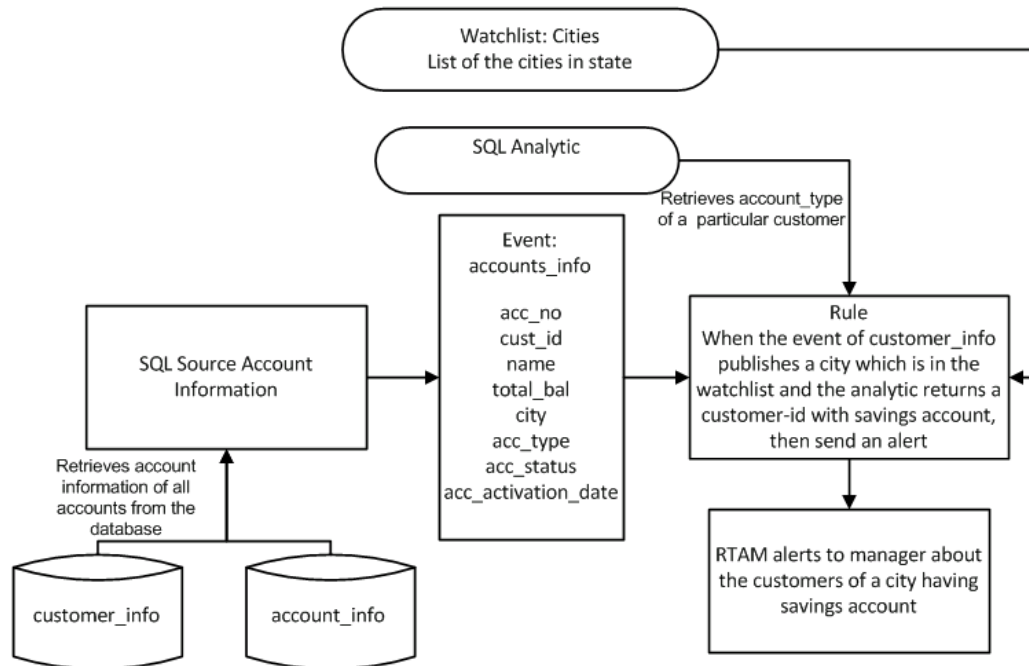
Alerting the Bank Manager about Customers with a Particular Account Type in a City Overview

Send an alert to the bank manager about customers who have a savings bank account in a particular city.

Lesson Concepts

In this lesson, you learn to create an advanced rule that uses a watchlist and an SQL analytic. The watchlist lists the cities in Florida. When you run SQL commands and queries as an analytic, you provide additional information in the data during rule processing that is otherwise not available in the event data. The SQL analytic retrieves the savings account information of customers in Florida. You also learn to view the dashboard to verify the configured object deployment and the activations. The lesson guides you in viewing the configured RTAM alert, customer alert, in the Real-Time Alert Manager.

The following figure represents the data model to retrieve information of customers who have a savings bank account:



The model representation uses an SQL source that connects to the database and retrieves account and customer information. The source publishes events on the *accounts_info* topic. The advanced rule uses a watchlist and an analytic to retrieve account information of customers living in Florida. When an event matches the rule condition, the manager receives an RTAM alert about customers who have saving bank accounts in Florida.

Lesson Objectives

In this lesson, you learn to perform the following tasks:

- Perform prerequisite tasks before you create the rule.
- Create an SQL analytic.
- Create a watchlist.
- Create an advanced rule to retrieve details of customers with savings account.
- Deploy the rule, source, and responder.
- View the RTAM alert.

Lesson Prerequisites

Before you start this lesson, you must complete the following prerequisites:

1. Create a topic, *customer_accounts*.
2. Create an SQL connection.
3. Create a Customer Accounts Source.
4. Create an RTAM responder.
5. Create an RTAM response.
6. Create a watchlist named *Cities in Florida*.

Step 1. Create an Analytic

Use analytics to analyze data within a system. Analytics use a data processing function. You can reference an analytic in rules that determines the activations.

1. On the **Design** tab, select **Analytics**.
2. In the right pane, click **Actions > New**.
3. Configure the following properties:

Field	Value
Name	cust_get_account_type
Description	Retrieves account type of the customer
Type	SQL Analytic
Connections	SQL Connection
SQL Query	select acc_type from account_info where cust_id=?

The following figure shows the analytic configurations in the **Analytic** view of the **Design** tab:

The screenshot shows the 'New Analytic' configuration form in the Informatica Rulepoint interface. The form is titled 'New Analytic' and has 'Save' and 'Cancel' buttons. It is divided into two main sections: 'Details' and 'Configurations'.
Details Section:
- **Name:** A text input field containing 'cust_get_account_type'. Below it is a placeholder: 'Enter Analytic name here'.
- **Description:** A text area containing 'Retrieves account type of the customer'. Below it is a placeholder: 'Enter Analytic description here'.
- **Type:** A dropdown menu with 'SQL Analytic' selected. Below it is a note: 'Please select one Analytic type from the list'.
- **Connection:** A dropdown menu with 'SQL Connection' selected. Below it is a note: 'Please select one Connection type from the list'.
Configurations Section:
- **SQL Query:** A text area containing 'select acc_type from account_info where cust_id=?'. Below it is a note: 'The SQL query to execute against the target database. Use ? as substitutions when using the SQL Service as an analytic.'
- **Condition Evaluation Required:** A dropdown menu with 'True' selected. Below it is a note: '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. The default value is True.'

4. Click **Save**.

Step 2. Create an Advanced Rule

Create a rule that uses a watchlist and an SQL analytic to retrieve customer details of a specific account type and send an RTAM response to specific customers in Florida.

1. On the **Design** tab, select **Rules**.
2. In the right pane, click **Actions > New > New Advanced Rule**.
3. Under **Details**, configure the following properties:

Field	Value
Name	Rule to demonstrate usage of Watchlist and SQL Analytic
Description	Demonstrate the usage of watchlist and SQL Analytic

4. Under **Rule Statement**, type the following rule:

```
when 1 customer_accounts c with watchlist: "Cities in Florida" contains c.city and
cust_get_account_type ( c.cust_id) = 'Savings' then "RTAM Response" with body="Customer $
{c.name} with ID ${c.cust_id} has account number ${c.acc_no} which is an account of type
${c.acc_type} ", subject = "Customer Alert" , to = "Administrator", priority=4,
channels="Florida Customers"
```

5. Click **Save and Deploy**.
6. Click **OK** when you are prompted to deploy.
7. Navigate to the dashboard and verify that the rule is deployed.

Step 3. Deploy the Rule, Source, and Responder

Deploy the configured objects in the run-time environment. After deployment, you can see the rule activations on the dashboard.

1. On the navigator, click **Actions > Rules, Sources & Responders > Deploy**.
2. Click **Next** in the wizard and in the Advanced Rules page.
3. Select the source **Customer Accounts Source**, and click **Next**.
4. Select the responder **RTAM Responder**, and click **Deploy**.
5. Navigate to the dashboard and verify that the rule is activated and the number of activations and aggregate count are updated.

Step 4. View the RTAM Alert

You can view the generated alert on the RTAM.

1. Log in to RTAM at: `http://host:port/RTAM`
2. Provide the username and password.

3. Under **Florida Customers**, verify if the alert **Customers Alert** displays.

CHAPTER 8

Alerting Customer Cash Withdrawal Details

This chapter includes the following topics:

- [Alerting Customer Cash Withdrawal Details Overview, 60](#)
- [Step 1. Create a Topic, 61](#)
- [Step 2. Create an SQL Source, 62](#)
- [Step 3. Create a Schedule, 63](#)
- [Step 4. Create an Advanced Rule, 63](#)
- [Step 5. Deploy the Rule, Source, and Responder, 63](#)
- [Step 6. View the RTAM Alert, 64](#)

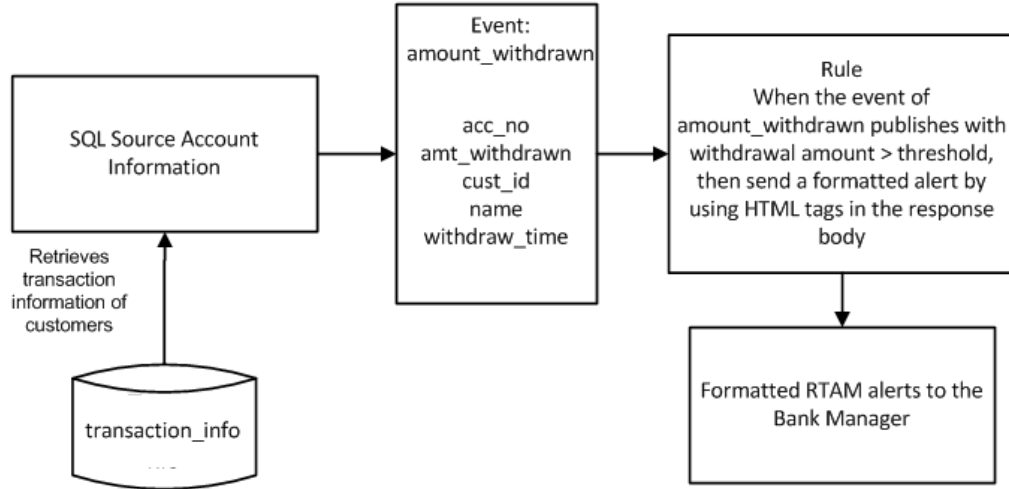
Alerting Customer Cash Withdrawal Details Overview

Check for amount withdrawn by a customer and send an alert to the manager if the amount withdrawn exceeds a defined limit.

Lesson Concepts

In this tutorial, you learn to create a wizard rule with formatted responses to check for customer cash withdrawal alerts to a manager when the amount withdrawn exceeds a defined limit. You learn to create a topic for amount withdrawal details and an SQL source to read data from the source. You also learn to configure a wizard rule, deploy the objects, and view RTAM alerts.

The following figure represents the model structure for sending an RTAM alert for customer cash withdrawals exceeding a threshold value:



The model representation uses an SQL source that connects to the database and retrieves transaction information. The source publishes events on the amount_withdrawn topic. The advanced rule checks for cash withdrawn that exceeds the specified limit and states that when an event matches the rule condition, send alerts to the bank manager.

Lesson Objectives

In this lesson, you will learn to perform the following tasks:

- Create a topic, amount_withdrawn.
- Create an SQL source to connect to the source.
- Create a schedule.
- Create an advanced rule.
- Deploy the rule, source, and responder.
- View the RTAM alert.

Lesson Prerequisites

Before you start this lesson, you must complete the following prerequisites:

- Create an SQL connection for the SQL source.
- Create an RTAM responder.
- Create an RTAM response.

Step 1. Create a Topic

Create a topic that contains event properties related with customer cash withdrawal details, account number and related details.

1. Select the project **Banking** under which you want to create all the artifacts related to banking.
2. On the **Design** tab, select **Topics**.

- In the right pane, click **Actions > New**.
- Under **Details**, configure the following properties:

Field	Value
Name	amount_withdrawn
Description	This topic contains event properties associated with amount withdrawn.
Expires in	600000
Responder access	ALL_PROPERTIES

- Under **Properties**, click **Bulk Create**.
- Configure the following properties in the editor, and click **Add**:
 - acc_no
 - amt_withdrawn
 - cust_id
 - name
 - withdraw_time
- Click **Save**.

Step 2. Create an SQL Source

Create an SQL source to retrieve details of the amount withdrawn by the customer.

- On the **Design** tab, select **Sources**.
- In the right pane, click **Actions > New**.
- Under **Details**, configure the following properties:

Field	Value
Name	Source to monitor amount withdrawn by the customer
Description	Monitors the amount withdrawn by the customer
Type	SQL Source
Connection	SQL Connection

- Under **Topic Information**, select the topic name, amount_withdrawn, and click **OK**.
- Under **Configuration**, type `select * from transaction_info` under the **SQL** field.
- Click **Save**.

Step 3. Create a Schedule

Create a schedule for the SQL source.

1. Under **Source to monitor amount withdrawn by the customer > Schedules**, click **Create New Schedule**.
2. Configure the following properties:

Field	Value
Schedule	Dynamic
Repeat Interval	120000 (In milliseconds)

3. Click **Add Schedule**.

Step 4. Create an Advanced Rule

Create an advanced rule to check for amount withdrawn by customer and send an alert to the customer and the manager if the amount withdrawn exceeds the limit.

1. On the **Design** tab, select **Rules**.
2. In the right pane, click **Actions > New > New Advanced Rule**.
3. Under **Details**, configure the following properties:

Field	Value
Name	Rule to check for amount withdrawn that exceeds the limit
Description	Checks for amount withdrawn that exceeds the limit

4. Under **Rule Statement**, type the following rule:

```
when amount_withdrawn t0 with t0.amt_withdrawn>25000 then "RTAM Response" with
subject="Transaction Alert", body="Amount withdrawn by the customer with <B> ID $
{t0.cust_id} </b> is greater than 25000. <br> <br> <b>Contact the customer immediately!
</b></br></br>", priority=5
```

5. Click **Save**.

Step 5. Deploy the Rule, Source, and Responder

1. On the navigator, click **Actions > Rules, Sources & Responders > Deploy**.
2. Under **Wizard and Advanced Rules**, select the rule **Rule to check for amount withdrawn that exceeds the limit**, and click **Next**.
3. Under **Source Instances**, select the source **Source to monitor amount withdrawn by the customer**, and click **Next**.

4. Under **Responder Instances**, select the responder **RTAM Responder**, and click **Deploy**.
5. Navigate to the dashboard and verify if the source, rule, and responders are deployed. You can see the source in the Default Source Controller.

Step 6. View the RTAM Alert

View the alert on the RTAM. The RTAM response is formatted to display the text as bold and in newline. You can add html tags in the rule response to improve the response further.

1. Log in to RTAM at: `http://host:port/RTAM`
2. Provide the username and password.
3. Under **Default Channel**, verify if the alert **Customers Alert Transaction Alert** displays.

CHAPTER 9

Alerting Stock Prices to DMAT Account Holders

This chapter includes the following topics:

- [Alerting Stock Prices to DMAT Account Holders Overview, 65](#)
- [Step 1. Create an Advanced Rule, 67](#)
- [Step 2. Deploy the Rule, Source, and Responder, 67](#)
- [Step 3. View the Email Alert, 68](#)

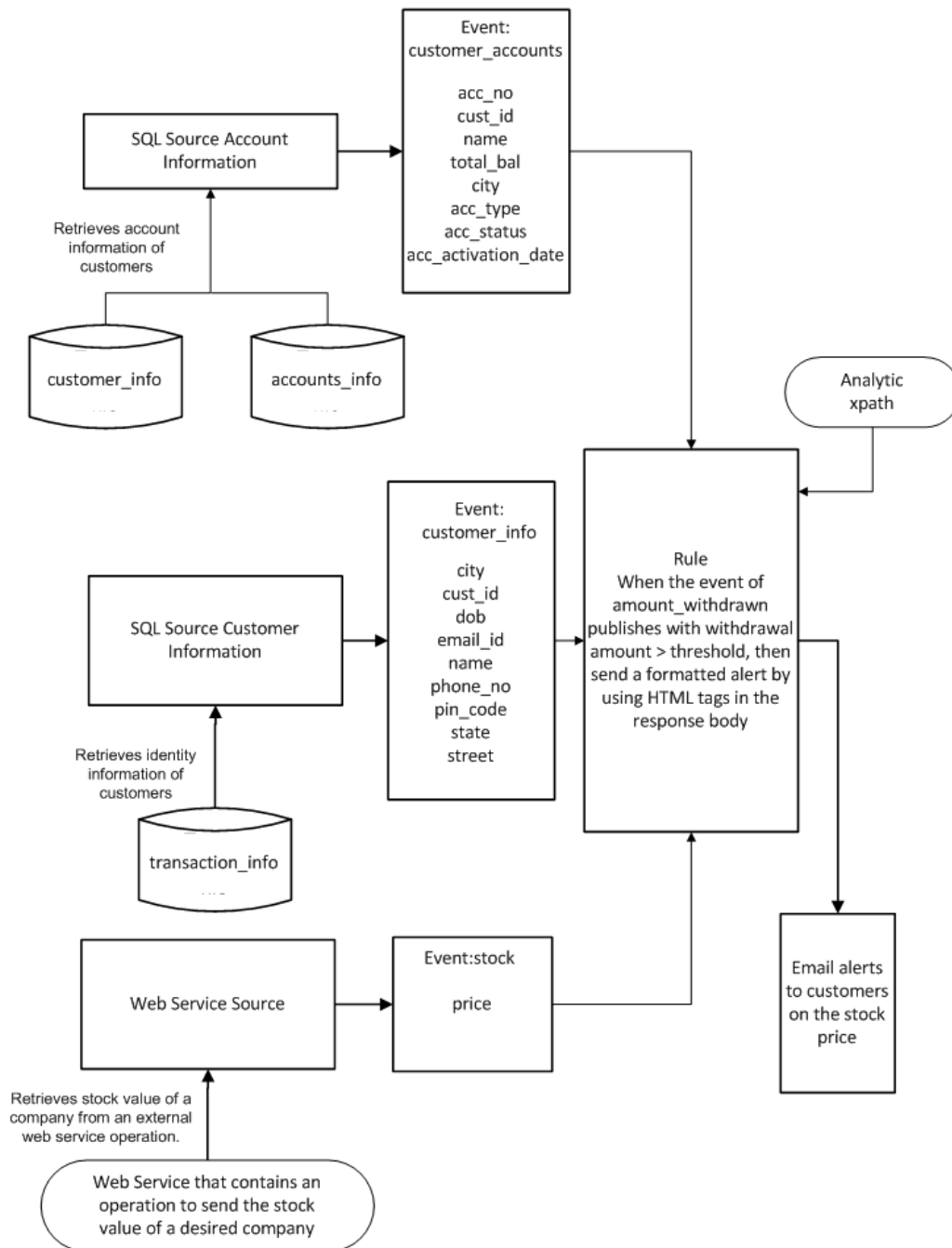
Alerting Stock Prices to DMAT Account Holders Overview

Send stock price alerts to DMAT account holders.

Lesson Concepts

In this lesson, you will learn to perform tasks necessary for sending stock alerts through email. The tasks involve using multiple event sources while creating an advanced rule. This event sources you use here are stock, customer_accounts, and customer_info. The rule condition is to send an alert to preferred customers who have a DMAT account when a specific stock price reaches a desired price.

The following figure represents the data model for sending an email alert to the DMAT account holder when a specific stock price of a company reaches a desired amount:



The model representation uses an SQL source that connects to the database and retrieves transaction, customer, and accounts information. The SQL source publishes matching events on corresponding customer_accounts and customer_info topics. The web service source retrieves the stock value of the specified company and publishes events on the stock topic. The rule uses an xpath analytic, which uses an XPath expression "GetQuoteResult" to find information for a specific stock attribute whose value is greater than 35 and returns the list of objects that match. When the events match the configured condition, the customer receives email alerts about the stock prices.

Lesson Objectives

In this lesson, you learn to perform the following tasks:

- Create an advanced rule to alert a DMAT account holder by email notifying the desired stock hike.
- Deploy the rule, source, and responder.
- View the email alert.

Lesson Prerequisites

Before you create the rule, create the following objects:

- Topics: stock, customer_accounts, and customer_info
- Source: Stock Source, Customer Accounts Source, and Customer Information Source
- Schedule: Dynamic schedule for the sources
- Responder: Email responder
- Response: Email response

Step 1. Create an Advanced Rule

Create a rule to send an alert to preferred customers who have DMAT account when a specific stock price reaches desired amount.

1. On the **Design** tab, select **Rules**.
2. In the right pane, click **Actions > Advanced Rule**.
3. Configure the following properties:

Field	Value
Name	Rule to check for stock price and send an alert to Demat account holder
Description	Demonstrate the using of multiple topics in a rule

4. Under **Rule Statement**, type the following rule:

```
when 1 stock s , 1 customer_accounts a, 1 customer_info c with a.cust_id = c.cust_id and  
a.acc_type='Demat' and xpath(s.getquoteresult, "/StockQuotes/Stock/Open") as price>35  
then "Email Response" with to="{c.email_id}", subject="Stock alert", body="The stock  
price of the company INFA is now more than $35 " remove all(s,a,c)
```

5. Click **Save**.

Step 2. Deploy the Rule, Source, and Responder

Deploy the configured objects and then view the activations in the dashboard

1. In the navigator, click **Actions > Rules, Sources & Responders > Deploy**.
2. Select the rule **Rule to check for stock price and send an alert to Demat account holder**, and click **Next**.

3. Select the source **Stock Source**, **Customer Accounts Source**, **Customer Information Source**, and click **Next**.
4. Select the responder **Email Responder**, and click **Deploy**.
5. Click the **Dashboard** tab, and verify if you can view the sources and rules.

Select the event processor in which you deployed the rule, and view the activated rule and counts in the **Rule** view in the right pane.

Step 3. View the Email Alert

1. Open your email application.
2. Verify if you have received an email.

CHAPTER 10

Checking Customer Cash Withdrawal Details for a Specified Duration

This chapter includes the following topics:

- [Checking Customer Cash Withdrawal Details for a Specified Duration Overview, 69](#)
- [Step 1. Create an Advanced Rule, 70](#)
- [Step 2. Deploy the Rule, Source, and Responder, 71](#)
- [Step 3. View RTAM Alert, 71](#)

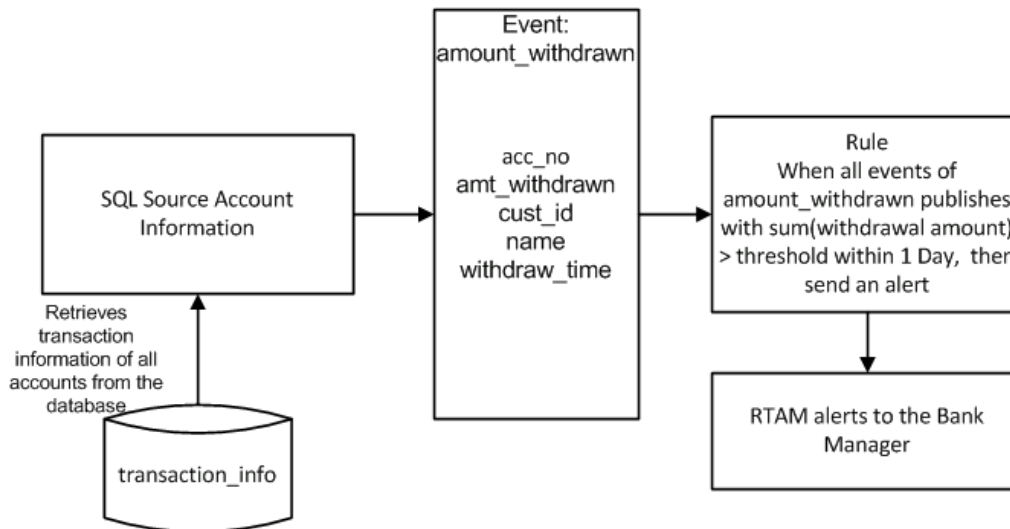
Checking Customer Cash Withdrawal Details for a Specified Duration Overview

Create an advanced rule to check for cash withdrawn by customers in a day is greater than the threshold and send an alert to the bank manager.

Lesson Concepts

In this lesson, you learn to create an advanced rule to monitor if the amount withdrawn by customers within one day exceeds the limits. You use a source that monitors cash withdrawal details of customers. You configure an RTAM responder and response for alerting.

The following figure represents the model structure for sending an RTAM alert for customer cash withdrawals that exceed a threshold value within a day:



The model representation uses an SQL source that connects to the database, retrieves transaction information, and publishes events on the configured amount_withdrawn topic. When the rule evaluates all events for amounts withdrawn in a day exceeds the specified threshold, the bank manager receives an alert.

Lesson Objectives

In this lesson, you learn to perform the following tasks:

- Create an advanced rule.
- Deploy the rule, source, and responder.
- View the RTAM alert.

Lesson Prerequisites

Before you create the rule, create the following objects:

- Topic: amount_withdrawn
- Connection: SQL connection for the source
- Source: SQL source to monitor amount withdrawn by the customer
- Schedule: Dynamic schedule for the source
- Responder: RTAM responder
- Response: RTAM response

Step 1. Create an Advanced Rule

Create an advanced rule to check for cash withdrawn by customers in one day is greater than the threshold and send an alert to the bank manager.

1. On the **Design** tab, select **Rules**.
2. In the right pane, click **Actions > New > New Advanced Rule**.

- Under **Details**, configure the following properties:

Field	Value
Name	Rule to monitor if the amount withdrawn by customers within 1 day exceeds the limits
Description	Monitors the total amount withdrawn by customers

- Under **Rule Statement**, type the following rule:

```
when amount_withdrawn a with sum(a.amt_withdrawn)as sum_amt > 100000 slide within 1 days
then "RTAM Response" with subject="Amount withdrawn exceeds the day's limit", body="The
total amount withdrawn by customers today exceeds the limit of 100000", channels="Amount
Withdrawn in 1 Day", priority=5
```

- Click **Save**.

Step 2. Deploy the Rule, Source, and Responder

Deploy the configured objects into the corresponding services in the run-time environment.

- On the navigator, click **Actions > Rules, Sources & Responders > Deploy**.
- Under **Wizard and Advanced Rules**, select the rule **Rule to monitor if the amount withdrawn by customers within 1 day exceeds the limits**, and click **Next**.
- Under **Source Instances**, select the source **Source to monitor amount withdrawn by the customer**, and click **Next**.
- Under **Responder Instances**, select the responder **RTAM Responder**, and click **Deploy**.
- Navigate to the dashboard and verify if you can view the deployed source, rule, and responder. You can view the source in the Default Source Controller. The dashboard also displays the objects associated with the source such as the topics and their details. Verify if the rule is activated.

Step 3. View RTAM Alert

View the alert on the RTAM.

- Log in to RTAM at the following URL:
`http://host:port/RTAM`
- Provide the user name and password.
- Under **Suspended accounts**, verify if the alert **Invalid Transactions** displays.

CHAPTER 11

Alerting Consequent Credit Card Transaction Rejections

This chapter includes the following topics:

- [Alerting Consequent Credit Card Transaction Rejections Overview, 72](#)
- [Step 1. Create an SQL Responder, 74](#)
- [Step 2. Create an Advanced Rule, 74](#)
- [Step 3. Deploy the Rule, Source, and Responder, 75](#)
- [Step 4. View JMS Messages, 75](#)
- [Step 5. View the RTAM Alert, 76](#)

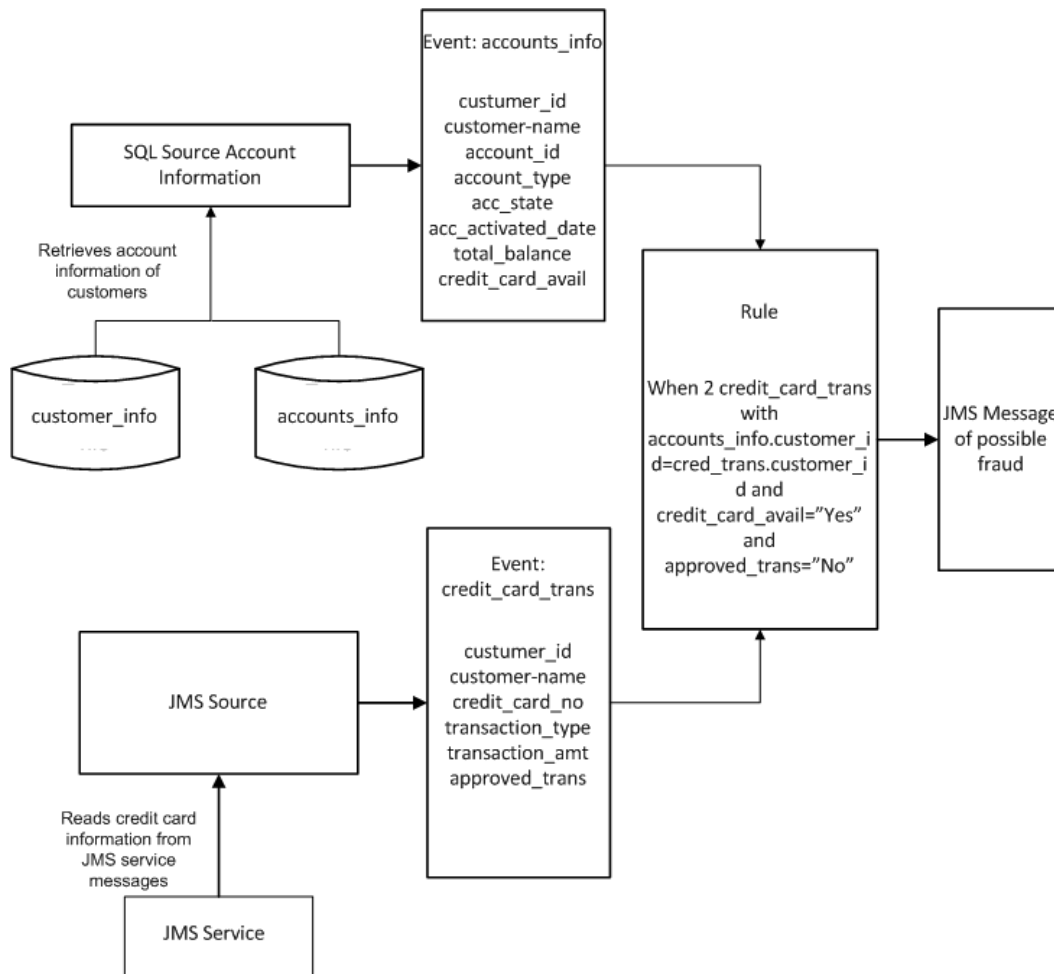
Alerting Consequent Credit Card Transaction Rejections Overview

Configure an alert to the bank manager when consequent credit card transaction events are rejected.

Lesson Concepts

In this lesson, you learn to create an advanced rule to send an alert to the manager when there are two credit card transactions with approval as "NO." You configure multiple responders, such as SQL responder and RTAM responder in the advanced rule. The SQL responder service responds to events by connecting to a database and executing SQL commands. The Real-Time Alert Manager responder service responds to events by sending alerts to a Real-Time Alert Manager server. You can simultaneously view the SQL and RTAM response.

The following figure represents the model structure for alerting managers when subsequent credit card transactions are not approved:



The model representation uses a JMS source to read messages from the JMS service and publishes events on the credit_card_trans topic. The SQL source connects to the database and retrieves customer and accounts information and publishes events on the accounts_info topic. When the rule evaluates all matching events of credit card transactions that are not valid, the account is suspended, and you can simultaneously view an RTAM alert and a JMS message of fraudulent transactions.

Lesson Objectives

In this lesson, you learn how to perform the following tasks:

- Create an SQL responder.
- Create an advanced rule.
- Deploy the rule, source, and responder.
- Send JMS messages.
- View RTAM alert.
- View SQL response on the dashboard.

Lesson Prerequisites

Before you start this lesson, you must complete the following prerequisites:

- Create a topic, customer_info and accounts_info.
- Create an SQL connection.

- Create a JMS source to monitor credit card transactions.
- Create an SQL source to retrieve account and customer information.
- Create a schedule for the sources.
- Create an RTAM responder.
- Create an RTAM response and SQL response.

Step 1. Create an SQL Responder

Create an SQL responder to send the response. Configure an SQL connection to connect to the target.

1. On the **Design** tab, select **Responders**.
2. In the right pane, click **Actions > New**.
3. Under **Responder**, configure the following properties:

Field	Value
Name	SQL Responder
Description	Sends SQL Response
Type	SQL Responder
Connection	SQL Connection

4. Under **Configuration**, configure the following properties:

Field	Value
SQL	update account_info set acc_status='Suspended' where acc_no=<<account_num>>
Parameters	account_num=0

5. Click **Save and Deploy**.
As the responder does not have a response associated with it, the response creation page appears.
6. Provide the name as SQL Response, and click **Save**.
A message appears that prompts you to confirm the deployment.
7. Click **OK**.

Step 2. Create an Advanced Rule

Create an advanced rule to send an alert to the manager when there are two credit card transaction events with approval as "NO." The advanced rule uses multiple responders.

1. On the **Design** tab, select **Rules**.

- In the right pane, click **Actions > New > New Advanced Rule**.
- Under **Details**, configure the following properties:

Field	Value
Name	Rule to check for credit card transaction that are rejected consecutively.
Description	Checks for credit card transactions that are rejected consecutively.

- Under **Rule Statement**, type the following rule:

```
when 2 credit_card_transaction cc with cc.jmstext contains 'NO' and match(cc.jmstext) and
current(cc.jmstext) as newtext !=0 and xpath(newtext,"//event/property[name='acc_no']/
value") as txt !=0 then "RTAM Response" with body="The account with number <B>${txt} </
B>has been suspended because of multiple invalid transactions", subject="Invalid
transactions",channels="Suspended accounts" and "sql response" with sql="update
account_info set acc_status='Suspended' where acc_no=<<acc_no>>",params="acc_no=${txt}"
```

- Click **Save**.

Step 3. Deploy the Rule, Source, and Responder

Deploy the configured objects into the services in the run-time environment.

- On the navigator, click **Actions > Rules, Sources & Responders > Deploy**.
- Under **Wizard and Advanced Rules**, select the rule **Rule to check for credit card transaction that are rejected consecutively**, and click **Next**.
- Under **Source Instances**, select the source **JMS Source to monitor Credit card Transactions**, and click **Next**.
- Under **Responder Instances**, select the responders **SQL Responder** and **RTAM Responder**, and click **Deploy**.
- Navigate to the dashboard and verify if you can view the deployed source, rule, and responder.
Verify that **JMS Source to monitor Credit Card Transactions** displays when you select **Default Source Controller**. The dashboard also displays the objects associated with the source such as the topics and their details. Check if you can view the activated rule.

Step 4. View JMS Messages

When an event is activated, you can view the alerts in ActiveMQ and the dashboard.

- Go to ActiveMQ. Open a browser and type:
`http://server name:port/admin/queues.jsp`
- Click **Send To** under the queue, `credit_card`.

3. Under **Message Body** of the queue, enter the following text, and click **Send**.

```
<event topic="CC_9678535853279753">
  <property>
    <name>Credit_Card_No</name>
    <value>9678535853279753</value>
  </property>
  <property>
    <name>Customer_Name</name>
    <value>John Smith</value>
  </property>
  <property>
    <name>Transaction_Type</name>
    <value>sale</value>
  </property>
  <property>
    <name>Merchant_Account</name>
    <value>SPINT65280378312</value>
  </property>
  <property>
    <name>Amount</name>
    <value>400</value>
  </property>
  <property>
    <name>Approval</name>
    <value>NO</value>
  </property>
  <property>
    <name>acc_no</name>
    <value>2</value>
  </property>
</event>
```

4. Repeat Step 2 to make sure that the JMS message is sent twice.
Note: As the rule evaluates two events with transactions that are not approved and particular to a specific account number, you need to publish the same data two times in the ActiveMQ.
5. Navigate to the RulePoint dashboard and verify if you can view the deployed source, rule, and responders.

Step 5. View the RTAM Alert

View the alert on the RTAM.

1. Log in to RTAM at the following URL:
`http://host:port/RTAM`
2. Provide the username and password.
3. Under **Amount Withdrawn in 1 Day**, verify if the alert **Amount withdrawn exceeds the day's limit** displays.

CHAPTER 12

Monitoring Balance Threshold of Customers

This chapter includes the following topics:

- [Monitoring Balance Threshold of Customers Overview, 77](#)
- [Step 1. Create a Template, 79](#)
- [Step 2. Create a Template Rule, 81](#)
- [Step 3. Create a Deployment Policy, 82](#)
- [Step 4. Deploy the Rule, Source, and Responder, 82](#)
- [Step 5. View the RTAM Alert, 83](#)
- [Step 6. Upgrade a Template, 83](#)
- [Step 7. Upgrade a Template to Include an Additional Parameter, 85](#)
- [Step 8. Update the Rule, 86](#)
- [Step 9. View the RTAM Alert, 87](#)
- [Step 10. Revert the Template History, 87](#)
- [Step 11. Create a Template from the Wizard Rule, 88](#)
- [Step 12. Create a Template Rule, 90](#)
- [Step 13. Deploy the Rule, Source, and Responder, 91](#)
- [Step 14. View the RTAM Alert, 91](#)

Monitoring Balance Threshold of Customers Overview

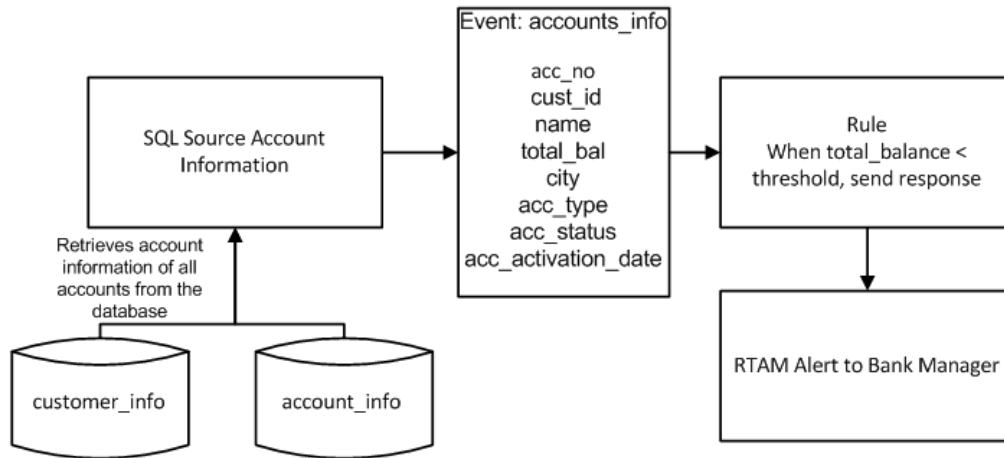
In this lesson, you perform tasks to monitor the balance threshold of customers.

Lesson Concepts

In this lesson, you will learn to create different types of templates. Templates help you to easily create new rules. To make templates easy to use, you can add specific user assistance where necessary. The lessons guide you to create a template to check for customers with a given balance. You learn how to create template rules from a configured template, create a deployment policy, deploy the objects, and view the corresponding rule activations and alerts when the event meets the required condition.

You also learn how to upgrade a template when you want to modify the parameters in that template. You learn to add a watchlist to the template so that you can monitor multiple parameters at a time and correspondingly update the rule. The lesson provides instructions to edit template rules and convert a wizard rule to a template.

The following figure represents the model structure to monitor the balance threshold:



The model representation uses an SQL source that retrieves customer and account information from the database and publishes events on the accounts_info topic. The advanced rule checks for events where the total balance is less than the specified threshold, and sends an RTAM alert to the bank manager

Lesson Objectives

In this lesson, you learn to perform the following tasks:

- Create a template to check for customers for a specified balance.
- Create a template rule to check whether a customer has the specified balance.
- Create a deployment policy.
- Deploy the rule, source, and responder, and correspondingly view the RTAM alert.
- Upgrade a template and then the rule, and correspondingly view the RTAM alert.
- Revert the template history.
- Create a template from the wizard rule, create a rule, and view the RTAM alert.
- Create a template rule.

Lesson Prerequisites

Before you start this lesson, you must complete the following prerequisites:

- Create a topic, accounts_info and customer_info.
- Create an SQL connection.
- Create an SQL source, Customer Accounts Source, to retrieve account information.
- Create a dynamic schedule for the source.
- Create an RTAM responder.
- Create an RTAM response to alert the bank manager.
- Create a watchlist that consists of the cities of Florida.

Step 1. Create a Template

Create a template to monitor the balance threshold of a specific customer.

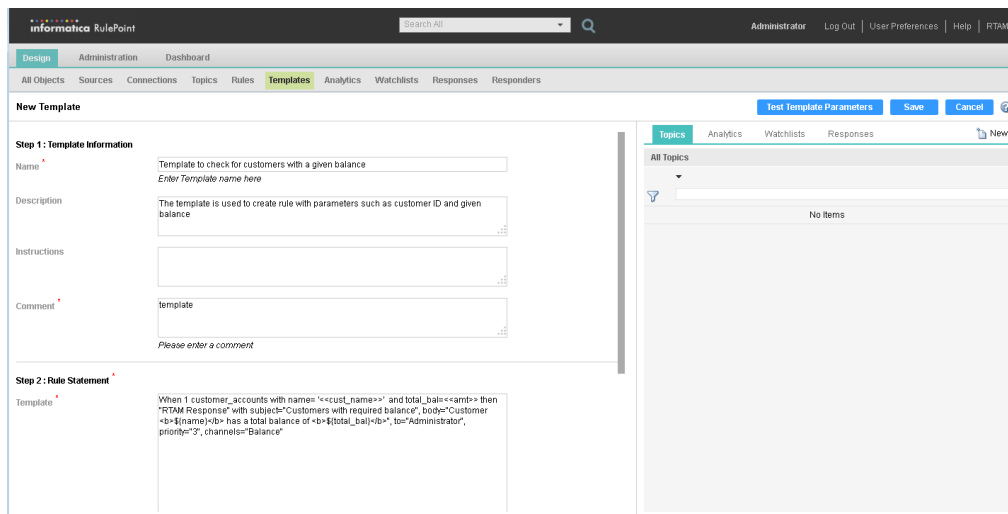
1. On the **Design** tab, select **All Objects**.
2. In the right pane, click **Actions > New > Template**.
3. In the **Template Information** section, configure the following properties:

Field	Value
Name	Template to check for customers with a given balance.
Description	The template is used to create rule with parameters such as customer ID and given balance.
Comment	template

4. In the **Rule Statement** section, type the following rule:

```
When 1 customer_accounts with name= '<<cust_name>>' and total_bal=<<amt>> then
"RTAM Response" with subject="Customers with required balance", body="Customer <b>${
{name}</b> has a total balance of <b>${total_bal}</b>", to="Administrator",
priority="3", channels="Balance"
```

The following figure displays the required configurations for creating a new template in the **Design > New Templates** view:



5. Click **Update Template Parameters**.
6. In the **Template Parameters** section, configure the following properties for **cust_name**:

Field	Value
Field Name	cust_name
Field Type	Single Line Text
Max Length	40

Field	Value
Test Value	John Smith

The following figure displays the template parameters for cust_name in the **Design > New Templates** view:

Step 3 : Template Parameters

cust_name

Field Name
Enter parameter name here

Description

Field Type

Max Length

Test value

Set as default value

Custom Validation

Instructions for User

amt

7. In the **amt** section, configure the following properties:

Field	Value
Field Name	amt
Field Type	Integer Number
Min Value	10000
Max Value	50000
Test Value	15000

The following figure displays the template parameters for amt in the **Design > New Templates** view:

Step 3 : Template Parameters

cust_name

amt

Field Name * amt
Enter parameter name here

Description

Field Type * Integer Number

Min Value 10000
Enter minimum range for test parameter value here

Max Value 50000
Enter maximum range for test parameter value here

Test value * 15000

Set as default value

8. Click **Test Template Parameters**.
9. When a message appears indicating that the template rule validation is successful, click **OK**.
10. Click **Save**.

Step 2. Create a Template Rule

Create a template rule to monitor the balance threshold of a specific customer. Specify the customer name and the threshold amount in the template rule.

1. On the **Design** tab, select **Rules**.
2. In the right pane, click **Actions > New > New Template Rule**.
 - a. Select **Template to check for customers with a given balance_rule**.
 - b. Click **Next**.
3. Configure the following properties:

Field	Value
Name	Rule to check whether a customer has the given balance .
Description	Template Rule

4. Under **Parameters**, configure the following properties:

Field	Value
cust_name	John Smith
amt	15000

5. Click **Test Template Rule**.
6. When a success message appears indicating that the template rule validation is successful, click **OK**.

7. Click **Save**.

The following figure shows the New Template Rule configurations:

The screenshot displays the 'New Template Rule - Step 2 of 2' configuration interface. At the top, there is a navigation bar with 'Design', 'Administration', and 'Dashboard' tabs. Below this, a breadcrumb trail shows 'All Objects > Sources > Connections > Topics > Rules > Templates > Analytics > Watchlists > Responses > Responders'. The main content area is titled 'New Template Rule - Step 2 of 2' and includes buttons for '< Back', 'Save and Deploy', 'Test Template Rule', 'Save', and 'Cancel'. The 'Details' tab is active, showing a form with the following fields:

- Name:** Rule to check whether a customer has the given balance (with a subtext: 'Enter the Template rule name here')
- Description:** Template Rule (with a subtext: 'Enter the Template rule description here')
- Step 2: Parameters:**
 - cust_name:** John Smith
 - amt:** 15000

Step 3. Create a Deployment Policy

Create a deployment policy for the rule.

1. On the **Design** tab, click **Templates**.
2. Select the rule, and then select **Create Deployment Policy** from the menu on the right side.
3. Click **Save**.

Step 4. Deploy the Rule, Source, and Responder

Deploy the objects to the services in the run-time environment.

1. On the navigator, click **Actions > Rules, Sources & Responders > Deploy**.
2. Under **Wizard and Advanced Rules**, select the rule **Template to check for customers with a given balance_rule**, and click **Next**.
3. Under **Source Instances**, select the source **Customer Accounts Source**, and click **Next**.
4. Under **Responder Instances**, select the responder **RTAM Responder**, and click **Deploy**.
5. Navigate to the dashboard and verify if you can view the deployed source, rule, and responder. View the activated rule, and the count of the number of activations, the aggregate count, and the related object counts.

Step 5. View the RTAM Alert

View the alert on RTAM.

1. Log in to RTAM using the following URL:
`http://host:port/RTAM`
2. Provide the user name and password.
3. Under the channel **Balance**, verify if the alert **Customers with required balance** displays.

Step 6. Upgrade a Template

When you want to edit a parameter value, upgrade the template so that associated template rules also upgrade to reflect the modified template.

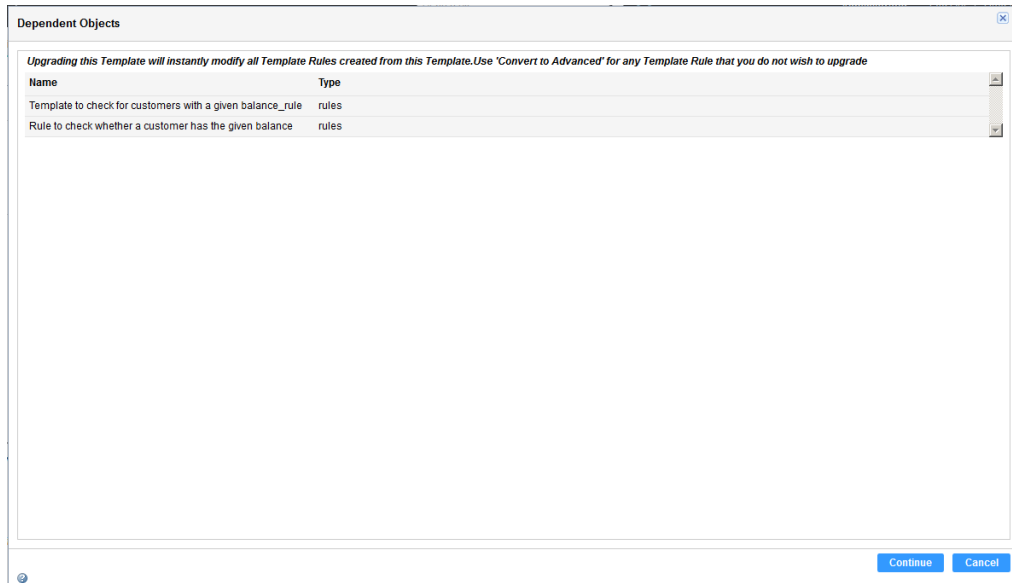
1. On the **Design** tab, click **Templates**.
2. Select the template **Template to check for customers with a given balance**, and click **Edit** from the menu on the right side.
3. Under **Edit Template**, type *Updating the min value of parameter to 5000* in the comment.

The following figure shows the configurations for editing an existing template:

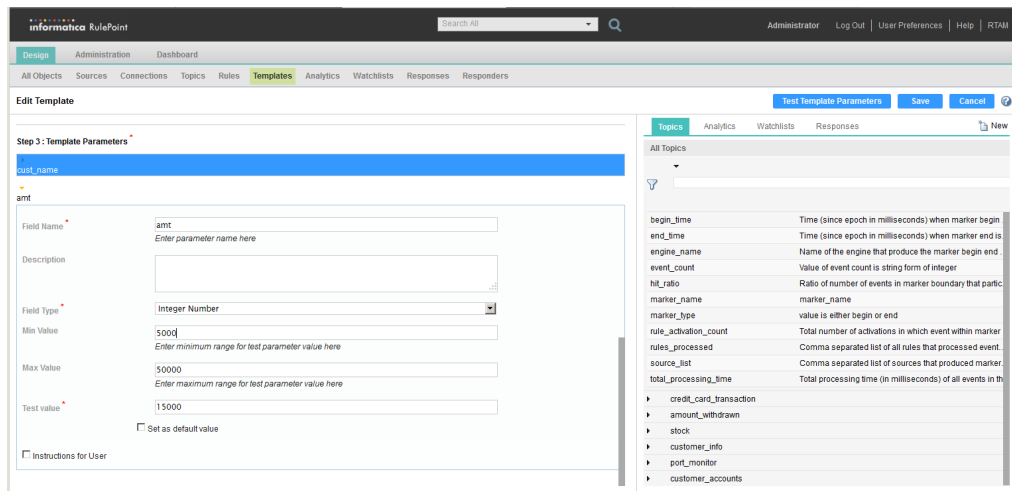
The screenshot displays the 'Edit Template' window in Informatica RulePoint. The top navigation bar includes 'Design', 'Administration', and 'Dashboard'. The main menu has 'All Objects', 'Sources', 'Connections', 'Topics', 'Rules', 'Templates', 'Analytics', 'Watchlists', 'Responses', and 'Responders'. The 'Edit Template' section has a 'Comment' field containing 'Updating the min value of parameter to 5000'. Below it is a 'Step 2: Rule Statement' field with a text area containing a rule definition. A note at the bottom states: 'Most fields are locked for edit, because this Template has rules associated with it. In order to change locked values of the Template, you must perform a Template Upgrade.' On the right, the 'Topics' panel shows a tree view with 'system_markers' expanded, listing parameters such as begin_time, end_time, engine_name, event_count, hit_ratio, marker_name, marker_type, rule_activation_count, rules_processed, source_list, total_processing_time, credit_card_transaction, amount_withdrawn, stock, and customer_info.

4. Under **Rule Statement**, click **Upgrade Template**, and in the **Dependent Object** window, click **Continue**.

The following figures displays all the dependent objects:



- Under **Template Parameters**, expand the **amt** parameter, and edit the **Min Value** to 5000.
The following figure shows the template parameters that you need to replace:



- Click **Save**.
- Navigate to **Design > Templates**, and select **Details View** to verify that you can view the previous state of the template.

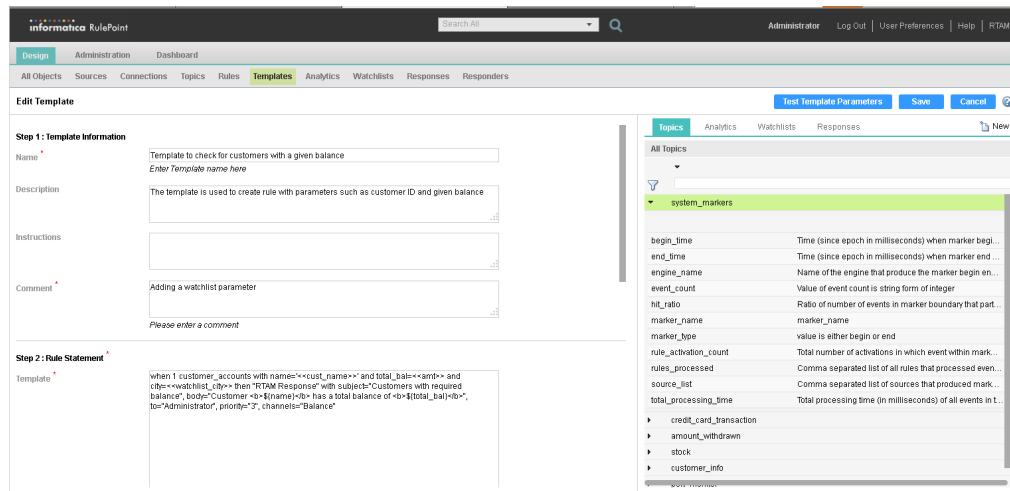
The template history shows the previous state of the template.

Step 7. Upgrade a Template to Include an Additional Parameter

Include an additional parameter in the template, such as a picklist.

1. On the **Design** tab, click **Templates**, and select the template **Template to check for customers with a given balance**.
2. Select **Edit** from the menu on the right side.
3. Under **Template Information > Comment**, type **Adding a watchlist parameter**.

The following information displays the configurations for including a watchlist in an existing template:



4. Under **Rule Statement**, click **Upgrade Template**.
5. On the **Dependent Objects** window, click **Continue**, and then perform the following tasks:

- a. Add the following rule statement:

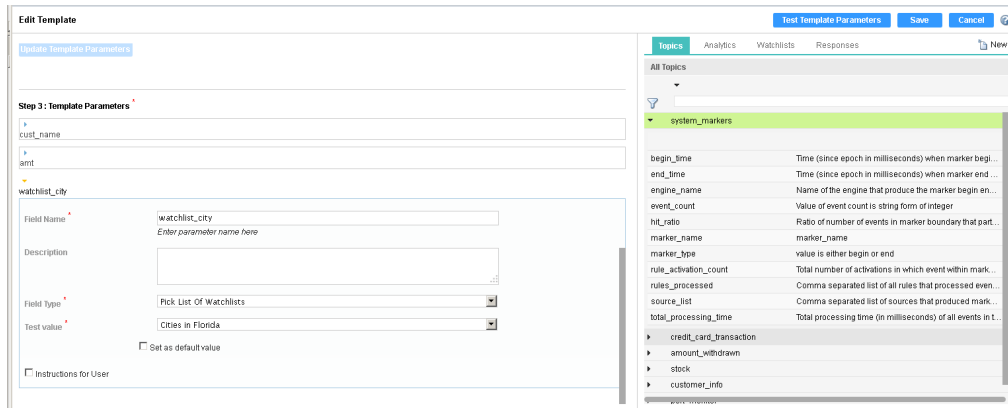
```
when 1 customer_accounts with name='<<cust_name>>' and total_bal=<<amt>> and
city=<<watchlist_city>> then "RTAM Response" with subject="Customers with
required balance", body="Customer <b>${name}</b> has a total balance of <b>${
total_bal}</b>", to="Administrator", priority="3", channels="Balance"
```

- b. Click **Update Template Parameters**.

6. Under **Template Parameters**, expand the parameter **watchlist_city**, and configure the following properties:

Field	Value
Field Name	watchlist_city
Field Type	Pick List of Watchlists
Test value	Cities in Florida

The following information displays the template for including a picklist in an existing template:



7. Click **Test Template Parameters**, and on the success message, click **OK**.
8. Click **Save**.
9. Under **Templates**, click the **Details View** on the right pane to verify that the history of the template is updated accordingly.

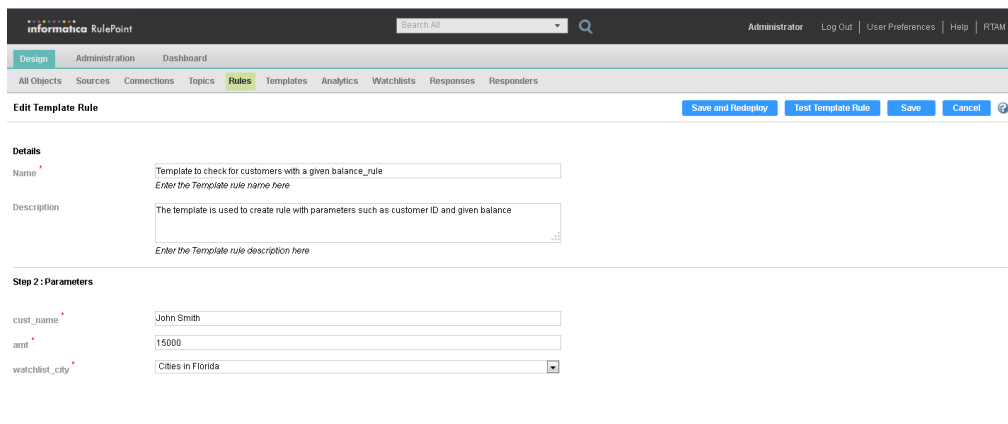
Note: The validity of the rule created from the template becomes false and it goes to NEEDS_DEPLOYMENT state.

Step 8. Update the Rule

Edit the template rule and provide the parameter value.

1. On the **Design** tab, click **Rules**, and select the template rule **Template rule to check for customers with a given balance**.
2. From the menu on the right-hand corner, select **Edit**.
3. Under the parameter, **watchlist_city**, select **Cities in Florida**.
4. Click **Save and Redeploy**.

The following figure shows the configurations for editing a template rule:



A message appears that prompts you to confirm the redeployment.

5. Click **OK**.

6. Navigate to the dashboard and verify that the rule is activated and the count is updated accordingly.

Step 9. View the RTAM Alert

View the alert on the RTAM.

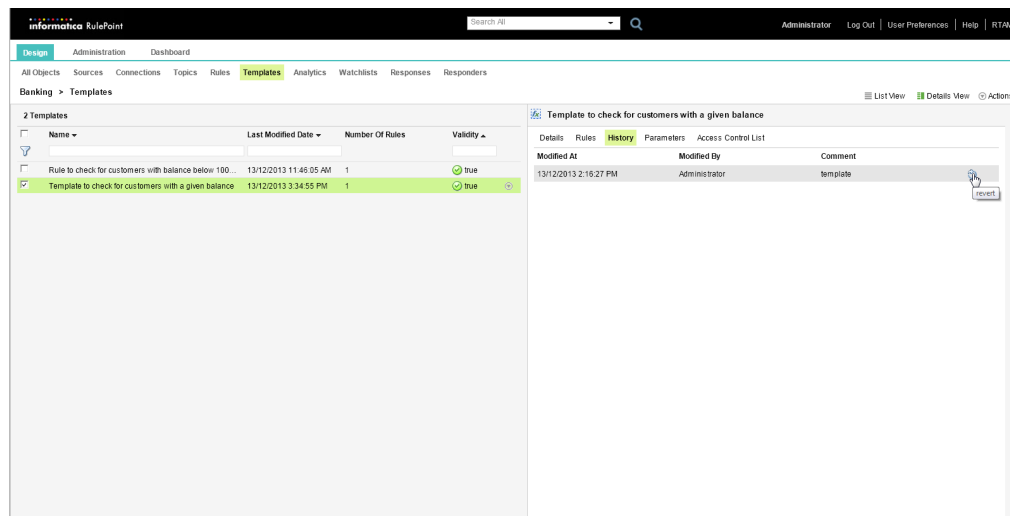
1. Log in to RTAM at the following URL:
`http://host:port/RTAM`
2. Provide the user name and password.
3. Under **Balance**, verify if the alert **Customers with required balance** displays.

Step 10. Revert the Template History

Revert the template to its initial state.

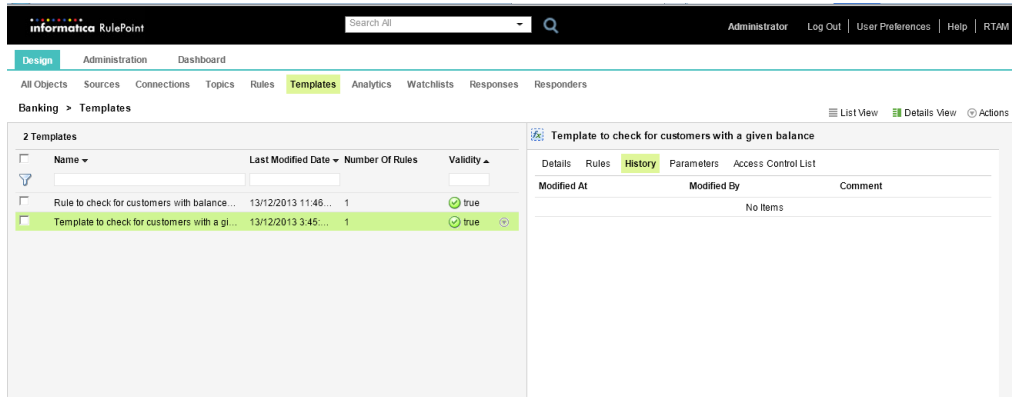
1. On the **Design** tab, click **Templates**, and select the template **Template to check for customers with a given balance**.
2. Click **Details View**, and then click **History** available in the right pane.
3. Click the **Revert History** icon, and on the Revert Confirmation message, click **OK**.

The following figure shows the history for the selected template in the right pane:



4. Click the **Revert History** icon again and on the Revert Confirmation message, click **OK**.
5. Verify that the template goes to the initial state and contains no history.

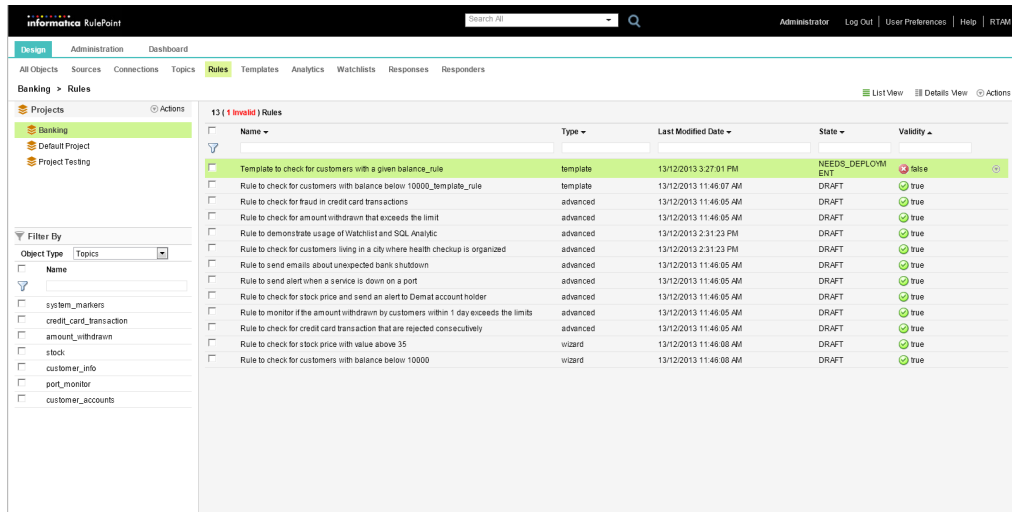
The following figure shows the initial state of the template:



6. Verify that the template rule updates accordingly and the state reflects as NEEDS_DEPLOYMENT.

Note: The template rule goes into invalid state when you add or delete a parameter in the template. After you upgrade or revert the template, you must edit the rule to make it valid.

The following figure shows the updated rule and its state:

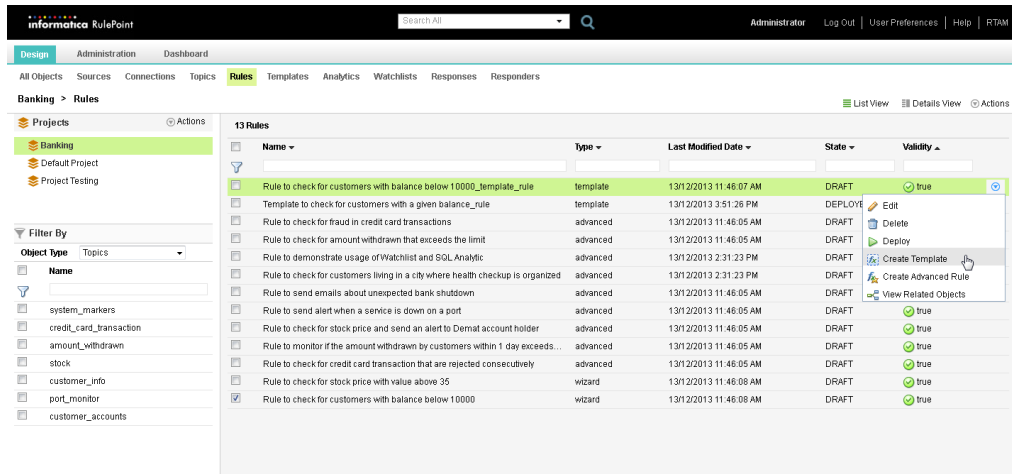


Step 11. Create a Template from the Wizard Rule

Convert the wizard rule to a template and then create a template rule from the template.

1. On the **Design** tab, click **Rules**.
2. Select the wizard rule **Rule to check for customers with balance below 10000**, and click **Create Template** from the menu on the right.

The following figure shows the selected rule from which you create a template:



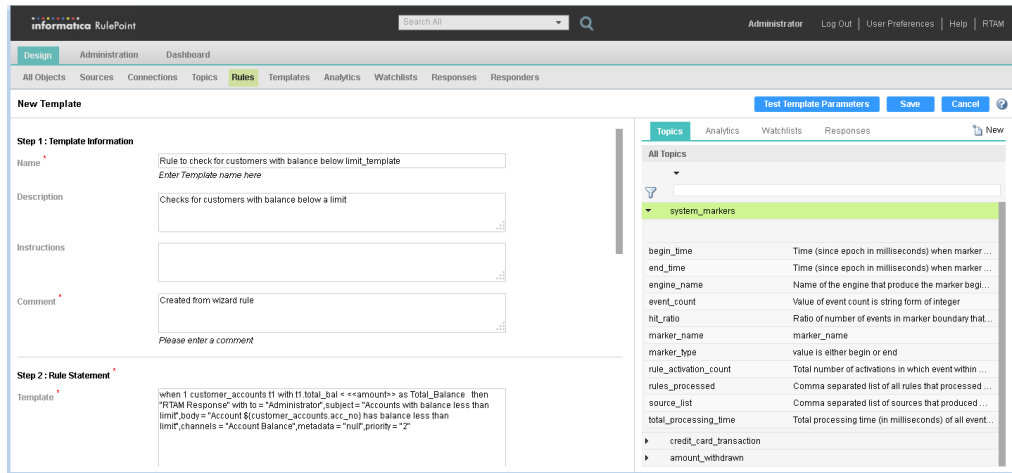
3. Configure the following properties:

Field	Value
Name	Rule to check for customers with balance below limit_template.
Comment	Created from wizard rule

4. In the **Rule Statement** section, replace **10000** with **amount**, and add the following value for the template:

```
when 1 customer_accounts t1 with t1.total_bal < <<amount>> as Total_Balance then
"RTAM Response" with to = "Administrator",subject = "Accounts with balance less than
limit",body = "Account ${customer_accounts.acc_no} has balance less than
limit",channels = "Account Balance",metadata = "null",priority = "2"
```

The following figure shows the template configurations in the **Design > Rules > New Template** view:

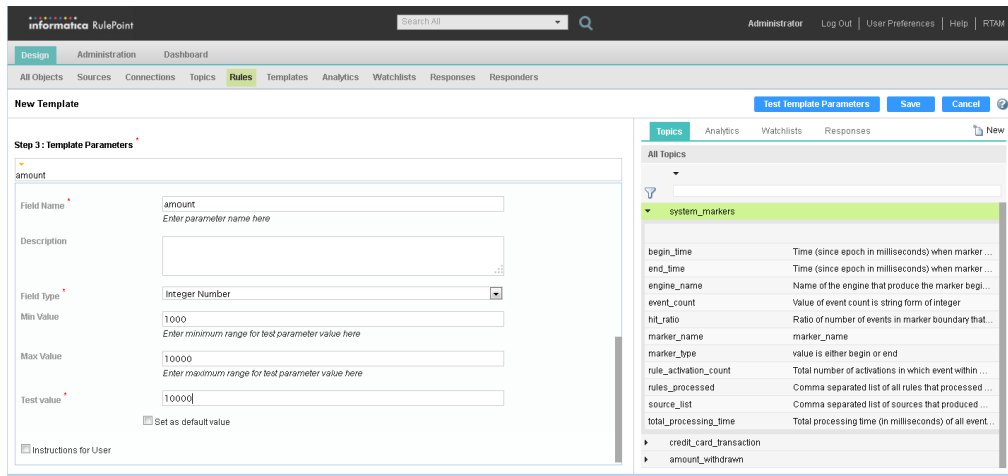


5. Under **Template Parameters**, configure the following properties:

Field	Value
Field Type	Integer number
Min value	1000

Field	Value
Max value	10000
Test value	10000

The following figure shows the **Template Parameters** section:



6. Click **Save**.

Step 12. Create a Template Rule

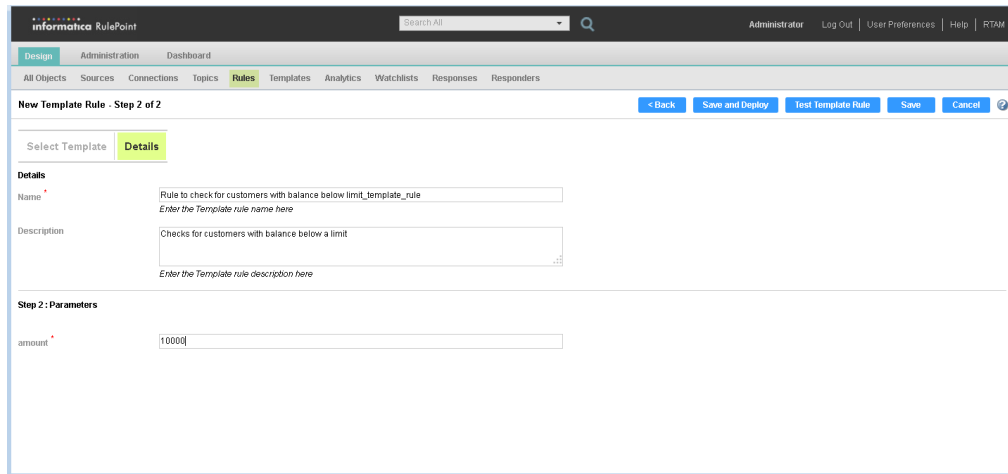
Create a template rule to check for customer accounts with balance that is below the specified limit.

1. On the **Design** tab, click **Rules**.
2. In the right pane, click **Actions > New > New Template Rule**.
3. Select the template: Rule to check for customers with balance below limit_template
4. Click **Next**.
5. Under **New Template Rule**, configure the following properties:

Field	Value
Name	Rule to check for customers with balance below limit_template_rule.
Description	Template Rule

6. In **Parameters**, enter 10000 under amount.

The following figure shows the **New Template Rule** configurations in the **Design > Rules** view



7. Click **Test Template Rule**.
8. When a success message appears indicating that the template rule validation is successful, click **OK**.
9. Click **Save**.

Step 13. Deploy the Rule, Source, and Responder

Deploy the objects in the run-time environment.

1. Click the **Design** tab, and deploy the Template rule, Rule to check for customers with balance below limit_template_rule, the source Customer Accounts Source and responder, RTAM Responder.
2. Verify that the dashboard displays the source, template rule, and the responder.
3. Verify that you view the updated count for the rules, responders, and sources.

Step 14. View the RTAM Alert

View the alert on the RTAM.

1. Log in to RTAM at the following URL:
`http://host:port/RTAM`
2. Provide the user name and password.
3. Under **Account Balance**, verify if the alert **Accounts with balance less than limit** displays.

CHAPTER 13

Using the Dashboard Functions

This chapter includes the following topics:

- [Using the Dashboard Functions Overview, 92](#)
- [Step 1. Perform Tasks for Deployed Sources, 93](#)
- [Step 2. Perform Tasks for Deployed Topics, 94](#)
- [Step 3. Perform Tasks for Deployed Rules, 96](#)
- [Step 4. Perform Tasks for Deployed Responses, 97](#)
- [Step 5. Set the Time Line Filter, 98](#)
- [Step 6. Purge an Application Service, 98](#)
- [Step 7. Purge the Topology, 99](#)
- [Step 8. View Errors, 99](#)

Using the Dashboard Functions Overview

In this lesson, you understand the various functions you can perform on the dashboard.

Lesson Concepts

After you create the objects during the design time, you need to deploy the objects so that they start functioning in the corresponding services in the run-time environment. The sources and supporting objects are deployed in the source controller, the rules and supporting objects are deployed in the event processor, and the responders and supporting objects are deployed in the responder controller. After you deploy the objects, you can perform various tasks for the deployed objects. This lesson gives you an understanding of the tasks you can perform using the features on the dashboard.

Lesson Objectives

In this lesson, you learn the following tasks:

- Perform tasks for sources deployed on the source controller
- Perform tasks for topics deployed on the source controller
- Perform tasks for rules deployed on the event processor
- Perform tasks for responses deployed on the responder controller
- General functions on the dashboard, such as setting the time line, purging the topology and services, and viewing errors, if any.

Lesson Prerequisite

Before you start this lesson, you must perform the following tasks:

- Navigate to the dashboard and verify if the default application services are running. During this time, no objects appear in the right pane of the dashboard. You can see the default source controller, the default event processor, and the default responder controller. All services are started on the default node .
- Deploy the rule, source, and responder. Go to the dashboard and verify that the objects are deployed. You can view the source in the Default Source Controller. You can also view the objects associated with the source such as the topic and its details. You can view the rules and the associated objects when you select the event processor. You can view the responders and the associated objects when you select the responder controller.

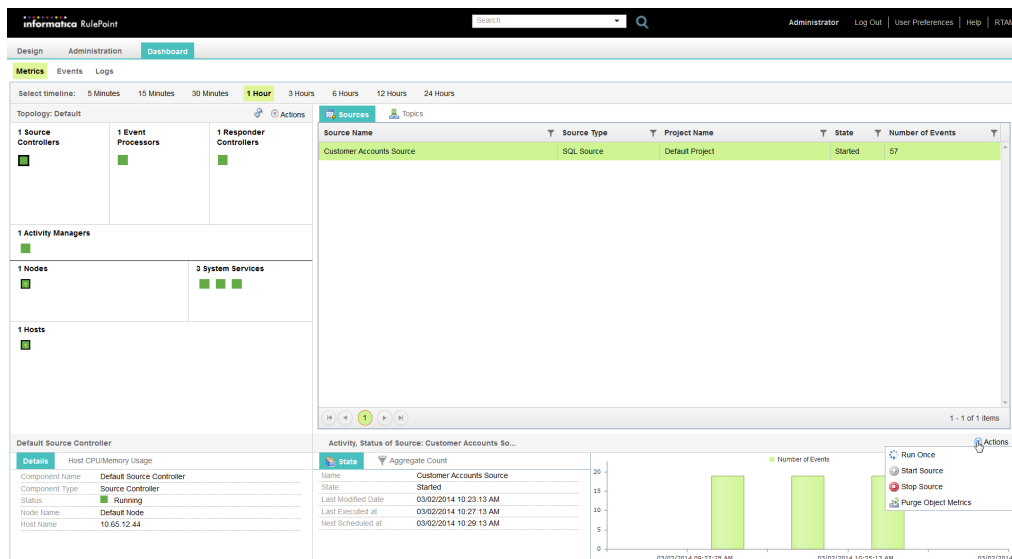
Step 1. Perform Tasks for Deployed Sources

After you deploy the sources and supporting objects, the features on the dashboard allow you to purge the run-time data, start or stop the source, or run the source once.

1. On the **Dashboard** tab, click the **Metrics** view.
2. Select the source controller where you have deployed the source and supporting objects.

The **Sources** and **Topics** tabs deployed in that source controller appears on the right pane. You can view the functions on the lower-right pane.

The following figure shows the deployed sources when you select the source controller in the **Sources** view of the contents panel:



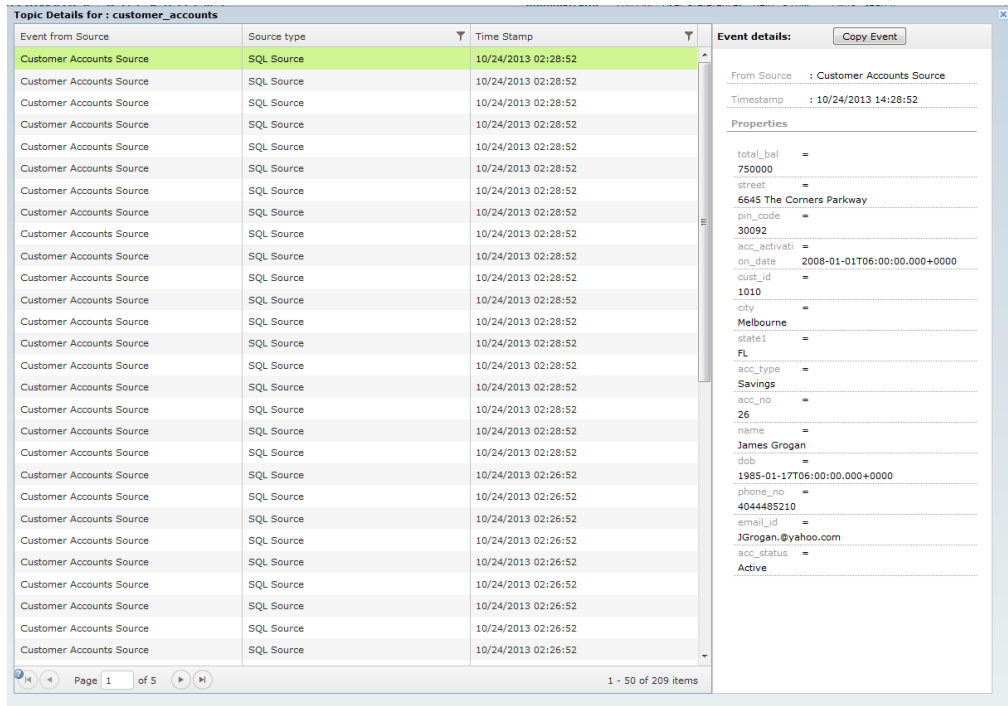
3. To purge the run-time data for the source, perform the following tasks:
 - a. From the **Actions** menu on the lower-right pane, select **Purge Object Metrics**.
 - b. Verify if all the run-time data such as aggregate count and the number of events resets to 0.

4. To run the source once, perform the following tasks:
 - a. From the **Actions** menu on the lower-right pane, select **Run Once**.
A success message appears.
 - b. Click **OK**.
 - c. From the **Actions** menu on the upper-right pane, select **Refresh**, and verify that the number of events are updated.
5. To stop the source, perform the following tasks:
 - a. From the **Actions** menu on the lower-right pane, select **Stop Source**.
A success message appears.
 - b. Click **OK**.
 - c. From the **Actions** menu on the upper-left pane, select **Refresh**.
 - d. On the contents pane, verify if the state of the source changes to **Stopped** and it does not fetch any event data.
6. To start the source that was stopped, perform the following tasks:
 - a. From the **Actions** menu on the lower-right pane, select **Start Source**.
 - b. A success message appears.
 - c. Click **OK**.
 - d. From the **Actions** menu on the upper-left pane, select **Refresh**.
You can also click the **Refresh** icon on the upper-left pane.
 - e. On the contents pane, verify that the state of the source changes to Started and it starts to fetch event data.

Step 2. Perform Tasks for Deployed Topics

After you deploy the sources and supporting objects, you can view the topic details, copy an event, or create an event.

1. On the **Dashboard** tab, select the source controller where you deployed the source.
The **Topics** tab appears on the right pane. You can view the functions in the **Actions** menu on the lower-right pane.
2. To view the topic details, such as the event from the source, the source type, and the time stamp, perform the following tasks:
 - a. Select **View Topic** from the **Actions** menu in the lower-right pane.
The Topic Details page appears, displaying the topic details of the selected event.The following figure shows the Topics Details page:



3. To copy event details and properties for a topic, perform the following tasks:
 - a. In the **View Topic Details** page, select any event, and click **Copy Event**.
The Create Event page appears.
 - b. If you want to create an event after you add or remove an event property from a topic, click **Create Event**.
 - c. When a success message, click **OK**. Close the **Topic Details** page, and click the **Refresh** icon in the upper-left pane.
You can also use refresh from the **Actions** menu.
The number of events appears updated.
4. To publish an event for a topic with the specified values, or add or remove the properties for the event you want to publish, click **Create Event** from the **Actions** menu in the lower-right pane.
5. Enter the following values, and click **Create Event**:
 - acc_no: 14879
 - acc_status : Active
 - acc_type : Savings
 - city : Melbourne
 - name : John Smith

You can choose to provide only one value.

The following figure shows the properties dialog box, where you can add or delete event properties.

Name	Value		
acc_activation_date		✖	✔
acc_no	14879	✖	✔
acc_status	Active	✖	✔
acc_type	Savings	✖	✔
city	Melbourne	✖	✔
cust_id		✖	✔
dob		✖	✔
email_id		✖	✔
name	John Smith	✖	✔
phone_no		✖	✔
pin_code		✖	✔
state1		✖	✔
street		✖	✔
total_bal		✖	✔

Buttons: Create Event, Cancel

Step 3. Perform Tasks for Deployed Rules

After you deploy the rules, you can view the rules and supporting objects in the event processor.

1. On the **Dashboard** tab, click the **Metrics** view.
2. On the left pane, select the event processor where you deployed the rules.
The **Rules** tab appears on the right pane. You can view the functions on the lower-right pane.
3. To purge the run-time data for a rule, perform the following tasks from the **Actions** menu on the lower-right pane:
 - a. Select **Purge Object Metrics**.
A success message appears.
 - b. Click **OK**.
 - c. Verify that the run-time data, such as the aggregate count and the number of activations resets to 0.
4. To stop a rule from executing, perform the following tasks from the **Actions** menu on the lower-right pane:
 - a. Select **Stop Rule**.
 - b. When a success message appears, click **OK**.

- c. Click **Refresh** from the **Actions** menu on the upper-right pane, and verify that the state of the rule changes to Stopped and you cannot see any rule activations.
5. To start the rule that you stopped, perform the following tasks from the **Actions** menu on the lower-right pane:
 - a. Select **Start Rule**.
 - b. When a success message appears, click **OK**.
 - c. Click the **Refresh** icon.

You can also click **Refresh** from the **Actions** menu in the upper-right pane.

Verify that the state of the rule changes to Started and you can view the rule activations.
6. To listen to event details of a rule and append it to a report for troubleshooting purposes, perform the following tasks:
 - a. Select **Enable Rule Tracing**.

A success message appears.
 - b. Click **OK**.
 - c. Navigate to the source controller, and select **Run Once** from the **Actions** menu, and refresh the data.

The trace report might take some time to generate.
7. To view the event tracing report summary for a rule, perform the following tasks:
 - a. Click **View Tracing**.
 - b. In the trace report, select the event to view the details. The summary provides the number of evaluations and activations for the rule. You can also view the configured rule, analytic, and response.
 - c. Close the trace report.
8. To disable rule tracing for a rule, perform the following tasks:
 - a. Click **Disable Rule Tracing**.
 - b. Click **OK** when a success message appears.

Step 4. Perform Tasks for Deployed Responses

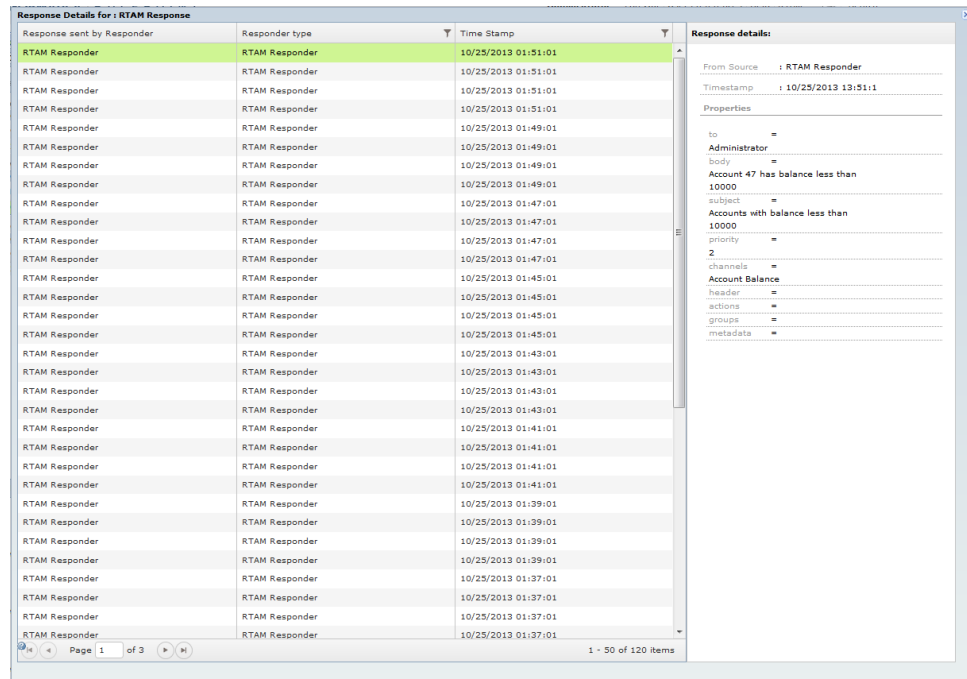
After you deploy the responders and supporting objects, you can view the responders and responses in the responder controller.

1. On the **Dashboard** tab, click **Default Responder Controller** on the left pane.

The **Responses** tab appears on the right pane. You can view the functions in the **Actions** menu in the lower-right pane.
2. To view the details for a response, such as the responder from which the response is sent, the responder type, and the time stamp, perform the following tasks:
 - a. Select **View responses** from the **Actions** menu in the lower-right pane.

- b. Verify that you can view the details of the response, such as the responder type and the time stamp.

The following figure shows the **Response Details** dialog box:



Step 5. Set the Time Line Filter

You can select the time line to see the metrics of objects in the run time. You can set it anywhere between five minutes and 24 hours. For example, if you set the time line at five minutes, you can see the metrics for the last five minutes. The default time line is one hour.

1. Select the time line as 30 minutes.
2. Verify that the dashboard displays the run-time data for the last 30 minutes.

Step 6. Purge an Application Service

You can purge the run-time data for all objects deployed on a source controller, event processor, or responder controller.

1. Select the source controller on the left pane, and click **Purge Controller Metrics** from the **Actions** menu in the upper-left pane.
2. Click **Ok** when prompted to confirm.
3. Verify that the event and aggregate count for both the sources and topics becomes 0 and the schedule data does not show any data.

Step 7. Purge the Topology

You can purge the run-time data for the entire topology.

1. Select any of the services on the left pane, and click **Purge Topology Metrics** from the **Actions** menu on the upper-left pane.
2. When prompted to confirm, click **OK**.
3. Verify that the event count and aggregate count across the topology is 0.

Step 8. View Errors

When one object attempts to connect to another object that does not exist, the dashboard displays an error for that object.

1. Click the **Design** tab, and edit the SQL source.
2. Under **Configuration**, edit the SQL query to:

```
select * from customer c,account_info a where c.cust_id=a.cust_id
```

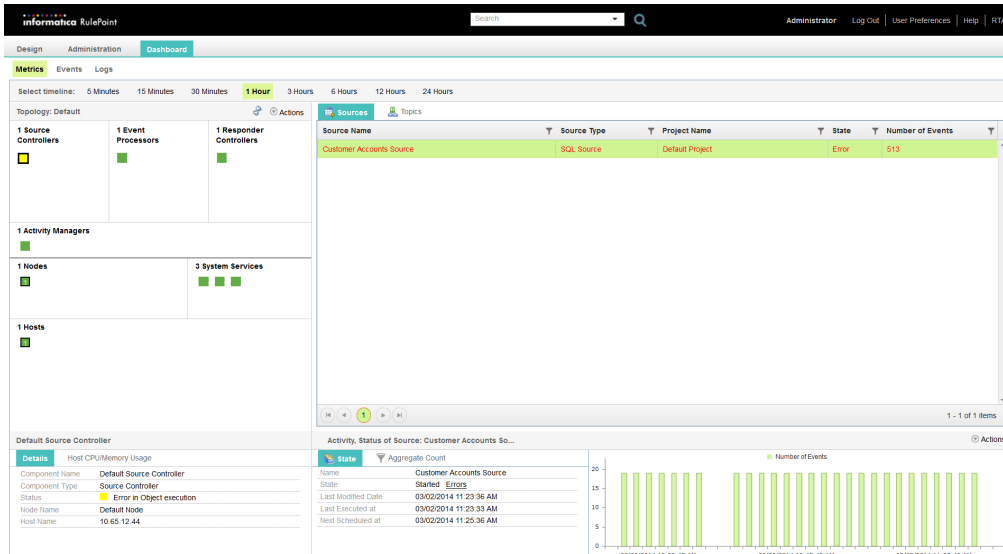
The following figure shows the **Edit Source** configurations in the **Sources** view of the **Design** tab:

Name	Description
customer_accounts	
acc_activation_date	
acc_no	
acc_status	
acc_type	
city	
cust_id	
name	
system_markers	
begin_time	Time (since epoch in milliseconds) when marker begin is ...
end_time	Time (since epoch in milliseconds) when marker end is ...
engine_name	Name of the engine that produce the marker begin end e...
event_count	Value of event count is string form of integer
hit_ratio	Ratio of number of events in marker boundary that partici...

3. Click **Save and Redeploy**, and click **OK** to confirm.
A message appears that redeployment is successful and if you want to preview the events.
4. Click **Cancel**.

- Navigate to the dashboard and verify that the status of the source controller where you deployed the source shows the status as error.

The following figure shows the error for the **Customer Account Source** on the dashboard for the source controller:



- To view the logs, perform one of the following tasks:

- On the **Dashboard** tab, click the **Logs** view.

The view displays the node, the log level, the component type, the message, and the log details. You can specify the start and end date to view the logs for that period.

- Select the source with the error in the contents panel of the dashboard, click the **State** view in the **Activity and Status** section on the lower pane, and then click the **Errors** link.

Verify that you can view the details, such as the source name, source type, timestamp, error code, and error message. On the right-hand side, you can view the error trace. After you complete viewing the details, close the **Error Details** window.

- On the **Design** tab, edit the SQL source to change the SQL query to:

```
select * from customer_info c,account_info a where c.cust_id=a.cust_id
```

- Click **Save and Redeploy**.

- Navigate to the dashboard and verify that the Source Controller error flag is cleared and you can view the status as **Last Errors** on the **State** view on the lower-right pane.

CHAPTER 14

Managing Banking Users and Roles

This chapter includes the following topics:

- [Managing Banking Users and Roles Overview, 101](#)
- [Step 1. Create a Banking Role, 102](#)
- [Step 2. Create a User, 102](#)
- [Step 3. Log In to RulePoint with New User, 103](#)
- [Step 4. Provide Project-Level Permissions, 103](#)
- [Step 5. Edit the Role, 103](#)

Managing Banking Users and Roles Overview

In this lesson, you create users for the banking domain in addition to the default administrators created during RulePoint installation.

Lesson Concepts

You can customize a role by associating a set of privileges to roles. When you associate a user with one or more roles, that user acquires all the privileges attached to the assigned roles. In this lesson, you perform tasks to provide permissions that reflect the banking role. After you create a role, you associate that role with the user, in this case, the banking user.

Lesson Objectives

In this lesson, you will learn how to perform the following tasks:

- Create a role.
- Create a user.
- Log in to RulePoint as new user.
- Provide project level permissions.
- Edit the role.

Lesson Prerequisite

Before you start this lesson, verify that the design time is running.

Step 1. Create a Banking Role

Create a banking role that contains privileges for a user to create and view objects and perform a specific set of administrative tasks.

1. Open a browser, and go to `http://<hostname>:8080/rulepoint`.
2. Provide the login credentials.
3. On the **Administration** tab, click the **User Management** view, and then click **Roles**.
4. In the navigation pane, click the **Actions** menu, and then select **New Role**.
5. In the **Name** field of the **Details** view, enter a name `BNK_ROLE` for the role.
6. Click **Next**.
7. Select the following privileges that you want to assign to the role:
PRIV_CREATE_PROJECT, PRIV_CREATE_SOURCE, PRIV_VIEW_SOURCE, PRIV_CREATE_CONNECTION,
PRIV_VIEW_CONNECTION, PRIV_CREATE_TOPIC, and PRIV_VIEW_TOPIC
8. Click **Save**.

Step 2. Create a User

After you create a banking role, you must associate that role with a user.

1. On the **Administration** tab, click the **User Management** view, and then click **Users**.
2. In the navigation pane, click **Actions > New User**.
3. Configure the following properties:

Field	Value
Username	bankuser1
Password	bankuser1
First Name	bank
Last Name	user1

4. Click **Next**.
5. Under **Available Roles**, select the `BNK_ROLE` to associate with the `bankuser1`, and click **Save**.

Step 3. Log In to RulePoint with New User

Use a web browser to access the RulePoint user interface. Use the credentials that you created for the banking user.

1. Open a browser, and go to the following URL:

```
http://host:port/rulepoint
```

2. Provide the user credentials.
3. Verify that the **Administration** tab is not visible for the user.
The **Administration** tab is visible only if the user has PRIV_SUPER_USER privileges for the associated role.
4. Navigate to the **Topics** view, and create a new topic. Click **Actions > New**.
Verify that you can create a new topic.
5. On the **Design** tab, click the **Responder** view, and verify that you cannot create a responder.
You do not have enough privileges to view or create a responder.

Note: By default, a permission that you provide for a role reflects on the project **Default Project** and the objects associated with it.

Step 4. Provide Project-Level Permissions

Provide read and write permissions for the banking user to be able to view and create objects in RulePoint.

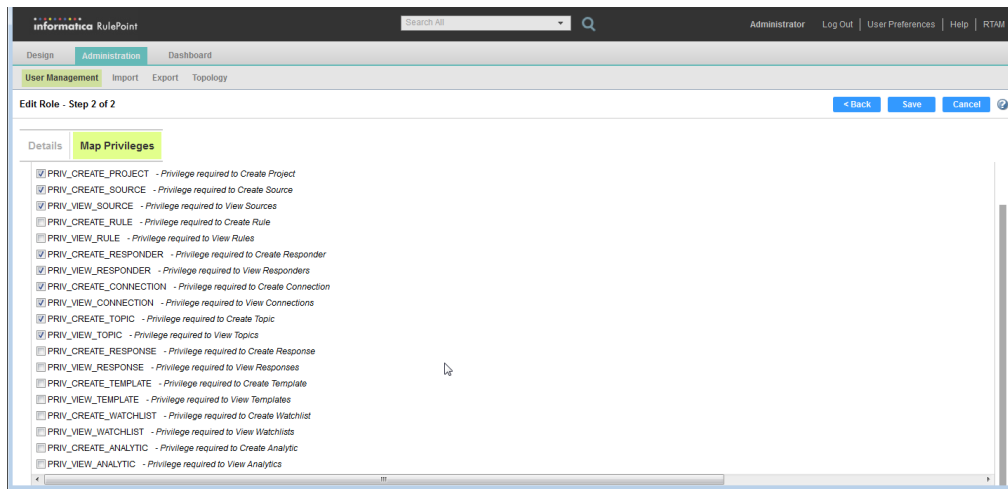
1. Log out from the user bankuser1.
2. Log in as Administrator/Administrator1.
3. Select the project **Banking**, and from the **Actions** menu in the navigator, select **Edit Direct Permissions**.
4. Click the **Add** icon, type the name bankuser1, and search. Click **OK** when the name displays.
5. Select the name bankuser1, and provide the READ and WRITE permissions.
6. Log in as bankuser1 and verify that the user can view and create the objects as defined by the role BNK_ROLE under the project Banking.

Step 5. Edit the Role

Add additional privileges to the banking role. A user cannot edit one's own direct permissions. For any role that does not have administrator privilege, that user cannot view or edit the ACL.

1. Log in as Administrator.
2. On the **Administration** tab, navigate to **Roles**.
3. Select **BNK_ROLE**, and click **Actions > Edit Role**.
4. Click **Next** in the **Details** page, and provide the privilege PRIV_VIEW_RESPONDER.

The following figure displays the privileges that you can map to a role:



5. Log in back as Administrator, edit the role BNK_ROLE, and provide the privilege PRIV_CREATE_RESPONDER.
6. Log in as bankuser1 and verify that you can create the responders.

CHAPTER 15

Importing and Exporting Objects

This chapter includes the following topics:

- [Importing and Exporting Objects Overview, 105](#)
- [Step 1. Export Selected Objects, 105](#)
- [Step 2. Import Objects, 107](#)

Importing and Exporting Objects Overview

In this lesson, you learn to import and export objects to and from a project.

Lesson Concepts

The import and export options allow you to import and export the configured RulePoint objects across projects. Use the import option to import configured objects into the RulePoint design time. Use the export option to export configured RulePoint objects from a project to an XML file. You can later import the XML file into a RulePoint project when required. In this lesson, you learn to export selected objects that you have created in a project. You also learn how to upload a file and then import a file into a project.

Lesson Objectives

In this lesson, you learn how to perform the following tasks:

- Export selected objects.
- Import objects.

Lesson Prerequisites

Before you start this lesson, verify that the design time is running.

Log in into RulePoint with the user name Administrator and password Administrator1.

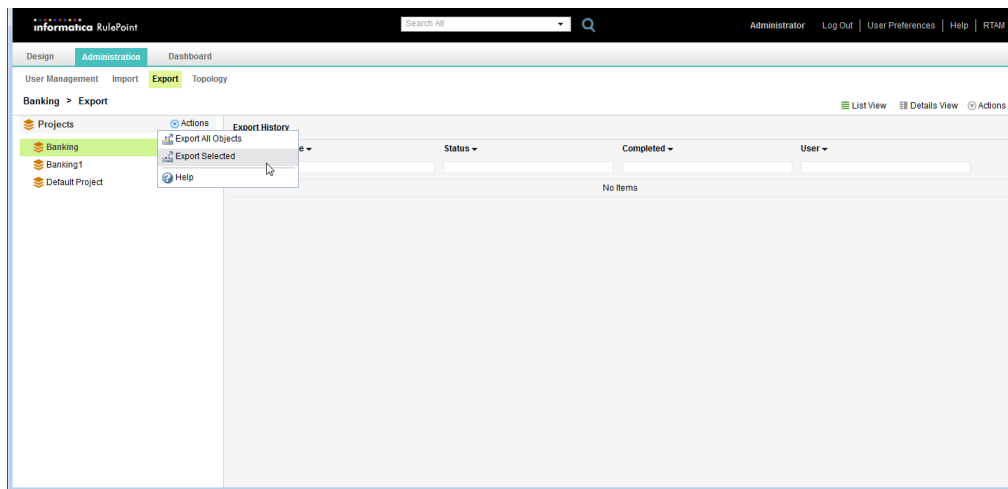
Step 1. Export Selected Objects

You can choose to export specific objects either into other projects, create a backup, or use in other Rulepoint instances. You can export the dependent objects only if those objects have an associated primary object for export. You can choose to export all objects by using the **Export All** option.

1. On the **Administration** tab, click the **Export** view.

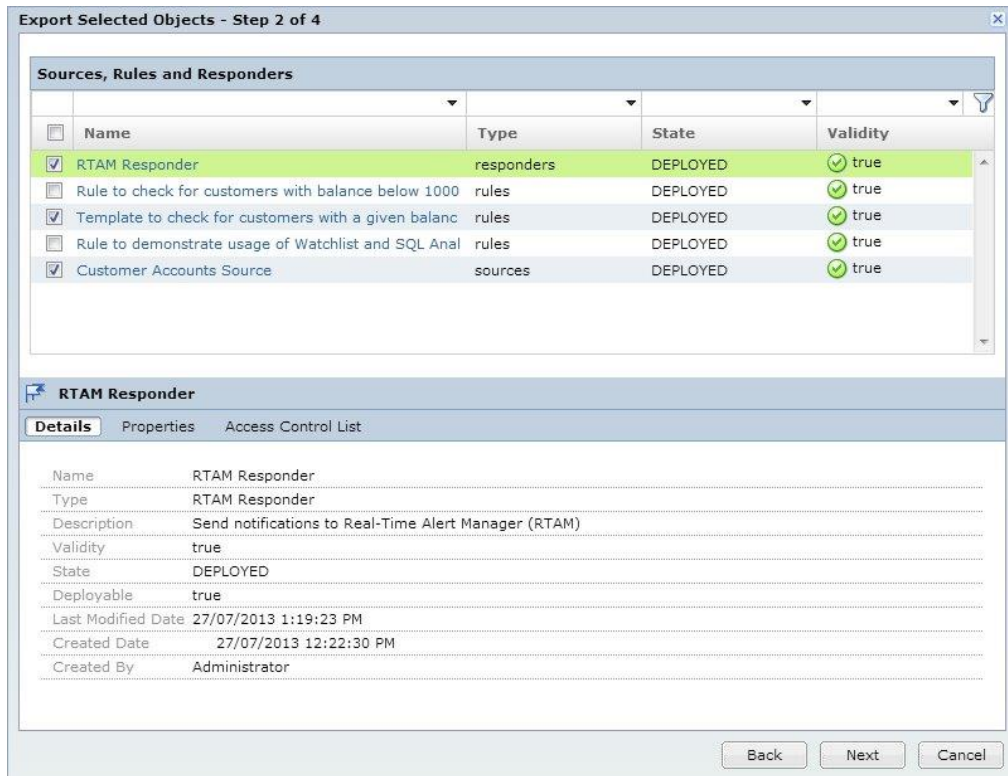
- In the navigator pane, click **Actions > Export Selected**.

The following figure shows the export options you can use for exporting objects from a project:



- Under **File Name**, type a name for the XML file, `Export_selected.xml`
- Select the following options:
 - Include invalid objects
 - Include ACLs in Export Set
- Click **Next**.

The following image shows the **Export Selected Objects - Step 2 of 4** dialog box:



- Select the objects that you want to export, and then click **Next**.

7. Review the summary, and then click **Next**.
8. On the **Users** page, select the user bankuser1, and click **Export**.
9. If the export process succeeds, a success message appears.

Step 2. Import Objects

You can import configured objects into a project in RulePoint. When you import objects, you can choose to fail, skip, or update objects.

1. Create a project, Import_Test.
2. Select the created project, and on the **Administration** tab, click the **Import** view.
3. In the navigator pane, click **Actions > Upload File**.
4. In the **Upload File** dialog box, click **Browse**, and then navigate to the XML file that you want to import.
You can import the xml file which you downloaded after the export in the previous lesson.
5. Click **Actions > Upload File**.
A message appears, which indicates that the file is successfully uploaded to the server.
6. Click **OK**.
7. In the right pane, click **Available Files**, and then select the file that you want to import.
8. In the right pane, click **Actions > Start Import**.
9. In the **Import** dialog box, under **On Collision**, specify what you want to do if there is a collision, and then click **Import**.
The preferable option is to update, as bankuser1 exists in the system.
10. When the success message appears, click **OK**.

CHAPTER 16

Setting Up High Availability

This chapter includes the following topics:

- [Setting Up High Availability Overview, 108](#)
- [Step 1. Add a Node, 109](#)
- [Step 2. Create an Event Processor for High Availability, 110](#)
- [Step 3. Verify the High Availability Settings, 111](#)
- [Step 4. Update the Deployment Policy, 112](#)
- [Step 5. Reassign a Rule, 113](#)

Setting Up High Availability Overview

In this lesson, you learn how to enable high availability for the event processor.

Lesson Concepts

In a single node setup, all the service controllers and the rule engine run on a single node. In this lesson, you learn to add multiple nodes and assign the service controllers and rule engine to different nodes. You can run the service controllers and the event processor in a standalone mode or in a high availability mode. In standalone mode, if the node fails, the server takes some time to bring up the node, the controllers, and the event processor running on that node. In the meantime, events may be lost. This can also result in a loss of alerts or duplication of alerts.

A high availability enabled mode has a primary node and a backup node. The primary node takes care of all the evaluation and processing activities, while the backup node mimics the activity of the primary node. When the primary node goes down, the backup node becomes the primary node and processes events and rules. The primary node becomes the backup node and no events are lost.

In this lesson, you also learn to reassign rules to different event processors.

Lesson Objectives

In this lesson, you learn how to perform the following tasks:

- Add a node.
- Create a high availability event processor.
- Verify the high availability settings.
- Update the deployment policy.
- Reassign a rule.

Lesson Prerequisite

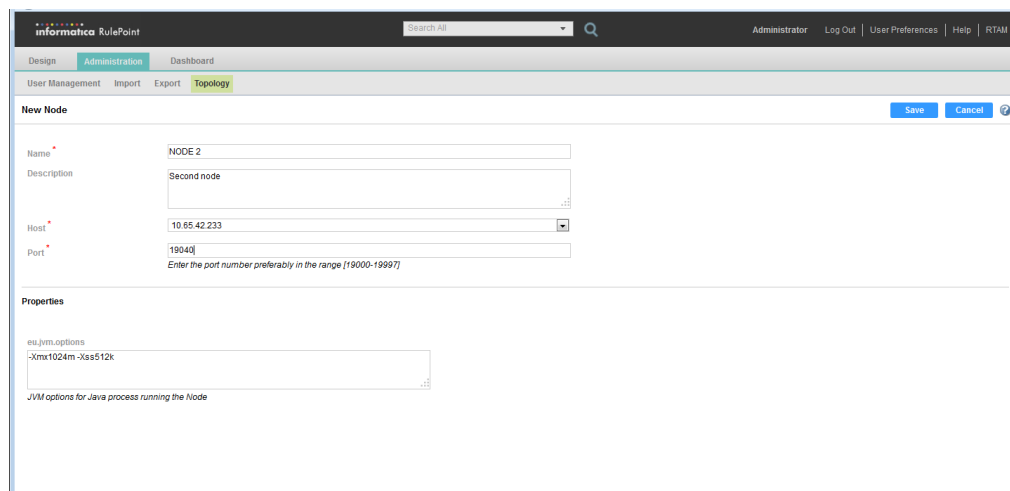
Before you add a node and start the lesson, verify that you shut down the run-time instance in RulePoint.

Step 1. Add a Node

You must configure a node in addition to the default node so that you can configure one of the nodes as the backup node for high availability.

1. On the **Administration** tab, select **Topology**.
2. Select **Nodes** from the Topology tree on the left pane.
3. In the right pane, click **Actions > Add Node**.

The following figure shows the configuration fields for adding a node in the topology:



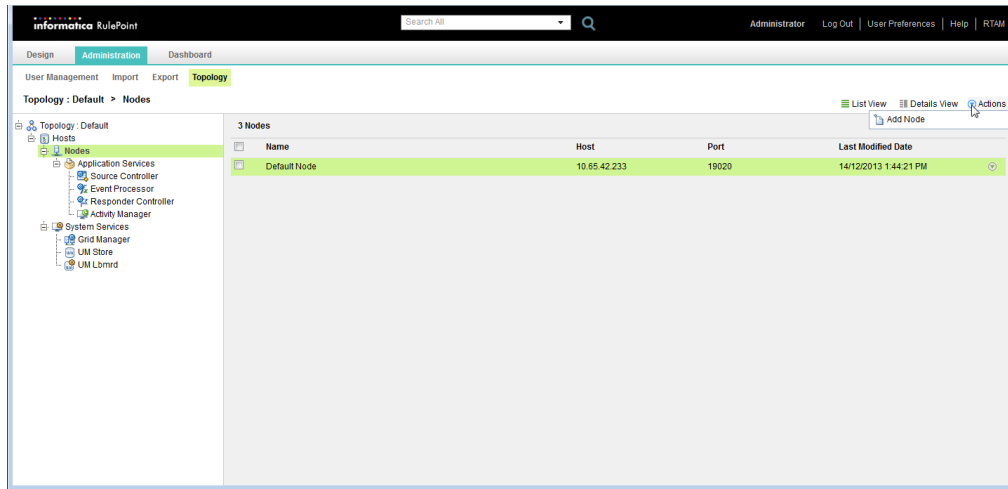
The screenshot shows the Informatica RulePoint Administration console. The 'Administration' tab is selected, and the 'Topology' sub-tab is active. The 'New Node' form is displayed with the following fields and values:

- Name:** NODE 2
- Description:** Second node
- Host:** 10.65.42.233
- Port:** 19043
- Properties:** -Xms1024m -Xss512k

4. Configure the following properties:

Field	Value
Name	NODE 2
Description	Second node
Host	Select the host from the menu
Port	Provide a unique number between 19000 and 19997

The following figure shows the configured node in the topology:



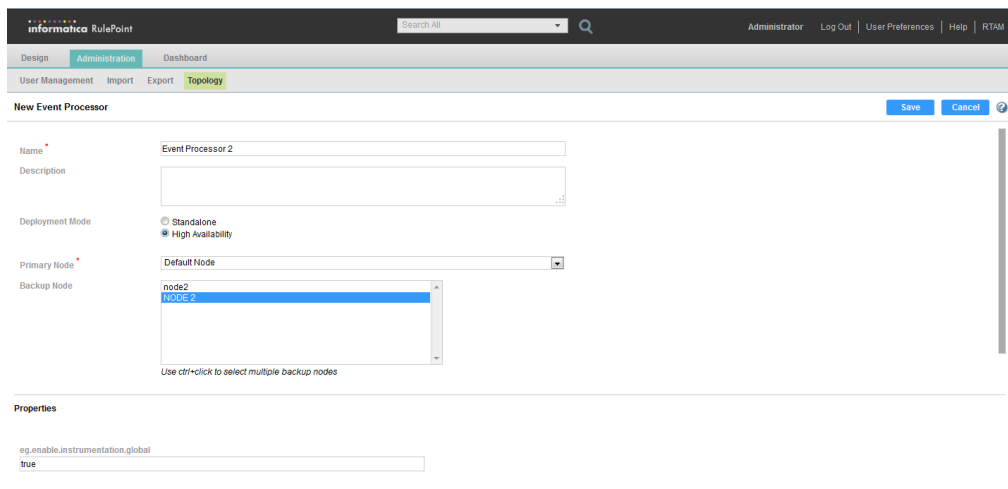
5. Click **Save**.

Step 2. Create an Event Processor for High Availability

Configure an event processor in high availability mode so that there is a primary node and a backup node.

1. Select **Event Processor** from the Topology tree on the left pane.
2. In the right pane, click **Actions > Add Event Processor**.

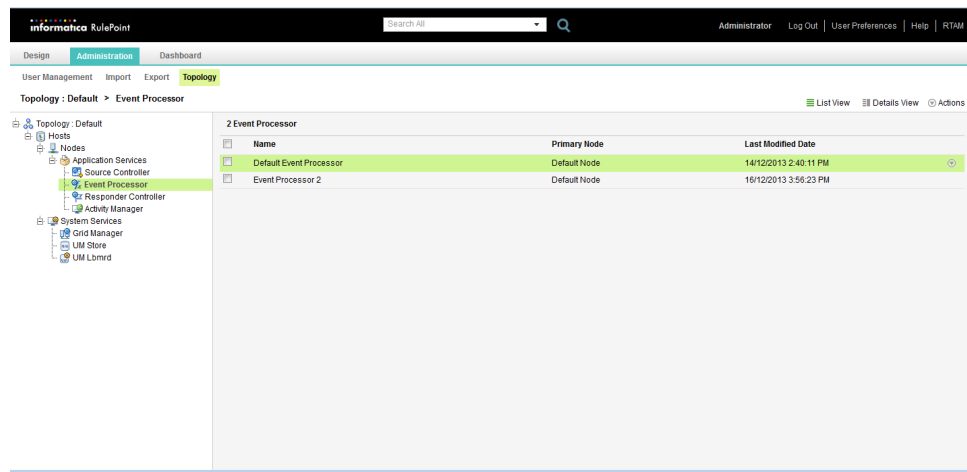
The following figure shows the configurations to create a new event processor:



3. Configure the following properties:
 - a. Provide a name, Event Processor 2
 - b. Select the Deployment Mode as High Availability.

- c. Select the node NODE2 as the Primary Node, and Default Node as the Backup Node.

The following figure shows the newly configured event processor in the **Topology > Event Processor** view:



4. Click **Save**.
5. Start the host agent.
6. Start the topology.
7. From the dashboard, when you select Event Processor 2, verify that two nodes are highlighted. One is the Default Node and the other NODE 2.

Note: If the primary node, NODE 2, goes down, the server makes the backup node, Default Node, as the primary, while the previous node, NODE 2, as the backup.

Step 3. Verify the High Availability Settings

After you configure high availability for the event processor, deploy the objects, and verify that the dashboard reflects the deployment changes accordingly.

1. On the **Design** tab, click **Actions > Deploy > Rules, Sources & Responders**.
2. Deploy the following rules:
 - Rule to check for customers with balance below 10000
 - Rule to demonstrate usage of Watchlist and SQL Analytic
3. Click **Next**.
4. Under **Edit Mapping of Rules to Event Processors**, select **Event Processor 2**, and click **Next**.
5. In the **Template Rules** page, select the template rule, Template rule to check for accounts with balance below the limit, and click **Next**.
6. Select the source, Customer Accounts Source, and click **Next**.
7. Select **RTAM Responder**, and click **Deploy**.
8. Navigate to the dashboard, and verify if the source and responder are deployed and the counts for various objects are updated accordingly.

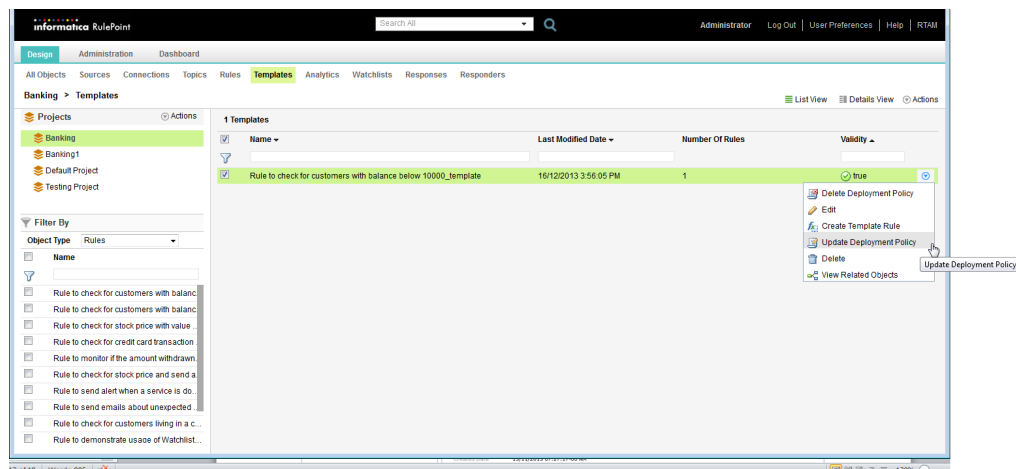
9. Select **Event Processor 2**, and verify that the **Rules** view on the right pane displays the Wizard rule, the Advanced rule, and the corresponding rule activation count.
10. Select **Default Event Processor**, and verify that the template rule displays in the **Rules** view on the right pane.
11. Log in to RTAM at `http://host:port/RTAM`, and verify that you can view the alerts.
12. Start the Task Manager, and kill the process for the configured port. This is the primary node for Event Processor 2.
13. Refresh the dashboard and verify that all nodes are running. Also verify that the Default Node displays as 1, which is the primary node. Verify that you can view the events generating.
14. Log in to RTAM and view the rule activations.

Step 4. Update the Deployment Policy

By default, the template deployment policy maps to the default event processor. If you configure multiple event processors, you can map the deployment policy across the available event processors.

1. On the **Design** tab, click **Templates**.
2. Select the template, Template to check for account balance below 10000, and click **Update Deployment Policy** from the menu on the right side.

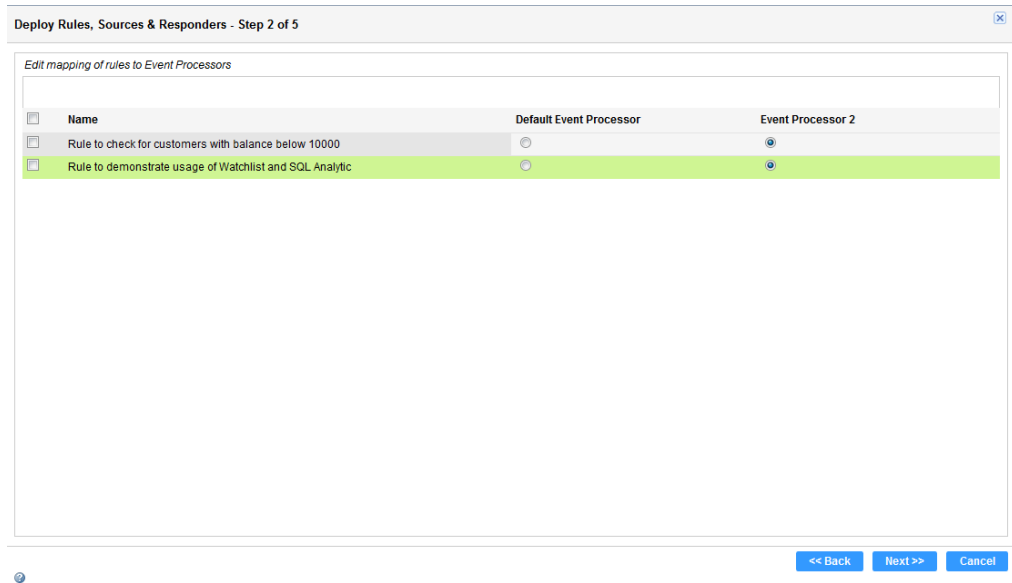
The following figure shows the **Update Deployment Policy** option for the selected template:



3. Change the mapping of the selected engine instances to the template. Clear **Default Event Processor** and select **Event Processor 2**.

Note: If you select both the event processors, the templates map to both the engines. Processing of rules occurs in a round-robin approach.

The following figure shows the rule mapped to Event Processor 2:



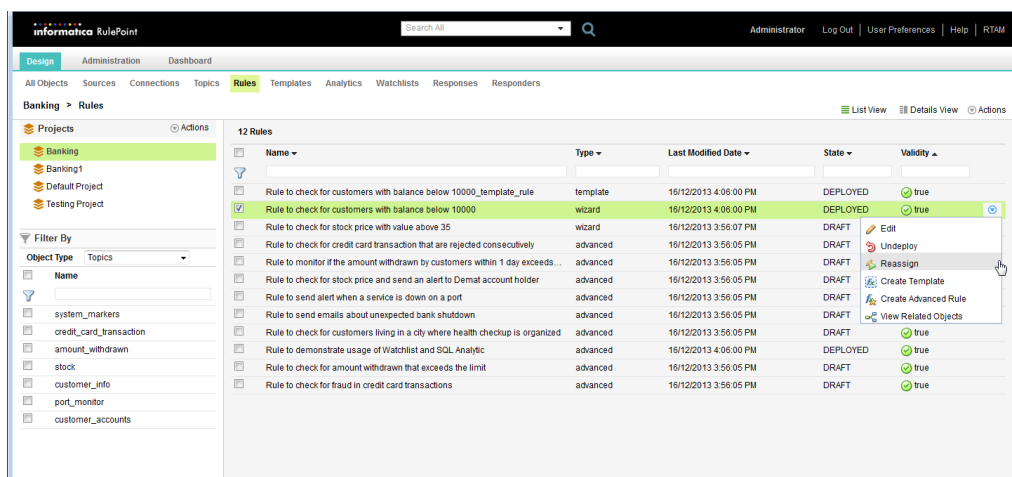
4. Click **Save**.
5. Click the **Dashboard** tab, and click Event Processor 2 under **Event Processors**.
You can view the deployed template rule in the **Rules** view on the right pane.

Step 5. Reassign a Rule

You can reassign objects to run on multiple service controllers or event processors. When you enable the event processor for high availability, you can reassign the rules to run on this event processor.

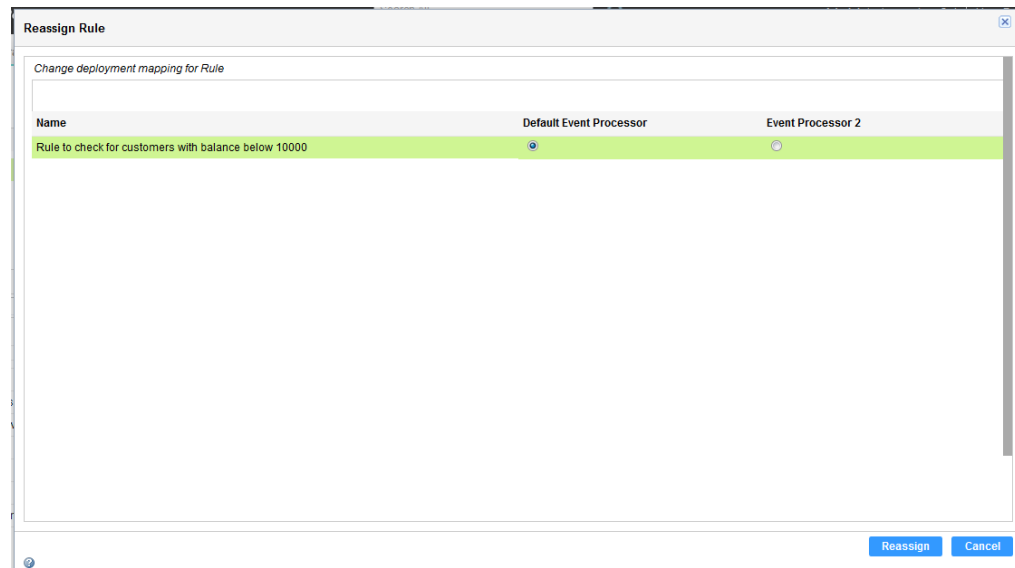
1. Navigate to the **Design** tab, and click **Rules**.
2. Select a deployed rule that you want to reassign, and click **Reassign** from the **Actions** menu.

The following figure shows the **Reassign** option for a selected rule in the **Rules** view of the **Dashboard** tab:



3. In the **Reassign** dialog box, select the rule and the event processor to which you want to reassign the rule, and click **Reassign**.

The following figure shows the screen where you can reassign the rule to a specific event processor:



4. Click the **Dashboard** tab, and in the **Default** view, select the event processor, Default Event Processor. The **Rules** view in the right pane displays the rule **Rule to check for customers with balance below 10000**, along with the activations and details.

CHAPTER 17

Using Custom Services

This chapter includes the following topics:

- [Using Custom Services Overview, 115](#)
- [Step 1. Create the JAR File, 116](#)
- [Alert Server Downtime Using Custom Source Overview, 116](#)
- [Notify Unexpected Bank Shutdown Using Custom Analytics Overview, 120](#)

Using Custom Services Overview

In this lesson, you understand the prerequisites before creating any custom services.

Lesson Concepts

You can use the RulePoint SDK to build your own custom pluggable services in RulePoint. The Custom Service API provides a set of classes that you can extend to develop pluggable custom services. You can create custom source, analytic, responder, connection, and marshaller services.

In this lesson, you learn how to configure and create the necessary JAR files. The lessons also guide you through tasks necessary for creating a custom source and analytic.

Lesson Objectives

In this lesson, you learn how to perform the following tasks:

- Create a JAR file.
- Use a custom source.
- Use a custom analytic

Lesson Prerequisites

Before you start this lesson, you must perform the following tasks:

- You must have a JDK compiler to build the classes. The Java Development Kit (JDK) contains the jar packaging tool.
- You must have a basic knowledge of OSGi to write the manifest file.
- Set the environment variable, RULEPOINT_HOME, to the RulePoint 6.2 installation location.
- Copy the `Banking` and `commonlibs` directories from `<INSTALLER_HOME>\samples` to your work location. The files contain the libraries required to create the JAR files.
- You must have Apache ANT installed on your machine to create the JAR files, as the instructions provided in this lesson uses Apache ANT. You can choose to download an application of your choice.

- Update the environmental variable, PATH, to the <ANT installation>/bin directory.

Step 1. Create the JAR File

Configure the required files to create a custom service jar.

If you want to import the banking XML that is part of the RulePoint installation package, you need to copy the jars, `portmonitorservice.jar` and `startswithanalytic.jar` from `C:\RulePoint_6.2\samples\Banking\build` to `C:\RulePoint_6.2\custom` directory.

1. Open Command Prompt.
2. Navigate to your workspace where you have copied the Banking folder.
3. Type `ant`, and press **ENTER** to create the JAR files.

The `build.xml` creates the required JARs, `portmonitorservice.jar` and `startswithanalytic.jar` in the `Build` folder, for example, `C:\Banking\build`.

4. Add the created jar in the `C:\RulePoint_6.2\custom` directory.

Note: You do not need to restart the design time after placing the jars if the server is running.

Alert Server Downtime Using Custom Source Overview

In this lesson, you learn to create objects required for sending an alert when a service on a port is down. You learn to create a custom source.

Lesson Concepts

Source services connect to external systems to fetch data and publish the events on topics. You can create schedulable and listener source services.

In this lesson, you create a custom source called Monitor Server Port to retrieve data from an SQL source. You learn to create an advanced rule with a condition stating that when the service on a port is down, send an RTAM alert notifying that the server is down.

Lesson Objectives

In this lesson, you learn how to perform the following tasks:

- Create a topic, `port_monitor`.
- Create an SQL source, Monitor Server Port.
- Create a schedule for the SQL source.
- Create an RTAM responder.
- Create an RTAM response.
- Create an advanced rule.
- Deploy objects and view events on dashboard.

- Shut down the server.
- View the RTAM alert.

Lesson Prerequisite

Before you start this lesson, you must place the created `portmonitorservice.jar` and `startswithanalytic` in the `RULEPOINT_HOME/custom` directory.

Step 1. Create a Topic

Create a topic **port_monitor**, and configure the required properties for the topic.

1. Log in to RulePoint using the login credentials.
2. Create a project named Custom Objects.
3. Select the project, Custom Objects, select **Topics** in the **Design** tab, and click **Actions > New**.
4. Under **Details**, configure the following properties:

Field	Value
Name	port_monitor
Expires in	600000
Responder access	ALL_PROPERTIES

5. Under **Properties**, click **Bulk Create**.
6. Type the following properties in the editor, and click **Add**:
 - host
 - message
 - port
 - service_up
7. Click **Save**.

Step 2. Create a Source

Create a source to monitor the service on a configured port.

1. On the **Design** tab, select **Sources**.
2. In the right pane, click **Actions > New**.
3. Under **Details**, configure the following properties:

Field	Value
Name	Monitor Server Port
Description	Monitor a Port on the Host
Type	Port Monitor Service

- Under **Topic Information**, select the topic name, port_monitor, and click **OK**.
- Under **Configuration**, configure the following properties:

Field	Value
Host	Provide the host name of the server.
Port	Provide a valid port number.

- Click **Save**.

Step 3. Create a Schedule

Create a schedule for the source.

- Under **Monitor Server Port**, click **Create Schedule**.
- Configure the following properties:

Field	Value
Schedule Type	Dynamic
Repeat Interval	120000 (In milliseconds)

- Click **Add Schedule**.

Step 4. Create a Responder

Create an RTAM responder to send notifications when events match the configured condition.

- On the **Design** tab, select **Responders**.
- In the right pane, click **Actions > New**.
- Under **Responder**, configure the following properties:

Field	Value
Name	RTAM Responder
Description	Send notifications to Real-Time Alert Manager (RTAM)
Type	RTAM Responder

- Under **Configuration**, configure the following properties:

Field	Value
To	Administrator
Subject	rtam alert

Field	Value
Body	rtam alert
Message Priority	3

5. Click **Save**.

Step 5. Create a Response

Configure an RTAM response to view the triggered alerts.

1. On the **Design** tab, select **Responses**.
2. In the right pane, click **Actions > New**.
3. Configure the following properties:

Field	Value
Name	RTAM Response
Description	Sends alerts to Real-Time Alert Manager (RTAM)

4. Under **Responder Information**, select **RTAM Responder**.
5. Click **Save**.

Step 6. Create an Advanced Rule

Create a rule that sends an alert when a service on a port is down.

1. On the **Design** tab, select **Rules**.
2. In the right pane, click **Actions > New > New Advanced Rule**.
3. Configure the following properties:

Field	Value
Name	Rule to send alert when a service is down on a port
Rule Statement	when 1 port_monitor t1 with t1.service_up = "NOT OK" as service_check then "RTAM Response" with body="Cannot connect to ActiveMQ server using the port \${t1.port}. Please check if the service is down and restart the service. ", subject = "ActiveMQ service is down!!", to = "Administrator", priority=5, channels="Monitor_Service"

4. Click **Save**.

Step 7. Deploy Objects

Deploy the objects to see the rule activations.

1. In the navigator, click **Actions > Rules, Sources & Responders > Deploy**.
2. Select the advanced rule, and click **Next**.

3. Select the source, and click **Next**.
4. Select the responder, and click **Deploy**.

View the deployed objects in the respective controllers in the dashboard.

Step 8. View Events on the Dashboard

After an event is published, you can view the topic details, such as the event from the source, the source type, and the time stamp.

1. On the **Dashboard** tab, select **Default Source Controller** on the left pane, and then select the **Topics** tab on the right pane.
2. Click **View Events** from the **Actions** menu on the lower-right pane.

The Topic Details page displays the topic details of the selected event. Verify that the server is running on the configured port.

Step 9. Shut Down the Server

Shut down the server so that the service is not available on the configured port and you can see the rule activation.

1. Open the ActiveMQ server Console.
2. Stop the server. Type **Ctrl+C**.

Step 10. View the RTAM Alert

View the alert displayed on the RTAM.

1. Log in to RTAM at `http://host:port/RTAM`.
2. Provide the login credentials, and click **Log In**.
3. Under **Monitor_Service**, verify that you can view the alert **ActiveMQ service is down!!**.

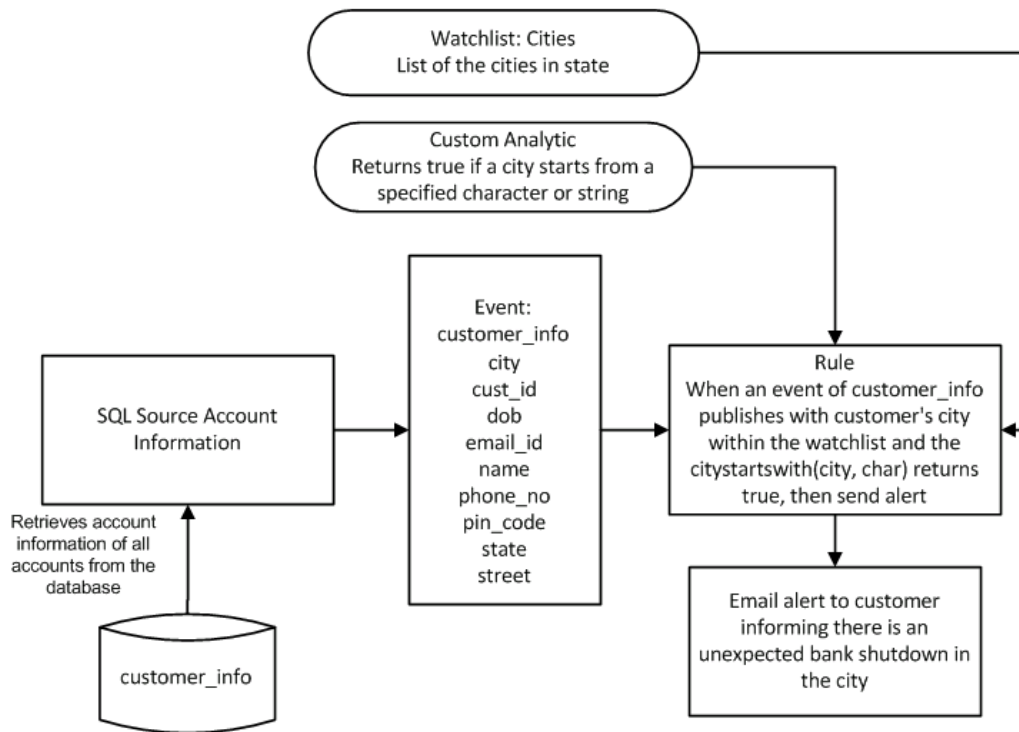
Notify Unexpected Bank Shutdown Using Custom Analytics Overview

In this lesson, you learn how to use a custom analytic in an advanced rule to notify customers of an unexpected shutdown.

Lesson Concepts

In this lesson, you create an advanced rule that uses a custom analytic to verify if the customer city name starts with a specified character. You use this analytic in a rule to send emails about an unexpected bank shutdown. You also learn to create the required objects to run this rule.

The following figure represents the model structure to notify unexpected shutdown:



The model representation shows an SQL source that retrieves customer information and publishes events on the customer_info topic. The rule uses a watchlist and a custom analytic to evaluate events and alert customers included in the watchlist by email when there is an unexpected bank shutdown.

Lesson Objectives

In this lesson, you learn to perform the following tasks:

- Create a topic.
- Create an SQL connection.
- Create an SQL source.
- Create a schedule.
- Create an analytic.
- Create an email connection.
- Create an email response.
- Create an advanced rule.
- Deploy objects.
- View email alerts

Lesson Prerequisite

Before you start this lesson, you must place the created `customservices.jar` in the `RULEPOINT_HOME/custom` directory.

Step 1. Create a Topic

Create a topic customer_info, and configure the required properties for the topic.

1. Log in to RulePoint using the login credentials.
2. Create a project named Custom Objects.
3. Select the project, Custom Objects, select **Topics**, and in the right pane, click **Actions > New**.
4. Under **Details**, configure the following properties:

Field	Value
Name	customer_info
Expires in	600000
Responder access	ALL_PROPERTIES

5. Under **Properties**, click **Bulk Create**.
6. Type the following properties in the editor, and click **Add**:
 - city
 - cust_id
 - dob
 - email_id
 - name
 - phone_no
 - pin_code
 - state1
 - street
7. Click **Save**.

Step 2. Create an SQL Connection

Create an SQL connection to connect to the database.

1. On the **Design** tab, select **Connections**.
2. From the menu in the right pane, click **Actions > New**.
3. Under **Connection**, configure the following properties:

Field	Value
Name	SQL Connection
Description	Connection to connect to the database
Type	SQL Connection

- Under **Configurations**, configure the following properties:

Field	Value
Driver Class	com.informatica.jdbc.oracle.OracleDriver
Connection URL	jdbc:informatica:oracle://infa-server:1521;databaseName=infaorcl
User Name	rulepoint
Password	rulepoint

- Click **Test Connection** to verify if the connection is successful.
- Click **Save**.

Step 3. Create a Source

Create an SQL source that retrieves customer information from the database.

- On the **Design** tab, select **Sources**.
- In the right pane, click **Actions > New**
- Under **Details**, configure the following properties:

Field	Value
Name	Customer Accounts Source
Description	Retrieves customer account details
Type	SQL Source
Connection	SQL Connection

- Under **Topic Information**, select the topic name, customer_info, and click **OK**.
- Under **Configuration**, configure the following property:

Field	Value
SQL	select * from customer_info

- Click **Save**.

Step 4. Create a Schedule

Create a schedule for the source.

- Under **Customer Accounts Source**, click **Create New Schedule**.

2. Configure the following properties:

Field	Value
Schedule Type	Dynamic
Repeat Interval	120000 (in milliseconds)

3. Click **Add Schedule**.

Step 5. Create an Analytic

Create an analytic for use in a rule such that it checks if the text starts with the string you specify.

1. On the **Design** tab, select **Analytics**.
2. In the right pane, click **Actions > New** from the menu.
3. Under **Details**, configure the following properties:

Field	Value
Name	citystartswith
Description	Usage: citystartswith(string,char) return true if the string starts with char
Type	StartsWith Analytic

Note: You can view the StartsWith Analytic option after you place the created `customservices.jar` in the `RULEPOINT_HOME/custom` directory.

4. Under **Configurations**, configure the following properties:

Field	Value
Condition Evaluation Required	True
Cache Duration	Usage: citystartswith(string,char) return true if the string starts with char
Type	0

5. Click **Save**.

Step 6. Create an Email Connection

Create an email connection to connect to the mail server.

1. On the **Design** tab, select **Connections**.
2. In the right pane, click **Actions > New**.

- Under **Connection**, configure the following properties:

Field	Value
Name	Email Connection
Description	Connection to connect to the mail server
Type	Email Connection

- Under **Configurations**, configure the following property:

Field	Value
Email Server	infa-server.com

- Click **Test Connection** to verify if the connection is successful.
- Click **Save**.

Step 7. Create an Email Responder

Create an email responder to send the configured alert to the hmail server.

- On the **Design** tab, select **Responders**.
- In the right pane, click **Actions > New**.
- Under **Responder**, configure the following properties:

Field	Value
Name	Email Responder
Description	Send Email Response
Type	Email Responder

- Under **Configuration**, configure the following properties:

Field	Value
To	student01@infa-server.com
Subject	Email Alert
Body	Bank Alerts
Content Type	html
From	admin@infa-server.com

- Click **Save**.

Note: Verify that you start the hmail server before you deploy the email responder.

Step 8. Create an Email Response

Create an email response to receive the alert.

1. On the **Design** tab, select **Responses**.
2. In the right pane, click **Actions > New**.
3. Configure the following properties:

Field	Value
Name	Email Response
Description	Sends Email Response

4. Under **Responder Information**, select **Email Responder**.
5. Click **Save**.

Note: Verify that you start the hmail server before you deploy the email responder.

Step 9. Create an Advanced Rule

Create a rule that uses the custom analytic to verify if the customer city name starts with a specified character.

1. On the **Design** tab, select **Rules**.
2. In the right pane, click **Actions > New > New Advanced Rule**.
3. Configure the following properties:

Field	Value
Name	Rule to send emails about unexpected bank shutdown
Description	Using custom analytic to verify if a city starts with a character
Rule Statement	when 1 customer_info c with citystartswith(c.city, "Mel") = true then "Email Response" with to="{c.email_id}", from = "admin@infa-server.com", subject = "bank closed due to bad weather", body=" Dear \${c.name} </br></br>We are sorry to inform that the Bank will be closed due to bad weather from 19th Nov, 2013 - 24th Nov, 2013. We apologise for the inconvenience. </br> </br> Regards, </br>Bank Manager"

4. Click **Save**.

Step 10. Deploy Objects

Deploy the configured objects to the corresponding services. You must start the hmail server before you deploy the email responder.

1. In the navigator, click **Actions > Rules, Sources & Responders > Deploy**.
2. Select the advanced rule, **Rule to send emails about unexpected bank shutdown**, and click **Next**.
3. Select the source, **Customer Accounts Source**, and click **Next**.
4. Select the responder, **Email Responder**, and click **Deploy**.

Step 11. View Email Alerts

You can configure the rule to include mails to more addresses that are available in hmail database. To do this, you need to configure the addresses in your mail box.

1. Open Windows Live Mail from the Start Menu.
2. If prompted for user name and password, configure the following properties:

Field	Value
User Name	student01@infa-server.com
Password	student01

3. Verify that you get a mail alert from admin@infa-server.com.

CHAPTER 18

Using the REST APIs

This chapter includes the following topics:

- [Using the REST APIs Overview, 128](#)
- [Step 1. Log In to RulePoint Using the REST Call, 129](#)
- [Step 2. Get Project ID to Create Objects in a Project, 130](#)
- [Step 3. Create a Topic, 131](#)
- [Step 4. Create a JDBC Connection, 132](#)
- [Step 5. Create an SQL Source, 132](#)
- [Step 6. Add Dynamic Schedule to the Source, 133](#)
- [Step 7. Create an RTAM Responder, 133](#)
- [Step 9. Create an RTAM Response, 134](#)
- [Step 10. Create an Advanced Rule, 135](#)
- [Step 11. Deploy the Responder, 135](#)
- [Step 12. Deploy the Rule, 136](#)
- [Step 13. Deploy the Source, 136](#)
- [Step 14. View the Objects in the Dashboard, 136](#)

Using the REST APIs Overview

In this lesson, you learn how to use API calls to perform RulePoint tasks.

Lesson Concepts

You can use REST calls to create, update, retrieve, and delete projects, objects, and users. You can also use the REST calls to perform deployment tasks. This lesson covers instructions to create and deploy objects such as topic, source, advanced rule, responder, and response. You can access the REST APIs through the following base URL, `http://host:port/rulepoint/api`.

Lesson Objectives

In this lesson, you learn to perform the following tasks:

- Login to RulePoint using the REST call.
- Get the project ID for creating objects in a project.
- Create a topic.

- Create a JDBC connection.
- Create an SQL source.
- Create a schedule for the source.
- Create an RTAM responder.
- Create an RTAM response.
- Create an advanced rule.
- Deploy the responder.
- Deploy the rule.
- Deploy the source.
- View the deployed objects in the dashboard.

Lesson Prerequisites

Before you start this lesson, you must perform the following tasks:

- You must complete the RulePoint training before you start with this lesson.
- Verify that there is a running instance of RulePoint 6.2.
- Verify that you have a RESTClient installed on the Firefox browser.
- Log in to RulePoint and create a project RESTapiTest.

Step 1. Log In to RulePoint Using the REST Call

Use the POST method to log in to RulePoint from a RESTClient application.

1. Open the RESTClient plug-in in the Firefox browser.
2. Navigate to **Headers > Custom Headers**.
3. In the **Request Header** window, configure the following properties:

Field	Value
Name	Accept
Value	application/json

4. Create another custom header. Configure the following properties:

Field	Value
Name	Content-Type
Value	application/x-www-form-urlencoded

- In the **Request Method** section, configure the following properties:

Field	Value
Method	POST
URL	http://<host>:8080/rulepoint/dtlogin

- In the **Request Body** section, configure the following property:

Field	Value
Body	j_password=Administrator1&j_username=Administrator

- Click **SEND**.
- In the **Response** section, select **Response Headers**.
Verify that you get the status code: 200 OK, which indicates that the login is successful.
- In the **Response** section, select **Response Body**.
- Save the value of the field session ID to use for further tests. For example, "sessionId": "A9C5ED4643006CE33612C252BEB4DEEE"

Step 2. Get Project ID to Create Objects in a Project

Use the GET method to retrieve the ID from a project in RulePoint.

- In the RESTClient, navigate to **Headers > Custom Headers**.
- In the **Request Header** window, configure the following properties:

Field	Value
Name	Content-Type
Value	application/json

- Create another header with the following properties:

Field	Value
Name	Cookie
Value	JSESSIONID=A9C5ED4643006CE33612C252BEB4DEEE

Note: The session ID is the value copied from the login response.

- Remove the header, Content-Type, created for login.

- In the **Request Method** section, configure the following properties:

Field	Value
Method	GET
URL	http://localhost:8080/rulepoint/api/projects

- Click **SEND**.
- In the **Response** section, select the **Response Body** and verify that you receive all the details of the projects in the RulePoint application. Copy the project RESTapiTest (for example, "id": "65536") to use for other REST calls.

Step 3. Create a Topic

Use the POST method to create a topic, customer_accounts.

- In the RESTClient **Request Method** section, configure the following properties:

Field	Value
Method	POST
URL	http://localhost:8080/rulepoint/api/projects/65536/topics

Note: 65536 is the project ID of the project RESTapiTest.

- In the **Request Body** section, enter the following value:

```
{ "name": "customer_accounts", "description": "Sql Test Topic",
  "responderAccess": "ALL_PROPERTIES", "properties": { "name": "acc_activation_date", "description": "",
  { "name": "acc_no", "description": "" }, { "name": "acc_status", "description": "" },
  { "name": "acc_type", "description": "" }, { "name": "city", "description": "" },
  { "name": "cust_id", "description": "" }, { "name": "name", "description": "" },
  { "name": "total_bal", "description": "" } ] }
```

- Click **SEND**.
- Verify that the topic is created. The Response Header shows the status as created.
- Select **Response Body**, and copy the href for future REST calls. For example, "href": "api/projects/65536/topics/0ac88e0b-f160-4359-ba62-6fea71f81d47"

Step 4. Create a JDBC Connection

Use the POST method to create a JDBC connection.

1. In the RESTClient **Request Method** section, configure the following properties:

Field	Value
Method	POST
URL	http://localhost:8080/rulepoint/api/projects/65536/connections

2. In the **Request Body** section, configure the following property:

```
{ "name": "SQL Connection", "type": "connections", "description": "A SQL Connection", "connectionType": "jdbcConnection", "config": { "driverClass": "com.informatica.jdbc.oracle.OracleDriver", "jdbcString": "jdbc:informatica:oracle://infa-server:1521;databaseName=infaorcl", "username": "rulepoint", "password": "rulepoint", "initialPoolSize": "5", "acquireIncrement": "1", "maxPoolSize": "10", "minPoolSize": "2", "loginTimeOut": "5000", "retryCount": "3", "retryDelay": "1000", "maxIdleTime": "36000", "acquireRetryAttempts": "0", "acquireRetryDelay": "5000", "checkoutTimeout": "2000" }
```

3. Click **SEND**.
4. Verify that the connection is created. The Response Header shows the status as created.
5. Select the **Response Body** tab, and copy the href for future REST calls. For example, href: "api/projects/65536/connections/5ee466ec-b79e-4bcd-9007-7adbc0f35a03"

Step 5. Create an SQL Source

Use the POST method to create an SQL source. Provide the HTTP request and response attributes for the SQL source.

1. In the RESTClient **Request Method** section, configure the following properties:

Field	Value
Method	POST
URL	http://localhost:8080/rulepoint/api/projects/65536/sources

2. In the **Request Body** section, configure the following property:

```
{ "name": "Customer Accounts Source", "valid": true, "type": "sources", "topics": [ { "rel": "topics", "href": "api/projects/65536/topics/0ac88e0b-f160-4359-ba62-6fea71f81d47", "title": "customer_accounts", "id": "0ac88e0b-f160-4359-ba62-6fea71f81d47" } ], "connection": { "rel": "connection", "href": "api/projects/65536/connections/5ee466ec-b79e-4bcd-9007-7adbc0f35a03", "title": "SQL Connection", "id": "5ee466ec-b79e-4bcd-9007-7adbc0f35a03", "sourceType": "SqlSource", "config": { "sql": "select * from customer_info c, account_info a where" }
```

```
c.cust_id=a.cust_id","propertyMetadataMap":{"\java.util.HashMap\",
{}}","parameters":"","transactionSize":0,"bufferResult":"no"},"marshaller":
{"marshallerProperties":{},"marshallerThreadsafe":false}}
```

Note: Replace the href and ID of the topic and connection as per the ID and href of your topic and connection objects. You can obtain the ID from the last section of the href value.

3. Click **SEND**.
4. Verify that the source is created. The Response Header shows the status as created.
5. Select the **Response Body** tab and copy the href for future REST calls. For example, "href": "api/projects/65536/sources/898d9474-efa0-4102-8fd3-afba72927eec"

Step 6. Add Dynamic Schedule to the Source

Use the POST method to add a schedule to the SQL source.

1. In the RESTClient **Request Method** section, configure the following properties:

Field	Value
Method	POST
URL	http://localhost:8080/rulepoint/api/projects/65536/sources/898d9474-efa0-4102-8fd3-afba72927eec/schedules

Note: "898d9474-efa0-4102-8fd3-afba72927eec" is the source ID.

2. In the **Request Body** section, enter the following value:

```
{"repeatInterval":"12000","scheduleType":"DYNAMIC"}
```
3. Click **SEND**.
4. Verify that the schedule for the source is created. The Response Header shows the status as created.

Step 7. Create an RTAM Responder

Use the POST method to create an RTAM responder.

1. In the RESTClient **Request Method** section, configure the following properties:

Field	Value
Method	POST
URL	http://localhost:8080/rulepoint/api/projects/65536/responders

2. In the **Request Body** section, enter the following value:

```
{"name": "rtamresponder", "description": "", "responderType": "RTAMResponder",
"connection": null, "valid": true, "config": {"to": "Administrator", "groups":
null, "subject": "RTAM Alert Subject", "body": "RTAM Alert Body", "actions":
null, "channels": null, "header": null, "metadata": null, "priority": "3"}, "marshaller":
{"marshallerClassname": null, "marshallerProperties": {}}
```

3. Click **SEND**.
4. Verify that the RTAM Responder is created.
The Response Header shows the status as created.
5. Select the **Response Body** tab and copy the href for future REST calls. For example, "href": "api/projects/65536/responders/3cdd2b89-139d-4cf8-81a6-d8eaf28f3dd2"

Step 9. Create an RTAM Response

Use the POST method to create an RTAM response.

1. In the RESTClient **Request Method** section, configure the following properties:

Field	Value
Method	POST
URL	http://localhost:8080/rulepoint/api/projects/65536/responses

2. In the **Request Body** section, enter the following value:

```
{"name": "rtamresponse", "valid": true, "force": false, "responder": { "rel":
"responder", "href": "api/projects/65536/responders/3cdd2b89-139d-4cf8-81a6-
d8eaf28f3dd2", "title": "rtamresponder", "id": "3cdd2b89-139d-4cf8-81a6-
d8eaf28f3dd2"}, "config": {"to": "Administrator", "body": "RTAM Alert Body", "subject":
"RTAM Alert Subject", "priority": "3", "channels": null, "header": null, "actions":
null, "groups": null, "metadata": null }}
```

Note: Replace the href and ID with the RTAM Responder href and ID that you have captured.

3. Click **SEND**.
4. Verify that the RTAM response is created. The Response Header must show the status as created.

Step 10. Create an Advanced Rule

Use the POST method to create a rule to check for customer balance below 10000 and send an RTAM response to the administrator.

1. In the RESTClient **Request Header** section, configure the following properties:

Field	Value
Method	POST
URL	http://localhost:8080/rulepoint/api/projects/65536/rules

2. In the **Request Body** section, configure the following property:

```
{"name":"Rule to check for customers with balance below 10000","description":"Rule evaluating events from a SQL Source to a RTAM Responder, \ndynamic schedule for the source","drqlStatement":"when 1 customer_accounts t1 with t1.total_bal < 10000 as Total_Balance then rtamresponse with to = \"Administrator\", body = \"Account ${acc_no} has balance less than 10000\", priority = \"2\", subject = \"Accounts with balance less than 10000\", channels = \"Account Balance\",metadata = \" \"\", \"force\":\"true\"}
```

3. Click **SEND**.
4. Verify that the advance rule is created. The Response Header shows the status as created.
5. Select the **Response Body** tab and copy the href for future REST calls. For example, "href": "api/projects/65536/rules/e24296b5-7d5f-4440-aa09-0e8716629f11"

Step 11. Deploy the Responder

Use the POST method to deploy the responder into the default responder controller.

1. In the RESTClient **Request Method** section, configure the following properties:

Field	Value
Method	POST
URL	http://localhost:8080/rulepoint/api/projects/65536/deploy

2. In the **Request Body** section, enter the following value:

```
{"topologyId":1,"itemsToDeploy":[{"artifactId":"3cdd2b89-139d-4cf8-81a6-d8eaf28f3dd2","groupNames":["Default Responder Controller"]}]}
```

Note: Replace the value of artifactId with the RTAM Responder id that you have captured.

3. Click **SEND**.
4. Verify that the RTAM responder is deployed. The Response Header shows the status as successful.

Step 12. Deploy the Rule

Use the POST method to deploy the rule into the default event processor.

1. In the RESTClient **Request Method** section, configure the following properties:

Field	Value
Method	POST
URL	http://localhost:8080/rulepoint/api/projects/65536/deploy

2. In the **Request Body** section, enter the following value:

```
{"topologyId":1,"itemsToDeploy":[{"artifactId":"e24296b5-7d5f-4440-aa09-0e8716629f11","groupNames":["Default Event Processor"]}]}
```

Note: Replace the value of artifactId with the advance rule id that you have captured.

3. Click **SEND**.
4. Verify that the advance rule is deployed. The Response Header shows the status as successful.

Step 13. Deploy the Source

Use the POST method to deploy the source into the default source controller.

1. In the RESTClient **Request Method** section, configure the following properties:

Field	Value
Method	POST
URL	http://localhost:8080/rulepoint/api/projects/65536/deploy

2. In the **Request Body** section, enter the following value:

```
{"topologyId":1,"itemsToDeploy":[{"artifactId":"898d9474-efa0-4102-8fd3-afba72927eec","groupNames":["Default Source Controller"]}]}
```

Note: Replace the value of artifactId with the SQL source id that you have captured.

3. Click **SEND**.
4. Verify that the SQL source is deployed. The Response Header shows the status as successful.

Step 14. View the Objects in the Dashboard

After you use the API calls to create and deploy the objects, view the deployed objects in the dashboard.

1. Log in to RulePoint using the login credentials.
2. Verify that the dashboard displays the created objects under the project RESTapiTest.

CHAPTER 19

Using Java Adapter for REST API

This chapter includes the following topics:

- [Using Java Adapter For REST API Overview, 137](#)
- [Step 1. Configure the BankingAdapaterDemo.java File, 138](#)
- [Step 2. Run the Code, 138](#)

Using Java Adapter For REST API Overview

In this lesson, you learn how to use the Java Adapter for REST APIs, which is a Java software module that simplifies the task of interacting with the RulePoint REST API.

Lesson Concepts

The REST adapter includes JSON parsers and generators that are compatible with the JSON schema exposed by the RulePoint REST API. The API classes provide a simple programmatic interface for accessing the RulePoint functionality.

Lesson Objectives

In this lesson, you learn to perform the following tasks:

- Configure the `BankingAdapaterDemo.java` file
- Run the code.

Lesson Prerequisites

Before you start this lesson, you must perform the following tasks:

- You must complete the RulePoint training before you start with this lesson.
- Verify that there is a running instance of RulePoint 6.2.
- You must have installed ANT on your machine.

Step 1. Configure the BankingAdapaterDemo.java File

The libraries required to run the sample code is available in the `RulePoint_Java_Adapter_6.2.zip` of the SDK pack. Contact Customer Support for the SDK pack.

1. Copy the Banking folder from `<RULEPOINT_HOME>\samples` to your work space, for example `c:\Banking`.
2. Update the path environment variable to point to `<ant_installion_home>/bin`.
3. Download the `RulePoint_Java_Adapter_6.2.zip` from the SDK pack and unzip the file.
4. Copy the `\lib` directory from the unzipped `RulePoint_Java_Adapter_6.2` file to `Banking\source\RulePoint_Java_Adapter`.
5. Open the `BankingAdapaterDemo.java` file from the following location:
`C:\Banking\source\RulePoint_Java_Adapter\src\com\informatica\rulepoint\api\samples`.
6. In the `getRulePointAPI()` method, edit the RulePoint URL to reflect the URL of RulePoint that you have installed.
7. In the `createSqlConnection(ProjectContext projectContext)` method, edit the values to reflect the valid database URL , driver class, database user name, and password.
8. Save the file.

Step 2. Run the Code

Create a project in RulePoint, and then run the Java code. You can view the created objects and their deployment status in RulePoint.

1. Log in to RulePoint at: `http://host:port/rulepoint` using the administrator credentials, Administrator/Administrator1.
2. Create a project named EnablementAdapterProj.
 - a. In the navigator, click **Actions > New**.
 - b. In the **Name** field, enter `EnablementAdapterProj`, and click **Save**.
3. Open command prompt and navigate to `C:\Banking`.
4. Type the following command:

```
>ant adapter
```

A message appears indicating that the build is successful.

5. Verify that the objects are created and deployed in RulePoint.

The following image shows the status of the deployed objects in the **All Objects** view on the **Design** tab:

The screenshot displays the Informatica RulePoint interface. The top navigation bar includes 'Design', 'Administration', and 'Dashboard'. The main content area is titled 'All Objects' and shows a list of objects under the 'EnablementAdapterProj' project. The objects are listed in a table with columns for Name, Type, Last Modified Date, State, and Validity. The 'system_markers' object is highlighted in green.

Name	Type	Last Modified Date	State	Validity
system_markers	topics	18/12/2013 11:38:26 AM	DRAFT	true
Cities in Florida	watchlists	18/12/2013 11:39:09 AM	DRAFT	true
RTAM Responder	responders	18/12/2013 11:43:04 AM	DEPLOYED	true
Rule to check for customers with balance below 10000	rules	18/12/2013 11:43:04 AM	DEPLOYED	true
RTAM Response	responses	18/12/2013 11:43:04 AM	DEPLOYED	true
customer_accounts	topics	18/12/2013 11:43:04 AM	DEPLOYED	true
SQL Connection	connections	18/12/2013 11:43:11 AM	DEPLOYED	true
Customer Accounts Source	sources	18/12/2013 12:40:06 PM	DEPLOYED	true

INDEX

A

- add node
 - high availability [109](#)
- alert credit card rejections
 - overview [72](#)
- alert customer cash withdrawals
 - overview [60](#)
- alert customer details
 - overview [19](#)
- alert server downtime
 - overview [116](#)

B

- banking
 - data model [13](#)
 - scenario [11](#)
- before you begin
 - overview [14](#)

C

- check customer cash withdrawal for a day
 - overview [69](#)
- create advanced rule
 - custom analytic [126](#)
 - custom service [119](#)
- create alert
 - email alert [126](#)
- create analytic
 - cust_get_account_type [57](#)
 - custom analytic [124](#)
- create connection
 - email [36](#)
 - email connection [124](#)
 - JMS [43](#)
 - SQL [22](#), [122](#)
 - web service [34](#)
- create deployment policy
 - rule [82](#)
- create event processor
 - high availability [110](#)
- create jar file
 - custom service [116](#)
- create Java project
 - Java Adapter for REST API [138](#)
- create project
 - banking [20](#)
- create responder
 - email [36](#)
 - email responder [125](#)
 - JMS [45](#)
 - RTAM [25](#), [118](#)

- create responder (*continued*)
 - SQL [74](#)
- create response
 - email [37](#)
 - JMS [46](#)
 - RTAM [26](#), [119](#)
- create role
 - banking [102](#)
- create rule
 - advanced [70](#)
 - advanced rule [46](#), [53](#), [58](#), [63](#), [74](#)
 - template rule [81](#), [90](#)
 - wizard [38](#)
 - wizard rule [27](#)
- create schedule
 - SQL source [52](#), [63](#), [123](#)
 - stock source [35](#)
- create source
 - custom source [117](#)
 - JMS [44](#)
 - SQL [23](#)
 - SQL source [51](#), [62](#), [123](#)
 - web service source [35](#)
- create template
 - rule [79](#)
 - wizard rule [88](#)
- create topic
 - amount_withdrawn [61](#)
 - credit_card_transaction [42](#)
 - customer_accounts [21](#)
 - customer_info [51](#), [122](#)
 - port_monitor [117](#)
 - stock [34](#)
- create user
 - banking [102](#)
- create watchlist
 - cities in Florida [52](#)
- creating database tables
 - Oracle [15](#)
- credit card transactions
 - overview [41](#)
- custom services
 - overview [115](#)

D

- dashboard tasks
 - overview [92](#)
- database tables
 - create [15](#)
 - property definitions [15](#)
- deploy objects
 - custom services [119](#)
- deployed responses
 - dashboard tasks [97](#)

- deployed rules
 - dashboard tasks [96](#)
- deployed sources
 - dashboard tasks [93](#)
- deployed topics
 - dashboard tasks [94](#)

E

- edit role
 - banking [103](#)
- export
 - selected objects [105](#)

G

- GET method
 - get project ID [130](#)

I

- import and export
 - overview [105](#)
- import and upload XML file
 - RulePoint objects [107](#)

L

- login
 - banking user [103](#)

M

- monitor balance threshold
 - overview [77](#)

N

- notify stock prices
 - overview [32](#)
- notify unexpected bank shutdown
 - overview [120](#)
- notifying free health checkup
 - overview [49](#)

P

- POST method
 - add dynamic schedule [133](#)
 - create advanced rule [135](#)
 - create RTAM responder [133](#)
 - create RTAM response [134](#)
 - create topic [131](#)
 - deploy responder [135](#)
 - deploy rule [136](#)
 - deploy source [136](#)
 - JDBC connection [132](#)
 - log in [129](#)
 - SQL source [132](#)
- prerequisites [17](#)

- problem
 - overview [12](#)
- provide permissions
 - project level [103](#)
- purge
 - application service [98](#)
 - topology [99](#)

R

- reassign rule
 - high availability [113](#)
- REST APIs
 - overview [128](#)
- retrieve customer details
 - overview [55](#)
- revert history
 - template [87](#)
- RulePoint solution
 - banking [12](#)
- run Java code
 - Java Adapter for REST API [138](#)

S

- set timeline filter
 - dashboard tasks [98](#)
- set up high availability
 - overview [108](#)

U

- update mapping
 - deployment policy [112](#)
- update rule
 - template [86](#)
- upgrade
 - template [83](#), [85](#)
- use cases
 - overview [12](#)
- users and roles
 - overview [101](#)
- using Java Adapter for REST API
 - overview [137](#)

V

- verify settings
 - high availability [111](#)
- view alert
 - custom source [120](#)
 - RTAM [58](#), [64](#), [71](#), [76](#), [83](#), [87](#), [91](#)
- view errors
 - dashboard [99](#)
- view messages
 - JMS [75](#)
- view objects
 - REST calls [136](#)
- view response
 - email [40](#), [54](#)
 - JMS [48](#)

W

wizard rule

check for customer accounts [27](#)