

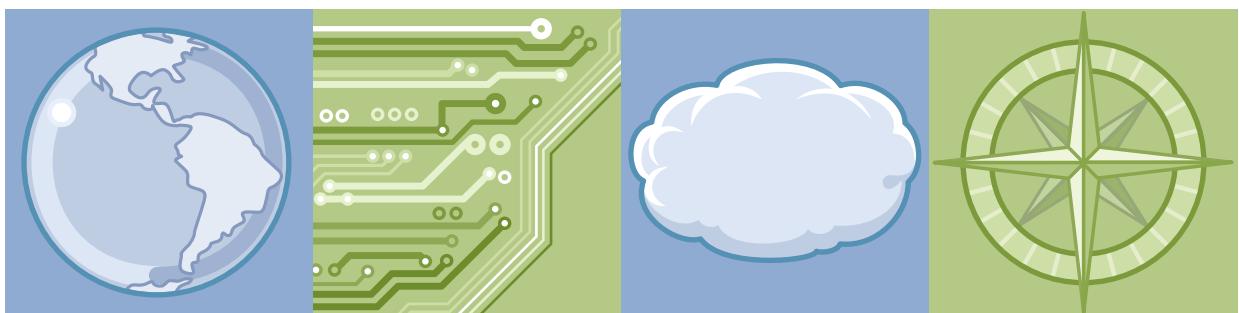


IBM Training

Student Exercises

Developing Rule Solutions in IBM Operational Decision Manager V8.8

Course code WB395 ERC 1.2



IBM Cloud
Middleware

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

CICS®	developerWorks®	Express®
ILOG®	Insight™	Insights™
Orchestrate®	PartnerWorld®	Redbooks®
SPSS®	Tivoli®	WebSphere®
z/OS®		

Intel is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

March 2016 edition

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Contents

Trademarks	ix
Exercises description	xi
General exercise information.....	xv
Exercise 1. Operational Decision Manager in action	1-1
Section 1. Running the Miniloan web application	1-4
1.1. Setting up your environment	1-4
1.2. Running the application	1-5
Section 2. Managing business rules in Decision Center	1-8
2.1. Editing rules in Decision Center Business console	1-8
2.2. Reviewing the rule history	1-14
2.3. Changing the credit score requirements	1-15
Section 3. Validating rule changes	1-19
Section 4. Deploying updated rules from Decision Center to Rule Execution Server	1-21
4.1. Deploying the decision service	1-21
Section 5. Monitoring the deployed rules in Rule Execution Server	1-24
Section 6. Viewing the effects in the Miniloan application	1-28
Section 7. Exploring the development environments in Rule Designer	1-29
7.1. Opening Rule Designer	1-29
7.2. Taking a quick tour of Rule Designer	1-31
Section 8. Shutting down the modules	1-33
Exercise 2. Setting up decision services	2-1
Section 1. Creating a standard rule project	2-3
1.1. Setting up your environment for this exercise	2-3
1.2. Creating a standard rule project with a BOM	2-4
Section 2. Importing the XOM and setting up the BOM	2-6
2.1. Importing the execution object model (XOM)	2-6
2.2. Creating a reference to the XOM in the rule project	2-9
2.3. Creating the BOM	2-10
2.4. Exploring the BOM	2-13
Section 3. Creating the main rule project for the decision service	2-16
3.1. Identifying the next tasks in decision service development	2-18
Section 4. Creating and defining the decision operation	2-20
4.1. Creating the decision operation	2-20
4.2. Creating ruleset variables for ruleset parameters	2-22
4.3. Binding the variables to ruleset parameters	2-24
4.4. Creating a local ruleset variable	2-25
Section 5. Creating rule packages	2-27
Section 6. Publishing from Rule Designer to Decision Center	2-29
6.1. Publishing the rule project to Decision Center	2-29
6.2. Opening the project in the Decision Center Business console	2-32
6.3. Opening the project in the Decision Center Enterprise console	2-34
6.4. Modifying the ruleset variable in Decision Center	2-38
6.5. Synchronizing Rule Designer with Decision Center	2-38
Section 7. Deleting a decision service from Decision Center	2-41

Section 8. Importing a decision service from Decision Center	2-42
8.1. Creating a project in Rule Designer from Decision Center	2-42
8.2. Finalizing the rule project in Rule Designer	2-44
Exercise 3. Working with the BOM	3-1
Section 1. Finalizing the XOM	3-2
1.1. Setting up your environment for this exercise	3-2
1.2. Understanding the problem	3-4
1.3. Finish implementing the Borrower class	3-5
1.4. Finish implementing the Test class	3-10
Section 2. Creating a rule project with a BOM	3-12
Section 3. Working with the vocabulary that is required to author rules	3-14
3.1. Creating the default vocabulary	3-14
3.2. Examining the verbalization for members of the Borrower class	3-16
3.3. Editing the default verbalization	3-19
Exercise 4. Refactoring	4-1
Section 1. Refactoring rule artifacts after vocabulary changes	4-2
1.1. Setting up your environment for this exercise	4-2
1.2. Exploring the default verbalization	4-3
1.3. Changing the default verbalization of the getBankruptcyAge method	4-5
1.4. Changing the default verbalization of the duration member	4-6
Section 2. Maintaining consistency between the BOM and the XOM	4-8
2.1. Adding a XOM class that is missing in the BOM	4-8
2.2. Adding a XOM method that is missing in the BOM	4-10
2.3. Creating a rule to use this new attribute	4-11
2.4. Deprecating BOM members	4-12
Exercise 5. Working with ruleflows	5-1
Section 1. Exploring a ruleflow diagram	5-2
1.1. Setting up your environment for this exercise	5-2
1.2. Exploring the ruleflow	5-3
1.3. Exploring action tasks and transitions	5-7
1.4. Using ruleset parameters and variables in rules	5-10
Section 2. Creating a ruleflow	5-12
Section 3. Defining a main ruleflow	5-16
Exercise 6. Exploring action rules	6-1
Section 1. Exploring rule structure	6-2
1.1. Setting up your environment for this exercise	6-2
1.2. Exploring the rules	6-3
Section 2. Exploring rule behavior	6-5
Section 3. Working with automatic variables	6-8
Exercise 7. Authoring action rules	7-1
Section 1. Authoring action rules in the Intellirule editor	7-2
1.1. Setting up your environment for this exercise	7-2
1.2. Authoring rules with the Intellirule editor	7-4
Section 2. Authoring actions rules in the Guided editor	7-9
Section 3. Authoring rules with ruleset parameters	7-11
Section 4. Creating an action rule template	7-15

4.1. Creating the template	7-15
Section 5. Authoring action rules from your action rule template	7-17
5.1. Publishing the BOM and the rules to Decision Center	7-17
5.2. Authoring the rule in Decision Center	7-17
5.3. Deleting projects	7-19
Exercise 8. Authoring decision tables and decision trees	8-1
Section 1. Authoring a decision table	8-2
1.1. Setting up your environment for this exercise	8-2
1.2. Creating the table	8-3
1.3. Completing the table cells with values	8-9
Section 2. Authoring a decision tree	8-17
2.1. Creating the tree	8-17
Exercise 9. Working with static domains	9-1
Section 1. Exploring a collection domain	9-2
1.1. Setting up your environment for this exercise	9-2
1.2. Exploring the domain	9-3
1.3. Testing the rule	9-5
Section 2. Working with an enumeration of literals	9-9
2.1. Creating the domain	9-9
2.2. Testing the domain in a rule	9-12
Section 3. Defining an enumeration of static references	9-14
3.1. Creating the static references Java class	9-14
3.2. Authoring an action rule that uses a domain of static references	9-21
Exercise 10. Working with dynamic domains	10-1
Section 1. Creating a dynamic domain in Rule Designer	10-3
1.1. Setting up your environment for this exercise	10-3
1.2. Creating the domain	10-4
1.3. Examining the dynamic domain in Rule Designer	10-13
Section 2. Using the dynamic domain in a rule	10-15
Section 3. Updating the dynamic domain	10-20
Section 4. Updating the XOM	10-24
Section 5. Publishing the BOM and rule projects to Decision Center	10-25
5.1. Publishing the decision service	10-25
Section 6. Examining rules in Decision Center	10-27
6.1. Opening the published projects in Decision Center	10-27
6.2. Creating a Resources smart folder	10-30
Section 7. Modifying the dynamic domain in Decision Center	10-32
7.1. Modifying the values of the dynamic domain	10-32
7.2. Modifying the domain value in the rule	10-33
Section 8. Updating Rule Designer from Decision Center	10-36
8.1. Synchronizing the rule project	10-36
Exercise 11. Working with queries	11-1
Section 1. Searching for rule artifacts	11-2
1.1. Setting up your environment for this exercise	11-2
1.2. Searching for specific criteria across the rules	11-2
1.3. Searching for dependencies between rules	11-4
Section 2. Querying rule projects	11-7

2.1. Searching for rules that may become applicable	11-7
2.2. Searching for rules that may lead to a state	11-9
2.3. Searching for rules that use a BOM member	11-10
2.4. Applying actions with queries	11-11
Exercise 12. Executing rules locally	12-1
Section 1. Executing rules locally by using a launch configuration	12-2
1.1. Setting up your environment for this exercise	12-2
1.2. Creating a launch configuration	12-2
1.3. Debugging with a launch configuration	12-6
Exercise 13. Debugging a ruleset	13-1
Section 1. Running the debug launch configuration	13-2
1.1. Setting up your environment for this exercise	13-2
1.2. Running the debug configuration	13-3
1.3. Setting breakpoints in the rules	13-4
Section 2. Debugging with breakpoints	13-6
2.1. Inspecting the agenda	13-6
Section 3. Debugging with breakpoints in the ruleflow	13-8
Section 4. Resolving the problem	13-10
Exercise 14. Enabling tests and simulations	14-1
Section 1. Validating the rule project	14-3
1.1. Setting up your environment for this exercise	14-3
1.2. Validating the rule project	14-3
1.3. Choosing a constructor	14-4
1.4. Editing argument names for column headings in the template	14-6
Section 2. Generating the scenario file template	14-8
Section 3. Populating the template and testing the scenario	14-11
3.1. Viewing the prepopulated scenario file	14-11
3.2. Running the tests with the populated scenario file	14-11
Section 4. Customizing input for the scenario file template	14-13
4.1. Removing columns from the template	14-13
4.2. Creating a virtual attribute to test complex objects	14-14
4.3. Regenerating the scenario file template	14-16
4.4. Testing the customized scenario file	14-17
Section 5. Enabling testing from Business console	14-19
Section 6. Testing from Business console	14-22
Exercise 15. Managing deployment	15-1
Section 1. Creating deployment configurations	15-2
1.1. Setting up your environment for this exercise	15-2
1.2. Creating a deployment configuration	15-2
Section 2. Deploying a RuleApp from Rule Designer	15-6
2.1. Deploying the RuleApp	15-6
Section 3. Adding a ruleset property to a deployment configuration	15-9
Section 4. Deploying the managed XOM from Rule Designer	15-11
4.1. Deploying the managed XOM from the rule project	15-11
4.2. Deploying a RuleApp associated with a managed XOM	15-13
Section 5. Exporting and importing deployment server definitions	15-15
5.1. Exporting deployment configurations	15-15

5.2. Importing deployment configurations	15-16
Exercise 16. Exploring the Rule Execution Server console	16-1
Section 1. Exploring the Rule Execution Server console	16-2
1.1. Setting up your environment for this exercise	16-2
1.2. Exploring the Rule Execution Server console pages	16-3
Section 2. Exploring the deployed RuleApps	16-5
Section 3. Working with deployed resources	16-7
Section 4. Exploring the Diagnostics and Server Info tabs	16-8
Section 5. Exploring the REST API tab	16-9
Section 6. Managing RuleApps and rulesets	16-10
6.1. Creating a RuleApp	16-10
6.2. Adding a ruleset to your RuleApp	16-10
6.3. Adding a managed XOM to your ruleset	16-13
6.4. Adding a ruleset property to the ruleset	16-14
6.5. Deleting the RuleApp	16-14
Section 7. Testing a deployed ruleset	16-16
7.1. Testing the ruleset	16-16
Exercise 17. Executing rules with Rule Execution Server in Java EE	17-1
Section 1. Building the web application	17-3
1.1. Setting up your environment for this exercise	17-3
1.2. Finalizing the client application code to call your rules for execution	17-4
1.3. Building the application for deployment	17-5
Section 2. Deploying the web application to WebSphere Application Server	17-7
2.1. Deploying the web application	17-7
2.2. Verifying application deployment to the application server	17-7
Section 3. Executing the web application with the RuleApp deployed	17-9
Exercise 18. Executing rules as a hosted transparent decision service (HTDS)	18-1
Section 1. Deploying the decision service to Rule Execution Server	18-2
1.1. Setting up your environment for this exercise	18-2
1.2. Deploying the RuleApp	18-2
Section 2. Getting the HTDS WSDL file from the deployed ruleset	18-5
2.1. Opening the WSDL in Rule Designer	18-7
Exercise 19. Auditing ruleset execution through Decision Warehouse	19-1
Section 1. Enabling ruleset monitoring	19-3
1.1. Setting up your environment for this exercise	19-3
1.2. Define the target server	19-4
1.3. Enabling monitoring	19-4
1.4. Deploying the updated RuleApp from Rule Designer	19-6
1.5. Running the client application	19-6
Section 2. Retrieving decision traces in the Rule Execution Server console	19-7
Section 3. Optimizing Decision Warehouse	19-10
3.1. Working with monitoring options	19-10
3.2. Specifying filters on the trace data	19-11
3.3. Removing BOM serialization	19-12
Section 4. Deleting trace information from the database	19-14
Exercise 20. Working with the REST API	20-1

Section 1. Using REST services to test ruleset execution	20-2
Section 2. Using REST services to deploy and execute rules	20-4
2.1. Setting up your environment for this exercise	20-4
2.2. Deploying the RuleApp resources	20-6
2.3. Testing ruleset execution	20-7
2.4. Viewing execution traces in Decision Warehouse	20-8
2.5. Cleaning the scenario	20-8

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

CICS®	developerWorks®	Express®
ILOG®	Insight™	Insights™
Orchestrate®	PartnerWorld®	Redbooks®
SPSS®	Tivoli®	WebSphere®
z/OS®		

Intel is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

Exercises description

Exercise objectives

After completing the exercises, students should be able to:

- Design a business rule solution that is based on decision services, including the implementation model (or XOM) for rule execution, and the business version of that model (or BOM) for rule authoring
- Orchestrate the flow of rule execution through ruleflows and by understanding rule engine algorithms
- Support business rule authors by setting up the rule authoring environment, collaborating on rule vocabulary changes, and maintaining synchronization between the business and development environments
- Set up the test and simulation environment for business users who work in Decision Center
- Package and deploy rule artifacts and the XOM to a managed execution environment in Rule Execution Server
- Use launch configurations to run and debug the rulesets
- Create client applications that request ruleset execution with Rule Execution Server in Java SE or in Java EE
- Deploy business rules as a web service, or hosted transparent decision service (HTDS), in Rule Execution Server
- Test and monitor ruleset execution, and audit decision traces

Exercise list

This course includes the following exercises:

- Exercise 1, "Operational Decision Manager in action"
- Exercise 2, "Setting up decision services"
- Exercise 3, "Working with the BOM"
- Exercise 4, "Refactoring"
- Exercise 5, "Working with ruleflows"
- Exercise 6, "Exploring action rules"
- Exercise 7, "Authoring action rules"
- Exercise 8, "Authoring decision tables and decision trees"
- Exercise 9, "Working with static domains"
- Exercise 10, "Working with dynamic domains"
- Exercise 11, "Working with queries"

- Exercise 12, "Executing rules locally"
- Exercise 13, "Debugging a ruleset"
- Exercise 14, "Enabling tests and simulations"
- Exercise 15, "Managing deployment"
- Exercise 16, "Exploring the Rule Execution Server console"
- Exercise 17, "Executing rules with Rule Execution Server in Java EE"
- Exercise 18, "Executing rules as a hosted transparent decision service (HTDS)"
- Exercise 19, "Auditing ruleset execution through Decision Warehouse"
- Exercise 20, "Working with the REST API"

Exercise structure

The exercises can be categorized into these groups:

Overview

Exercise 1, "Operational Decision Manager in action"

The first exercise is an overview of Operational Decision Manager components for business rules, including Decision Center, Rule Execution Server, and Rule Designer. This lab sets the stage for all the other labs by introducing you to the workflow between ODM modules. During the course, you work mainly with Rule Designer and Rule Execution Server console, but you also do some exercises with the Decision Center tools.

Enabling business rule authoring and governance

- Exercise 2, "Setting up decision services"
- Exercise 14, "Enabling tests and simulations"

Rule authoring

- Exercise 5, "Working with ruleflows"
- Exercise 6, "Exploring action rules"
- Exercise 7, "Authoring action rules"
- Exercise 8, "Authoring decision tables and decision trees"

Configuring rule vocabulary

- Exercise 3, "Working with the BOM"
- Exercise 4, "Refactoring"
- Exercise 9, "Working with static domains"
- Exercise 10, "Working with dynamic domains"

Deployment and ruleset execution

- Exercise 11, "Working with queries"
- Exercise 12, "Executing rules locally"
- Exercise 13, "Debugging a ruleset"
- Exercise 15, "Managing deployment"
- Exercise 16, "Exploring the Rule Execution Server console"
- Exercise 17, "Executing rules with Rule Execution Server in Java EE"
- Exercise 18, "Executing rules as a hosted transparent decision service (HTDS)"
- Exercise 19, "Auditing ruleset execution through Decision Warehouse"
- Exercise 20, "Working with the REST API"

Most of the exercises are independent and do not require you to complete a previous exercise. Exercises that you do in Rule Designer have “start” files and “answer” (or “solution”) files. You can use the “answer” files when you run out of time or are unable to complete the exercise correctly. If necessary, the start and answer files make it possible to skip exercises or do them out of sequence. However, you are encouraged to complete them in the order in which they are presented. In the exercise instructions, you can check off the line before each step as you complete it to track your progress.

General exercise information

This section provides general information about the exercises in this course. Review this section before starting the exercises.

User IDs and passwords

The following table lists user ID and password information for this course.

Entry point	User ID	Password
VMware image	administrator	websphere
Windows 2008 R2	administrator	websphere
Single-sign-on ID for ODM installation and user ID for WebSphere Application Server and Decision Server	odm	odm
Decision Center administrator	rtsAdmin	rtsAdmin
Decision Center business user	rtsUser1	rtsUser1
Decision Center configuration user	rtsConfig	rtsConfig
Decision Server administrator	resAdmin	resAdmin
Business console manager user	Paul	Paul
Business console rule author user	Bea	Bea
Business console test specialist user	Abu	Abu

How to follow the exercise instructions

Exercise structure

Each exercise is divided into sections with a series of numbered steps and lettered substeps:

- The numbered steps (1, 2, 3) represent actions to be done.
- The lettered substeps (a, b, c) provide detailed guidance on how to complete the action.



Information

If you already understand how to do the action in the numbered step, you can skip the specific guidance in the lettered substeps.

The following example comes from Exercise 1 of this course.



Example

Excerpt from Exercise 1

- 1. Edit the rule and change the debt-to-income ratio from 0.3 to 0.5.
 - a. Click **Edit** to open the rule editor.
 - b. In the **if** part of the rule, change `0.3` to: `0.5`

In this example, the numbered instructions prompt you to edit a rule. The “a” and “b” substeps provide specific guidance on how to edit the rule.

Text highlighting in exercises

Different text styles indicate various elements in the exercises.

Words that are highlighted in **bold** represent GUI items that you interact with, such as:

- Menu items
- Field names
- Icons

Words that are highlighted with a `fixed font` include the following items:

- Text that you type or enter as a value
- System messages
- Directory paths
- Code

Tracking your progress

As shown in the example step, you can see that an underscore precedes each numbered step and lettered substep.

You are encouraged to use these markers to track your progress by checking off each step as you complete it. Tracking your progress in this manner might be useful if you are interrupted while working on an exercise.

Required exercise sections

Most exercises include required sections that should always be completed. It might be necessary to complete these sections before you can start subsequent exercises.

Dependencies between exercises are listed in the exercise introduction.

Optional exercise sections

Some exercises might also include optional sections that you can complete when you have sufficient time and want an extra challenge.

File references for exercises

Exercise steps contain references to files or projects to open or import. Two directories are used in these references:

- <*InstallDir*>: This directory is the IBM Operational Decision Manager installation directory.
- <*LabfilesDir*>: This directory contains the files that are required during demonstrations, exercises, and the workshop steps, such as samples of code that you can copy and paste.

If you are using the VMware image that is provided with this course:

- <*InstallDir*> is: C:\Program Files\IBM\ODM88

This folder is the default IBM Operational Decision Manager installation directory on Windows.

- <*LabfilesDir*> is: C:\labfiles

If you are not using the VMware image that is provided with this course, ask the installer of your environment, or your instructor, where to find the <*InstallDir*> and <*LabfilesDir*> directories.



Stop

Make sure that you identify the <*InstallDir*> and <*LabfilesDir*> directories before you proceed with the exercises in this course.

Projects for exercises

Most of the exercises for this course are done in Rule Designer, which uses the Rule perspective of Eclipse.

The exercise projects are provided for you to import into Eclipse by using the Import wizard or the Samples Console perspective. For most of the exercises, you use the Samples console perspective, as described here.

To open Rule Designer, you click **Start > All Programs > IBM > Operational Decision Manager V8.8 > Rule Designer** (or you can use the shortcut on the desktop).

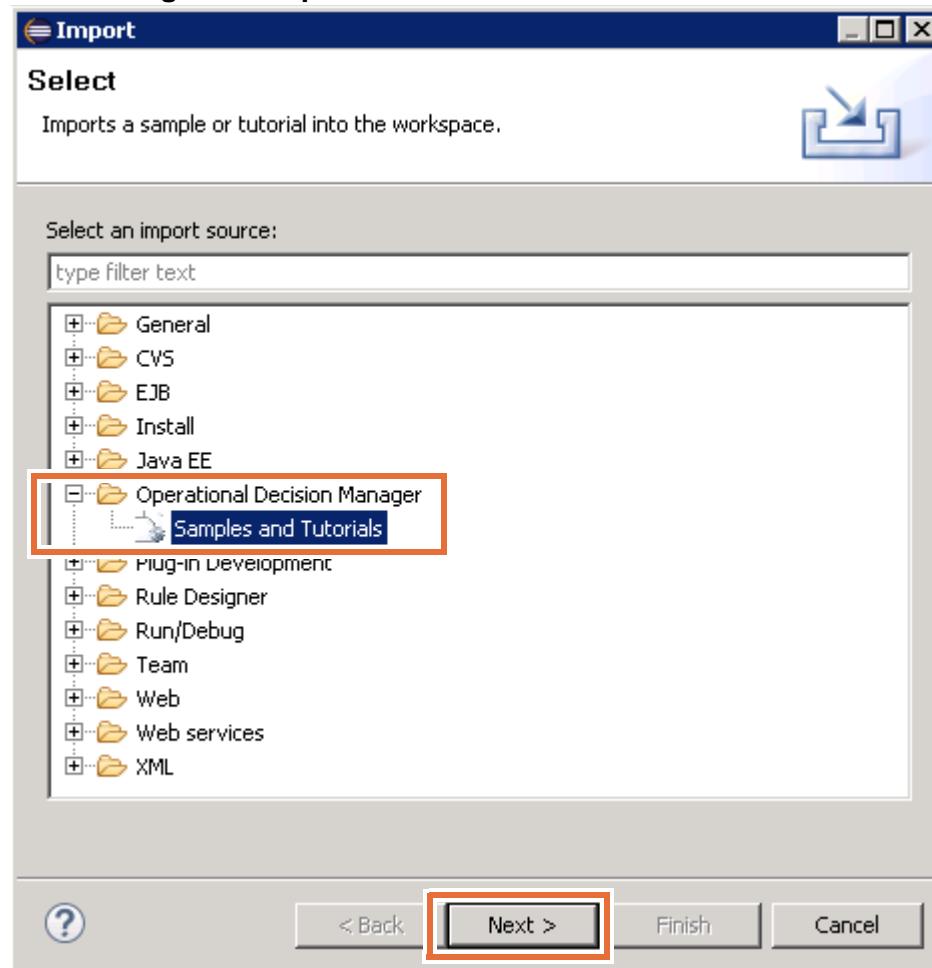


When prompted for a workspace, you can type the path directly in the Workspace Launcher, for example:

C:\labfiles\workspaces\myWorkspace

When you type a path, an empty workspace is created in the Rule perspective.

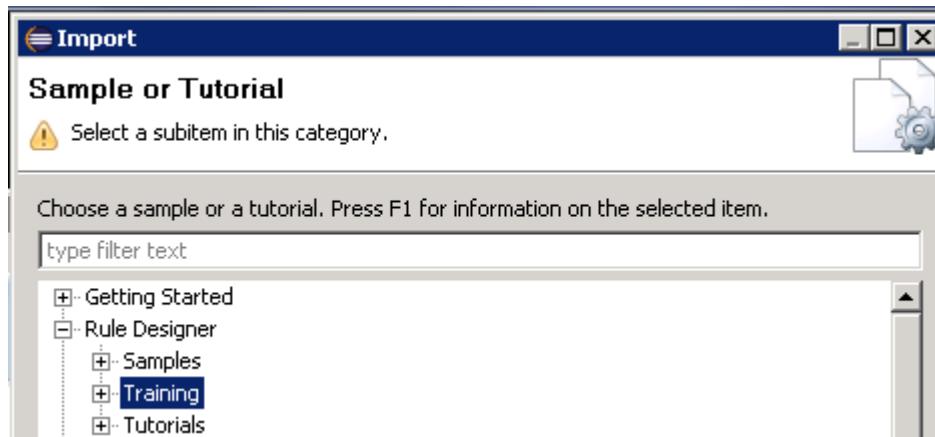
To start an exercise, you use the Samples and Tutorials import wizard from the **File > Import** menu. In the Import wizard, you select **Operational Decision Manager > Samples and Tutorials**.



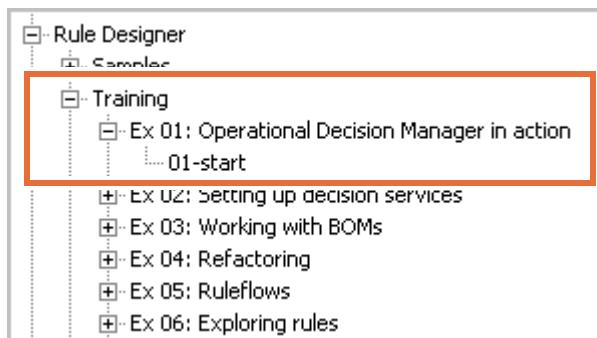
On the "Sample or Tutorial" page, you expand the **Rule Designer > Training** node to see a list of all exercises for this course.

**Hint**

You can collapse the **Rule Designer** node and then expand it to simplify the list.



To import an exercise, you expand the exercise name and select the start project. For example, expand **Training > Ex 01: Operational Decision Manager in action** and click **01-start**.



Both “start” and “solution” projects are provided for most exercises.

When you import the projects, Eclipse automatically switches to the Rule perspective. To give yourself more area to work in, you can close the Help view by clicking the X.



Sample server port

This course uses the default installation of Operational Decision Manager where Decision Center and Rule Execution Server are hosted on the sample server of Operational Decision Manager.

With the default installation, you use the following URLs to access the console of these two modules through a web browser:

- `http://localhost:PORT/decisioncenter`: to access the Decision Center Business console
- `http://localhost:PORT/teamserver`: to access the Decision Center Enterprise console
- `http://localhost:PORT/res`: to access the Rule Execution Server console

PORT is the required port. The port might be different in your environment.

If you are using the **VMware** image that is provided with this course:

- The value of *PORT* is: **9080**

This value is the default port with the default installation of Operational Decision Manager. This course assumes that this default value of *PORT* is used, and uses the following URLs:

- `http://localhost:9080/decisioncenter`: to access the Decision Center Business console
- `http://localhost:9080/teamserver`: to access the Decision Center Enterprise console
- `http://localhost:9080/res`: to access the Rule Execution Server console

If you are not using the **VMware** image that is provided with this course, find the value of *PORT* for your installation:

- a. Open the profile log folder in the installation directory for WebSphere Application Server. The default path to the profile log is:

`C:\Program Files\IBM\WebSphere\AppServer\profiles\ODMSample8800\logs`

- b. Open the file `AboutThisProfile.txt` with any text editor.
- c. Find the value of *PORT* at the end of the line that starts with:

`HTTP transport port:`



Stop

If you are not using the VMware image that is provided with this course, make sure that you identify the value of *PORT* before you proceed with the exercises in this course.

Using the product documentation

The product documentation is installed locally on the VMware image that is provided with this course.

To access the local documentation while working in Rule Designer, you must first start it by clicking **Start > All Programs > IBM > Operational Decision**

Manager V8.8 > Information Center for Operational Decision Manager (local).

You can also access the product documentation from a web browser at this web address:

www.ibm.com/support/knowledgecenter/SSQP76_8.8.0

**Information**

If you are not using the VMware image that is provided with this course, you must either install the local help as described in the product documentation, or use the online version.

Exercise 1. Operational Decision Manager in action

What this exercise is about

In this exercise, you see how the Operational Decision Manager modules work together to provide a comprehensive Business Rule Management System (BRMS) across the business and development environments.

What you should be able to do

After completing this exercise, you should be able to:

- Explain the general workflow in Operational Decision Manager for working with business rule projects
- Identify the Operational Decision Manager tools that apply to your role in your organization

Introduction

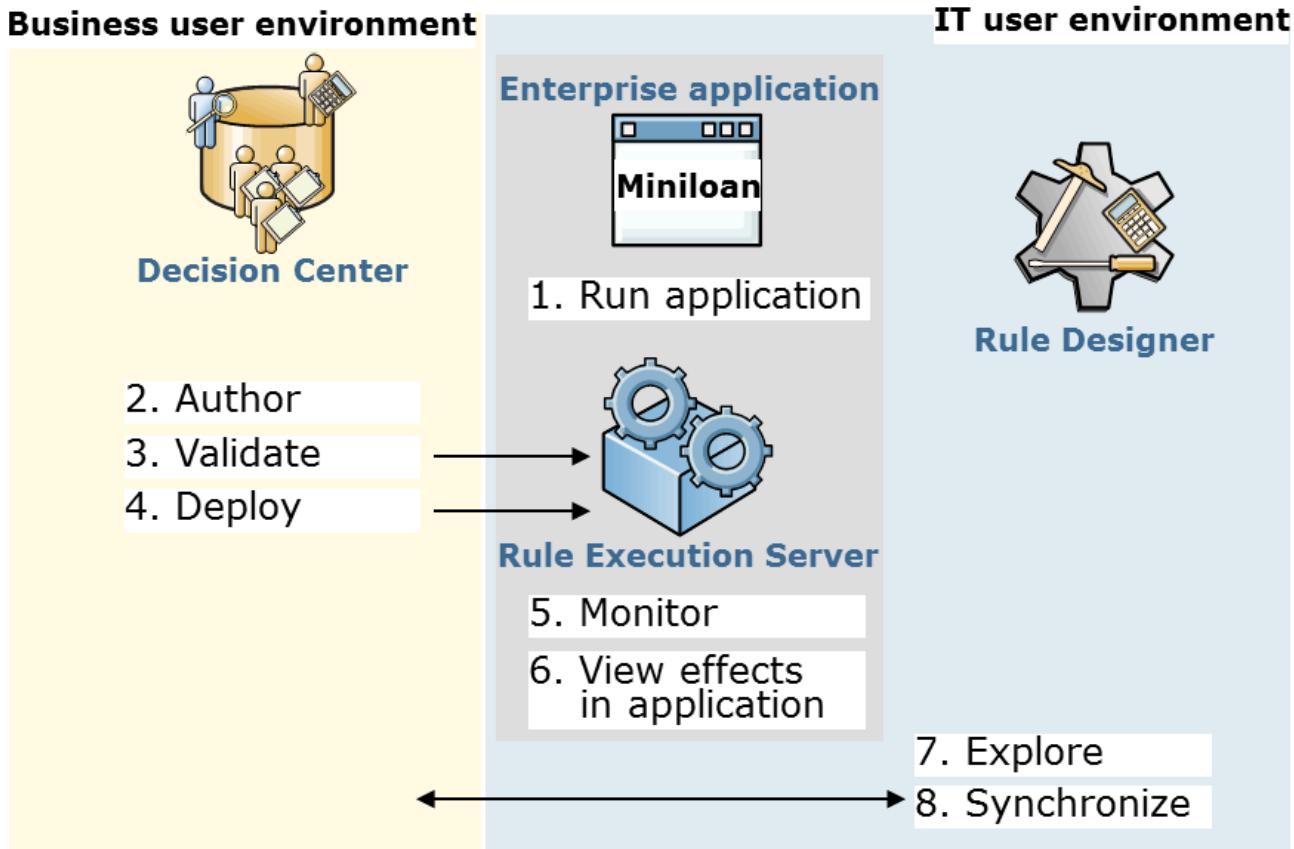
The exercise is based on a fictional financial institution, Miniloan, which provides online quotations for loan requests.

To see how Operational Decision Manager facilitates collaboration between business and IT teams, you take on business and technical roles to perform these tasks:

- "Running the Miniloan web application"
- "Managing business rules in Decision Center"
- "Validating rule changes"
- "Deploying updated rules from Decision Center to Rule Execution Server"
- "Monitoring the deployed rules in Rule Execution Server"
- "Viewing the effects in the Miniloan application"
- "Exploring the development environments in Rule Designer"
- "Shutting down the modules"

This exercise provides you with an introduction to the different Operational Decision Manager modules, and how they work together. Both IT and business tasks are included in this exercise to demonstrate the collaboration that occurs between the technical and business organizations.

The exercise workflow is shown here.



You start by running the Miniloan enterprise application. Then, you work in the business user environment by using Decision Center to author, validate, and deploy rules. Finally, you move to the IT user environment to monitor the rule changes, see how the changes affect the application, and synchronize the business and IT environments.

By completing tasks in the business and IT environments, you can see how both business and IT work together within Operational Decision Manager to manage business rules.

Requirements

This exercise has no specific requirements.



Important

The exercises in this course use a set of lab files that are installed in the product installation directory:

```
<InstallDir>\studio\training
```

The default directory for `<InstallDir>` is `C:\Program Files\IBM\ODM88`. If you are not using the provided lab environment for this course, make sure that you know the location of `<InstallDir>`.

Additional lab support files are stored in the C:\labfiles directory.

The exercises point you to the lab files as you need them.



Note

For IBM ODM on Cloud users

This exercise requires that you use the on-premises version of ODM. The exercise uses features that are not supported in IBM ODM on Cloud.

- Sample server: The sample server is provided with the on-premises version of ODM only.
- Miniloan web application: The Miniloan web application is run on the sample server. IBM ODM on Cloud does not include the sample server. However, you can deploy and run the Miniloan web application on an application server that you set up. For information about running the Miniloan web application, see the IBM ODM on Cloud documentation.

Section 1. Running the Miniloan web application

Before you explore the tools, you first look at how rules affect the user application.

Scenario

Joe is planning to buy a house and wants a loan for \$500,000. To find out whether he is eligible, he goes to the Miniloan website.

1.1. Setting up your environment

This exercise uses the sample server that is provided with Operational Decision Manager.

For IBM ODM on Cloud users

To set up your environment in ODM on Cloud, follow the directions in the “First steps in IBM ODM on Cloud” tutorial in the IBM Knowledge Center for IBM ODM on Cloud.

In the “First steps” tutorial, you work with the `MiniloanService` decision service and install the `miniloan-webapp` web application.

- 1. Start the sample server.
 - a. Click **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Start server** (or you can use the **Start server** shortcut on the desktop).



- b. Wait until the server is started.

The command window shows server trace messages as the server starts. A feedback message indicates when the server start is complete.

```
Start server
remoteSession init
[set-samples-settings] INFO: Authentication succeeded
[set-samples-settings] Feb 02, 2016 8:33:19 AM org.springframework.context.support.AbstractApplicationContext doClose
[set-samples-settings] INFO: Closing org.springframework.context.support.ClassPathXmlApplicationContext@e0937cfcd: startup date [Tue Feb 02 08:33:14 PST 2016]; root of context hierarchy
[set-samples-settings] Feb 02, 2016 8:33:19 AM org.springframework.beans.factory.support.DefaultSingletonBeanRegistry destroySingletons
[set-samples-settings] INFO: Destroying singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@573661b: defining beans [httpInvokerProxy,authenticationExecutor]; root of factory hierarchy

checkIfIsInstallingDC:

update.decisioncenter.db:

delete.new.installation.files:
[samples.echo] Feb 02, 2016 8:33:21 AM com.ibm.rules.sampleserver.Log info
[samples.echo] INFO: GBRPS0029I: start.server is completed.
[samples.echo] GBRPS0029I: start.server is completed.

BUILD SUCCESSFUL
Total time: 3 minutes 8 seconds
Press any key to continue . . .
```

- c. After the server is started, click in the command window, and press any key on your keyboard to close it.

1.2. Running the application

- 1. Open a web browser and enter the following URL:

<http://localhost:9080/miniloan-server>

The Miniloan application opens.

The screenshot shows a web-based application titled "Getting Started - Miniloan Server Application". The main title bar is dark blue with white text. Below it, a header bar has the text "Miniloan Validation" in a light blue background. The main content area has two sections: "Borrower Information" and "Loan Information".

Borrower Information		Loan Information	
Name:	Joe	Amount:	500000
Yearly Income:	80000	Duration (months):	240
Credit Score:	600	Yearly Interest Rate :	0.05

Below the tables are two buttons: "Use Rules" with a checkbox and "Validate Loan".

Scenario

Joe earns \$80,000 a year, has a credit score of 600, and would like to take a loan for \$500,000. He intends to repay the loan over a 20-year period.

Is Joe eligible for this loan?

- 2. Click **Validate Loan**.

Based on the information that Joe submitted, the rule engine returns a decision to reject the loan.

Miniloan Validation

Borrower Information		Loan Information	
Name:	Joe	Amount:	500000
Yearly Income:	80000	Duration (months):	240
Credit Score:	600	Yearly Interest Rate :	0.05

Use Rules

Validate Loan

The loan is rejected.

Messages:

The debt-to-income ratio is too big.

Scenario

Joe's loan request is rejected. Miniloan responds by reporting that the ratio of debt to income is too high.

- 3. You can either close the browser window or, if you prefer, you can leave it open because you come back to this application later in the exercise.

As you can see, decisions that are made by an application affect both your business and the lives of ordinary people. For that reason, it is important that experts in business policy, not just the IT team, can see and maintain the rules that are implemented. Instead of hardcoding the rules into the application, and requesting technical developers to make rule changes in the code, Operational Decision Manager separates the decision logic from the code. It empowers business users to access and manage the rules through Decision Center. Rules are stored in the Decision Center repository, and can be shared among various users through permission and locking mechanisms.

Next, you see how easily business policies can be updated in Decision Center.

Section 2. Managing business rules in Decision Center

Scenario

Several customers, including Joe, complained about having their loan requests rejected. You do not change the rules merely so that Joe can get a house, but a significant number of loan rejections might be an indication of errors in the rules. Some types of errors can be corrected early on, while in other situations, modifying rules might be a maintenance issue, such as ongoing rule adjustment for promotions or for different types of customers.

Miniloan decided to review its eligibility policies about debt-to-income ratios. Miniloan uses Decision Center to store and manage business rules, so next, you open Decision Center to see which rule to change to solve the loan rejection issue.

Decision Center supports auditability to trace the who, what, where, when, and why of rule updates. The rule administrators define roles and permissions in Decision Center to control access to the rules.

For this step, you sign in with the business user role of Rule Author to review the eligibility rules. You use Decision Center Business console, which is a collaborative rule authoring environment, to edit the rules in this section.

2.1. Editing rules in Decision Center Business console

1. Open the Business console from the Start menu by clicking **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Decision Center Business console**. You can also double-click the **Decision Center Business console** shortcut on the desktop.



Information

You can also open the Business console by entering the following URL in a browser:

<http://localhost:9080/decisioncenter>

Make sure that you use the correct URL and port for your environment.

For IBM ODM on Cloud users

Instead of going through your operating system, you start the Business console by logging in to the User Portal, then clicking **Launch** under **Decision Center Business console**.

The Business console is configured to open to the URL of your ODM on Cloud instance, so you do not need to enter a URL or port number.

- 2. When the Privacy message opens, click **Agree and Proceed**.

Privacy

Cookies are important to the proper functioning of a site. To improve your experience, we use cookies to remember log-in details, provide secure log-in and deliver content tailored to your interests. Click Agree and Proceed to accept cookies and go directly to the site.

Agree and Proceed



Troubleshooting

If you receive the Privacy message again, either later in this exercise or in other exercises, you can click **Agree and Proceed**.

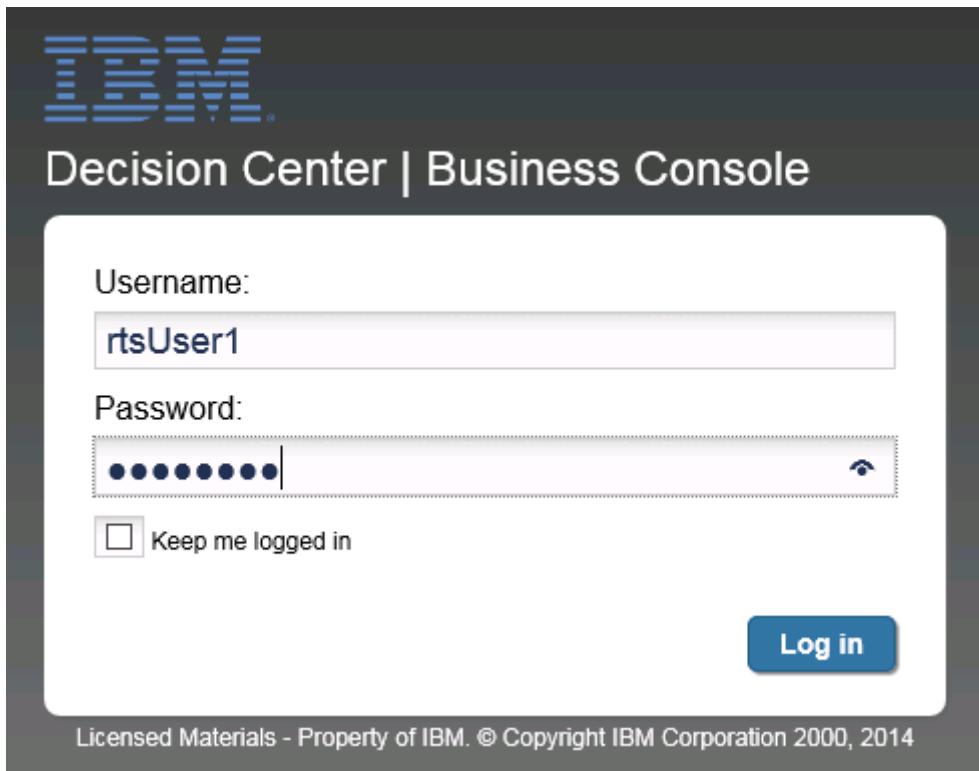
- 3. Sign in as a business user with the following values for the **Username** and **Password** fields.

- **Username:** rtsUser1
- **Password:** rtsUser1



Note

These values are case-sensitive.



The image shows the IBM Decision Center Business Console login page. At the top left is the IBM logo. Below it is the title "Decision Center | Business Console". The main area contains fields for "Username" (containing "rtsUser1") and "Password" (represented by a series of dots). There is also a checkbox for "Keep me logged in" and a blue "Log in" button. At the bottom of the form is a note: "Licensed Materials - Property of IBM. © Copyright IBM Corporation 2000, 2014".

- 4. Click **Log in**.

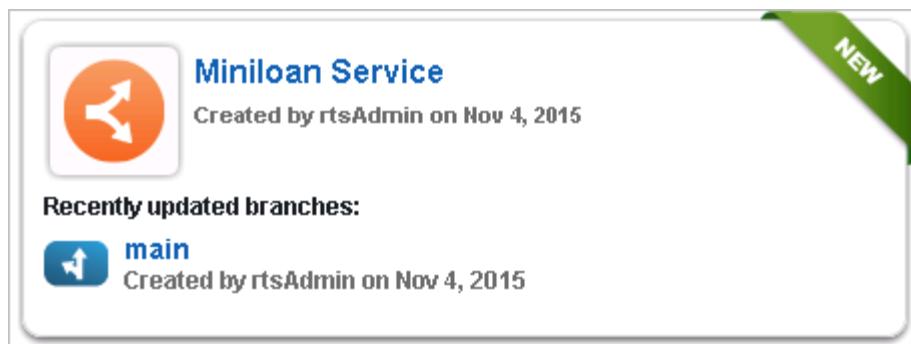


If you see a message to store your password, you can click **Not for this site**.

- 5. On the Decision Center menu, click **Library**.

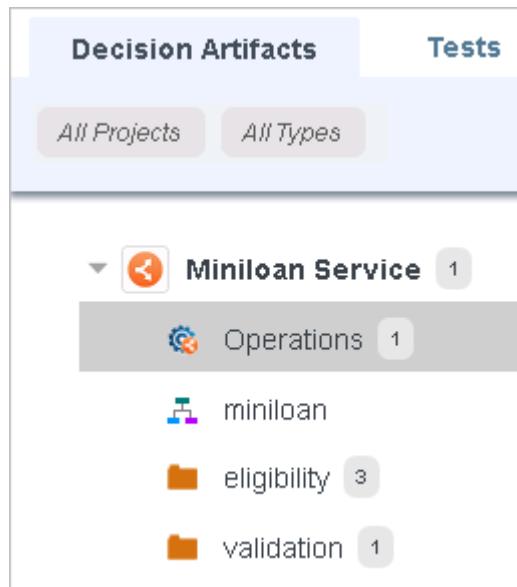


- 6. On the Decision Services page, you see the list of decision services that you can work on in Decision Center.
— 7. Click **Miniloan Service** (in the white space) and click **main**.



The **Decision Artifacts** tab opens and lists the contents of Miniloan Service, including these artifacts:

- **Operations** (artifact that defines input and output parameters for decision services)
- **miniloan** (ruleflow)
- **eligibility** (folder that contains rules)
- **validation** (folder that contains rules)



— 8. View the rules.

- a. Click the **eligibility** folder to see the rules that it contains.
- b. Click **minimum income** to review the minimum income policy, which is:

```
if
    the yearly repayment of 'the loan' is more than the yearly income of
    'the borrower' * 0.3
then
    add "Too big Debt-To-income ratio" to the messages of 'the
    loan' ;
    reject 'the loan' ;
```

The screenshot shows the 'minimum income' rule definition under the 'eligibility' folder. The rule is defined as follows:

```
if
    the yearly repayment of 'the loan' is more than the yearly
    income of 'the borrower'* 0.3
then
    add "Too big Debt-To-Income ratio" to the messages of
    'the loan';
    reject 'the loan';
```

Scenario

The minimum income rule implements the debt-to-income ratio policy that caused the loan rejection for Joe. Currently, if the debt is more than 30% of the borrower's income, the loan cannot be approved.

Miniloan business analysts decided to update their policy and change the ratio from 30% to 50%. Now, if the loan causes the borrower's debt to be up to 50% of the borrower's income, the loan can be approved.

- ___ 9. Edit the rule and change the debt-to-income ratio from 0.3 to 0.5.

- ___ a. Click **Edit** (the pencil icon) to open the rule editor.



- ___ b. In the **if** part of the rule, change 0.3 to: 0.5

if

the yearly repayment of '**the loan**' is more than the yearly income of '**the borrower**' * **0.5**

- ___ 10. Add a condition to the rule that tests whether the yearly income of the borrower is less than 500,000.

- ___ a. Place the cursor after 0.5, and press Enter to create a line.

if

the yearly repayment of '**the loan**' is more than the yearly income of '**the borrower**' * **0.5**

then

- ___ b. Enter the following condition text into the editor:

and the yearly income of 'the borrower' is less than **500000**



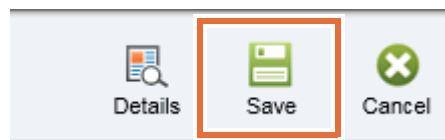
Note

As you type, the automatic completion menu might suggest terms that you can use to build your rule. You can also click these completion menu items to build the condition.

if

the yearly repayment of '**the loan**' is more than the yearly income of '**the borrower**' * **0.5**
and the yearly income of '**the borrower**' is less than **500000**
then

11. When you are finished with your changes, end the editing session and add a comment to provide version information for your changes.
- a. Click **Save**.



- b. In the **Create New Version** field, type a comment, such as:
Reject loans for low to mid-range income with debt-to-income ratio above 50%
- c. Click **Create New Version**.

Create New Version

A new version of this element will be created.
You can add a comment to associate with this version which can be viewed in the timeline.

Reject loans for low to mid-range income with debt-to-income ratio above 50%

Create New Version

The new rule details are shown in the “minimum income (v1.1)” window.

Miniloan Service > main

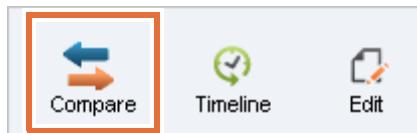
minimum income (v1.1) ☆ ⓘ

if
the yearly repayment of **'the loan'** is more than the yearly income of **'the borrower'** * 0.5
and the yearly income of **'the borrower'** is less than **500000**
then
add "**Too big Debt-To-Income ratio**" to the messages of **'the loan'**;
reject **'the loan'**;

You can now check the history of the rule that you modified, and compare the differences between the two versions.

2.2. Reviewing the rule history

- 1. Compare versions 1.0 and 1.1 of the minimum income rule.
 - a. On the toolbar, click **Compare**.



The “Compare minimum income Versions” window lists all versions of the rule. Since the rule has only two versions, both versions are selected by default.

- 2. Click **Compare**.

The screenshot shows a dialog box titled "Compare minimum income Versions". It contains two sections: "Current Version (1.1)" and "Version 1.0". Both sections have a green checkmark icon. Below each section is a summary of the rule's logic. At the bottom right is a large blue "Compare" button, which is also highlighted with a red box.

Compare minimum income Versions

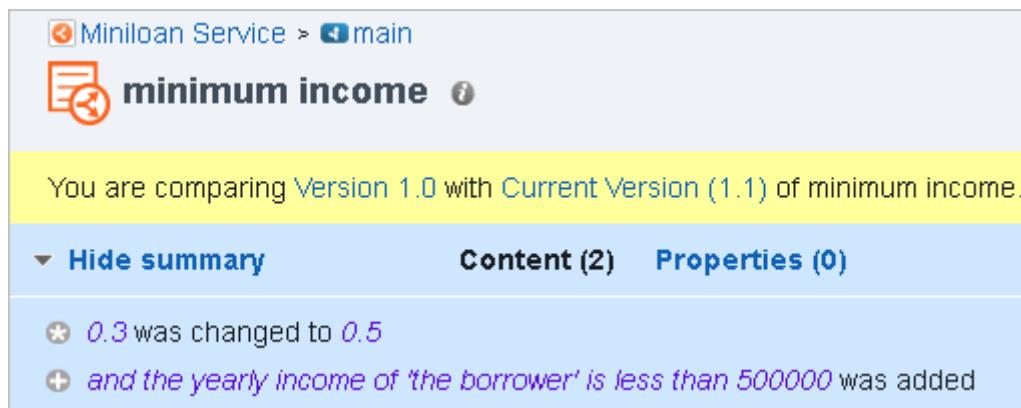
Select the two versions of this rule you want to compare.

Current Version (1.1)
Created by itsUser1 on Jan 19, 2016
Reject loans for low to mid-range income with debt-to-income ratio above 50%

Version 1.0
Created by itsAdmin on Nov 4, 2015

Compare

The two rule versions are shown side-by-side, with the newer version on the right. The summary lists the differences between the two versions.



The screenshot shows a rule comparison interface. At the top, it says "Miniloan Service > main" and "minimum income". A yellow bar at the top indicates "You are comparing Version 1.0 with Current Version (1.1) of minimum income." Below this, there are two tabs: "Hide summary" (with a downward arrow) and "Content (2) Properties (0)". Under "Content (2)", there are two items: a minus sign followed by "0.3 was changed to 0.5" and a plus sign followed by "and the yearly income of 'the borrower' is less than 500000 was added".

Miniloan Service > main

minimum income

You are comparing Version 1.0 with Current Version (1.1) of minimum income.

▼ Hide summary Content (2) Properties (0)

★ 0.3 was changed to 0.5

+ and the yearly income of 'the borrower' is less than 500000 was added

A red triangle indicates modified values, and additions to the rule are shown in purple, underlined text.

version 1.1 (current)

Created by itsUser1 on Jan 19, 2016

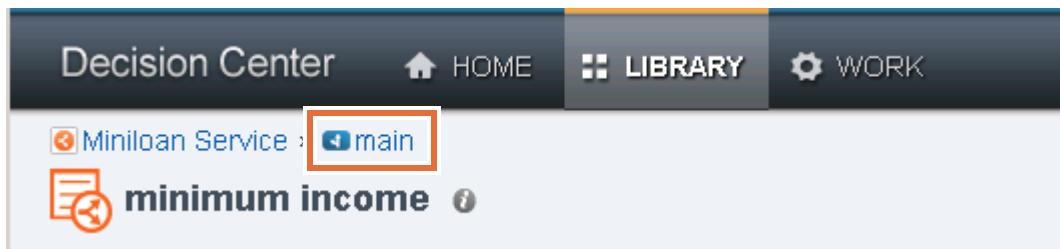
if

the yearly repayment of **'the loan'** is more than the yearly income of **'the borrower'** * **0.5**
and the yearly income of **'the borrower'** is less than **500000** ▲

then

add "**Too big Debt-To-Income ratio**" to the messages of **'the loan'**;
reject **'the loan'**;

- ___ 3. Return to the Miniloan Service decision service and open the **eligibility** folder.
 - ___ a. In the upper-right corner of the browser window, click the **main** breadcrumb.



- ___ b. In the left pane, click the **eligibility** folder to list its contents.

2.3. Changing the credit score requirements

Scenario

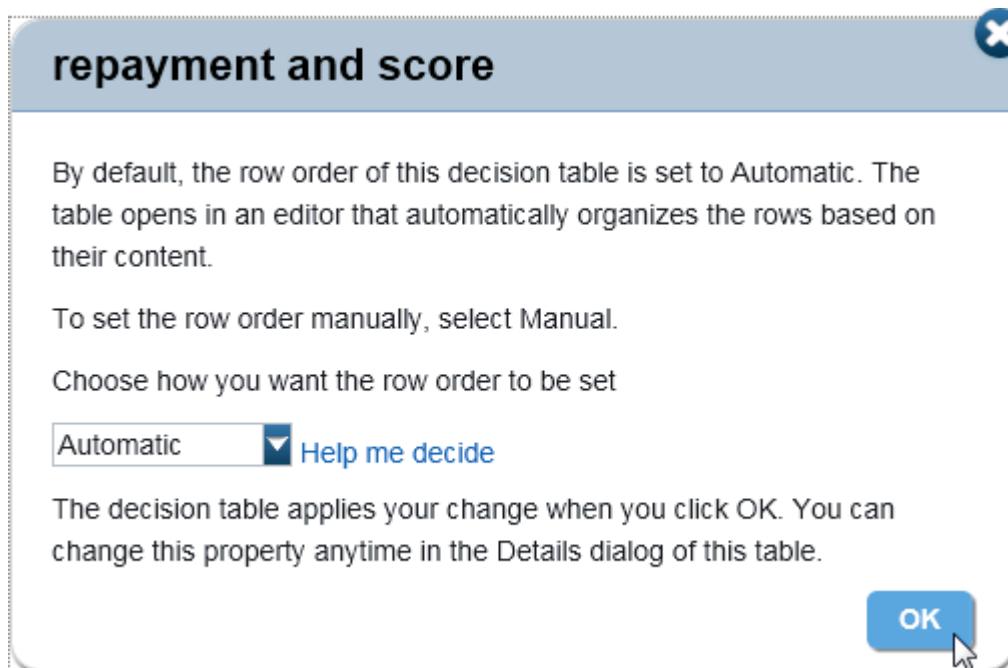
After reviewing the eligibility business policies, Miniloan decided to adjust its credit score requirements. Borrowers with a debt-to-income ratio that falls within the range of 45 and 50 are rejected when their credit score is less than 500. The credit score policies are implemented in a decision table.

- __ 1. Edit the repayment and score decision table to modify the credit score values.
- __ a. In the **eligibility** folder, hover your mouse on **repayment and score** and click the **Edit** icon to open the repayment and score decision table.

The screenshot shows the IBM ODM interface with the 'Miniloan Service' project selected. The 'eligibility' folder is highlighted. Inside, there are four rows: 'Name', 'minimum credit score', 'minimum income', and 'repayment and score'. The 'repayment and score' row is selected, indicated by a dashed blue border and a green pencil icon in the first column.

Name	Last Changed By	Last Change On
minimum credit score	rtsAdmin	November 4,
minimum income	rtsUser1	January 19, ..
<input checked="" type="checkbox"/> repayment and score	rtsAdmin	November 4,

- __ b. In the “repayment and score” window, leave the default row order set to **Automatic** and click **OK**.



- __ c. In row 5, double-click the cell in the credit score column with the value: [0: 600[

- ___ d. Delete 600 and type: 500

	debt to income		credit score	
	min	max	min	max
1	0 %	30 %	0	200
2	0 %	30 %	200	800
3	30 %	45 %	0	400
4	30 %	45 %	400	800
5	45 %	50 %	0	500
6	45 %	50 %	600	800
7	$\geq 50 \%$		0	800

- ___ e. Click another area of the table to make sure that the value changed.
 ___ f. Click **Save**.
 ___ g. In the Create New Version window, add a comment about the update that you made, such as:
 Row 5 value changed from 600 to 500
 ___ h. Click **Create New Version**.

After you create the version, your changes are saved and a gap error is detected in the table. The problem cells are highlighted with a gold triangle in the lower-left corner.

5	45 %	50 %	0	500	debt-to-income too high comp score
6	45 %	50 %	0	600	
7	$\geq 50 \%$		0	800	

2. Edit the table to resolve the gap error.
- Click **Edit** to open the rule editor.
 - In row 6, select the cell in the credit score column with the value [600; 800[.
 - Change the value from 600 to 500, and click **Save**.
 - In the Create New Version window, click **Create New Version**.

The table no longer shows any errors.

The screenshot shows a web-based application for managing loans. At the top, there's a navigation bar with links for 'Miniloan Service' and 'main'. Below the navigation is a title 'repayment and score (v1.2)' with a star icon and a refresh button. A section titled 'Preconditions' is expanded, showing a table with the following data:

	debt to		credit score		message	loan
	min	max	min	max		
1	0 %	30 %	0	200	debt-to-income too high compared to credit score	<input type="checkbox"/>
2	0 %	30 %	200	800	✗	<input checked="" type="checkbox"/>
3	30 %	45 %	0	400	debt-to-income too high compared to credit score	<input type="checkbox"/>
4	30 %	45 %	400	800	✗	<input checked="" type="checkbox"/>
5	45 %	50 %	0	500	debt-to-income too high compared to credit score	<input type="checkbox"/>
6	45 %	50 %	500	800	✗	<input checked="" type="checkbox"/>
7	≥ 50 %		0	800	debt-to-income too high compared to credit score	<input checked="" type="checkbox"/>

3. Click the **Stream** tab on the right view and notice that the rule stream was automatically updated. By clicking the **plus** icon , you can see the details of the version that you created.

Section 3. Validating rule changes

Scenario

Before making the updated business rules available to the Miniloan web application, the policy manager wants to make sure that the rules behave as expected in a test environment.

The business analyst works with the development team to identify what must be tested and to create the test scenarios. The scenarios are stored, along with their expected results, in a Microsoft Excel spreadsheet.

In this section, you run tests and simulations from the Decision Center Business console.

To run a scenario in Decision Center, you must create a test suite. For this exercise, the scenario file for the test suite is already created for you. The `miniloan-test.xls` scenario file is in this directory:

```
<InstallDir>\gettingstarted\BusinessConsole
```

The scenario file contains two scenarios with their corresponding expected results. The scenarios are listed on the **Scenarios** tab, and the anticipated results on the **Expected Results** tab.

Running the test

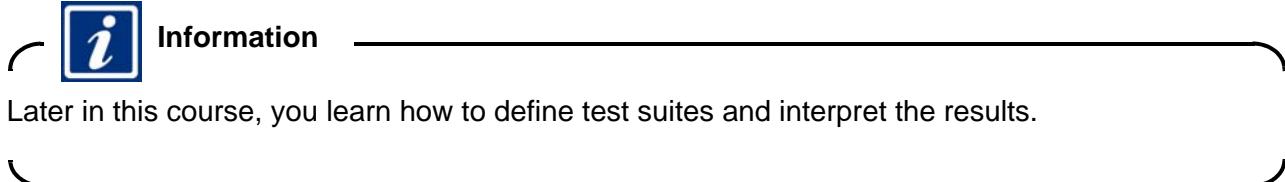
- ___ 1. Click the **main** breadcrumb to return to the Miniloan Service artifacts.
- ___ 2. Click the **Tests** tab.

A prepared test suite, Miniloan test suite, is available to test all the rules in the decision service. You can run this test to see whether your changes affect the outcome.

- ___ 3. Hover your mouse over **Miniloan test suite**, and click the **Run** icon.



-
- ___ 4. In the Run Test Suite window, click **OK** to close the window.
 - ___ 5. Wait for the test to complete and then click the report link to open the report.
Because you changed the rules, you should not expect a 100% success rate.
 - ___ 6. Click **Close** to close the report.



- ___ 7. Log out of the Business console.
 - ___ a. Click the down arrow next to your user name in the upper-right corner of the browser window.
 - ___ b. Click **Log out**.



- ___ c. Close the browser.

Section 4. Deploying updated rules from Decision Center to Rule Execution Server

Scenario

The tested rules are now ready to be deployed to the Miniloan web application.

Users with the appropriate access rights can deploy rulesets directly from the Decision Center. You can also deploy rulesets from Rule Designer.

In this task, you take the role of Rule Administrator to define and deploy the updated rules. The rules are packaged as a RuleApp.



Information

A RuleApp is a container for rulesets. The RuleApp is deployed to Rule Execution Server, making the rulesets available for execution.

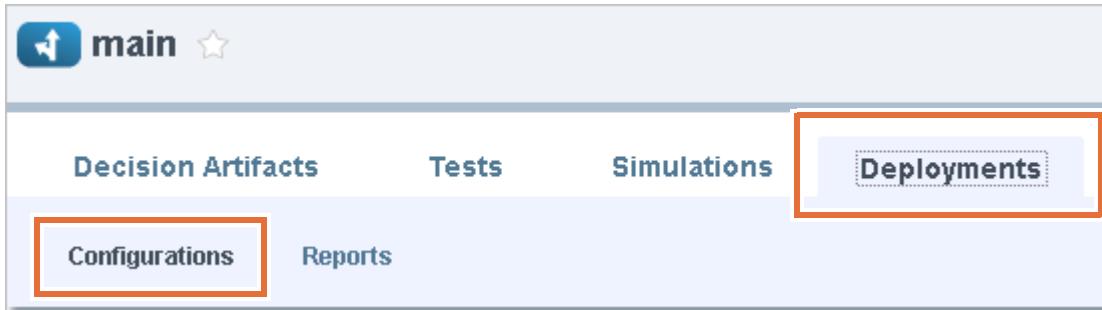
4.1. Deploying the decision service

- 1. Sign in to the Business console as a user with administrative privileges:
 - **User name:** rtsAdmin
 - **Password:** rtsAdmin
- 2. Click **Library** and open the **main** branch of the Miniloan Service decision service.
- 3. On the **Decision Artifacts** tab, click **Operations**, and click **Miniloan ServiceOperation**.
- 4. Open the Operation view by clicking the **Operation** icon in the upper-right corner.



- 5. In the upper-right corner of the page, click **Edit**.
- 6. Change the value in these fields:
 - **Name:** my_operation
 - **Ruleset name:** my_operation
- 7. Save your changes.
 - a. Click **Save** to exit the editor.
 - b. In the Create New Version window, click **Create New Version**.
- 8. Click the **main** breadcrumb to return to the Decision Service page.

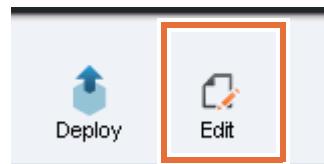
- ___ 9. Click the **Deployments** tab and make sure that you are on the Configurations page.



- ___ 10. Click the **Miniloan** deployment configuration.



- ___ 11. In the upper-right corner, click the **Edit** icon.



- ___ 12. On the **General** tab, edit the configuration by changing the **RuleApp name** field to: my_deployment

- ___ 13. Click the **Targets** tab and select **Sample**.

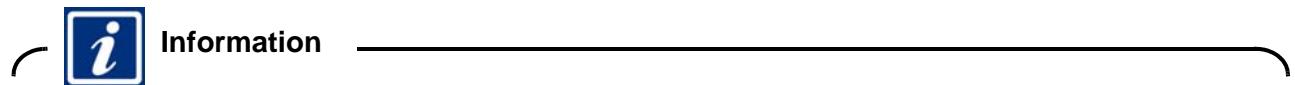


- ___ 14. Click **Save** and click **Create New Version**.

- ___ 15. In the upper-right corner, click the **Deploy** icon.



-
- ___ 16. In the “Deploy main” window, click **Deploy**.
 - ___ 17. In the “Deployment status” window, click **OK**.
 - ___ 18. When deployment completes, click the **Report** link.
 - ___ 19. View the report details, and when you are finished, click **Close**.
 - ___ 20. Log out of the Business console.



Later in this course, you learn how to define deployment configurations, define target servers, and manage deployment in Rule Designer and Rule Execution Server console.

Section 5. Monitoring the deployed rules in Rule Execution Server

You can now go to the Rule Execution Server console and see whether the RuleApp was properly deployed.

Rule Execution Server is an execution environment for rules (Java SE and Java EE) that interacts with the rule engine. Rule Execution Server handles the management, performance, security, and logging capabilities that are associated with the execution of your rules.

Viewing the deployed RuleApp

1. Open Rule Execution Server by entering the following URL in a browser:

`http://localhost:9080/res`



Information

You can also open the Rule Execution Server from the **Start** menu:

Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Rule Execution Server console.

For IBM ODM on Cloud users

Instead of going through your operating system, you start the Rule Execution Server console by logging in to the User Portal, then clicking **Launch** under **Rule Execution Server console**.

The Rule Execution Server console is configured to open to the URL of your ODM on Cloud instance, so you do not need to enter a URL or port number.

2. Sign in to the Rule Execution Server console with the following credentials.

- **User name:** resAdmin
- **Password:** resAdmin

-
3. Click the **Explorer** tab, and in the Navigator pane, expand **RuleApps**.

The screenshot shows the IBM Rule Execution Server console interface. The top navigation bar includes tabs for Home, Explorer (which is selected and highlighted in blue), Decision Warehouse, Diagnostics, and Services. Below the navigation bar, the URL is shown as Explorer > RuleApps.

The main area is titled "RuleApps View". On the left, there is a "Navigator" pane containing a tree view of resources. A red box highlights the "RuleApps (1)" node, which has two children: "/my_deployment/1.0 (1)" and "/my_operation/1.0". Other nodes in the tree include "Resources (1)", "Libraries (2)", and "Service Information".

The right side of the screen displays the "RuleApps" section. It shows a summary: "Total Number of RuleApps 1". Below this, a table lists the single rule app: "1 RuleApp(s)". The table columns are "Select All", "Name", "Version", and "Creation Date". The data row shows a checkbox next to a cube icon, the name "my_deployment", version "1.0", and creation date "Feb 2, 2016 11:48:57 AM GMT-08".

Select All	Name	Version	Creation Date
<input type="checkbox"/>	my_deployment	1.0	Feb 2, 2016 11:48:57 AM GMT-08

At the bottom of the table, it says "RuleApp 1 - 1 of 1". To the right, there are navigation links: "prev 10 next".

You see the ruleset that you deployed.

-
4. In the RuleApps list, click **my_deployment** to see the details in the RuleApp view.

RuleApp View

The screenshot shows the RuleApp View for the deployment '/my_deployment/1.0'. At the top, there are buttons for 'Add Ruleset', 'Add Property', 'Download Archive', and 'Edit'. Below this, the deployment details are listed:

Name	my_deployment
Version	1.0
Creation Date	Feb 2, 2016 11:48:57 AM GMT-08:00
Display Name	
Description	

Below the details, there is a link 'Show Properties (0)'. Underneath, a table lists the rulesets:

<input type="checkbox"/> Select All	Name	Version	Ruleset Path	Creation Date
<input type="checkbox"/>	my_operation	1.0	/my_deployment/1.0/my_operation/1.0	Feb 2, 2016 11:48

At the bottom, it says 'Ruleset 1 - 1 of 1' and has navigation links 'prev 10 next 10'.

5. In the list of rulesets, click **my_operation** to open it in the Ruleset view.

Notice that the status of the ruleset is **enabled**, which means that your newly deployed ruleset can be executed. The next time that the application calls for a decision, this ruleset is considered the most recent version, and will be used.

-
6. Scroll down and click **Show Properties** to open the list of properties for this ruleset.

 Ruleset Parameters

Direction	Name	Kind	XOM Type
	borrower	native	miniloan.Borrower
	loan	native	miniloan.Loan

Ruleset Parameters 1 - 2 of 2 [prev](#) [10](#) [next](#) [10](#)

 [Show Managed URIs \(1\)](#)

[Hide Properties](#)

 18 properties

<input type="checkbox"/> Select All	Name	Value
<input type="checkbox"/>	decisioncenter.url	 http://localhost:9080/decisioncenter
<input type="checkbox"/>	decisioncenter.version	 Decision Center 8.8.0.0
<input type="checkbox"/>	decisionservice.branch.id	 dsm.DsDeploymentBsln:109:109
<input type="checkbox"/>	decisionservice.branch.name	 Miniloan_2016-01-20T16_06_50Z
<input type="checkbox"/>	decisionservice.branch.type	 snapshot
<input type="checkbox"/>	decisionservice.branch.url	 http://localhost:9080/decisioncenter/t/library#overviewsnapshot?dataBaselineId=dsm.DsDeploymentBsln:109:109
<input type="checkbox"/>	decisionservice.deployer.id	 rtsAdmin
<input type="checkbox"/>	decisionservice.deployer.name	 rtsAdmin
<input type="checkbox"/>	decisionservice.deploymentConfiguration.id	 dsm.Deployment:12:24
<input type="checkbox"/>	decisionservice.deploymentConfiguration.name	 Miniloan



Information

Later in this course, you work with Rule Execution Server console.

-
7. Sign out and close the Rule Execution Server console window.

Section 6. Viewing the effects in the Miniloan application

Scenario

After testing and deploying the updated business rules to the Miniloan application, you can now see how the business policy changes that you made are reflected in the Miniloan application. You can also see how they affect the original request from Joe.

To run the Miniloan application with the updated ruleset:

1. Reopen the Miniloan application in a web browser window.
`http://localhost:9080/miniloan-server`
2. Select **Use Rules**.
3. Click **Validate Loan** with the default values.

The loan is now approved because of the deployed rule changes.

The screenshot shows the 'Miniloan Validation' application interface. At the top, there are two sections: 'Borrower Information' and 'Loan Information'. Under 'Borrower Information', the 'Name' field contains 'Joe', 'Yearly Income' is '80000', and 'Credit Score' is '600'. Under 'Loan Information', 'Amount' is '500000', 'Duration (months)' is '240', and 'Yearly Interest Rate' is '0.05'. Below these fields is a checkbox labeled 'Use Rules' which is checked. A large blue button labeled 'Validate Loan' is centered below the input fields. A green message box displays the text 'The loan is approved.' Below this message, a smaller text states 'The yearly repayment is 39,597.' and a 'Messages:' section is present. At the bottom, a blue header bar reads 'Rules Execution Summary [1 rule(s) executed.]' followed by a table row showing 'Rule(s) executed' and 'Rule eligibility.repayment and score 6 was executed in rule task miniloan#e'.

Notice that the **Rules Execution Summary** lists the decision table row that you edited as the rule that was used for this loan decision.

4. Close the browser.

Section 7. Exploring the development environments in Rule Designer

7.1. Opening Rule Designer

- 1. Click **Start > All Programs > IBM > Operational Decision Manager V8.8 > Rule Designer** (or you can use the shortcut on the desktop).



- 2. When prompted for a workspace, type the following path and click **OK**:

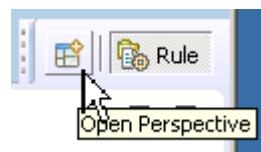
C:\labfiles\workspaces\intro

- 3. Close the Welcome view.

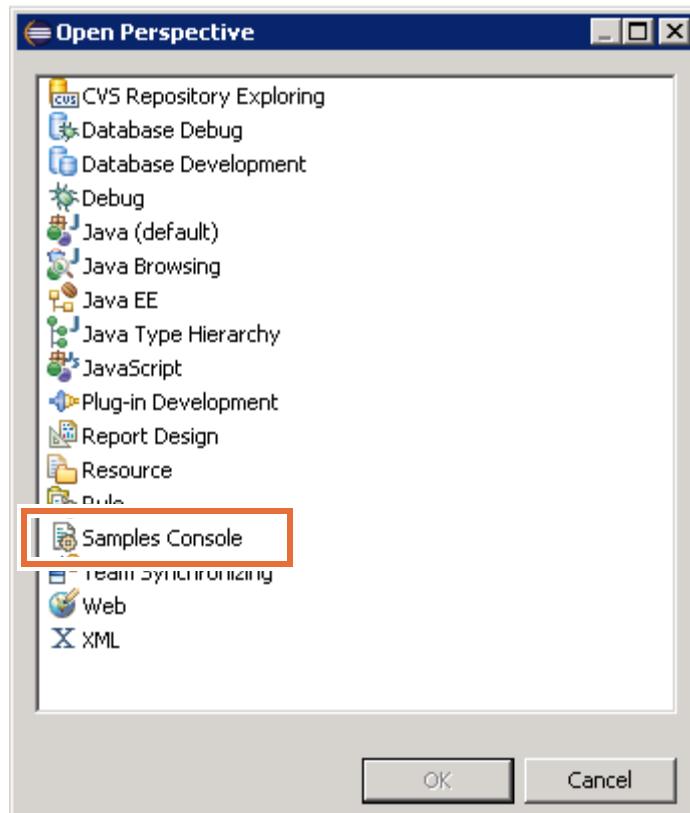
By default, the Rule Designer perspective opens in Eclipse.

- 4. Switch to the Samples Console and import the project for this exercise.

- a. In the upper-right corner, click the **Open Perspective** icon to switch from the Rule perspective.



- __ b. In the Open Perspective list, select **Samples Console**, and click **OK**.



- __ c. In the **Getting Started** section, expand **Decision Server**, and under **answer**, click **Import projects**.



The Eclipse perspective automatically switches back to the Rule perspective.

- __ d. Close the Help view by clicking the X.



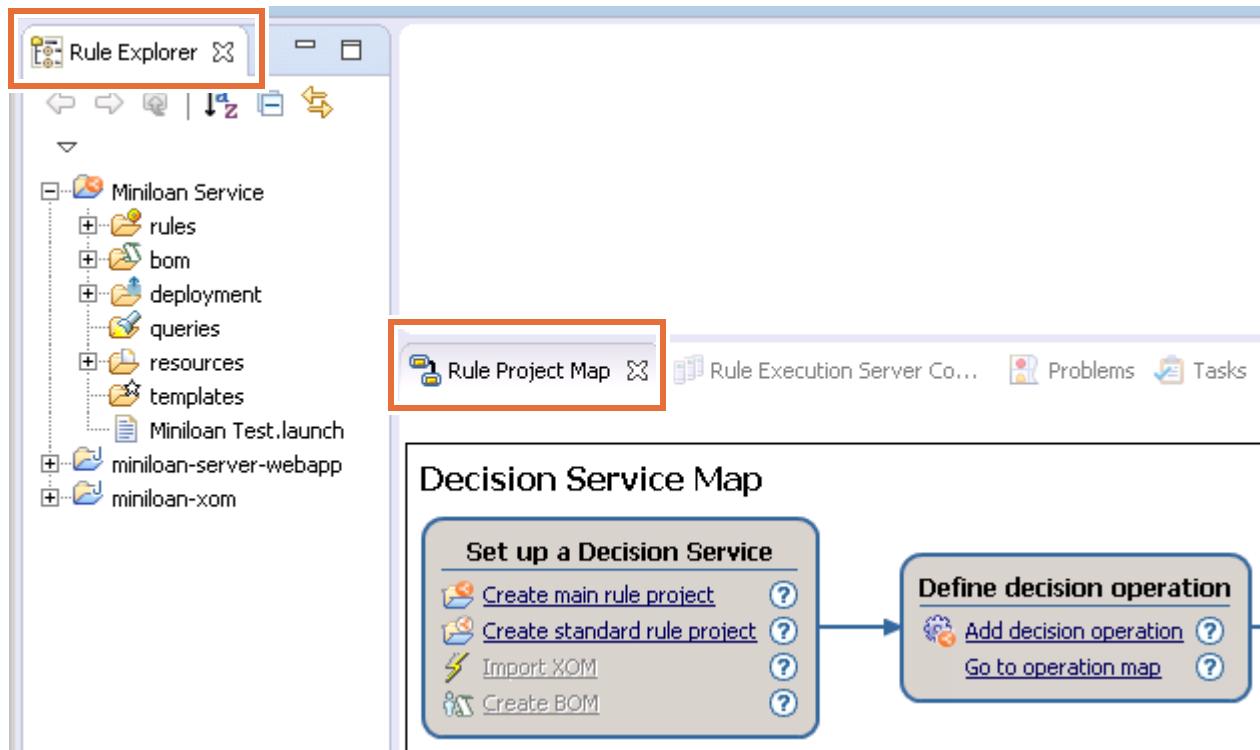
Hint

To give yourself more working area, you can close the views (such as Help or Welcome views) that open by default in Eclipse.

7.2. Taking a quick tour of Rule Designer

In Eclipse, the window for the Rule Designer development environment is called the *Rule perspective*.

- 1. Look at the various windows and panes that are open in the Rule perspective, including:
 - Rule Explorer: Navigate the contents of the projects that you are working with
 - Rule Project Map view: Use the Decision Service Map to guide you through development of a decision service



The Rule Explorer contains these projects:

- **Miniloan Service**: The main rule project that contains Miniloan business rules.
 - **miniloan-xom**: The Java project of the Miniloan application that is composed of the **borrower** and the **loan** Java classes.
- 2. In Rule Explorer, expand **Miniloan Service > rules** to see the rule artifacts.
 - a. Expand the **eligibility** and **validation** folders to see the rules that they contain, and notice that these folders are the same folders that you saw in Decision Center.
 - b. In the **eligibility** folder, double-click **repayment and score** to open the decision table.
 - c. Notice that the values in the **credit score** columns for row 5 and 6 do not reflect the changes that you made in Decision Center, where you changed 600 to 500.
 - d. Close the editing window for the decision table.
 - 3. Expand the **bom > miniloan > miniloan** folder to see **Borrower** and **Loan**.

-
- The **bom** folder contains all the vocabulary that is used in the rules.
- ___ a. Expand **Borrower** and **Loan** to see their members in Rule Explorer.
 - ___ b. Double-click either **Borrower** or **Loan** to open it in the BOM editor and view its properties and members.
- ___ 4. Open the **Outline** view (under Rule Explorer), and note the connection between this view and the contents of the **bom** folder.
- The **Outline** view lists all the members of the BOM.
- ___ a. In the Outline view, expand **Borrower** and **Loan** to see their members.
 - ___ b. Click the **Vocabulary** tab and expand **borrower** to see the vocabulary expressions that are assigned to the BOM members.
 - ___ c. Notice the similarities and differences between the expressions that are listed in the **Vocabulary** view and the **Outline** view.
- ___ 5. In Rule Explorer, go back to the **rules** folder to see how the rules use the vocabulary.
- ___ a. Go back to the **eligibility** folder and double-click **minimum income** to open this rule in the rule editor.



Questions

Does the wording in the rule seem to match the **Outline** view or the **Vocabulary** view?

Remember this comparison as a first look at the connection between rules and vocabulary.

- ___ b. Notice the rule statement. This rule also does not include your edits from Decision Center.

To get the latest version of the rules from Decision Center, you must synchronize the Rule Designer and Decision Center environments.



Information

Later in this course, you work with the Rule Designer synchronization tools.

Section 8. Shutting down the modules

These steps describe how to shut down the sample server and modules.

- 1. Close all browsers for the Miniloan application, Rule Execution Server, and Decision Center consoles.
- 2. Close Rule Designer and click **OK** when prompted to confirm your exit from Eclipse.
- 3. Stop the sample server.
 - a. Click **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Stop server.**
 - b. After the sample server is stopped, press any key to close the terminal window.

End of exercise

Exercise review and wrap-up

This exercise showed the general workflow of Operational Decision Manager for business rules.

Exercise 2. Setting up decision services

What this exercise is about

In this exercise, you learn how to set up decision services in Rule Designer.

What you should be able to do

After completing this exercise, you should be able to:

- Create main and standard decision service projects
- Set up the decision service to reference the execution object model (XOM)
- Generate a business object model (BOM) and a default vocabulary
- Create a decision operation
- Define ruleset variables and ruleset parameters
- Create rule packages
- Synchronize decision services with Decision Center

Introduction

In this exercise, you use Rule Designer to create the initial elements that are required to set up a business rule application. You set up the decision service structure, define the business and execution object models, and create a decision operation, which includes ruleset variables and ruleset parameters. Ruleset parameters are used to pass objects between the external application and the rule engine.

You also outline the basic structure for organizing rule artifacts by creating rule packages, which are also called folders in the business user environment.

This exercise includes the following sections:

- Section 1, "Creating a standard rule project"
- Section 2, "Importing the XOM and setting up the BOM"
- Section 3, "Creating the main rule project for the decision service"
- Section 4, "Creating and defining the decision operation"
- Section 5, "Creating rule packages"
- Section 6, "Publishing from Rule Designer to Decision Center"
- Section 7, "Deleting a decision service from Decision Center"
- Section 8, "Importing a decision service from Decision Center"

Requirements

This exercise uses the following files, which are installed in the `<InstallDir>\studio\training` directory:

- Start project: Dev 02 - Decision services\01-start
- Solution project: Dev 02 - Decision services\02-answer
 - You can use the solution project in "Exercise review and wrap-up" on page 2-40.

Section 1. Creating a standard rule project

In this section of the exercise, you set up the environment for the exercise and create a standard rule project for the decision service.

1.1. Setting up your environment for this exercise

- 1. Open Rule Designer by clicking **Start > All Programs > IBM > Operational Decision Manager V8.8 > Rule Designer**.
- 2. In the **Workspace** field of the Workspace Launcher window, create a workspace.
 - a. Type a workspace name:
`<LabfilesDir>\workspaces\decision_service`
 - b. Make sure that the **Use this as the default and do not ask again** check box is *not* selected.
 - c. Click **OK**.

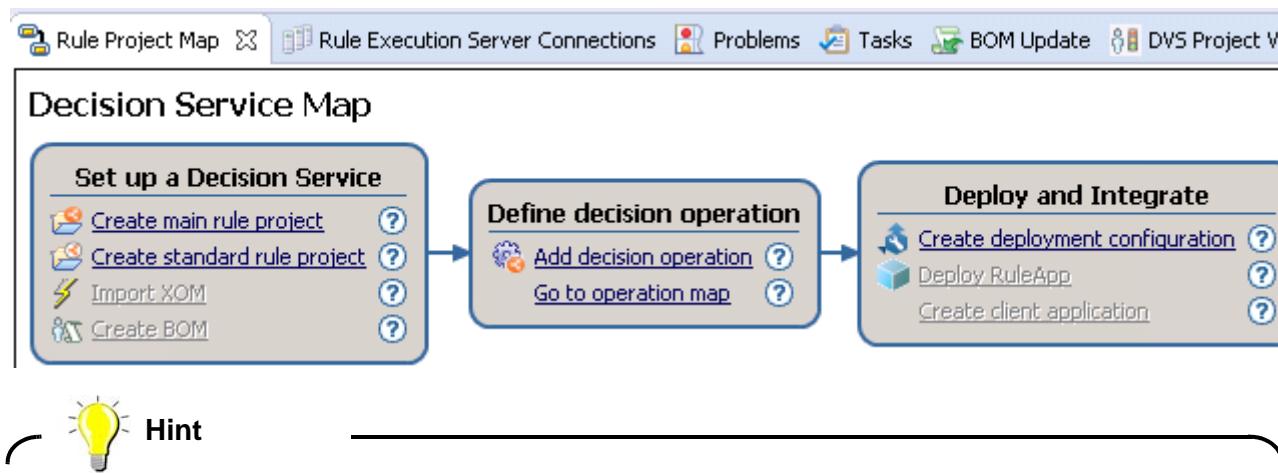
The `<LabfilesDir>\workspaces\decision_service` workspace is created.

- 3. Close the Welcome view.
- 4. Close the Help view.

The steps for setting up a decision service are outlined in the map tasks:

- **Set up a Decision Service**
- **Define decision operation**
- **Deploy and Integrate**

In this exercise, you go through these steps to create and deploy a decision service.



You can maximize the view and reset it to the default size and location by using the icons in the upper-right corner of the view. If you close the Rule Project Map view, you can open it by clicking **Window > Show View > Rule Project Map**.

1.2. Creating a standard rule project with a BOM

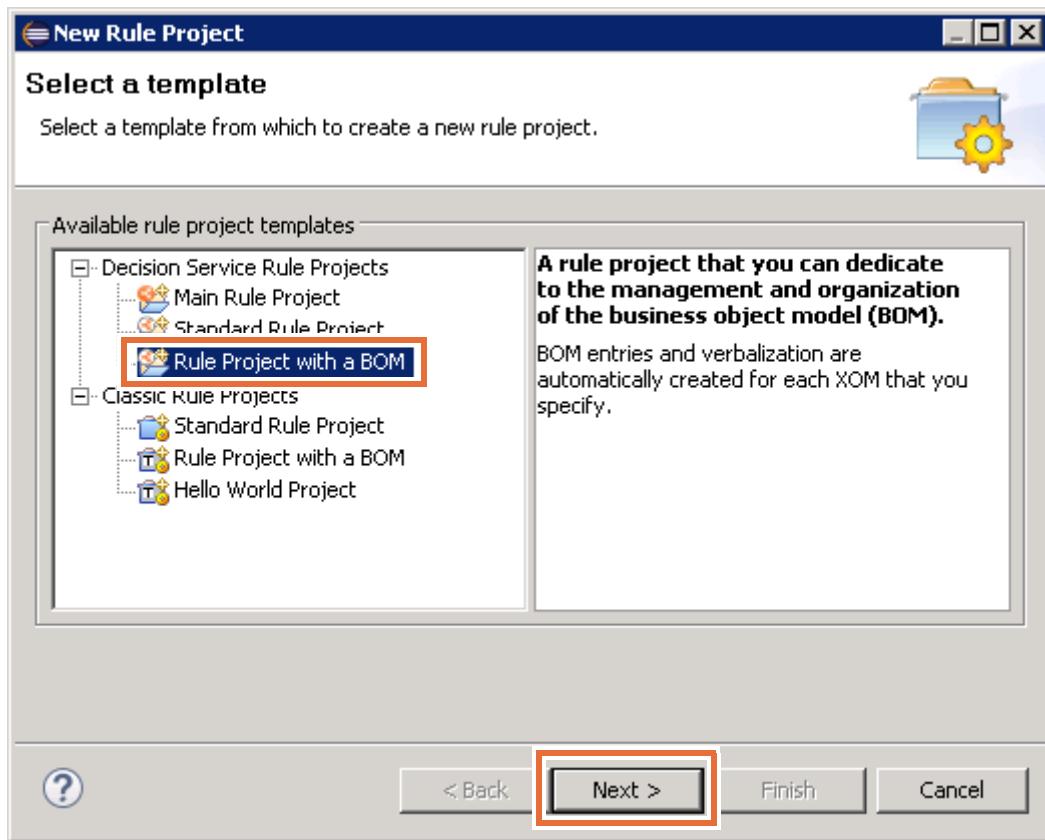
In this section of the exercise, you create an empty standard rule project with a BOM in preparation for the next section of the exercise, where you import the XOM and create the BOM.

The rule project that you create in this section eventually stores the BOM for the decision service.

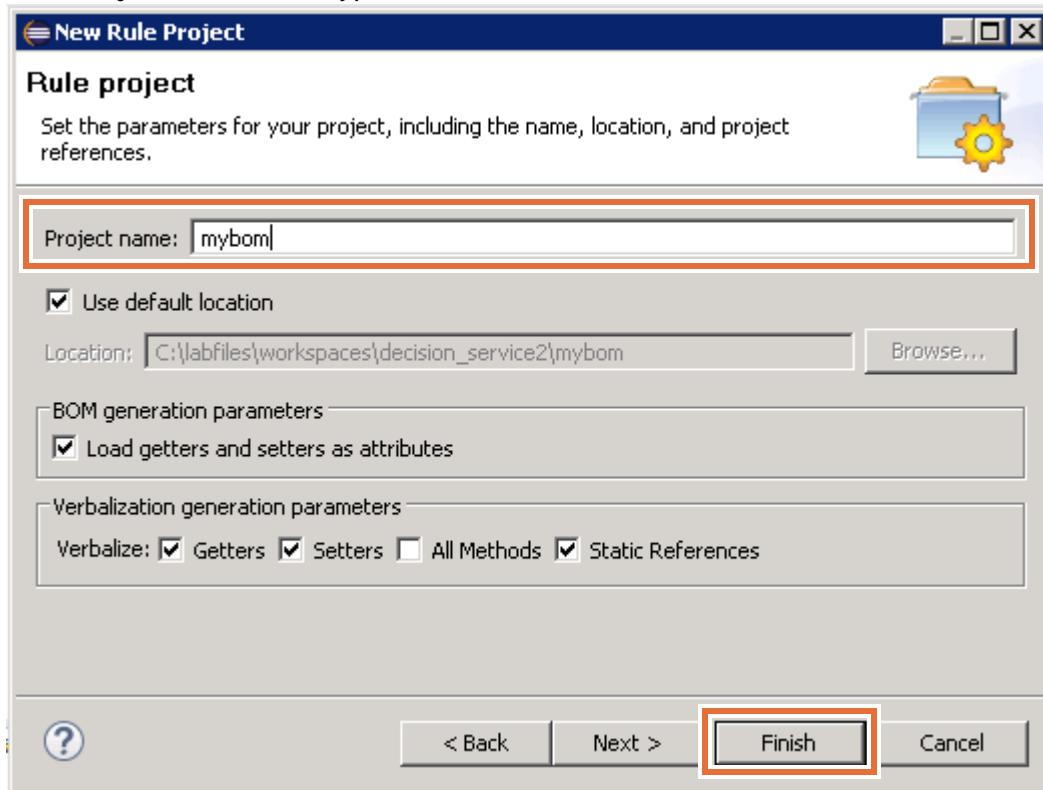
- 1. Go to the **Setup a Decision Service** part of the Decision Service Map, and click **Create standard rule project**.



- 2. In the **Decision Service Rule Projects** section of the New Rule Project window, click **Rule project with a BOM**, and click **Next**.



3. In the **Project name** field, type a name such as: mybom



4. Click **Finish**.

The **mybom** folder is now in the Rule Explorer pane.



Section 2. Importing the XOM and setting up the BOM

2.1. Importing the execution object model (XOM)

One of the first tasks when designing your decision service is to import a XOM into your project.

By importing a XOM, you create a reference between your rule project and a XOM project in your workspace. To be able to create such a reference, you must first have the XOM in your Rule Designer workspace.

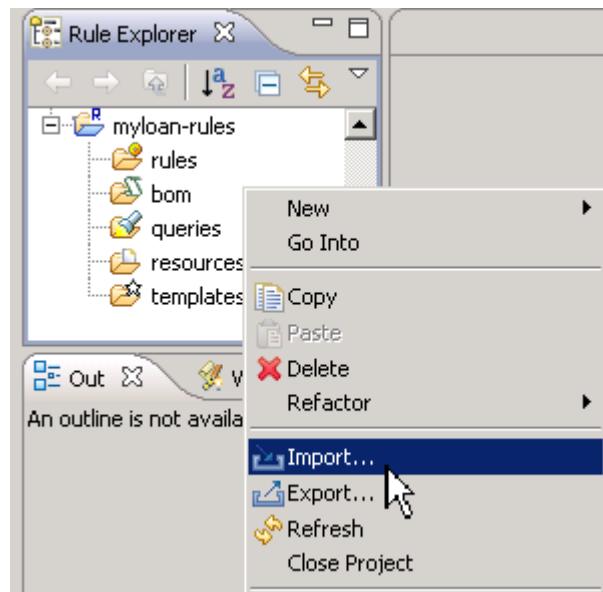


Information

The XOM is the code implementation of the model against which your business rules are executed. For this exercise, the XOM is provided for you. You learn more about XOMs later in this course.

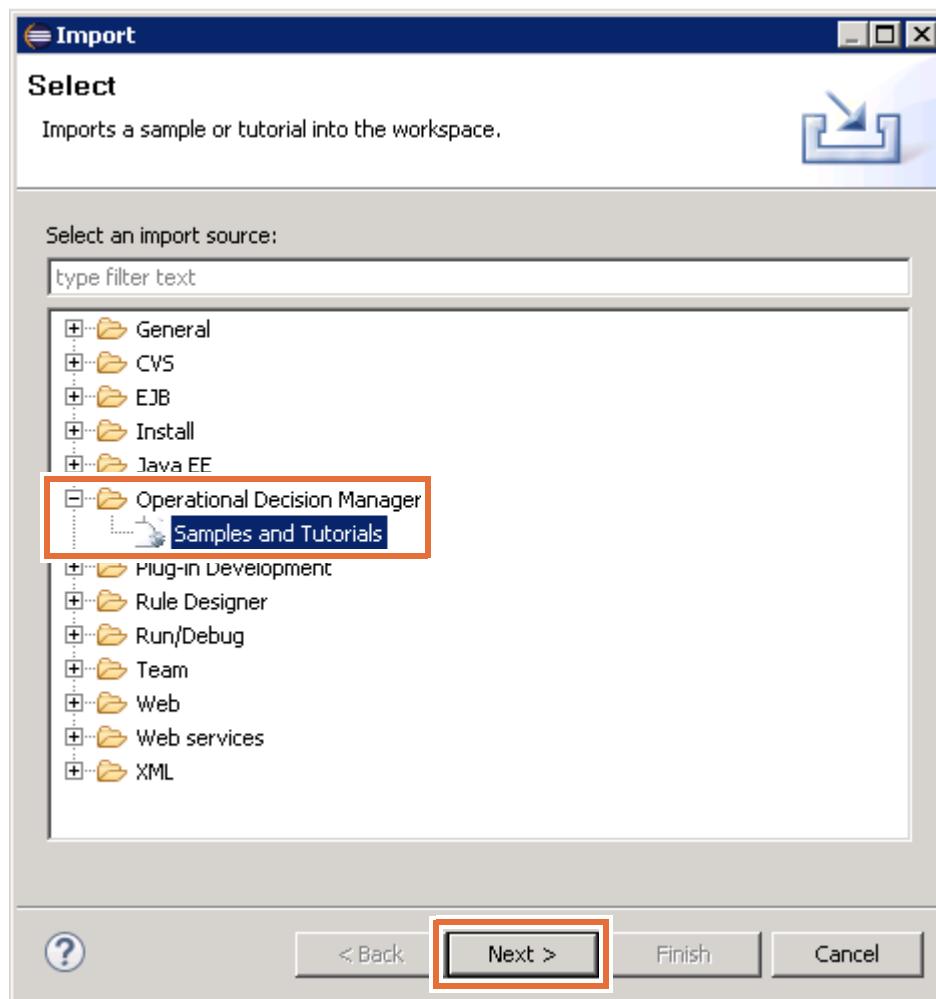
For this exercise, the required XOM is the `loan-xom` Java project and is provided for you. You import this XOM by importing the project **Ex 02: Setting up decision services > 01-start** by following the next procedure.

1. Right-click anywhere in the Rule Explorer and click **Import** to open the Import wizard in Eclipse.



The Select page of the Import wizard opens.

2. On the Select page, select **Operational Decision Manager > Samples and Tutorials**, and click **Next**.

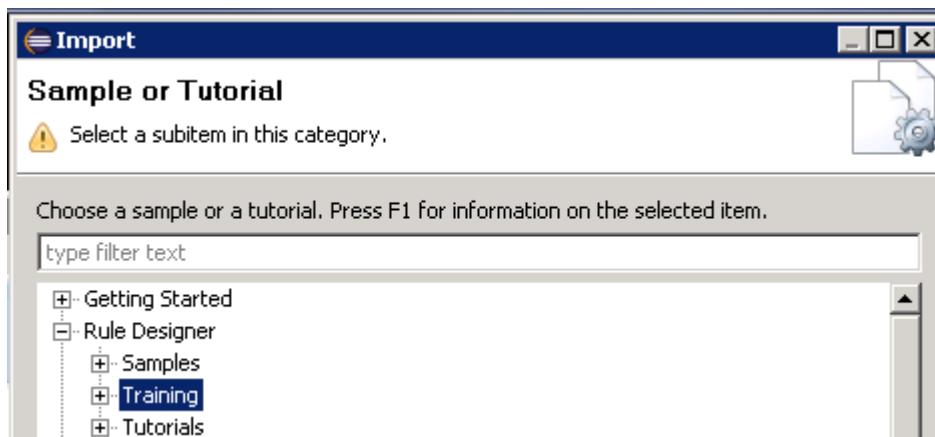


3. On the “Sample or Tutorial” page, expand the **Rule Designer > Training** node.

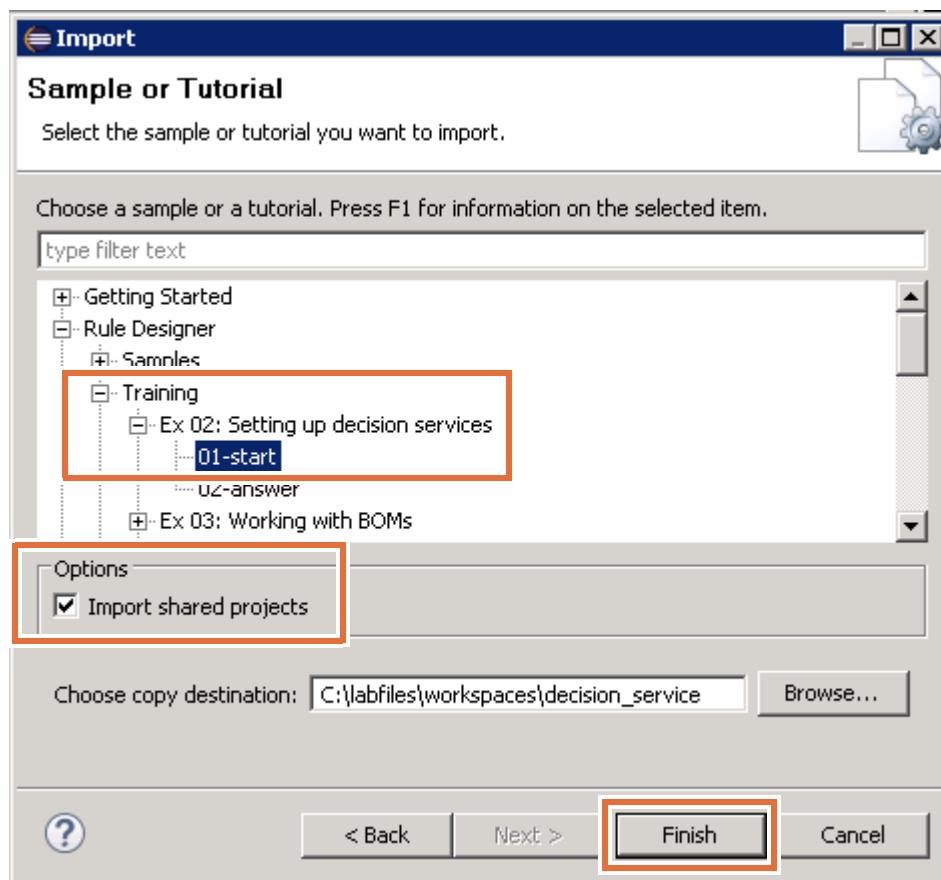


Hint

You can collapse the Rule Designer node and then expand it to simplify the list.



- 4. Select your start project from the **Rule Designer > Training** node.
 - a. In the Training node, expand the **Ex 02: Setting up decision services** node, and select **01-start**.
 - b. Make sure that the **Import shared projects** check box is selected, and click **Finish**.



Rule Designer automatically imports the content of the selected project into your workspace. In this case, Rule Designer imports the `loan-xom` Java project.

The `loan-xom` Java project is now available in Rule Explorer.



- ___ 5. After the workspace builds, close the Help view.



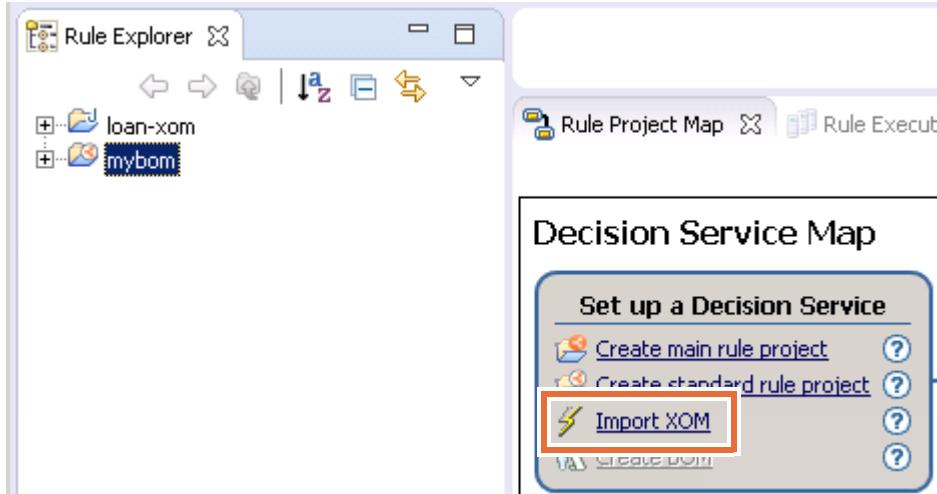
Important

For each exercise, you import projects into your workspace. The procedure for importing is also available at the beginning of this document, in "Projects for exercises" on page xvii.

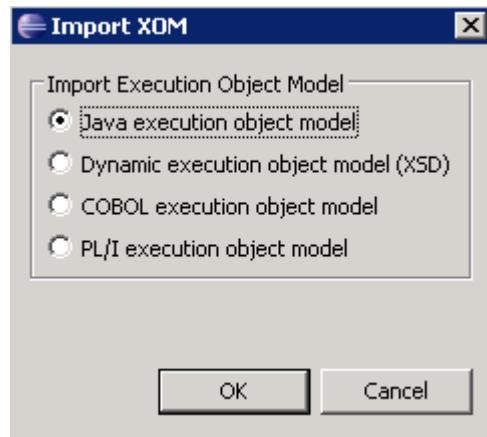
2.2. Creating a reference to the XOM in the rule project

Now that you have a XOM in your workspace, use the Decision Service Map to import it into your decision service.

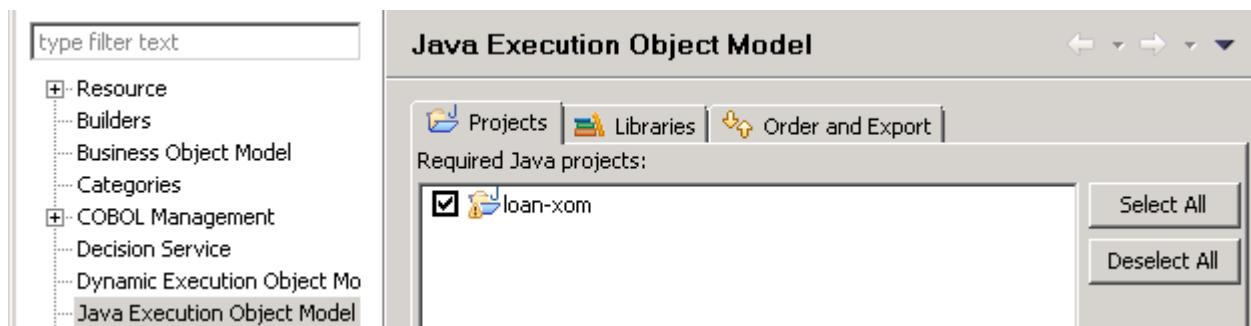
- ___ 1. Make sure that the `mybom` project is selected in the Rule Explorer.
- ___ 2. In the **Set up a Decision Service** part of the Decision Service Map, click the **Import XOM** link to import the XOM into your decision service.



- ___ 3. On the **Import XOM** page, select **Java execution object model**, and click **OK**.



- ___ 4. In the “Properties for myloan-rules” window, under **Java Execution Object Model**, select the **loan-xom** check box, and click **OK**.



By selecting **loan-xom**, the **mybom** project now references the **loan-xom** Java project.

You can expand the **loan-xom** project to view the Java code implementation.

2.3. Creating the BOM

In the **Set up a Decision Service** part of the Decision Service Map, the **Create BOM** link is enabled.



This link indicates that the next required task in designing your ruleset is to define the BOM and the vocabulary that you and business users need to author the business rules.

You now use this link to set up the BOM in your rule project.

- ___ 1. In the Rule Explorer, make sure that the **mybom** project is selected.
- ___ 2. In the **Set up a Decision Service** task, click the **Create BOM** link to create a BOM entry.

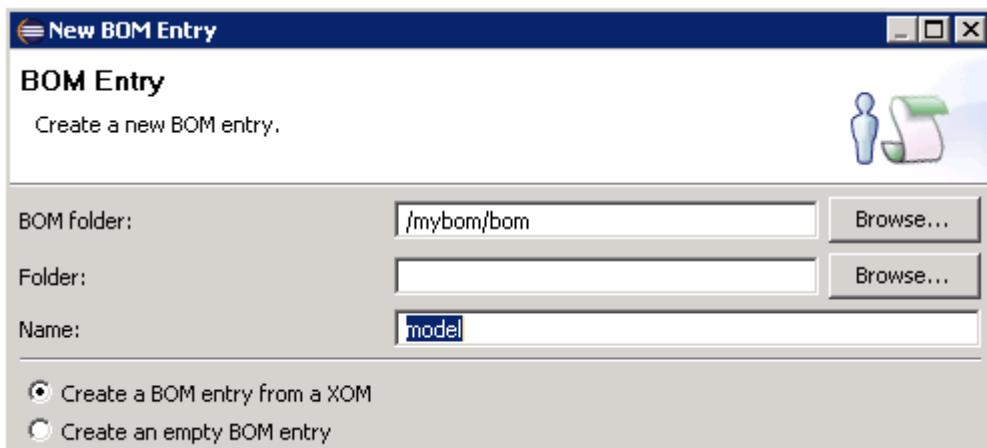
The **New BOM Entry** wizard opens.



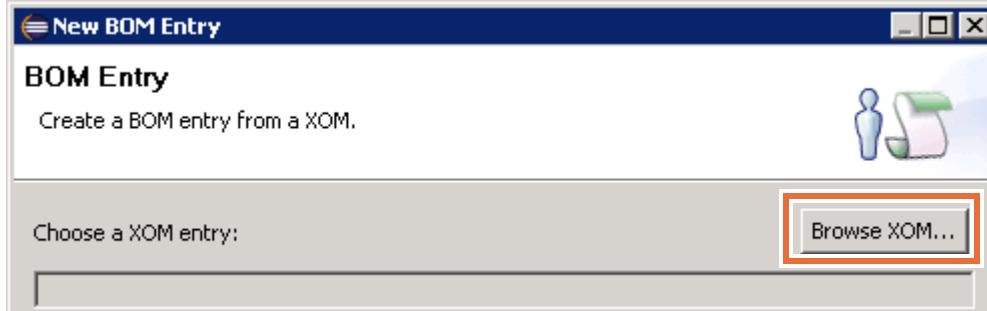
Information

BOM entries are the building blocks of the BOM. Each BOM entry defines a set of business elements in the BOM. You learn more about BOMs and BOM entries later in this course.

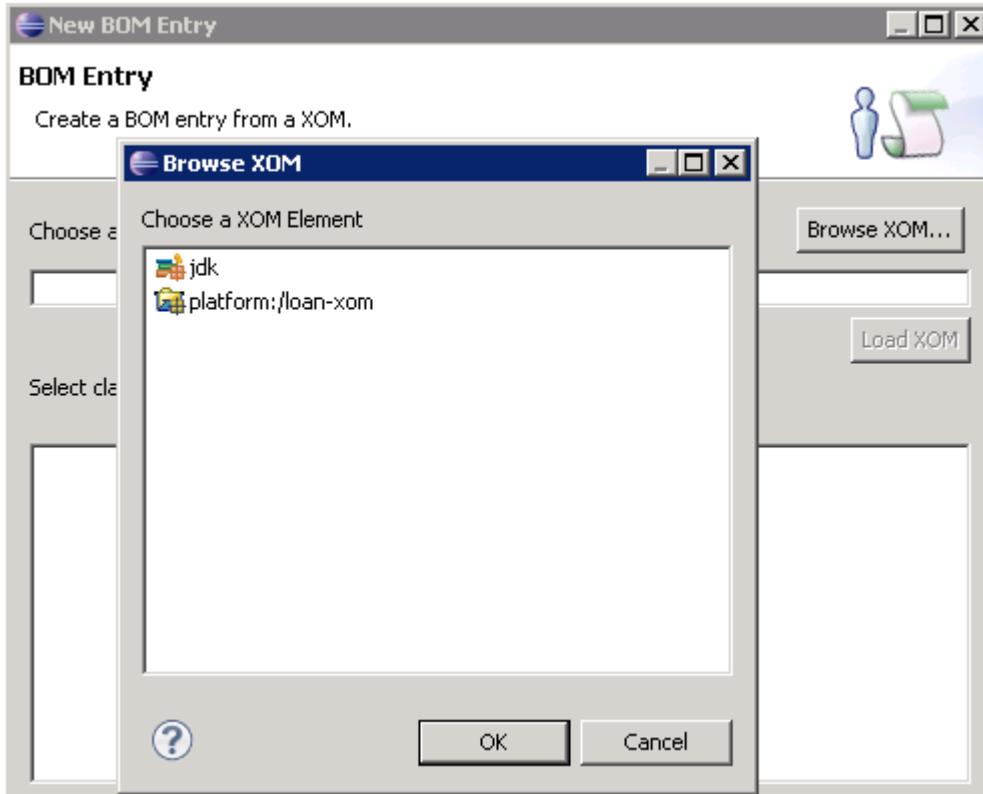
- 3. Make sure that the **Create a BOM entry from a XOM** option is selected, keep the default values for the other fields, and click **Next**.



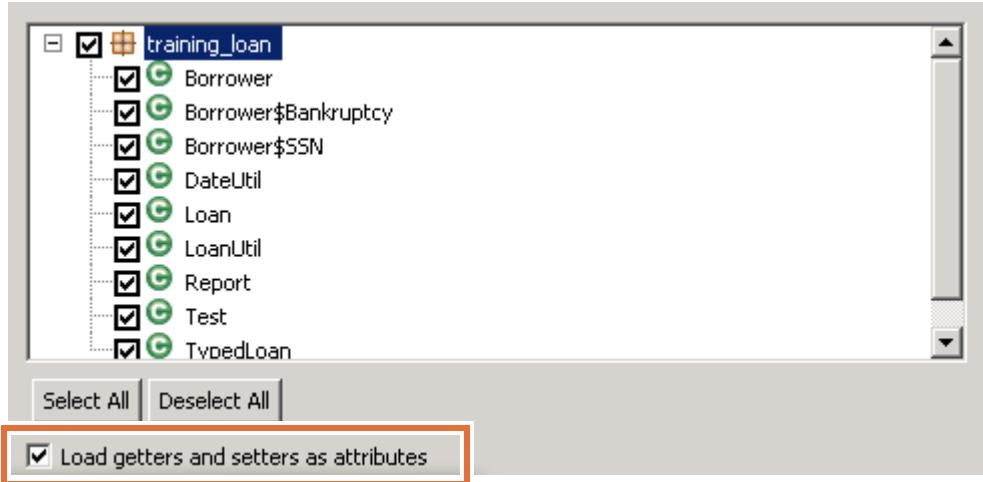
- 4. On the BOM Entry page, click **Browse XOM**.



- ___ 5. In the Browse XOM window, select **platform:/loan-xom**, and click **OK**.



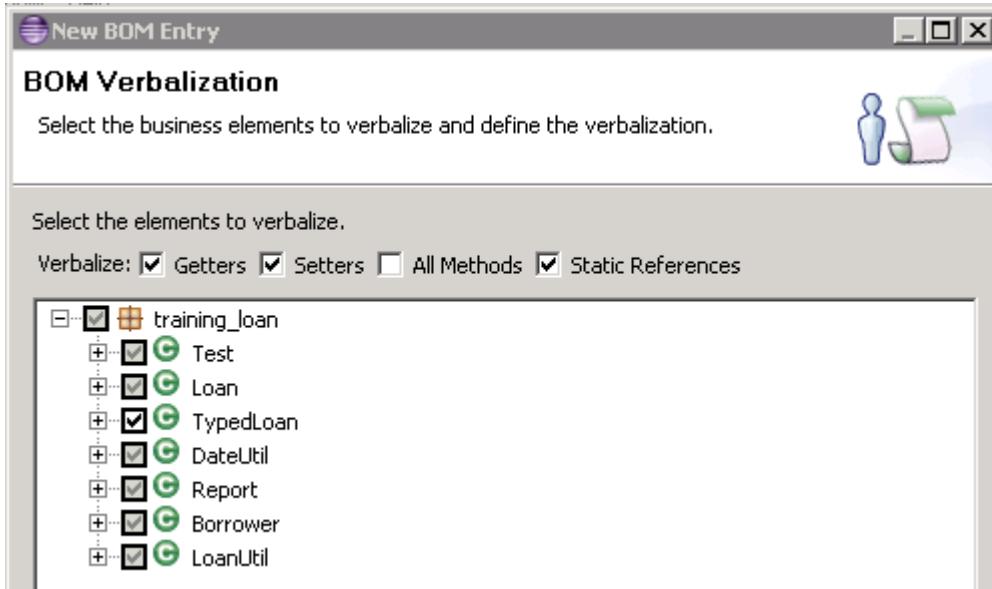
- ___ 6. Make sure that your BOM entry fully reflects the classes and methods in the XOM as follows:
- Expand **training_loan** and select all classes by clicking **Select All**.
 - Make sure that the **Load getters and setters as attributes** check box is selected.



This step ensures that all Java classes, attributes, and methods in the XOM are mapped to or have a corresponding BOM member.

- Click **Next**.

The BOM Verbalization page opens.



- 7. Click **Finish** to accept the default verbalization.



Information

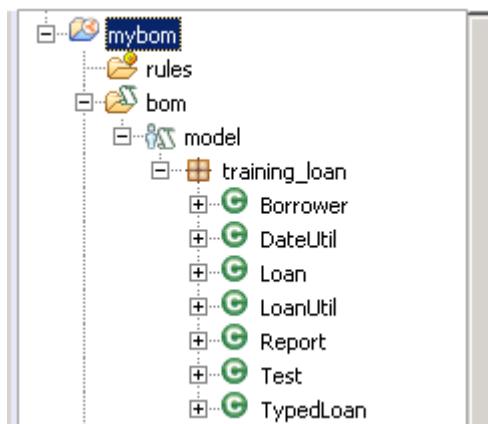
Verbalization attaches natural-language terms to the BOM elements to create the vocabulary for your business rules. You learn more about verbalization later in this course.

In your **mybom** project, the BOM entry that is called `model` is now available with a default vocabulary. You can view it in the Rule Explorer by going to **mybom > bom > model**.

2.4. Exploring the BOM

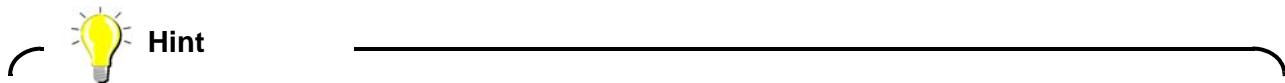
Next, you explore your BOM entry to see how the XOM elements are translated into the BOM entry.

- 1. In the `mybom` project, expand **bom > model > training_loan**.



The BOM entry contains a series of classes, including the `Borrower` class.

- ___ 2. Double-click the **Borrower** class in the Rule Explorer to open it in the BOM editor.



You can also right-click the **Borrower** class in the Rule Explorer, and click **Open With > BOM Editor**.

- ___ 3. In the **Class Verbalization** section, review the default verbalization of the **Borrower** class.

▼ Class Verbalization

the verbalization. the documentation.

Generate automatic variable

Term:

i the borrower, a borrower, the borrowers....

- ___ 4. In the **Members** section, select the **firstName** member of the **Borrower** class, and click **Edit** to open it.

▼ Members
Specify the members of this class.

- birthDate
- creditScore
- firstName**
- lastName
- latestBankruptcyChapter
- latestBankruptcyDate
- latestBankruptcyReason
- spouse
- SSN
- yearlyIncome

- ___ 5. In the **Member Verbalization** section, note the verbalization for **firstName** that is used in the **Navigation** section:

the first name of a borrower

▼ Member Verbalization

the verbalization.

a navigation phrase.

the subject used in phrases.

▼ Navigation : "the first name of a borrower" X

Template:

- ___ 6. Compare the navigation phrase to the **Template** field, and try to understand the use of braces.

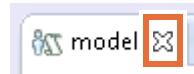
{first name} of {this}



Note

You learn more about the BOM, the BOM editor, the verbalization, and how to use the braces ({}) later in this course.

- ___ 7. Close the BOM editor.

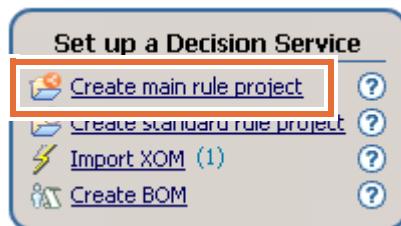


Section 3. Creating the main rule project for the decision service

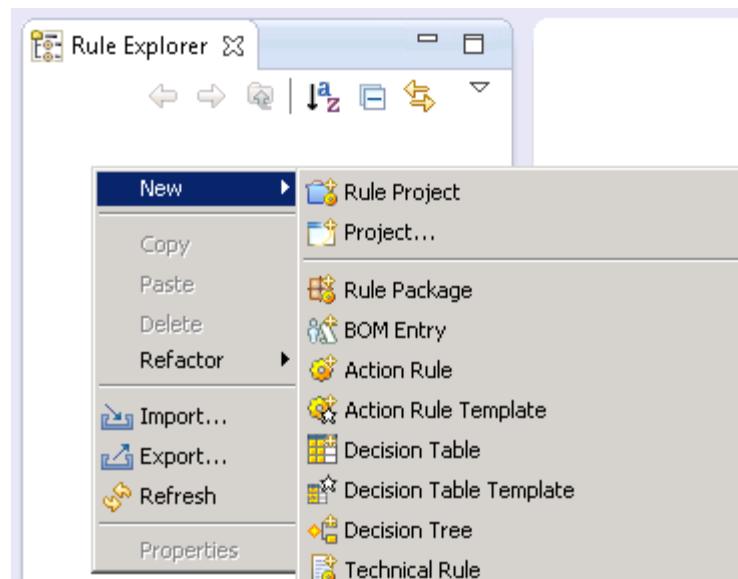
The main decision service project serves as the entry point to the decision service, and it references all of the other rule projects that are in the decision service.

In this section of the exercise, you create a main rule project for the decision service.

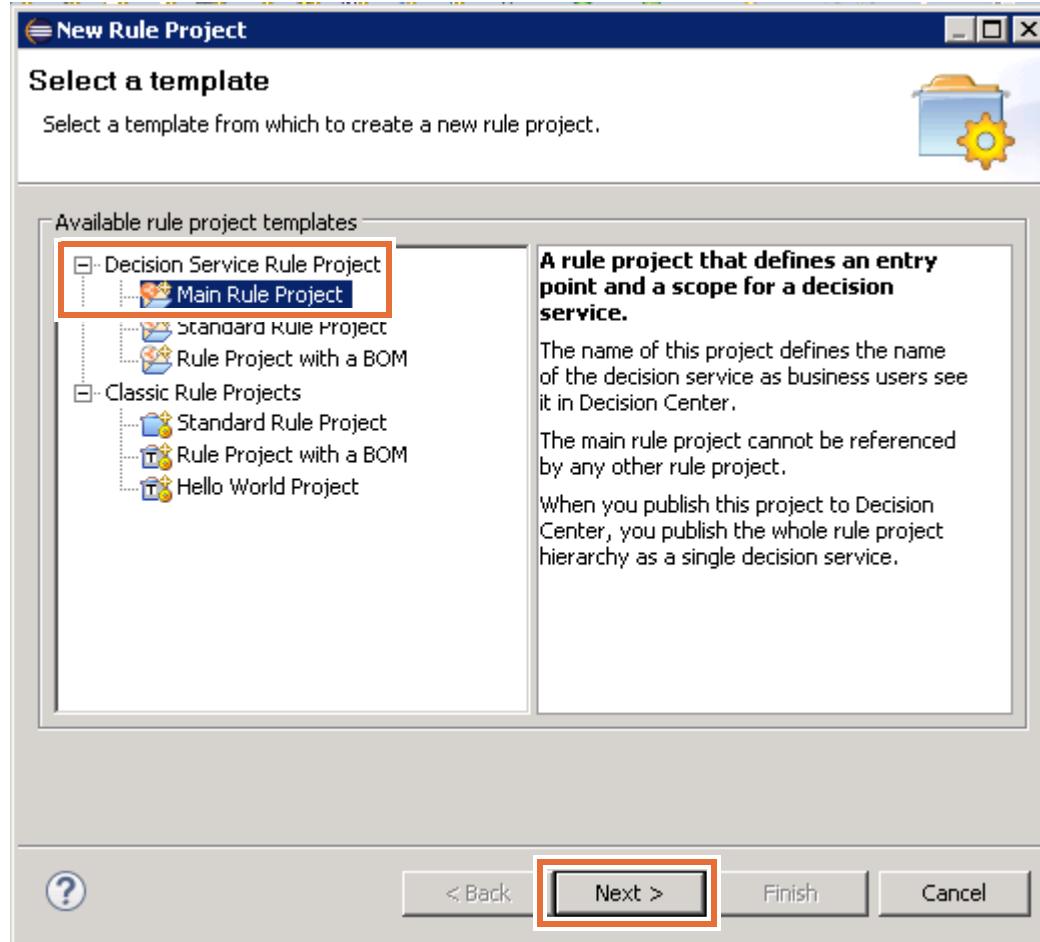
1. In the “Set up a Decision Service” task in the Decision Service Map pane, click **Create main rule project**.



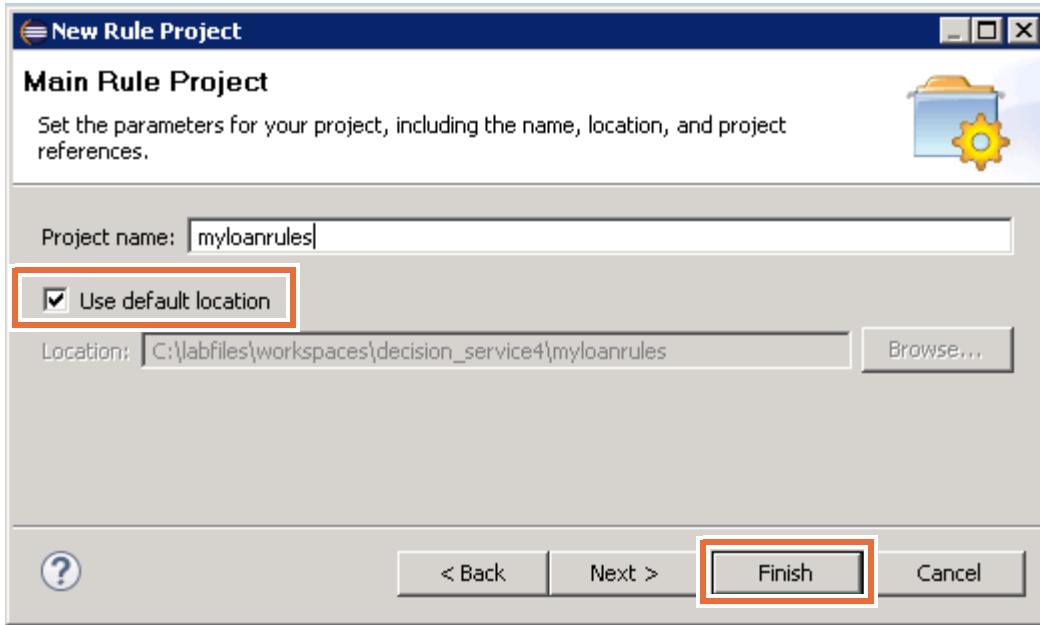
You can also open the New Rule Project window by going to **File > New > Rule Project** or by right-clicking anywhere in the Rule Explorer to open the menu and clicking **New > Rule Project**.



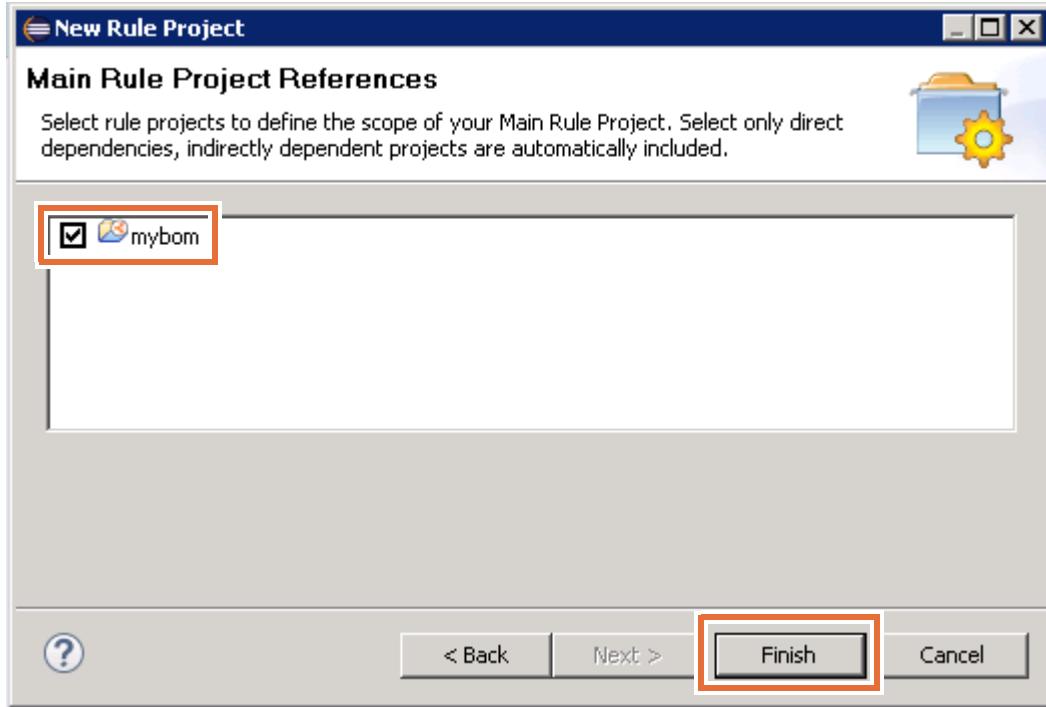
2. In the New Rule Project window, create a main rule project for a decision service.
- a. Select **Decision Service Rule Projects > Main Rule Project**, and click **Next**.



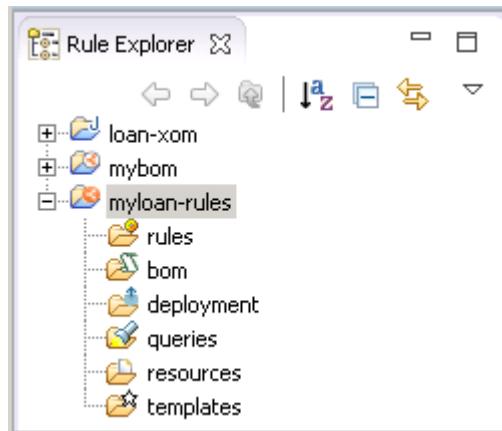
- b. In the **Project name** field, type: myloan-rules
- c. Make sure that the **Use default location** check box is selected, and click **Next**.



- __ d. On the Main Rule Project References page, select **mybom**, and click **Finish**.



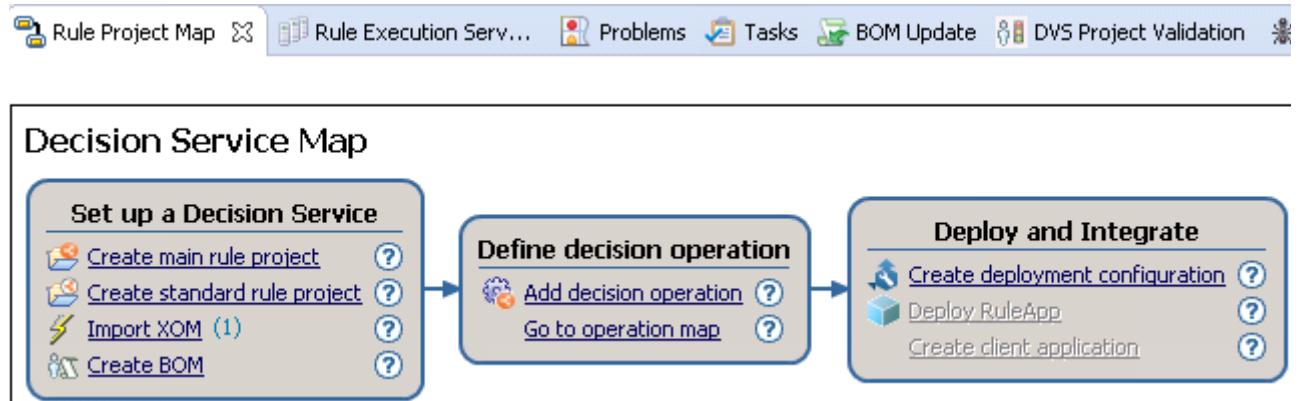
The `myloan-rules` project is now visible in the Rule Explorer.



3.1. Identifying the next tasks in decision service development

The decision service is the starting point of your business rule application development. Consider the next tasks of developing the application.

- __ 1. Select the `myloan-rules` decision service.
- __ 2. In the Decision Service Map, identify the newly enabled links:
 - In the **Define decision operation** part: **Add decision operation** and **Go to operation map**
 - In the **Deploy and Integrate** part: **Create deployment configuration**



Section 4. Creating and defining the decision operation

In this section, you work with the “Define decision operation” part of the Decision Service Map.

A decision operation defines the decision-making logic and the input and output data (or ruleset parameters) for a decision. The ruleset parameters of the decision service function as the interface between your decision service and the calling application.



Requirements

Discussions with the business analysts confirm that the business decision should be based on borrower and loan information. The decision output should update the loan information and provide a report.

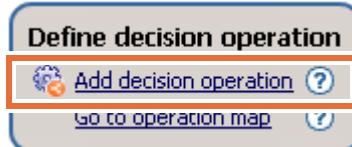
You are to implement the following ruleset parameters to pass the data between the calling application and the rule engine:

Name	Type	Direction	Verbalization
borrower	training_loan.Borrower	IN	the borrower
loan	training_loan.Loan	IN_OUT	the loan
report	training_loan.Report	OUT	the loan report

These parameters do not have default values.

4.1. Creating the decision operation

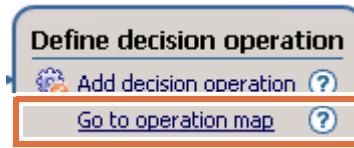
- 1. Make sure that the **myloan-rules** folder is selected.
- 2. In the “Define decision operation” part of the Decision Service Map, click **Add decision operation**.



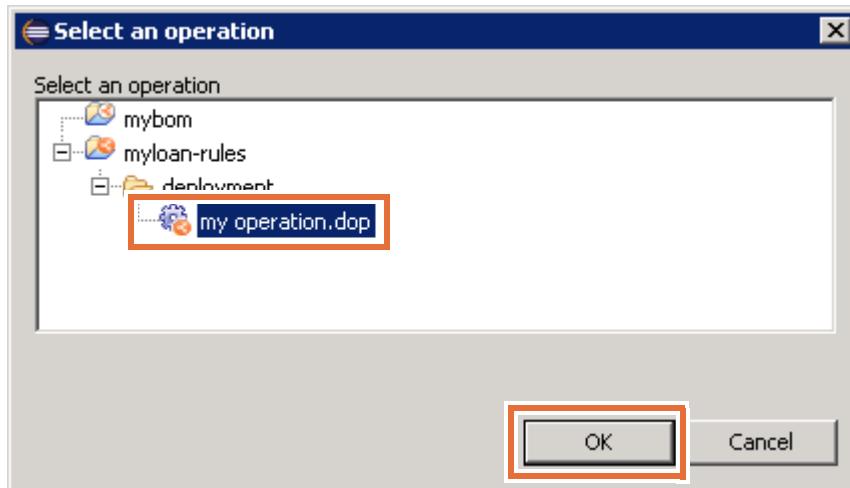
- 3. In the New Decision Operation wizard, in the **Name** field, type: **my operation** and click **Next**.
- 4. In the Source Rule Project window, ensure that **myloan-rules** is selected, and then click **Finish**.

You can see the **my operation** decision operation in the **myloan-rules > deployment** folder.

5. In the “Define decision operation” part of the Decision Service Map view, click **Go to operation map**.

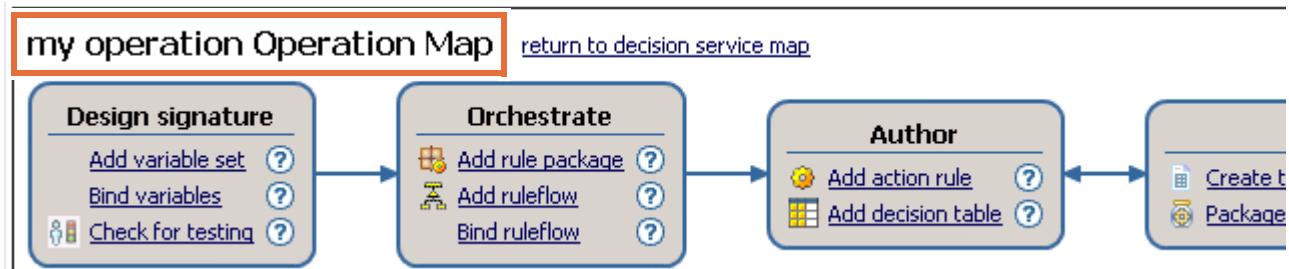


6. Select **myloan-rules > deployment > my operation.dop** and click **OK**.



The Decision Service Map view switches to the Operation Map view. The Operation Map has the following parts:

- Design signature
- Orchestrate
- Author
- Test



4.2. Creating ruleset variables for ruleset parameters

In this section of the exercise, you create ruleset variables for the decision service. You map these variables to ruleset parameters later in this exercise.

- 1. In the “Design signature” part of the Operation Map, click **Add variable set**.



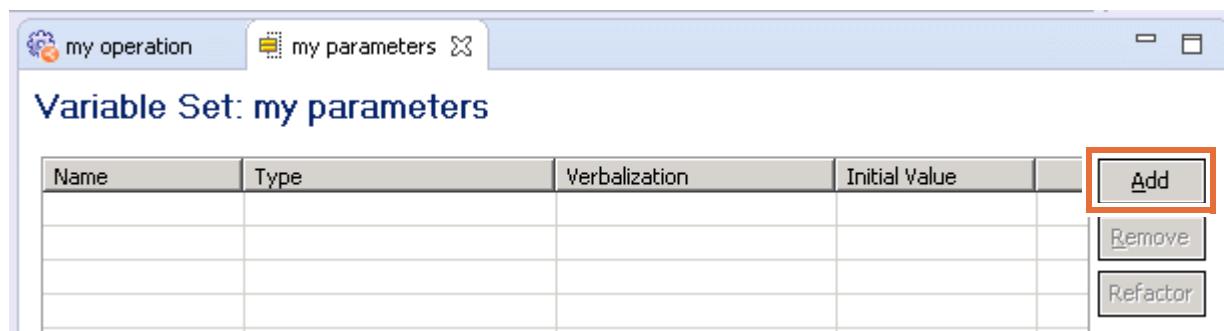
- 2. In the New Variable Set wizard, in the **Name** field, type: my parameters

- 3. Click **Finish**.

The Variable Set editor opens.

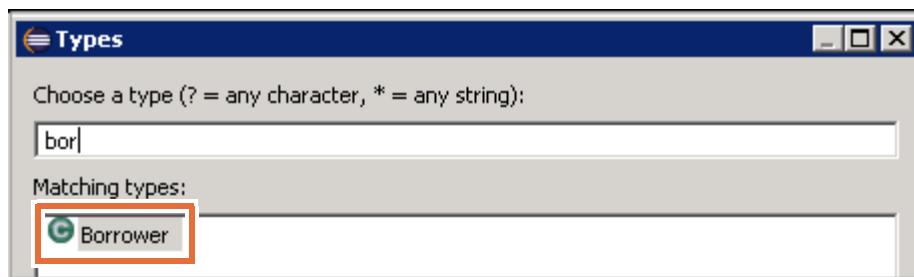
- 4. In the Variable Set editor, define the `borrower` parameter.

- a. Click **Add**.



A new row is displayed with default values. You can type over the values as described next.

- b. In the **Name** column, overwrite the value (`myVar`) by typing: `borrower`
- c. In the **Type** column, click the cell, and then click the ellipsis (...) to open the Types window.
- d. Start typing `borrower` to find the Borrower type in the Matching Types list, and then double-click **Borrower** to select it.



**Hint**

When you set the **Type** value, you can type Borrower and Rule Designer finds the training_loan.Borrower type for you.

- ___ e. In the **Verbalization** field, overwrite the value by typing: the borrower
- ___ 5. Define the loan parameter.
 - ___ a. Click **Add**.
 - ___ b. Change the variable values to:
 - **Name:** loan
 - **Type:** Loan
 - **Verbalization:** the loan

**Hint**

When you set the **Type** value, you can type Loan and Rule Designer finds the training_loan.Loan type for you.

- ___ 6. Define the report parameter.
 - ___ a. Click **Add**.
 - ___ b. Change the variable values to:
 - **Name:** report
 - **Type:** Report
 - **Verbalization:** the report

**Hint**

When you set the **Type** value, you can type Report and Rule Designer finds the training_loan.Report type for you.

- ___ 7. Save your work (Ctrl+S).

4.3. Binding the variables to ruleset parameters

In this section, you bind the variables that you defined to ruleset parameters.

- 1. In the “Design signature” part of the Operation Map, click **Bind variables**.



The **Signature** tab of the Decision Operation editor opens.

- 2. Map the variables to the input and output parameters.
 - a. Expand **my parameters** in the **Eligible variables** section of the Decision Operation Signature editor to see the `borrower`, `loan`, and `report` variables.
 - b. Drag `borrower` to the **Input Parameters** table.
 - c. Drag `loan` to the **Input-Output Parameters** table.
 - d. Drag `report` to the **Output Parameters** table.

The decision operation signature now has `borrower` in the Input Parameters table, `loan` in the Input-Output Parameters table, and `report` in the Output Parameters table.

Input Parameters

Define the parameters required to call the execution.

Parameter name	Verbalization	Type
<code>borrower</code>	the borrower	<code>training_loan.Borrower</code>

Input - Output Parameters

Define the parameters that are required, modified, and then returned by the execution.

Parameter name	Verbalization	Type
<code>loan</code>	the loan	<code>training_loan.Loan</code>

Output Parameters

Define the parameters that are initialized and returned by the execution.

Parameter name	Verbalization	Type
<code>report</code>	the report	<code>training_loan.Report</code>

- 3. Save your work, and close the **my operation** editor.

4.4. Creating a local ruleset variable

Ruleset parameters carry the objects that are passed between the calling application and the rule engine. These objects are evaluated by the rules during rule execution.

To avoid directly handling these objects, you can define ruleset variables to manipulate the objects during rule execution. When rule execution is complete, the final value of the ruleset variable can be assigned back to an output ruleset parameter and returned to the external application.



Requirements

In discussion with the business analysts, they clarify that the decision results should be returned in the `training_loan.Report` parameter (which is defined as an OUT ruleset parameter).

Rather than manipulating this parameter directly during rule execution, you implement the business request by creating a ruleset variable that is called `loanApproved` that can be updated during rule evaluation. When rule execution is complete, the value that is stored in `loanApproved` can be transferred to the `Report` parameter and passed back to the calling application.

To meet these requirements, you create a local ruleset variable that is called `loanApproved`, which has the following attributes:

- **Name:** `loanApproved`
- **Type:** `boolean`
- **Verbalization:** the loan is approved
- **Initial Value:** `true`



Note

To define the `loanApproved` variable, you can use the same ruleset variables table, **my parameters**, that you used to define the variables for the ruleset parameters.

However, you do not bind `loanApproved` to a ruleset parameter.

- 1. Go back to the **my parameters** ruleset variables table.
- 2. Add the `loanApproved` ruleset variable.
 - a. In the Variable Set editor, click **Add**.
 - b. In the **Name** column, type over `myVar` to enter: `loanApproved`
 - c. Click the **Type** field and click the ellipses (...).
 - d. In the Types window, start typing: `boolean`
 - e. Select the lowercase `boolean` type option.
 - f. In the **Verbalization** column, type over `myVar` to enter: the loan is approved
 - g. In the **Initial Value** column, type: `true`
- 3. Save your work (`Ctrl+S`).
- 4. Click the Problems view to make sure that no errors are listed.
- 5. Close the variable set editor by clicking the **X** for that window.



Variable Set: my parameters

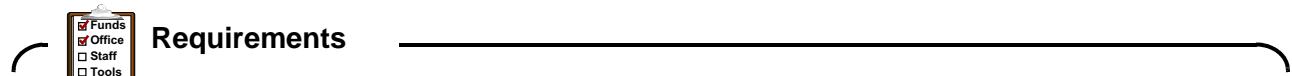
Name	Type	Verbalization	Initial Value
<code>borrower</code>	<code>training_Join.Borrower</code>	the borrower	
<code>loan</code>	<code>training_Join.Loan</code>	the loan	
<code>report</code>	<code>training_Join.Report</code>	the report	
<code>loanApproved</code>	<code>boolean</code>	the loan is approved	<code>true</code>

Section 5. Creating rule packages

Recall in the Operation Map, the Orchestrate part has a link called **Add rule package**.



Your next step is to add rule packages to the `myloan-rules` project.



The business analysts provide you with an initial outline of how the rules can be organized and the naming conventions to use. Based on these requirements, the rule artifacts should be logically managed within the following packages and subpackages:

- computation
- eligibility
- insurance
- validation
 - validation.borrower
 - validation.loan

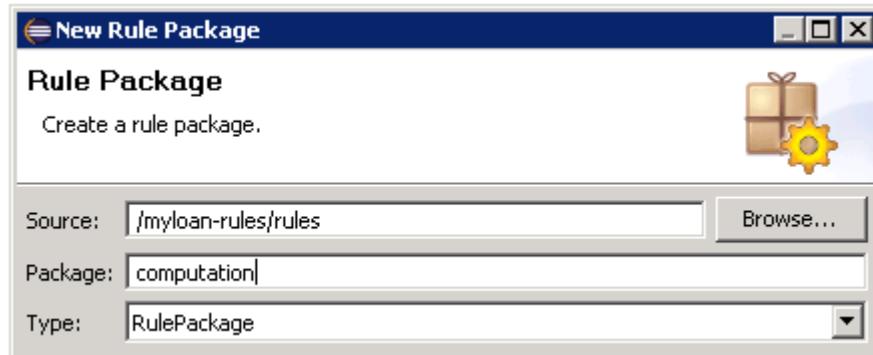
These rule packages also outline the smaller decisions that must be taken before determining whether to approve the loan.

You can add packages by using the Operation Map link or by clicking the **New > Rule Package** menu item.

- ___ 1. Click the **myloan-rules** project and make sure that the Operation Map is open.
 - ___ 2. Create the **computation** package.
- ___ a. In the Orchestrate part of the Operation Map, click the **Add rule package** link.

The New Rule Package window opens.

- __ b. In the **Package** field of the Rule Package window, enter `computation` and click **Finish** (or press Enter).



The `computation` package is created in your rule project.

- __ 3. Add the remaining packages by using the instructions in Step 2:

- `eligibility`
- `insurance`
- `validation`

- __ 4. Create the `borrower` and `loan` subpackages of the `validation` package.

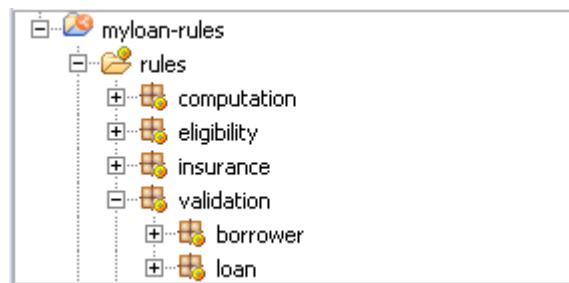
- __ a. In the Orchestrate part of the Operation Map, click the **Add rule package** link.
 __ b. In the **Package** field, type `validation.borrower` and press Enter.
 __ c. Add another package, type `validation.loan` in the **Package** field, and press Enter.



Note

The name of a package is its full path, with a **period** (.) to separate subpackages.

The rule project now contains all the required rule packages.



Notice the next link in the Orchestrate part of the Rule Project Map: **Add ruleflow**.

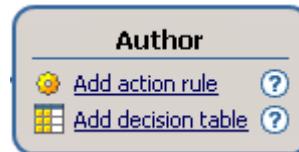


You work with ruleflows in Exercise 5, "Working with ruleflows".

Section 6. Publishing from Rule Designer to Decision Center

In this part of the exercise, you publish the decision service from Rule Designer to Decision Center. Notice the links in the Author part of the Operation Map.

- **Add action rule**
- **Add decision table**



The decision service is now ready for the rule authors to start working with rules for this project.

By making the decision service available to business users through the Decision Center consoles, rule authors can begin their work on the rules, and thus start the feedback loop on your implementation.

6.1. Publishing the rule project to Decision Center

In this section, you publish the `myloan-rules` decision service to Decision Center to create the corresponding project in Decision Center.

- 1. If the sample server is not already started, start it now by clicking **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Start server**.



Reminder

Decision Center consoles run on the sample application server that is provided for this course. The sample server must be running before you can access Decision Center.

You can check whether the sample server is running by opening one of the Decision Center consoles in a browser, for example: <http://localhost:9080/teamserver>

If the console page opens correctly, the server is running.

For IBM ODM on Cloud users

When you synchronize between Rule Designer and the ODM on Cloud environment, enter the URL of your IBM ODM on Cloud instance in the **URL** field instead of `localhost`.

- 2. In the Rule Designer Rule Explorer view, right-click the **myloan-rules** decision service, and click **Decision Center > Connect**.

The Decision Center configuration wizard opens.

3. Enter the following values in the Decision Center Configuration window fields.

- **URL:** `http://localhost:9080/teamserver`
- **User name:** `rtsAdmin`
- **Password:** `rtsAdmin`
- **Data source:** (Leave the field empty)

**Important**

Make sure that you use the correct URL and port for your environment.

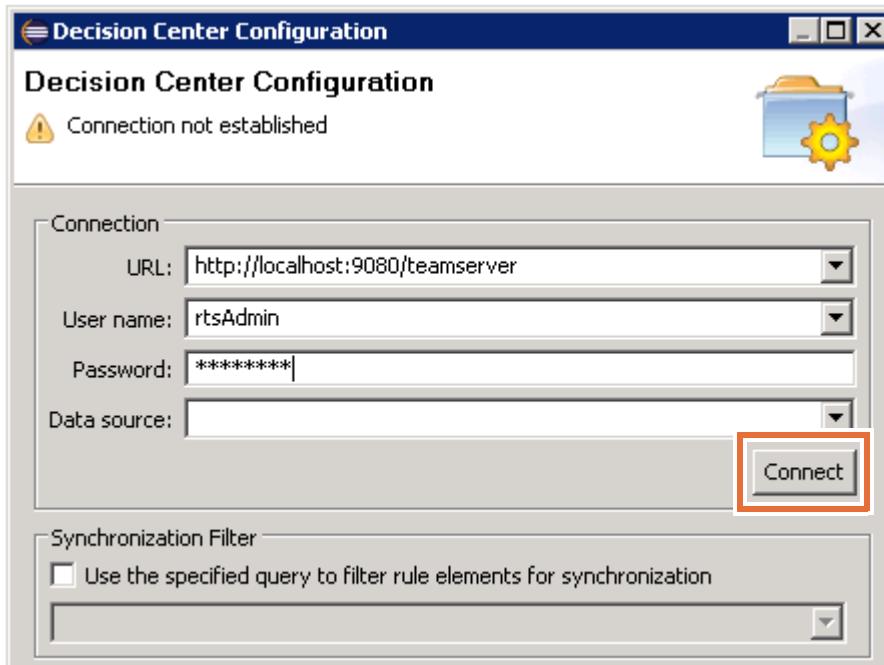
For IBM ODM on Cloud users

When you synchronize between Rule Designer and the ODM on Cloud environment, enter the URL of your IBM ODM on Cloud instance in the **URL** field instead of `localhost` and make sure to use “`https`”.

You must also specify which environment you are publishing to, either development, test, or production. The URL might look like the following example:

`https://odmdemo1.bpm.ibmcloud.com/odm/dev/teamserver`

4. Click **Connect**.



5. After the connection is successfully established, click **Next**.

6. In the Synchronization Settings window, click **Next**.

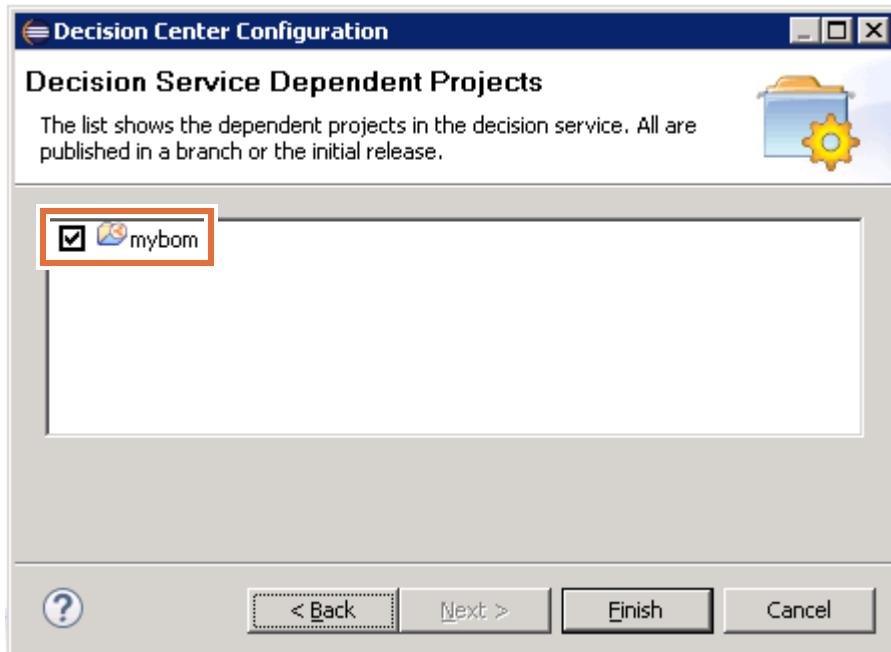


Information

In the Synchronization Settings window, you can choose whether the decision service will be governed in the Decision Governance Framework.

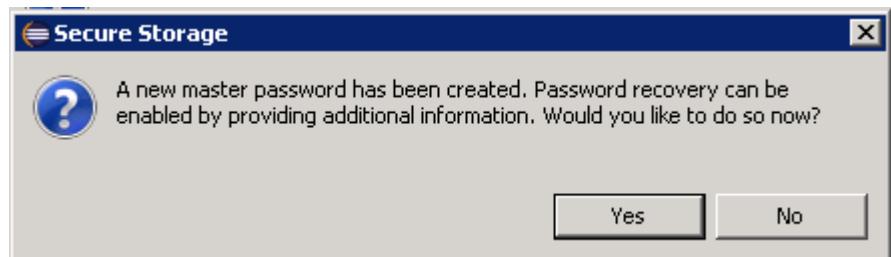
You do not select the **Use Decision Governance Framework** option for this exercise.

7. In the Decision Service Dependent Projects window, make sure that **mybom** is selected, and click **Finish**.



Troubleshooting

If you see a Secure Storage window, click **No**.

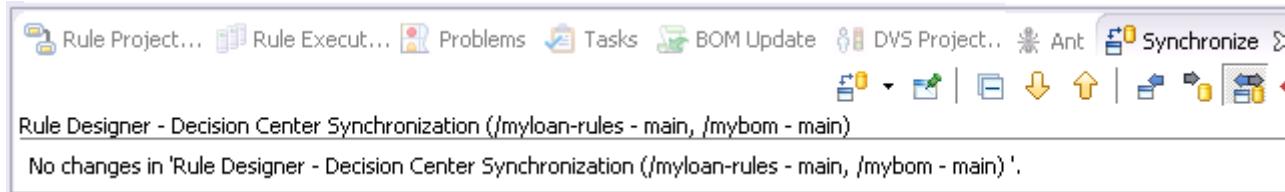


After synchronization completes, you are notified that no changes were found.



- ___ 8. Click **OK** to close the window.
- ___ 9. When prompted to switch to the Team Synchronizing perspective, click **No**.

You do not need to switch to the Team Synchronizing perspective because it is empty. However, you can see the Synchronize view in the lower part of the perspective.



Because you just created this project in Decision Center, no differences exist between the project in Rule Designer and the project in Decision Center.



Important

After you publish the `myloan-rules` decision service to Decision Center, do not disconnect from Decision Center so that you do not have to establish this connection again in later steps.

6.2. Opening the project in the Decision Center Business console

Sign in to the Business console to view the rule artifacts that you published from Rule Designer.

- ___ 1. Open the Decision Center Business console.
 - ___ a. Double-click the **Decision Center Business console** desktop shortcut or click **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Decision Center Business console**.



Information

Alternatively, you can open a web browser at the address that you used to connect Rule Designer and Decision Center:

`http://localhost:9080/decisioncenter`

Make sure that you use the correct URL and port for your environment.

- ___ b. Sign in with `rtsAdmin` as both the **User name** and the **Password**.
- ___ 2. If you are not on the **Library** tab, click the **Library** tab.
- ___ 3. On the **Library** tab, notice the new `myloan-rules` decision service is now listed.



Decision Services

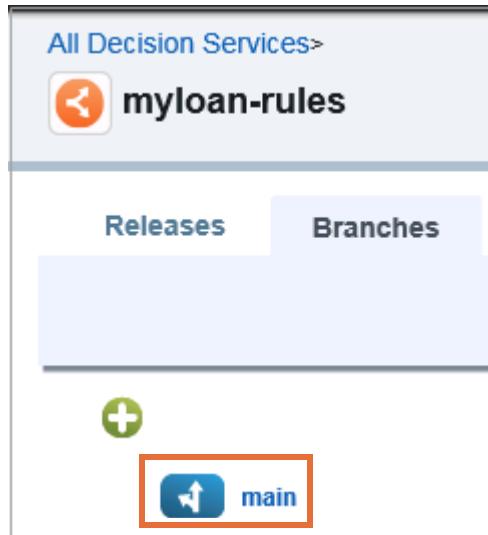
Date | Name

 myloan-rules Created by rtsAdmin on Jan 11, 2016	 Value Edi Created by rts
--	--

- ___ 4. Click **myloan-rules** to open it.

The `myloan-rules` decision service opens on the **Releases** tab. Because `myloan-rules` is not a governed decision service, it has no releases.

- ___ 5. Click **Branches**, and then click **main** to open the main branch of `myloan-rules`.



- ___ 6. If you are not on the **Decision Artifacts** tab, click **Decision Artifacts**.
- ___ 7. On the left pane of the **Decision Artifacts** tab, expand `myloan-rules`.

You can see the different rule packages (folders) that you created for the decision service.

The screenshot shows the IBM Operational Decision Manager Business console interface. At the top, there's a header with the project name "myloan-rules" and a "main" tab. Below the header, there are tabs for "Decision Artifacts", "Tests", "Simulations", "Deployments", and "Schemas". Under "Decision Artifacts", there are buttons for "All Projects" and "All Types". The left sidebar lists rule packages: "myloan-rules" (selected), "Operations" (with a count of 1), "computation" (selected), "eligibility", "insurance", and "validation". Below this is a "mybom" section. The main panel shows a table with columns for "Name" and "computation". There are icons for back, forward, add, edit, and star. A "Filter:" button is present. A message at the bottom says "There are no items to display".

- ___ 8. Click one of the rule packages, such as the **computation** folder.

You can see that the decision service does not contain any rules.

- ___ 9. Log out and close the Business console.

- ___ a. Click **rtsAdmin** in the upper-right corner.
- ___ b. Click **Log out**.
- ___ c. Close the Business console window.

The next task for a rule author is to complete the decision service by adding the rules. However, before doing anything further in this console, you switch to the Enterprise console to see the decision service in that environment.

6.3. Opening the project in the Decision Center Enterprise console

Sign in to the Enterprise console to view the decision service that you published from Rule Designer.

- ___ 1. Open the Decision Center Enterprise console.
 - ___ a. Double-click the **Decision Center Enterprise console** shortcut on the desktop, or click **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Decision Center Enterprise console**.



Information

Alternatively, you can open a web browser at the address that you used to connect Rule Designer and Decision Center:

`http://localhost:9080/teamserver`

Make sure that you use the correct URL and port for your environment.

- ___ b. Sign in by typing `rtsAdmin` in both the **Username** and **Password** fields.

On the **Home** tab of the Decision Center Enterprise console, you can see the different projects and decision services that are accessible to business users.

- ___ 2. Select the **myloan-rules** decision service.
 - ___ a. Select **Work on a decision service**.
 - ___ b. In the **Decision service in use** field, select **myloan-rules** (and leave the other fields unchanged).

- ___ 3. Click the **Explore** tab.
- ___ 4. In the Smart Folders list, expand the **validation** folder.

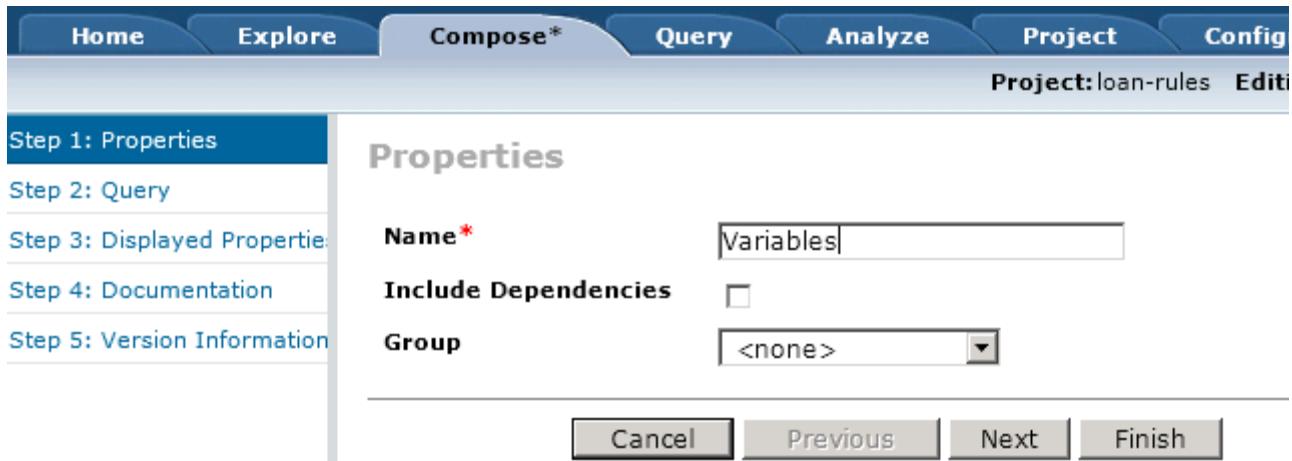
The **Explore** tab lists all the artifacts of the `myloan-rules` decision service. Artifacts are organized within smart folders. The project does not contain any rules yet.

By default, all business rules are under the **Business Rules** smart folder, and organized in subfolders that correspond to rule packages in Rule Designer.

Recall that you created a ruleset variable for this project, but that variable is not visible. You can create a smart folder to make the variable visible to the business users.

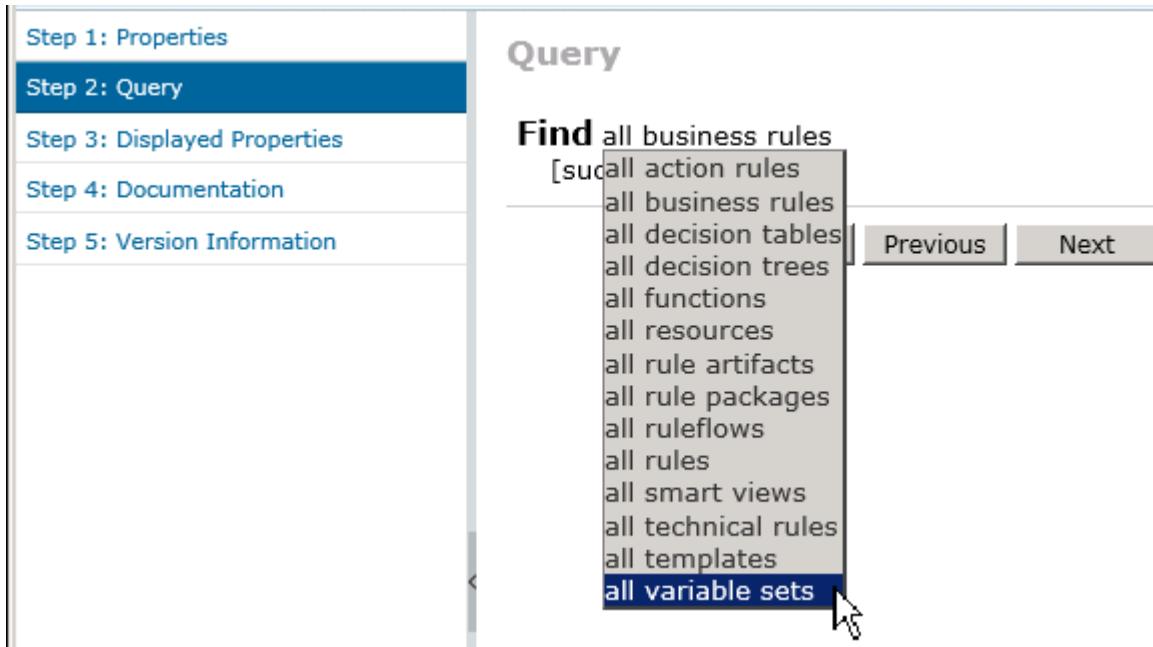
- ___ 5. Click the **Compose** tab.
- ___ 6. In the **Start from a type** pane, click **Smart Folder**, and click **OK**.

7. Name the smart folder Variables and click **Next**.



The smart folder is based on a query. You can choose which artifacts the query should list.

8. In the **Find** statement, click **all business rules** and select **all variable sets**.



9. Click **Finish**.

The ruleset variables that you defined on the rule project is now visible to the business users in the **Variables** smart folder on the **Explore** tab. Business users can now recognize it when they begin rule authoring.

6.4. Modifying the ruleset variable in Decision Center

- 1. On the **Explore** tab, select the **my parameters** variable set in the table and click **Edit**.

Variables	
<input type="button" value="New"/> <input type="button" value="Details"/> <input style="border: 2px solid red;" type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="Copy"/> <input type="button" value="Lock"/> <input type="button" value="Unlock"/> <input type="button" value="Release lock"/>	
Display by 10	
<input checked="" type="checkbox"/>	Actions Name
<input checked="" type="checkbox"/>	<input type="button" value="my parameters"/>
	Last Changed By rtsAdmin

The **Compose** tab opens.

- 2. Click **Next** to open the Variables page.
— 3. Select the **loanApproved** variable and click **Edit**.

Step 1: Properties					
Step 2: Variables					
Step 3: Documentation					
Step 4: Version Information					
<input type="checkbox"/>	Name	Type	Type	Verbalization	Initi
<input type="checkbox"/>	borrower	Variable	training_loan.Borrower	the borrower	
<input type="checkbox"/>	loan	Variable	training_loan.Loan	the loan	
<input type="checkbox"/>	report	Variable	training_loan.Report	the report	
<input checked="" type="checkbox"/>	loanApproved	Variable	boolean	the loan is approved	true
<input type="button" value="New"/> <input style="border: 2px solid red;" type="button" value="Edit"/> <input type="button" value="Delete"/>					
<input type="button" value="Cancel"/> <input type="button" value="Previous"/> <input type="button" value="Next"/> <input type="button" value="Finish"/>					

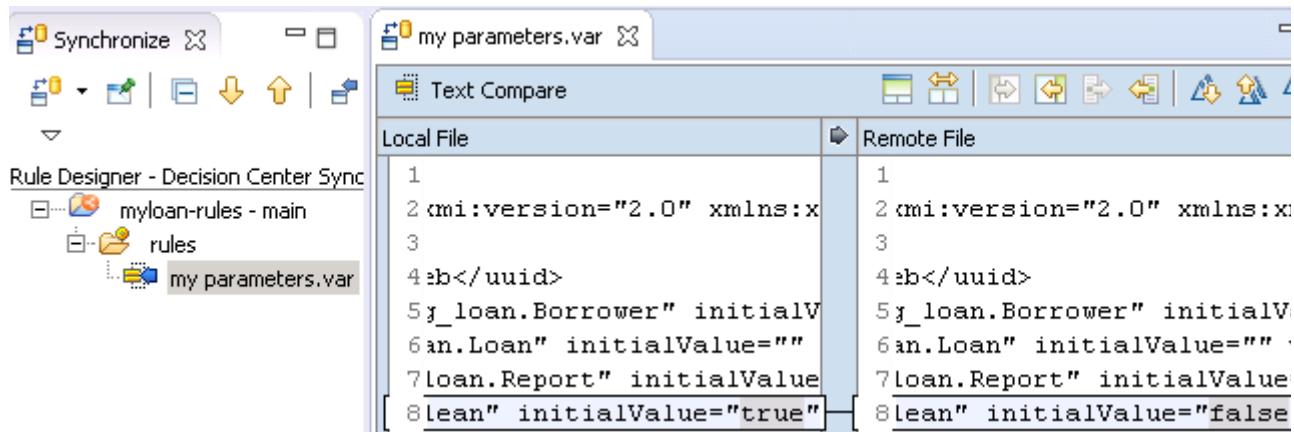
- 4. In the **Initial Value** field, change the value from `true` to `false` and click **Apply**.
— 5. Click **Finish**.

The variable is updated and opens in the Details page.

6.5. Synchronizing Rule Designer with Decision Center

- 1. Go back to Rule Designer in the Rule perspective.
— 2. Synchronize the `myloan-rules` decision service with Decision Center.
 — a. Right-click `myloan-rules` and click **Decision Center > Synchronize with Decision Center**.
 — b. In the Synchronization Settings window, click **Finish**.
 — c. When prompted to switch to the Team Synchronizing perspective, click **Yes**.

- ___ 3. In the Synchronize view, expand **myloan-rules - main > rules** to see that your ruleset variable is changed.
- ___ 4. Double-click the variable to open it in the Text Compare view and check your changes.



- ___ 5. Close the Text Compare view.



Information

In the Synchronize view, entries show if rule artifacts were modified locally, remotely, or both concurrently. The color and direction of the arrow in the entry icon indicates where the modification occurred, and what type of action is possible.

Entry	Modification source	Expected actions
Black entries with an arrow that points to the right	A change occurred in Rule Designer.	<p>Publish to Decision Center what is in Rule Designer: Right-click the rule artifact, and click Publish.</p> <p>Select Override and Update when you want to override the changes, keep the version from Decision Center, and update Rule Designer.</p>
Blue entries with an arrow that points to the left	A change occurred in Decision Center.	<p>Update Rule Designer with what is in Decision Center: Right-click the rule artifact, and click Update.</p> <p>Select Override and Publish when you want to override the changes, keep the version from Rule Designer, and publish it again to Decision Center.</p>
Red entries with double arrows	Changes occurred in both Rule Designer and in Decision Center.	<p>Automatic merging is not possible. Decide how to handle this conflict.</p> <p>If you want to update Rule Designer with the changes made in Decision Center, select Override and Update.</p> <p>If you want to keep the changes from Rule Designer and publish them to Decision Center, select Override and Publish.</p>

In this case, after consulting with the rule author to confirm the reasons for this change, you agree that the default value should return to `true`.

- 6. Override the change by right-clicking the variable and clicking **Override and Publish**.

If prompted about conflicts, click **Yes** to confirm that you want to overwrite the changes.

After the publication is finished, no changes are listed in the Synchronize view.

Section 7. Deleting a decision service from Decision Center

When synchronization is not a viable option, or for some reason you must delete an entire decision service, you can erase the decision service from the Decision Center database.

- ___ 1. If you closed the Decision Center Enterprise console, sign in again with `rtsAdmin` as the user name and password.
- ___ 2. On the **Home** tab, make sure that **Work on a decision service** is selected, then select the `myloan-rules` decision service as the decision service in use.
- ___ 3. To verify that your project was successfully updated from Rule Designer, click the **Explore** tab, and in the **Variables** folder, click **my parameters** to open the Details page.
In the table of variables, you can see the initial value for the `loanApproved` variable is set to `true`.
- ___ 4. Go to the **Configure** tab and in the **Administration** section, click **Erase Current Decision Service**.
- ___ 5. Click **Yes** in the window to confirm the project deletion.
- ___ 6. Sign out and close the Enterprise console.

Section 8. Importing a decision service from Decision Center

In this part of the exercise, you import a decision service in to Rule Designer from an existing decision service in Decision Center.

 Requirements

Business analysts indicate that they worked with another team and developed a complementary decision service, called Miniloan Service. The latest version of Miniloan Service is in Decision Center. Business analysts ask you to import this decision service in Rule Designer to look at it.

8.1. Creating a project in Rule Designer from Decision Center

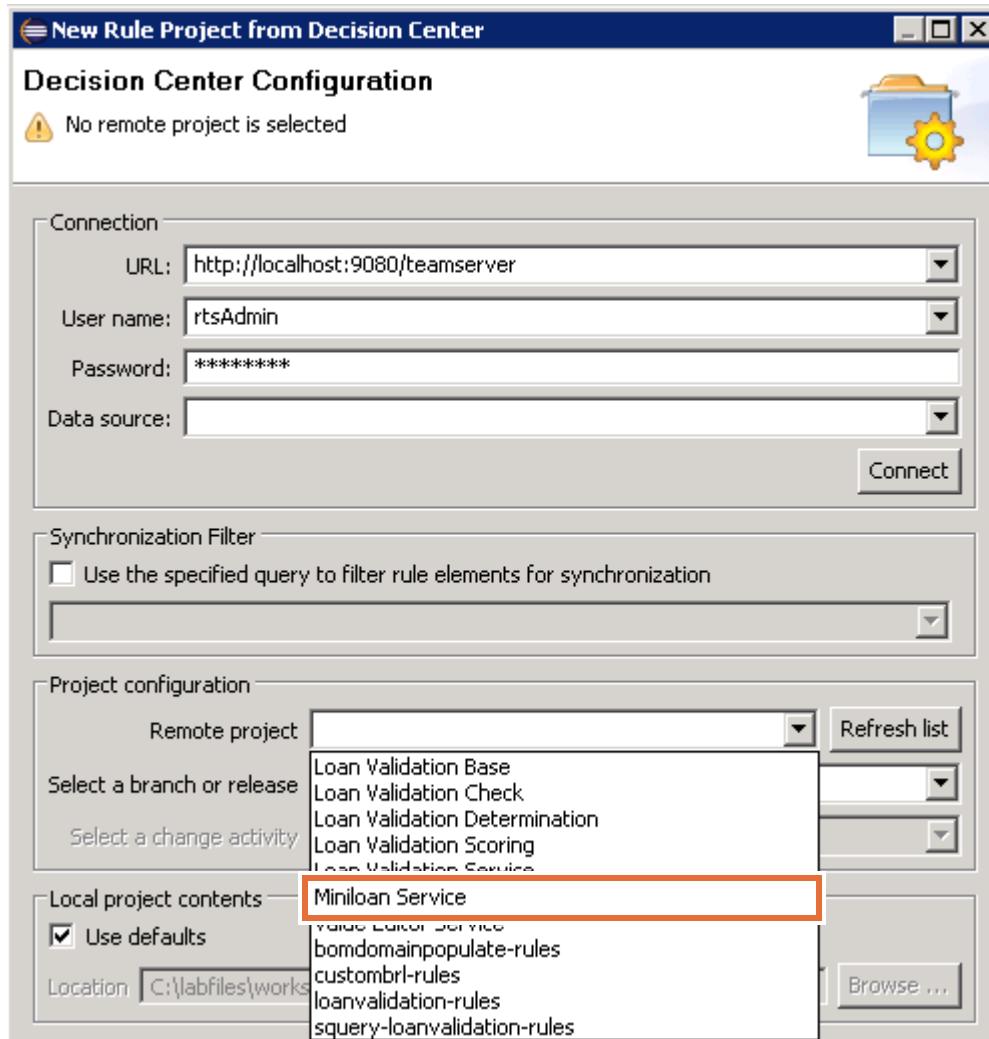
- ___ 1. Go back to Rule Designer and make sure that you are in the Rule perspective.
- ___ 2. Click **File > New > Project**, select **Rule Project from Decision Center**, and click **Next**.



The New Rule Project from Decision Center wizard opens.

- ___ 3. Verify the connection entries, which should still be available from the previous exercise.
 - **URL:** `http://localhost:9080/teamserver`
 - **User name:** `rtsAdmin`
 - **Password:** `rtsAdmin`
- ___ 4. Click **Connect**.

5. In the **Project configuration** section, select **Miniloan Service** from the **Remote project** list to import it in Rule Designer.



Note

If you cannot see the project in the list, click **Refresh List**.

The **Remote project** list shows all the rule projects in Decision Center for which you have the **View** permission. This permission depends on the **user name** that you used to connect.

Here, because you are connected as `rtsAdmin`, the Decision Center administrator, you can see all the rule projects in Decision Center.

6. In the **Select a branch or release** menu, select the `main` branch.
 7. Click **Finish**.

Rule Designer imports all the items in the rule project into your workspace and creates **Miniloan Service** in Rule Designer based on its content in Decision Center.



Troubleshooting

You can ignore any errors for now. When you get a synchronization error message, click **OK** to close the window. You see the synchronization error message because of errors in the BOM. You learn about BOM errors in the next section of this exercise.

- ___ 8. When prompted to switch to the Team Synchronizing perspective, click **No** to remain in the Rule perspective.
- ___ 9. When the Synchronize Complete window opens to notify you that no changes are found, click **OK** to close this window.
- ___ 10. Look at the created decision service in Rule Designer, and relate its content to the content of the corresponding decision service in Decision Center.
- ___ 11. Disconnect Rule Designer from Decision Center for Miniloan Service.
 - ___ a. Right-click the Miniloan Service and click **Decision Center > Disconnect**.
The “Disconnect from Decision Center” window opens.
 - ___ b. Select the **Keep the Decision Center entries files** option, and click **Yes**.
 - ___ c. Click **OK** to close the Decision Center synchronization error window.



Reminder

You learn about the BOM errors in the next section of this exercise.

- ___ 12. Stop the sample server by double-clicking the **Stop server** desktop shortcut or by clicking **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Stop server**.

8.2. Finalizing the rule project in Rule Designer

After you successfully retrieve a rule project from Rule Designer, your work is not necessarily complete. In many cases, as you learn now with Miniloan Service, you still have a few steps to follow.

- ___ 1. In the Rule perspective, wait for Miniloan Service to finish building.

2. Look at the messages in the Problems view, and notice that two of the issues are prefaced with "B2X".

Description	Resource	Path	Location
Errors (5 items)			
[B2X] Cannot find execution class "miniloan.Borrower" for transl	miniloan.bom	/Miniloan Service...	/Miniloan S...
[B2X] Cannot find execution class "miniloan.Loan" for translating	miniloan.bom	/Miniloan Service...	/Miniloan S...
The deployment configuration 'Miniloan' only references Java SE	Miniloan.dep	/Miniloan Service...	44b9b9ad...
The deployment configuration 'Miniloan' references a target 'Sim	Miniloan.dep	/Miniloan Service...	44b9b9ad...
The project path platform:/miniloan-xom cannot be resolved.	Miniloan Service		Unknown

3. Open the **Miniloan Service** properties.
- a. In the Rule Explorer, right-click **Miniloan Service** and click **Properties** from the menu.
 - b. In the Properties window, select **Business Object Model**.
 - c. In the Business Object Model page, expand the list in the Required BOM entries to see the origin or source for the **Miniloan Service** BOM.



Questions

Why does your new project have errors?

Answer

To build a rule project in Rule Designer that is created from Decision Center, you must also have its referenced projects. These referenced projects also include any executable elements (such as XOMs, .jar files, or libraries) that are not in Decision Center.



Questions

Which executable elements are required?

Answer

The Miniloan Service decision service cannot build because it requires the `miniloan-xom` project, which is not present in your Rule Designer workspace.



Stop

For this exercise, you are not required to retrieve the missing XOM.

- d. Click **Cancel** to close the Properties window.

You successfully created the initial elements that are required to start developing a business rule application, and learned how the Rule Project Map guides you through development. You also learned how to publish and synchronize the rule authoring environment with Decision Center.

End of exercise

Exercise review and wrap-up

This exercise looked at how you start developing a business rule application. You set up a decision service in Rule Designer and finished by publishing the rule environment to Decision Center to make it accessible to business users through the Business console or the Enterprise console.

As an optional step when you are finished with this exercise, you can import the exercise solution file into a new workspace by using the Samples Console and review the solution.

- 1. *(Optional)* To view the exercise solution, switch to a new workspace.
 - a. Go to **File > Switch Workspace > Other**.
 - b. In the **Workspace** field, enter a name for the solution workspace, such as:
`<LabfilesDir>\workspaces\decision_service_answer`
 - c. When the new workspace opens, close the **Welcome** tab.
- 2. Open the Samples Console perspective.
 - a. Click the **Open Perspective** icon.
 - b. Select **Samples Console**, and click **OK**.
- 3. Import the solution project for this exercise.
 - a. Go to **Rule Designer > Training > Ex 02: Setting up decision services > 02-answer**.
 - b. Click **Import projects**.
 - c. When the import is complete, close the Help pane.
- 4. Review the solution for this exercise.



Note

The solution project does not include the imported Miniloan Service decision service.

Exercise 3. Working with the BOM

What this exercise is about

This exercise describes how to create a BOM from a XOM.

What you should be able to do

After completing this exercise, you should be able to:

- Generate a BOM from an existing XOM
- Verbalize the BOM with natural-language vocabulary

Introduction

In this exercise, you work with the Java code to finalize the implementation of the XOM. Next, you create a BOM and vocabulary that is based on the completed XOM.

The exercise includes these tasks:

- Section 1, "Finalizing the XOM"
- Section 2, "Creating a rule project with a BOM"
- Section 3, "Working with the vocabulary that is required to author rules"

Requirements

This exercise uses the following files, which are installed in the `<InstallDir>\studio\training` directory:

- Start project: Dev 03 - BOM\01-start
- Solution project: Dev 03 - BOM\02-answer
 - You use the solution project in "Exercise review and wrap-up" on page 3-21

Section 1. Finalizing the XOM

As part of your role as a developer, you create the implementation code, including the XOM, for the business rule project. The `loan-xom` Java project does not compile, so your task is to complete the XOM implementation.

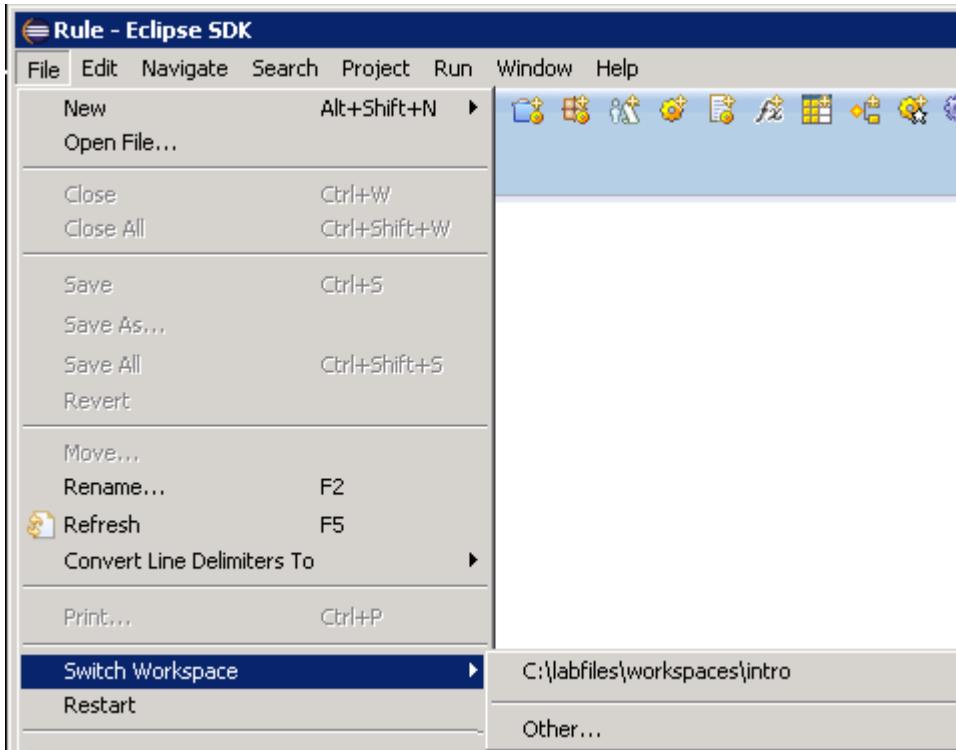


Note

The purpose of this exercise is not to teach you how to create Java classes, but to illustrate the types of tasks that developers are required to do when implementing the XOM.

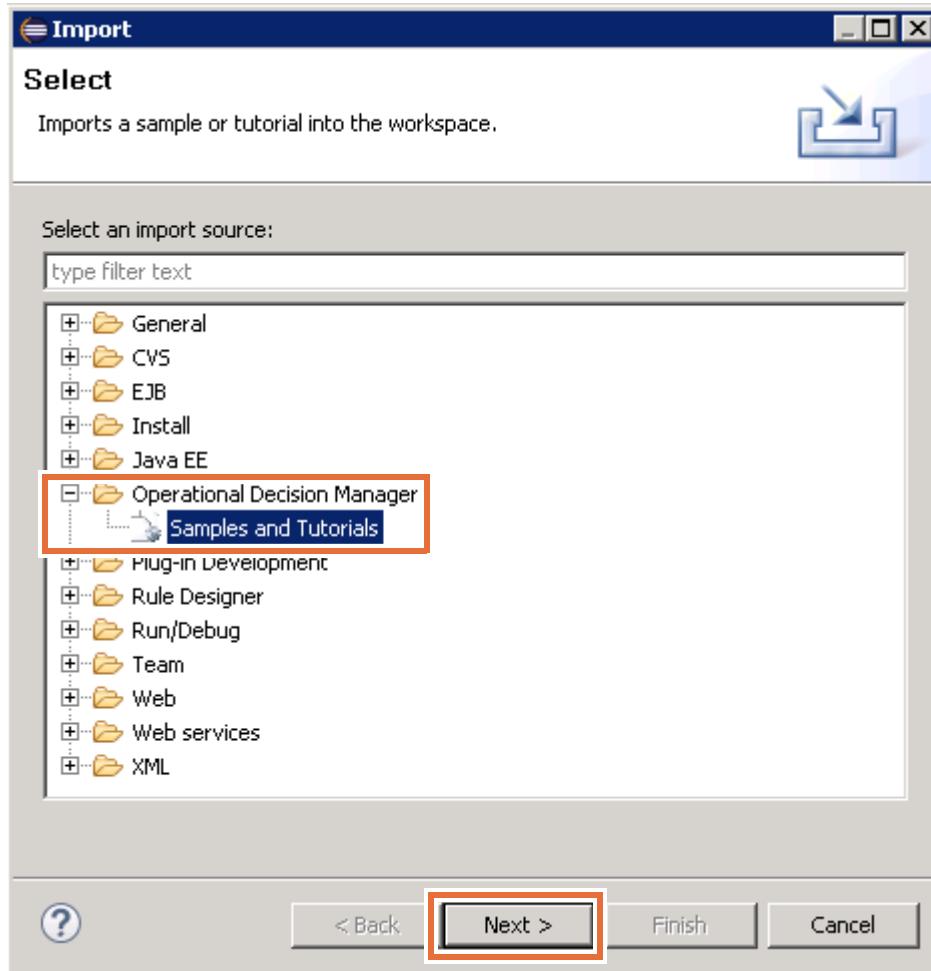
1.1. Setting up your environment for this exercise

- 1. In Rule Designer, switch to a new workspace by following these steps:
 - a. From the **File** menu, click **Switch Workspace > Other**.



- b. In the Workspace Launcher window, enter the path:
`<LabfilesDir>\workspaces\bom`
- c. Ignore **Copy Settings**, and click **OK**.
- 2. Close the Welcome view.
- 3. Use the **File > Import** menu to import the exercise start project.
 - a. In Eclipse, click **File > Import**.

- ___ b. In the Select window, expand **Operational Decision Manager**, select **Samples and Tutorials**, and click **Next**.

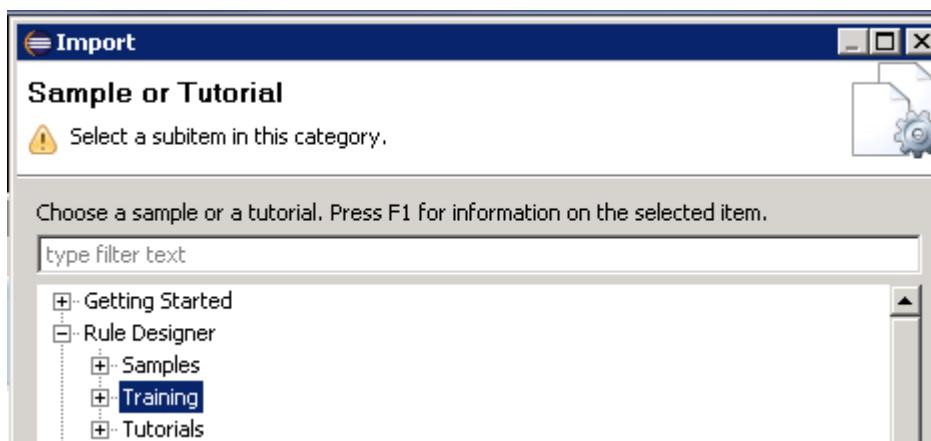


- ___ c. On the “Sample or Tutorial” page, expand the **Rule Designer > Training** node.



Hint

You can collapse the Rule Designer node and then expand it to simplify the list.



- __ d. Expand the **Ex 03: Working with BOMs** node, and select **01-start**.
- __ e. Make sure that the **Import shared projects** check box is selected, and click **Finish**.
- __ 4. When the workspace finishes building, close the Help view.
- __ 5. In the Rule Explorer, notice that you now have the `loan-xom` Java project available in your workspace.

The `loan-xom` project shows the warning icon.



1.2. Understanding the problem

- __ 1. In the Rule perspective, click the **Problems** tab to open the Problems view (at the bottom of the perspective) and expand **Warnings** to review the problem.

Description	Resource	Path	Location	Type
⚠ Warnings (1 item)				
⚠ The value of the field Borrower.spouse is not used	Borrower.j...	/loan-xom/src/tr...	line 27	Java Problem

Notice the description of the problem states that the `spouse` field of the `Borrower` class is not used.

- __ 2. Click the **Tasks** tab to open the Tasks view.

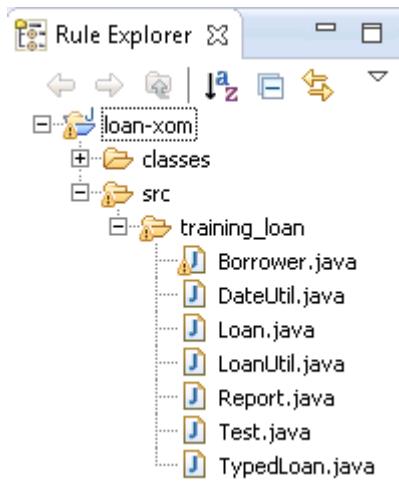
	Description	Resource	Path	Location	Type
1	TODO : Uncomment when the getters and sette...	Test.java	/loan-xom/src/tr...	line 19	Java Task
2	TODO : Uncomment when the getters and sette...	Test.java	/loan-xom/src/tr...	line 35	Java Task
3	TODO: Create a getter called getBirthDate for t...	Borrower.j...	/loan-xom/src/tr...	line 18	Java Task
4	TODO: Create the getters (not the setters) for t...	Borrower.j...	/loan-xom/src/tr...	line 16	Java Task
5	TODO: Create the getters and the setters for t...	Borrower.j...	/loan-xom/src/tr...	line 17	Java Task

The Tasks view indicates a series of tasks that are required to finalize this XOM.

Next, you complete the tasks that are listed in the Tasks view to finish implementing the classes in the `loan-xom` Java project.

1.3. Finish implementing the Borrower class

- 1. In Rule Explorer, expand **loan-xom > src > training_loan** and see the **Borrower.java** class file.



- 2. Double-click **Borrower.java** to open this file in the code editor.

Next, you generate the “getter” code for some of the attributes in this class.



Example

This code shows you an example of what a getter method should look like for the `firstName` attribute:

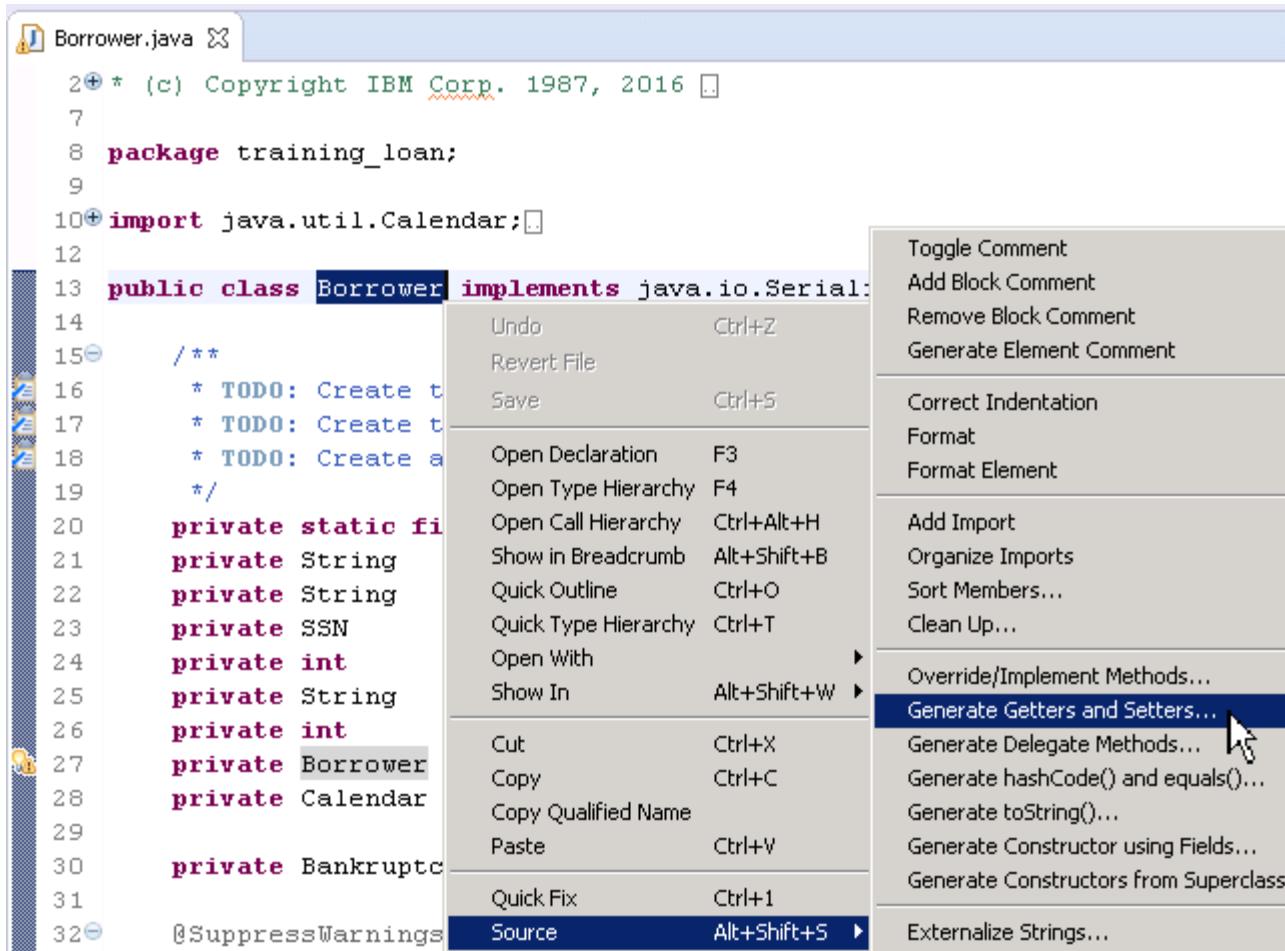
```
/**  
 * @return the firstName  
 */  
public int getFirstName() {  
    return firstName;  
}
```

Rule Designer can automatically generate these types of methods for you, as you see next.

3. In the `Borrower` class, create the getters (not the setters) for these attributes:

- `firstName`
- `lastName`
- `SSN`

a. In the `Borrower` class code, select the `Borrower` class name, and right-click to click **Source > Generate Getters and Setters**.



```

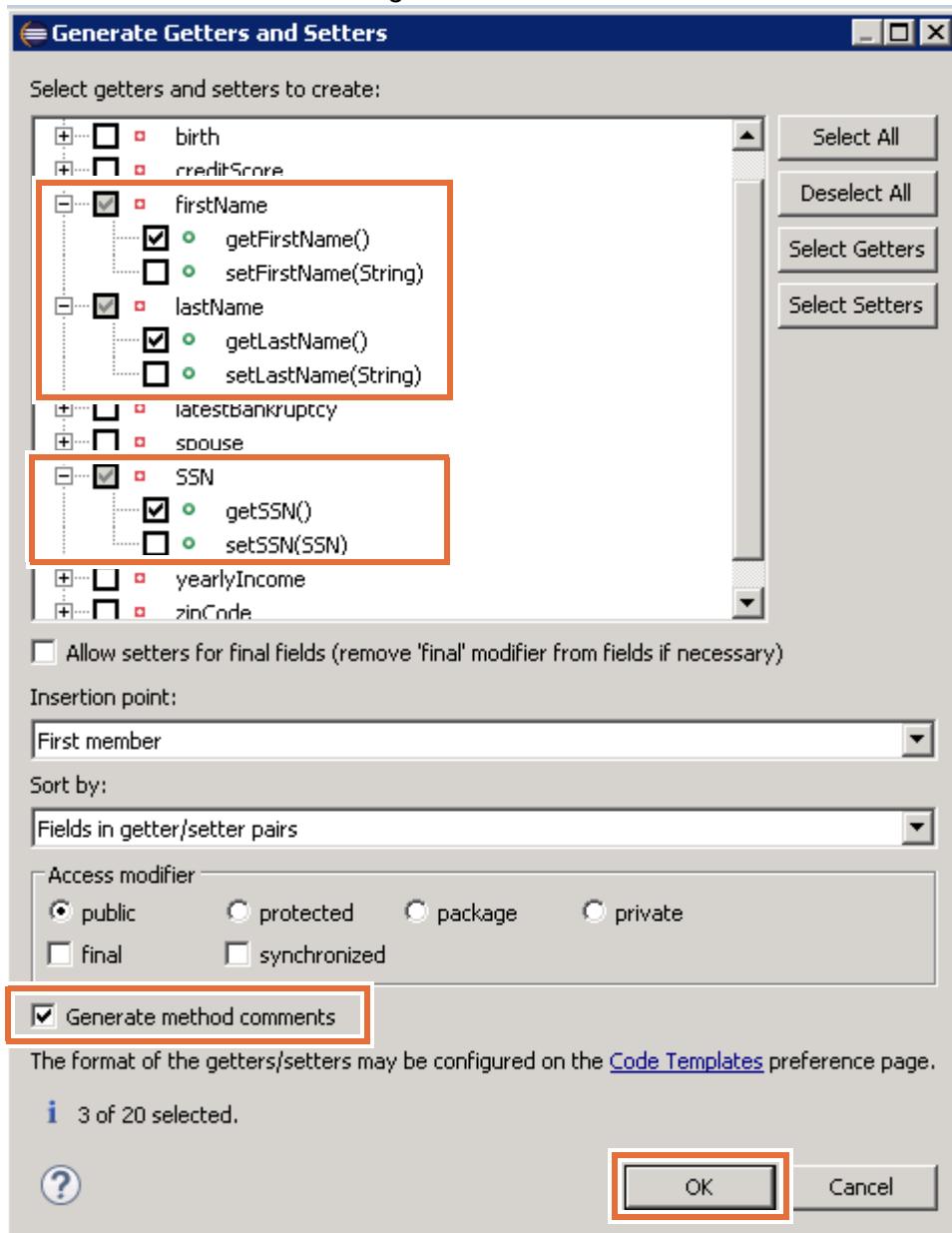
1 Borrower.java X
2 * (c) Copyright IBM Corp. 1987, 2016
3
4 package training_loan;
5
6 import java.util.Calendar;
7
8 public class Borrower implements java.io.Serializable {
9
10    /**
11     * TODO: Create t
12     * TODO: Create t
13     * TODO: Create a
14     */
15    private static final long serialVersionUID = 1L;
16    private String firstName;
17    private String lastName;
18    private String SSN;
19    private int age;
20    private String gender;
21    private int zipCode;
22    private Borrower();
23    private Calendar birthDate;
24    private Bankruptcy bankruptcies;
25
26    @SuppressWarnings("S")
27
28    @Override
29    public String toString() {
30        return "Borrower{" +
31                "firstName=" + firstName +
32                ", lastName=" + lastName +
33                ", SSN=" + SSN +
34                ", age=" + age +
35                ", gender=" + gender +
36                ", zipCode=" + zipCode +
37                ", birthDate=" + birthDate +
38                ", bankruptcies=" + bankruptcies +
39                '}';
40    }
41
42    @Override
43    public int hashCode() {
44        return Objects.hash(firstName, lastName, SSN, age, gender, zipCode, birthDate, bankruptcies);
45    }
46
47    @Override
48    public boolean equals(Object o) {
49        if (this == o) return true;
50        if (o == null || getClass() != o.getClass()) return false;
51        Borrower borrower = (Borrower) o;
52        return Objects.equals(firstName, borrower.firstName) &&
53               Objects.equals(lastName, borrower.lastName) &&
54               Objects.equals(SSN, borrower.SSN) &&
55               age == borrower.age &&
56               gender.equals(borrower.gender) &&
57               zipCode == borrower.zipCode &&
58               Objects.equals(birthDate, borrower.birthDate) &&
59               Objects.equals(bankruptcies, borrower.bankruptcies);
60    }
61
62    @Override
63    protected void finalize() throws Throwable {
64        super.finalize();
65    }
66}

```

The screenshot shows the Eclipse IDE interface with a Java file named `Borrower.java` open. A context menu is displayed over the class name `Borrower`. The menu has several sections:

- File**: Undo (Ctrl+Z), Revert File, Save (Ctrl+S)
- Source**: Open Declaration (F3), Open Type Hierarchy (F4), Open Call Hierarchy (Ctrl+Alt+H), Show in Breadcrumb (Alt+Shift+B), Quick Outline (Ctrl+O), Quick Type Hierarchy (Ctrl+T), Open With, Show In (Alt+Shift+W)
- Refactor**: Add Import, Organize Imports, Sort Members..., Clean Up...
- Generate**: Override/Implement Methods..., **Generate Getters and Setters...** (highlighted with a cursor), Generate Delegate Methods..., Generate hashCode() and equals()..., Generate toString()..., Generate Constructor using Fields..., Generate Constructors from Superclass
- Externalize**: Externalize Strings...

- ___ b. In the “Generate Getters and Setters” window, expand the **firstName**, **lastName**, and **SSN** attributes and select the “get” method for each.



- ___ c. Select **Generate method comments**.
___ d. Click **OK**.

- __ e. In the Borrower.java file, look for the newly added getter methods.

```
    /**
     * @return the firstName
     */
    public String getFirstName() {
        return firstName;
    }

    /**
     * @return the lastName
     */
    public String getLastname() {
        return lastName;
    }

    /**
     * @return the sSN
     */
    public SSN getSSN() {
        return SSN;
    }
```



Note

The new methods were appended where your mouse was positioned in the file. You can select all these methods and move them to the end of the file, just before the final closing brace ()).

- __ 4. In the Borrower class, create both the getters and the setters for the following attributes:

- creditScore
- spouse
- yearlyIncome
- zipCode



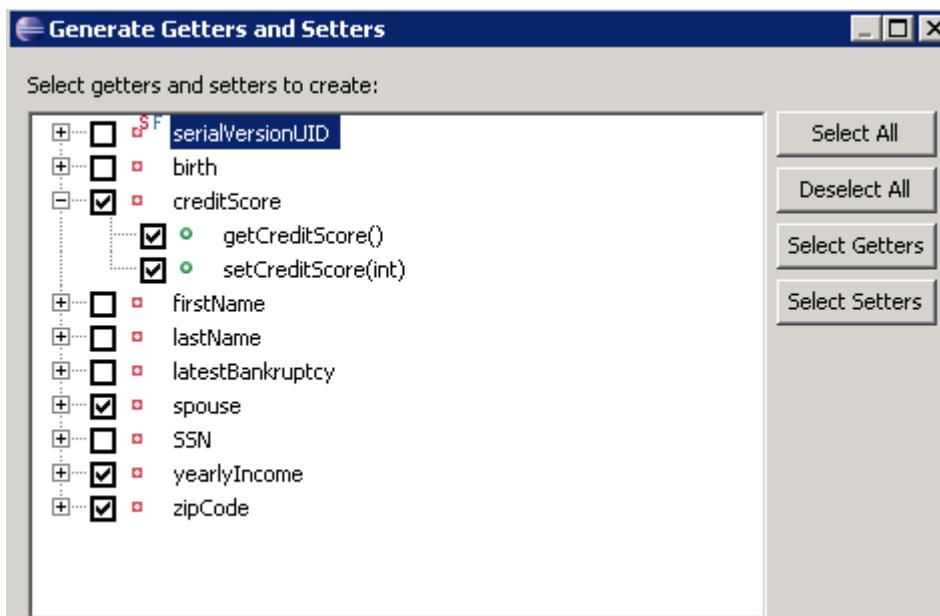
Example

The following examples show what the getter and setter methods should look like for the `yearlyIncome` attribute.

```
/**
 * @return the yearlyIncome
 */
public int getYearlyIncome() {
    return yearlyIncome;
}

/**
 * @param yearlyIncome the yearlyIncome to set
 */
public void setYearlyIncome(int yearlyIncome) {
    this.yearlyIncome = yearlyIncome;
}
```

- __ a. Reopen the “Generate Getters and Setters” window from the `Borrower.java` source code editor by right-clicking the `Borrower` class name, and clicking **Source > Generate Getters and Setters**.
- __ b. Select `creditScore`, `spouse`, `yearlyIncome`, and `zipCode`, which selects both the “get” and “set” methods for these attributes.



- __ c. Click **OK**, and review the source code to see these newly added methods.
- __ d. Save your work by pressing **Ctrl+S**.

**Note**

The new methods for the `spouse` attribute should resolve the warning that you saw earlier.

- ___ 5. In the `Borrower` class, create a getter that is called `getBirthDate` for the `birth` attribute.

This method must return `birth.getTime()`, which is an instance of the `java.util.Date` class.

Rule Designer cannot automatically generate this method, so you must manually edit the `Borrower.java` file.

- ___ a. Scroll to the end of the file to find the final closing brace `}`.
- ___ b. Just before the final brace, add the following lines of code:

```
/**  
 * @return the birth date  
 */  
public Date getBirthDate() {  
    return birth.getTime();  
}
```

- ___ 6. In the `Borrower` class, look for the following comments and delete them:

```
/**  
 * TODO: Create the getters (not the setters) for the following attributes:  
 firstName, lastName, and SSN.  
 * TODO: Create the getters and the setters for the following attributes:  
 zipCode, yearlyIncome, creditScore, and spouse.  
 * TODO: Create a getter called getBirthDate for the birth attribute.  
 */
```

- ___ 7. Save the `Borrower.java` file by pressing **Ctrl+S**.

1.4. Finish implementing the Test class

- ___ 1. In Rule Explorer, double-click **Test.java** to open it in the code editor and follow the instructions of the `TODO` comments.
- ___ a. Look through the `Borrower` class to find the following method calls and delete the forward slashes `(//)` to obtain these lines of code:

```
b1.setCreditScore(600);  
b1.setYearlyIncome(100000);  
r1.setCorporateScore(b1.getCreditScore());
```

These lines are in various places in the file.

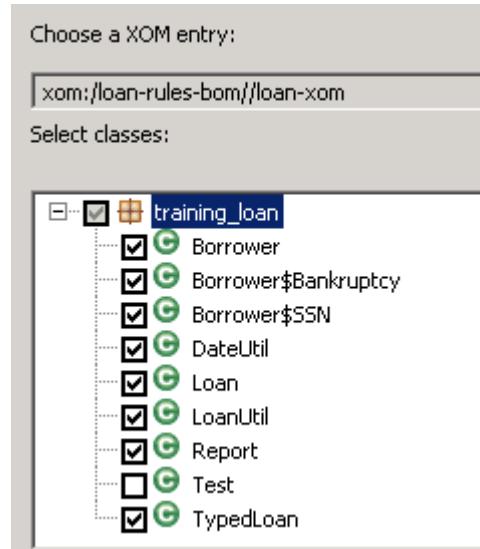
```
Borrower b1 = new Borrower("John", "Doe", DateUtil.makeDate(1968, Calendar.MAY,  
// TODO : Uncomment when the getters and setters of the Borrower's attributes are  
b1.setCreditScore(600);  
b1.setYearlyIncome(100000);  
b1.setLatestBankruptcy(DateUtil.makeDate(1990, Calendar.JANUARY, 01), 7, "Unemp.");  
System.out.println(b1);  
  
Borrower b2 = new Borrower("John", "Doe", DateUtil.makeDate(1970, Calendar.MAY,  
System.out.println(b2);  
  
Borrower b3 = new Borrower("John", "Doe", DateUtil.makeDate(1970, Calendar.MAY,  
System.out.println(b3);  
  
Loan l1 = new Loan(DateUtil.makeDate(2005, Calendar.JUNE, 1), 60, 100000, 0.70);  
System.out.println(l1);  
  
Report r1 = new Report(b1, l1);  
// TODO : Uncomment when the getters and setters of the Borrower's attributes are  
r1.setCorporateScore(b1.getCreditScore());  
r1.addCorporateScore(l2);
```

- b. Delete the TODO comments in the Test class.
- 2. Press Ctrl+Shift+S ("Save All") to save the Borrower class and the Test class.
- 3. Close the editor windows for both the Borrower and the Test classes.
- 4. Verify that the Problems view and the Tasks view are empty, which indicates that the loan-xom Java project compiled successfully.

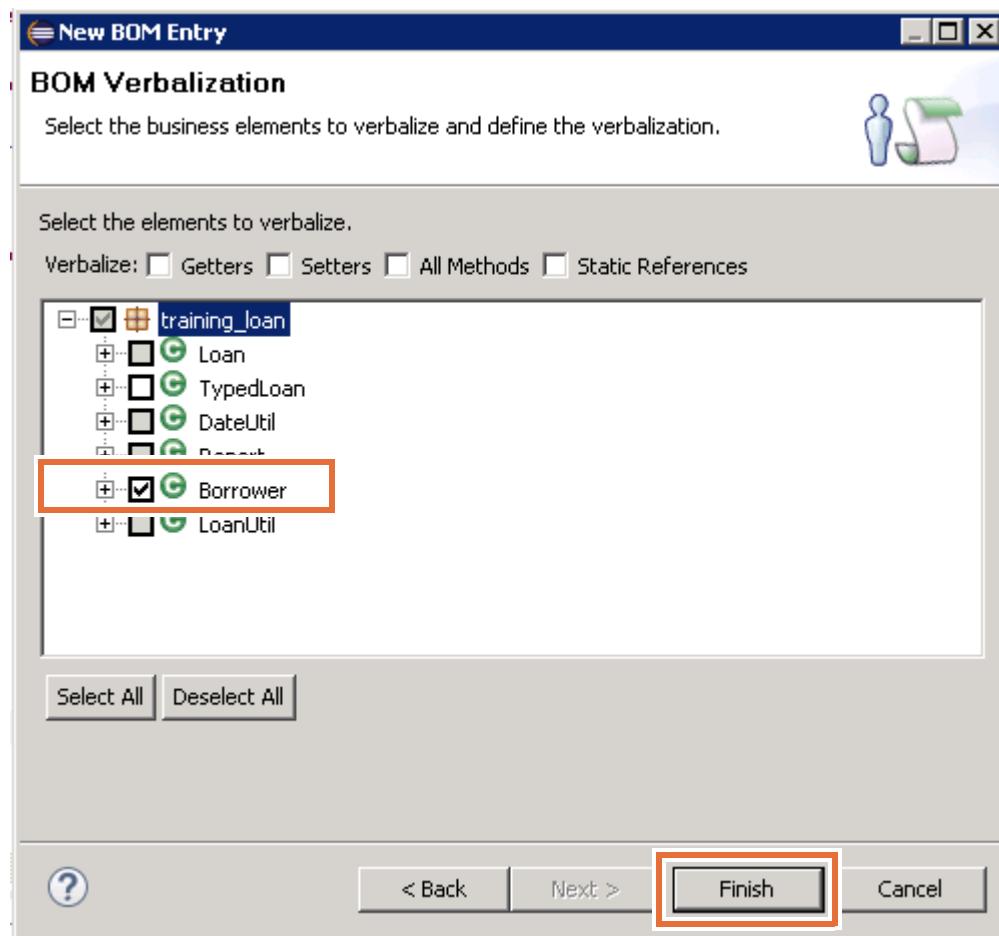
Section 2. Creating a rule project with a BOM

Now that you have a complete XOM, you create a standard rule project, and then you create a BOM from the XOM.

- 1. Create a standard rule project with a BOM.
 - a. Right-click anywhere in the Rule Explorer and click **New > Rule Project**.
 - b. On the “Select a template” page, in the **Decision Service Rule Projects** section, select **Standard Rule Project** and click **Next**.
 - c. In the **Project name** field, type `loan-rules-bom` and click **Finish**.
 - d. Right-click **loan-rules-bom** and click **Properties**.
- The new rule project opens in the rule editor.
- 2. In the Properties wizard, click **Java Execution Object Model**, select **loan-xom**, and click **OK**.
- 3. Create a BOM entry in the `loan-rules-bom` project.
 - a. In Rule Explorer, right-click **loan-rules-bom > bom** and click **New > BOM Entry**.
 - b. In the New BOM Entry window, keep `model` in the **Name** field, select the **Create a BOM entry from a XOM** option, and click **Next**.
 - c. For the **Choose a XOM entry** field, click **Browse XOM**.
 - d. In the Browse XOM window, select **platform:/loan-xom**, and click **OK** to close the window.
 - e. In the **Select classes** section, expand **training_loan**, select the check boxes for all classes except `Test`, and click **Next**.



- ___ f. On the BOM Verbalization page, click **Deselect All** to clear all the check boxes, select **Borrower**, and click **Finish**.



The `bom` folder of the `loan-rules` project now contains a BOM entry called `model`. This model is the BOM that you generated from the XOM.

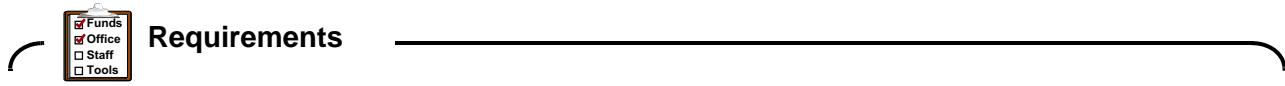
Only the `Borrower` class is verbalized.

- ___ 4. Make sure that everything is saved by pressing **Ctrl+Shift+S**.

Section 3. Working with the vocabulary that is required to author rules

In the previous part of the exercise, you learned how to create the default vocabulary of your BOM at the time you created the BOM itself. In this part of the exercise, you learn more about the BOM editor and create the vocabulary that is required to author rules by using the Verbalization wizard in Rule Designer. You also review the various concepts that are involved, including business terms, phrases, and placeholders.

3.1. Creating the default vocabulary

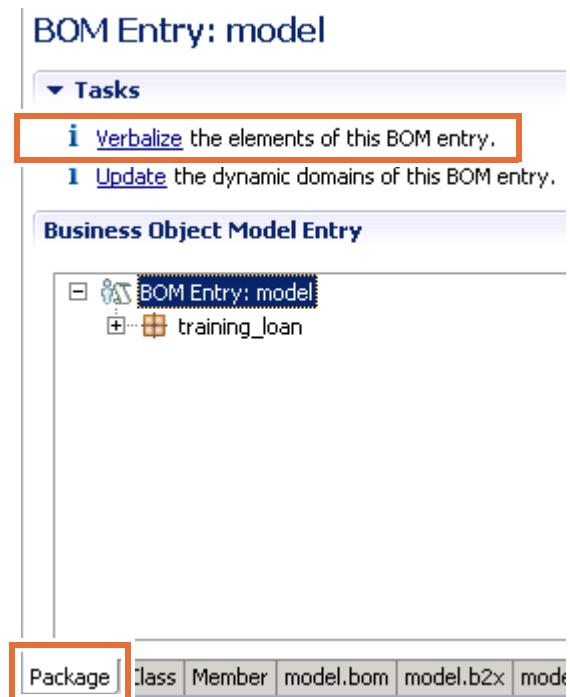


Business analysts examined the BOM that you created for the project and found that some elements must be reworked. They now ask for a complete verbalization of the `Report` class and of the `Loan` class. They also ask you to revise the vocabulary that is associated with the `Borrower` class.

In this section, you learn now how to create or modify the vocabulary in the BOM editor.

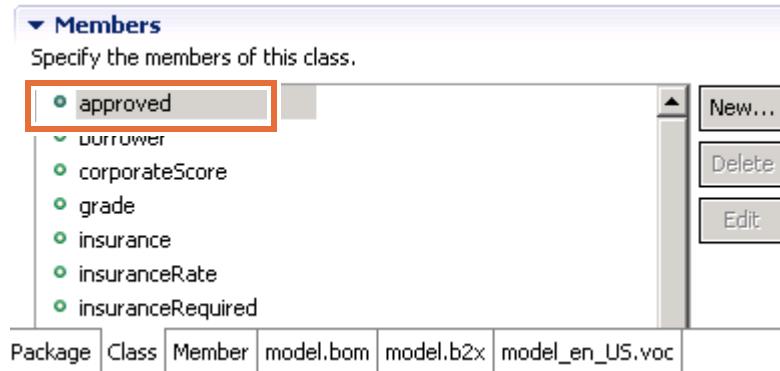
- ___ 1. In Rule Explorer, expand `loan-rules-bom > bom > model > training_loan`.
- ___ 2. Double-click **Report** to see this class in the BOM editor, and note that this member does not yet have a verbalization.

- ___ 3. In the BOM editor, verbalize the `Report` class and all its members with the default verbalization.
- ___ a. Click the **Package** tab to open the Package page of the BOM editor, and click the **Verbalize** task.



The Verbalize BOM wizard opens.

- ___ b. On the Configure Verbalization page of the Verbalize BOM wizard, click **Deselect All**, and then select **Report**.
You want to verbalize the `Report` BOM class and all its members.
- ___ c. Click **Finish**.
- ___ d. Save your work (Ctrl+S).
- ___ e. In the **Business Object Model Entry** section of the **Package** tab of the BOM editor, expand **training_loan** and double-click **Report** to open it in the Class page of the BOM editor.
- ___ f. In the **Members** section of the Class page, double-click **approved** to open it in the Member page.



The default verbalization for the setter of the Boolean `approved` member is:

`make it {approved} that {this} is approved`

- ___ 4. Return to the **Package** tab and double-click **Loan** to open it in the **Class** tab.
- ___ 5. In the **Class Verbalization** section, click **Create a default verbalization**.

This screenshot shows the 'Class Verbalization' section of a software interface. It includes a note: 'This class is not verbalized' with a link to 'Create a default verbalization'. There is also an unchecked checkbox for 'Generate automatic variable'.



The `Loan` class itself is now verbalized, but not its members.

In the next steps, you verbalize some members individually in the corresponding **Member Verbalization** section of the BOM editor.

- ___ 6. Create the default verbalization for the `duration` member of the `Loan` class.
 - ___ a. On the Class page for the `Loan` class, double-click the `duration` member to edit it in the Member page of the BOM editor.
- The Member page opens. In the **Member Verbalization** section of the Member page, you can see that the `duration` member is not verbalized.
- ___ b. In the **Member Verbalization** section of the Member page, click **Create a default verbalization**.

This screenshot shows the 'Member Verbalization' section of a software interface. It includes a note: 'This member is not verbalized' with a link to 'Create a default verbalization'. There is also an unchecked checkbox for 'Generate automatic variable'.

- ___ c. Notice the default verbalization:
 - Navigation phrase: "the duration of a loan"
 - Template field: `{duration}` of `{this}`
- ___ 7. Save your work (Ctrl+S).

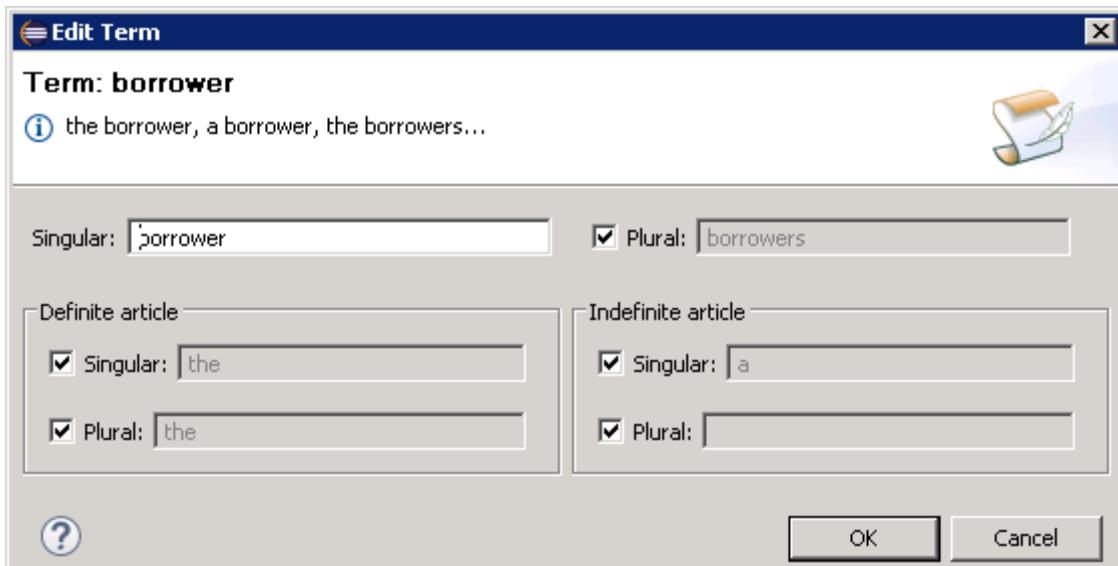
3.2. Examining the verbalization for members of the Borrower class

- ___ 1. On the **Package** tab of the BOM editor, double-click the `Borrower` class to open it in the **Class** tab.

2. Click **Edit term** in the **Class Verbalization** section.

The screenshot shows the 'Class Verbalization' section of a software interface. It includes options to 'Remove' or 'Edit' the verbalization, a checkbox for 'Generate automatic variable', and a text input field for 'Term' containing 'borrower'. A tooltip for the 'Term' field says 'the borrower, a borrower, the borrowers...'. A hand cursor is hovering over the 'Edit term.' button, which is highlighted with a red border.

The Edit Term window opens.



- a. Notice that you can edit the `borrower` term to set the vocabulary for singular and plural forms, and for the definite and indefinite articles.
 - b. Click **Cancel** to close the Edit Term window.
3. From the **Class** tab, open the `hasLatestBankruptcy` method of the `Borrower` class and examine its default verbalization:

`{this} is has latest bankruptcy`



Questions

Do you think that the default verbalization of the `hasLatestBankruptcy` method is suitable vocabulary for business rules?

Answer

No, the verbalization ("`{this} is has latest bankruptcy`") would not make sense in a sentence, so a rule that uses this phrase would be confusing. The sentence would not be grammatically correct.

Also, the word “bankruptcy” is incorrectly spelled as: *bankrupcy*



Important

When you create a BOM from a XOM, or when you define the default verbalizations, you retain the way that the method is written in the XOM.

In the XOM for this exercise, “bankruptcy” is incorrectly written as “*bankrupcy*” in the `hasLatestBankrupcy` method name. As a result, the name and the default verbalization of the `hasLatestBankrupcy` method in the BOM also have the same incorrect spelling.

- 4. From the **Class** tab, open the `setLatestBankruptcy` method of the `Borrower` class and examine its default verbalization in the **Template** field.

```
{this}.setLatestBankruptcy({0}, {1}, {2})
```

▼ Member Verbalization

✖ Remove the verbalization.

✚ Create an action phrase.

▼ Action : "a borrower.setLatestBankruptcy(a date, a number, a string)" ✖

Template:



Questions

Do you think that the default verbalization of the `setLatestBankruptcy` method is suitable for business rules?

Answer

No, the verbalization of the `setLatestBankruptcy` method is not suitable for business rules because it does not look like natural language. In a business rule, this type of programming-language vocabulary makes the rule awkward and unreadable.

**Questions**

Can you identify what the placeholders in the default verbalization of the `setLatestBankruptcy` method stand for?

Answer

The placeholders correspond to the three arguments of this method, as shown in the **Arguments** section on the **Member** tab.

▼ Arguments
Edit the arguments of this member.

Name	Type	Domain
arg1	java.util.Date	
arg2	int	
arg3	java.lang.String	

Add...
Remove...
Up
Down
Edit...

3.3. Editing the default verbalization

The default verbalization of the `hasLatestBankruptcy` method is incorrect. The default verbalization of the `setLatestBankruptcy` method also does not look like natural language.

You must correct this situation by changing the vocabulary that is associated with these methods.

- 1. Return to the Member page for the `hasLatestBankruptcy` method, and change the verbalization in the **Template** field of the **Member Verbalization** section to the following text:

{this} has latest bankruptcy

**Note**

The new verbalization is more natural and also corrects the way “bankruptcy” is written so that you and business users can easily author rule artifacts later.

You can leave the name of the BOM method unchanged and consistent with the name of the corresponding XOM method, without any effect on authored rules.

- ___ 2. Return to the Member page for the `setLatestBankruptcy` method and change the verbalization in the **Template** field to match the following text:
set the latest bankruptcy of {this} to date {0}, chapter {1} and reason {2}
Notice that the BOM editor notifies you of missing placeholders as you edit the **Template** field.
- ___ 3. Save your work.
- ___ 4. Close the BOM editor.

End of exercise

Exercise review and wrap-up

To see a solution to this exercise, switch to a new workspace and import the project **Ex 03: Working with BOMs > 02-answer**.

The first part of this exercise looked at how you can create a BOM from an existing XOM, and create the default vocabulary at the same time.

The second part of this exercise looked at how you can use the BOM editor to create or modify the vocabulary that is associated with a specific BOM element.

Exercise 4. Refactoring

What this exercise is about

This exercise describes how to manage inconsistencies within the project as the XOM, BOM, and vocabulary evolve.

What you should be able to do

After completing this exercise, you should be able to:

- Refactor vocabulary changes
- Manage inconsistency issues after updating the XOM and BOM

Introduction

In this exercise, you modify the XOM, BOM, and vocabulary to support rule authoring requirements from the business users. You also learn how these changes can affect the project in terms of consistency, and you resolve the consistency issues.

The exercise includes these tasks:

- Section 1, "Refactoring rule artifacts after vocabulary changes"
- Section 2, "Maintaining consistency between the BOM and the XOM"

Requirements

This exercise uses the following files, which are installed in the `<InstallDir>\studio\training` directory:

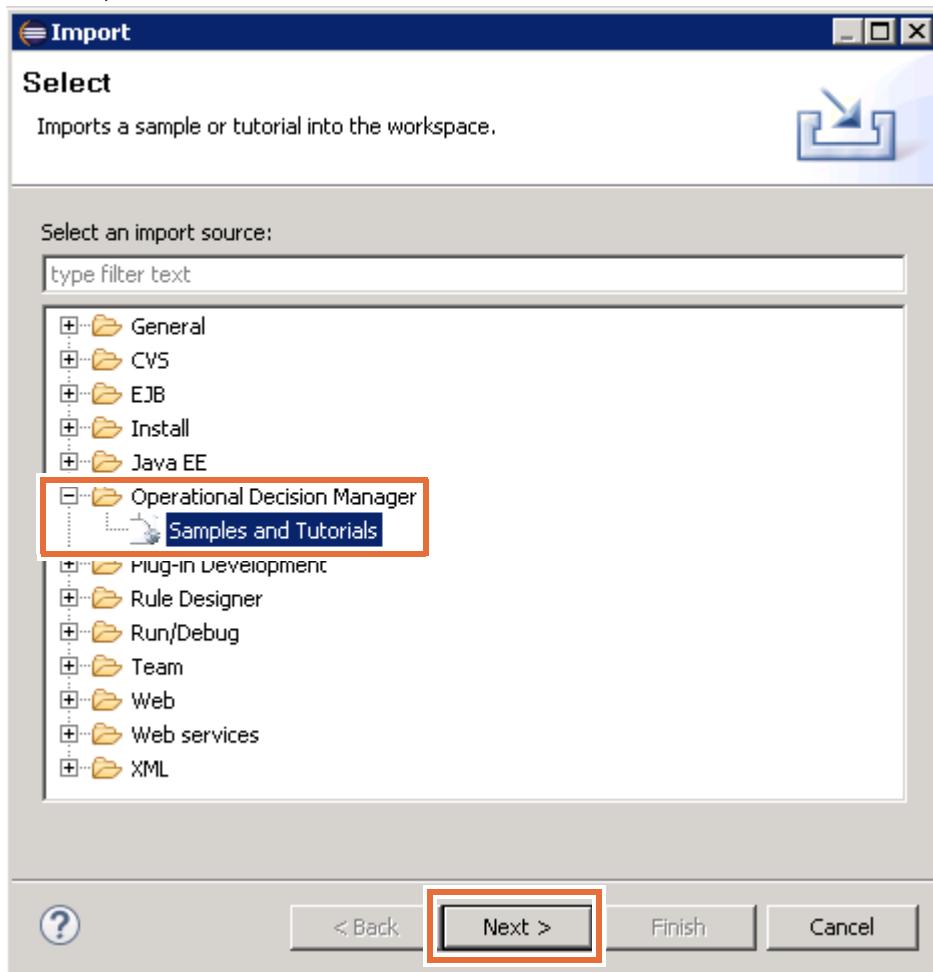
- Start project: Dev 04 - Refactoring\01-start
- Solution project: Dev 04 - Refactoring\02-answer
 - You can use the solution project in "Exercise review and wrap-up" on page 4-16.

Section 1. Refactoring rule artifacts after vocabulary changes

In this part of the exercise, you see how a change in the vocabulary affects rule artifacts, and learn more about the relationship between the rule artifacts and the vocabulary in the BOM.

1.1. Setting up your environment for this exercise

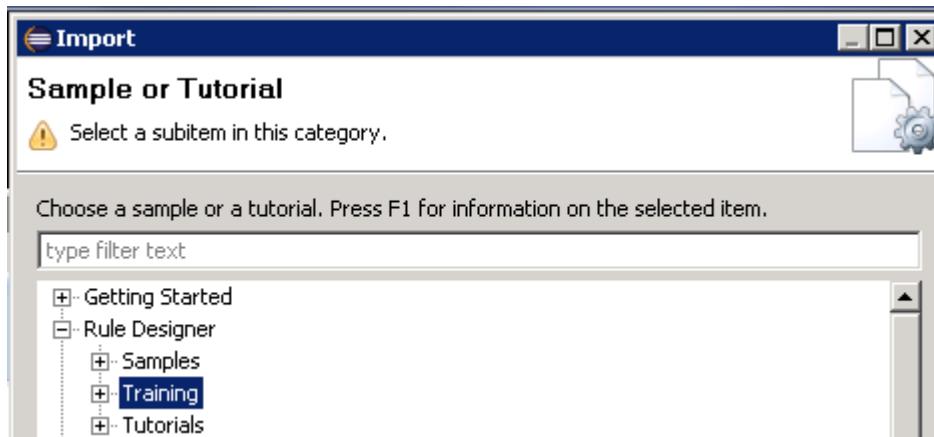
- ___ 1. In Rule Designer, switch to a new workspace:
 - ___ a. From the **File** menu, click **Switch Workspace > Other**.
 - ___ b. In the Workspace Launcher window, enter the path:
<LabfilesDir>\workspaces\refactoring
- ___ 2. Close the Welcome view.
- ___ 3. Use the **Import** menu to import the exercise start project.
 - ___ a. In Eclipse, click **File > Import** or right-click the Rule Explorer and click **Import**.
 - ___ b. In the Select window, expand **Operational Decision Manager**, select **Samples and Tutorials**, and click **Next**.



- ___ c. On the "Sample or Tutorial" page, expand the **Rule Designer > Training** node.

**Hint**

You can collapse the Rule Designer node and then expand it to simplify the list.



- ___ d. Expand the **Ex 04: Refactoring** node, and select **01-start**.
- ___ e. Make sure that the **Import shared projects** check box is selected, and click **Finish**.
- ___ 4. When the workspace finishes building, close the Help view.

1.2. Exploring the default verbalization

- ___ 1. In Rule Explorer, expand **loan-rules > bom > model > training_loan > Borrower**.
- ___ 2. Double-click the **Borrower.getBankruptcyAge** method to see its default verbalization:
`{bankruptcy age} of {this}`
- ___ 3. To see how this verbalization is displayed in a rule, open the **checkLatestBankruptcy** rule in the **rules** folder of the **loan-rules** project.
 - ___ a. Expand **loan-rules > rules**.
 - ___ b. Double-click **checkLatestBankruptcy**.

The condition statement uses this vocabulary:

```
if the bankruptcy age of 'the borrower' is less than 365
```

**Questions**

Is this rule condition easy to understand and unambiguous?

Answer

The `checkLatestBankruptcy` action is ambiguous because a borrower might have more than one bankruptcy. Rule authors might be unsure about how to maintain this rule.



Questions

How can you solve this ambiguity issue?

Answer

Change the vocabulary to use clear language.



Important

You must discuss usability requirements with the business analysts and rule authors before changing vocabulary, since they are best placed to explain vocabulary requirements.



Requirements

The business analysts request that you change the verbalization for these terms:

- The current “bankruptcy age” term should indicate the *latest* bankruptcy.
- The navigation template for the `duration` member of the `Loan` class should indicate that the duration is in years.

1.3. Changing the default verbalization of the getBankruptcyAge method

After you change the default verbalization in the next steps, you also learn the effects of such a change and how to resolve them.

- ___ 1. Edit the `getBankruptcyAge` BOM member verbalization to become: age of the latest bankruptcy
- ___ a. Switch back to the **model** tab.

Member getBankruptcyAge (class: training_loan.Borrower)

General Information

Name: `getBankruptcyAge`
Type: `int`
Class: `training_loan.Borrower`

Static Final
 Deprecated Update object state

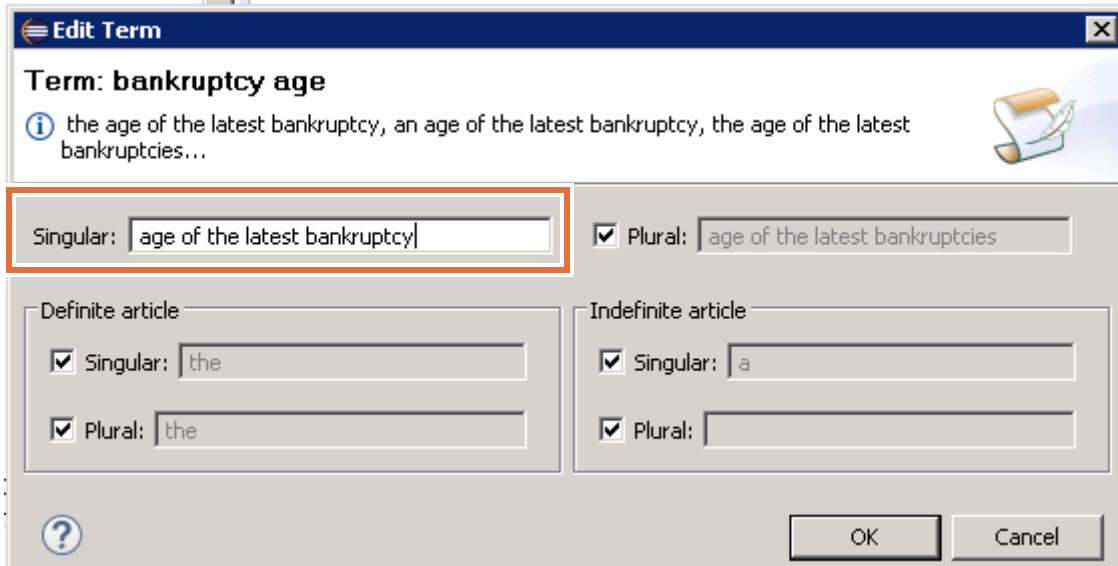
Member Verbalization

Remove the verbalization.
 Create a navigation phrase.
Edit the subject used in phrases.

Navigation : "the bankruptcy age of a borrower"

Template: `{bankruptcy age} of {this}`

- ___ b. Click the **Edit the subject used in phrases** link in the **Member Verbalization** section of the `getBankruptcyAge` method.
- ___ c. Look at the Edit Term window, and try to figure out the purpose of its fields.
- ___ d. In the **Singular** field of the Edit Term window, enter:
age of the latest bankruptcy



- ___ e. Click **OK**.

Note the changes to the navigation phrase and the template of `Borrower.getBankruptcyAge`.

▼ Navigation : "the age of the latest bankruptcy of a borrower" 

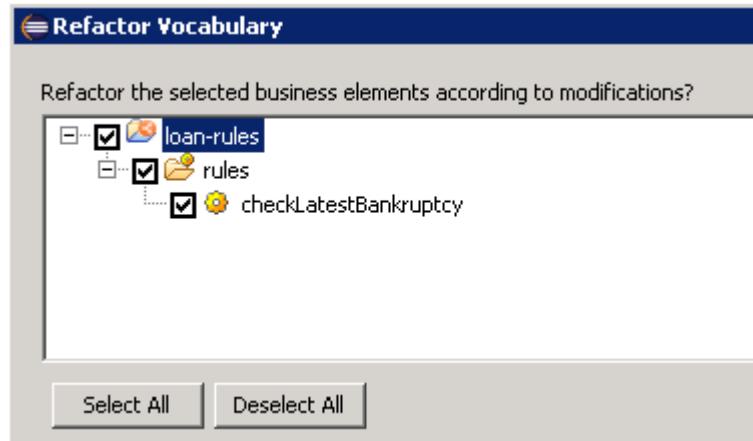
Template: `{age of the latest bankruptcy} of {this}` 

 Changes of this verbalization may impact business rules.

A warning is displayed that indicates which rules the new vocabulary affects.

- ___ 2. Save your work (Ctrl+S).

The Refactor Vocabulary window opens.



Because the previous vocabulary term is no longer valid, you must update any rule artifact that uses that BOM member.

The Refactor Vocabulary window automatically lists the rules that are affected so you can update them all at the same time.

- ___ 3. In the Refactor Vocabulary window, make sure **Select All** is selected and click **Yes**.
- ___ 4. Wait for the workspace to finish building, then switch back to the `checkLatestBankruptcy` rule to see how it was modified.

1.4. Changing the default verbalization of the duration member

- ___ 1. In Rule Explorer, expand **bom > model > training_Loan**, double-click the `loan.duration` BOM member, and look at the current verbalization.



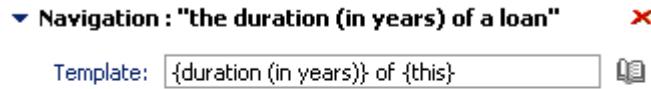
Before you change anything, which rule artifacts would be affected if you change the verbalization of the `duration` member to "duration in years"?

Answer

If you change the verbalization of the `duration` member of the `Loan` class, only the `checkDuration` action rule is no longer valid.

- 2. Repeat the steps from Section 1.3, "Changing the default verbalization of the `getBankruptcyAge` method" to edit the term that is used in the navigation template for the `Loan.duration` member to:

{duration (in years)} of {this}



Changes of this verbalization may impact business rules.

- 3. Save your BOM.

The Refactor Vocabulary window opens again for you to select the business rule elements that you want to update.

- 4. In the Refactor Vocabulary window, click **Select All** and click **Yes** to automatically refactor the selected rule artifacts.
— 5. Open the `checkDuration` rule to see how it was modified.
— 6. Close the editors for the `checkLatestBankruptcy` and `checkDuration` rules.

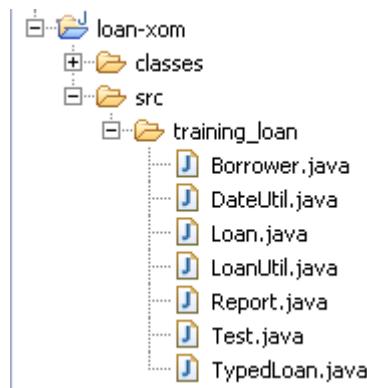
Section 2. Maintaining consistency between the BOM and the XOM

In this part of the exercise, you make sure that the XOM and the BOM of the rule project are consistent.

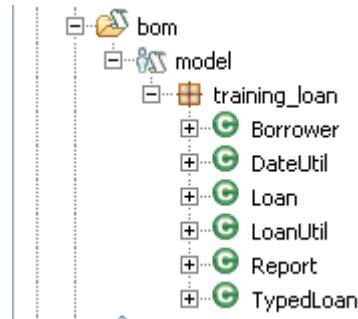
2.1. Adding a XOM class that is missing in the BOM

In this section, you use the BOM Update view to determine which classes in the XOM are not present in the BOM, or vice versa, and to make sure that the two object models are consistent.

- ___ 1. In Rule Explorer, expand **loan-xom > src > training_loan** and notice the list of classes.



- ___ 2. Expand the BOM in the **loan-rules** project (**loan-rules > bom > model > training_loan**).



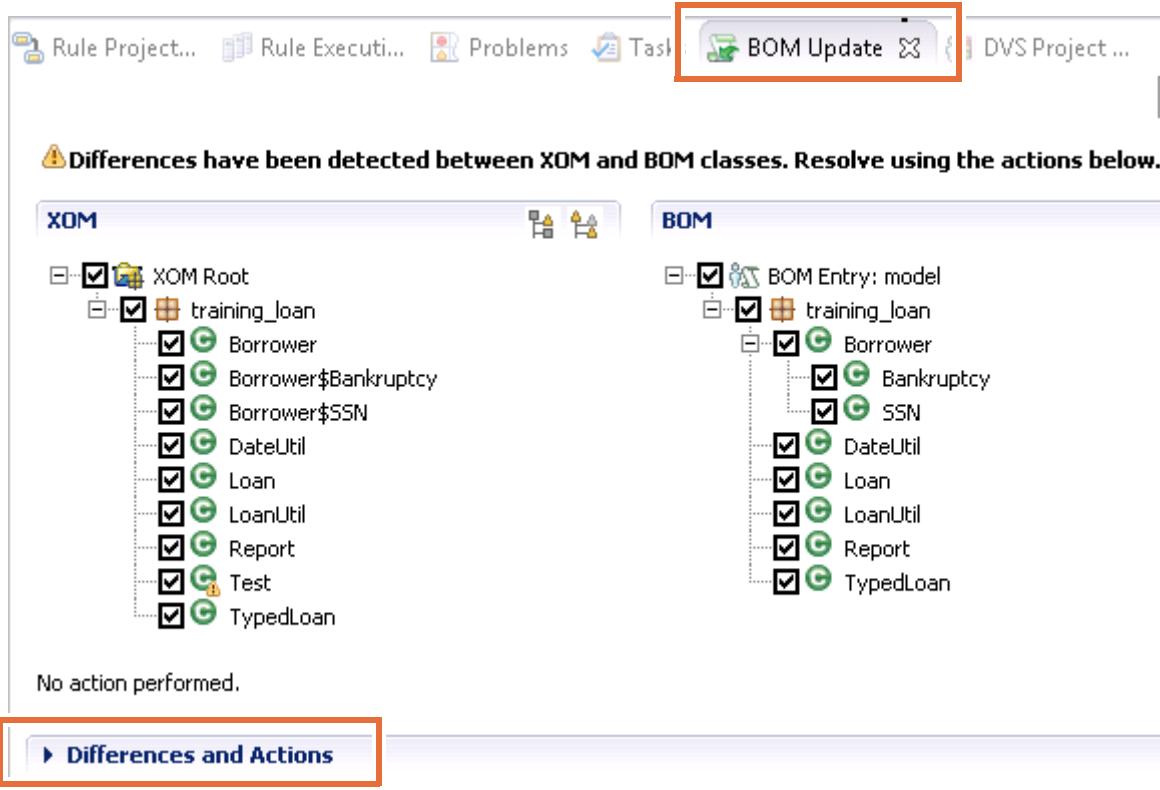
Questions

Do the lists match?

Notice that the `training_loan.Test` class exists in the `loan-xom` Java project, but is not present in the BOM of the `loan-rules` project.

- ___ 3. Right-click `model` in the `bom` folder of the `loan-rules` project, and click **BOM Update**.

The BOM Update view opens in the lower part of the perspective.

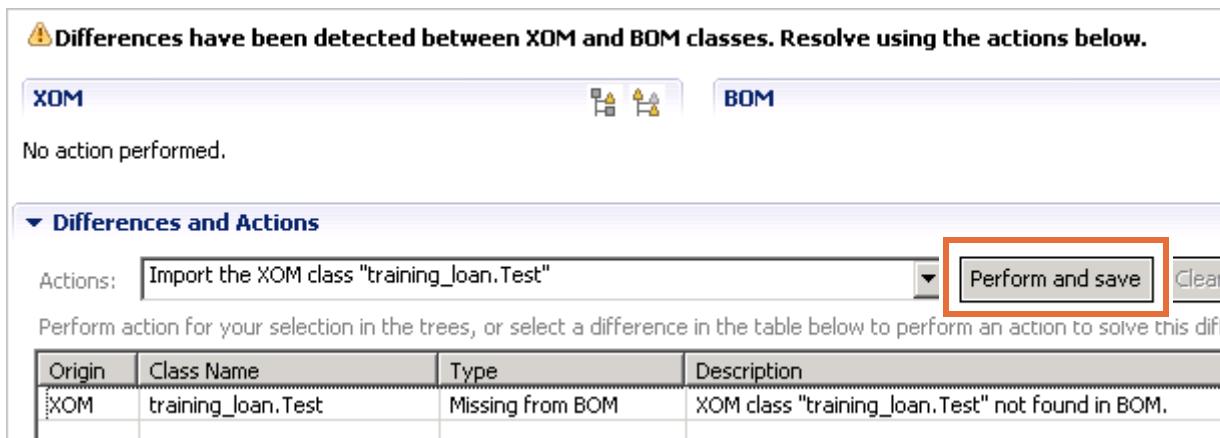


- 4. In the BOM Update view, expand **Differences and Actions**.

Notice the lines that state that the `training_loan.Test` XOM class is not found in the BOM.

Only one suggested action is listed: Import the `training_loan.Test` XOM class.

- 5. Select the suggested action, and click **Perform and save** to apply it.

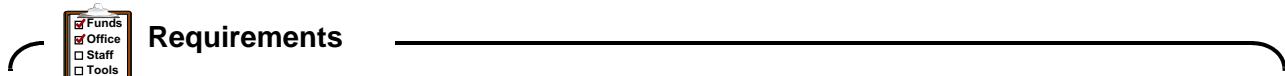


The BOM is automatically updated, and the `training_loan.Test` BOM class is created.

Rule Designer also prompts you to create a default verbalization for this new class in the Verbalize BOM window.

- ___ 6. In the Verbalize BOM window, click **Select All** and click **Finish** to create a default verbalization for the new class.
- ___ 7. Notice the presence of the new `training_loan.Test` class in the BOM of the `loan-rules` decision service.

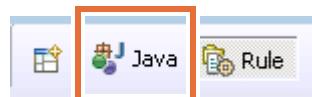
2.2. Adding a XOM method that is missing in the BOM



Following a meeting with business analysts, you must now create a method that is called `getFullName` in the `Borrower` XOM class of the `loan-xom` project, and author an action rule that lists the full names of all borrowers.

Applying this requirement affects consistency between the XOM and the BOM. In this section, you learn how to resolve this type of inconsistency.

- ___ 1. In Rule Designer, switch to the Java perspective.



- ___ a. If the Java perspective is not available in the toolbar, click the **Open Perspective** icon.



- ___ b. Select **Java (default)** from the perspectives list, and click **OK**.
- ___ 2. In the Package Explorer of the Java perspective, expand `loan-xom > src > training_loan` and double-click `Borrower.java` to edit it.
- ___ 3. Add a method called `getFullName` to the `Borrower` class of the `loan-xom` project.

The method should concatenate the first name and the last name. You use this code to implement the method:

```
public String getFullName() {
    return getFirstName() + " " + getLastName();
}
```

You can append this method at the end of the `Borrower` class before the final closing brace `}`.

- ___ 4. Save your work.
- ___ 5. Switch back to the Rule perspective.

- ___ 6. Go to the BOM Update view, and manage the new difference between the BOM and the XOM.

- ___ a. Refresh the BOM Update view by clicking Refresh () (upper-right corner).



- ___ b. Expand **Differences and Actions** and notice the new line that states that the XOM attribute `training_loan.Borrower.fullName` was not found in the `training_loan.Borrower` BOM class.

- ___ c. In the **Actions** field, select the following line and click **Perform and save**.

`Update the BOM class "training_loan.Borrower"`

- ___ d. When the Verbalize BOM window opens, click **Select All** and click **Finish** to create a default verbalization for the new member.

The BOM is automatically updated, and the `Borrower` BOM class contains a new member called `fullName`.

Members
Specify the members of this class.

- age
- birthDate
- creditScore
- firstName
- **fullName**
- lastName
- latestBankruptcyChapter
- latestBankruptcyDate
- latestBankruptcyReason
- spouse
- SSN

New...
Delete
Edit

- ___ e. Refresh the BOM Update view, and notice that no differences between the XOM and the BOM are listed.

2.3. Creating a rule to use this new attribute

- ___ 1. Add an action rule to the `rules` folder of the `loan-rules` project:

- ___ a. Right-click the `rules` folder, and click **New > Action Rule**.

- ___ b. In the **Name** field, type: `displayFullName`

- ___ c. Click **Finish**.

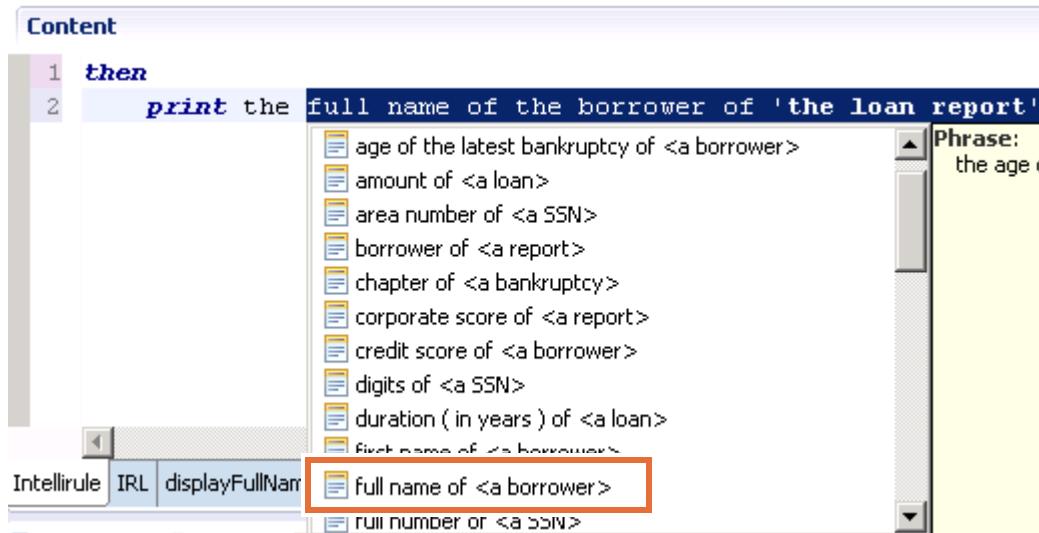
The Intellirule editor opens.

- ___ 2. In the **Content** section of the rule editor, type the following rule:

then

```
print the full name of the borrower of 'the loan report';
```

- ___ 3. Notice that the new “full name” attribute is immediately available as vocabulary for rule authoring.



Information

For this exercise, you want to test only your new BOM member, so no ***if*** statement is required in your rule. As you learn later, condition statements are optional in rules.

- ___ 4. Save your work.

2.4. Deprecating BOM members

The BOM Update view also shows the differences when the BOM has a member that is not in the XOM, and also suggests different corrective actions.



Requirements

Following a subsequent meeting, business analysts now indicate that the `getFullName` method in the `Borrower` class of the `loan-xom` project is superfluous and must be removed.

Applying this requirement can affect consistency between the XOM and the BOM, and affect the rule project. In this section, you learn how to resolve these consistency issues.

- ___ 1. Delete the `getFullName` method that you created in "Adding a XOM method that is missing in the BOM" on page 4-10.
 - ___ a. Stay in the Rule Perspective and expand `loan-xom > src > training_loan`.
 - ___ b. Open the `Borrower.java` file, and delete the code that you added for the `getFullName` method.
 - ___ c. Save your work.

2. Open the Problems view, and notice that the `Borrower` class in the BOM of the `loan-rules` project now has an error:

[B2X] Cannot find attribute "fullName" in execution class
`"training_loan.Borrower"`

Description		Resource
<input type="checkbox"/> Errors (1 item)	<input checked="" type="checkbox"/> [B2X] Cannot find attribute "fullName" in execution class "training_loan.Borrower"	model.bom

3. Refresh the BOM Update view and look at the description of the difference.

The difference states that the `training_loan.Borrower.fullName` BOM attribute cannot be found within the `training_loan.Borrower` XOM class.

4. Open the list of proposed actions to solve it.

Differences and Actions

Actions:	Deprecate the BOM attribute "training_loan.Borrower.fullName"	Perform and save	Clear table
Perform action:	Deprecate the BOM attribute "training_loan.Borrower.fullName"	an action to solve this difference.	
Origin	Delete the BOM attribute "training_loan.Borrower.fullName"		
XOM	training_loan.Borrower	Modified	BOM attribute "training_loan.Borrower.fullName" not found within XOM

With the possible actions, you can either *delete* or *deprecate* the BOM attribute:
`training_loan.Borrower.fullName`

5. Select: **Deprecate the BOM attribute "training_loan.Borrower.fullName"** and click **Perform and save**.



Information

You can ignore the error icon for the `Borrower.fullName` BOM member.

You learn how to resolve this error in later steps.

6. View the deprecation result in the `Borrower.fullName` BOM member in the BOM editor.
 a. Expand **loan-rules > bom > model > training_loan > Borrower**.

- __ b. Double-click the **fullName** attribute.

Member fullName (class: training_loan.Borrower)

General Information

Name:	fullName	
Type:	java.lang.String	Browse...
Class:	training_loan.Borrower	Browse...
<input type="radio"/> Read/Write	<input checked="" type="radio"/> Read Only	<input type="radio"/> Write Only
<input type="checkbox"/> Static	<input type="checkbox"/> Final	
<input checked="" type="checkbox"/> Deprecated		
<input type="checkbox"/> Update object state		
<input type="checkbox"/> Ignore for DVS		

Notice how this `fullName` attribute is marked as deprecated.

- __ 7. View the deprecation result in the `displayFullName` rule.

- __ a. Expand **loan-rules > rules**.
- __ b. Reopen the `displayFullName` action rule.
- __ c. Hover your mouse over the **Warning** icon.

Action Rule: displayFullName

General Information

Name : displayFullName

Documentation

Content

1 **then**

2 **Warning** : 'training_loan.Borrower: fullName' is deprecated.

- __ 8. View the result in the Problems view.

- __ a. Notice the error:

Cannot find attribute "fullName" in execution class
"training_loan.Borrower"

- __ b. Notice the warning messages and the resources to which they apply:

- The deprecated messages for the `displayFullName` action rule
- The deprecated message for the `fullName` attribute of the `Borrower` BOM class

**Questions**

Can you identify how to update the projects in your workspace to resolve the problems and warnings that are currently identified?

Answer

Several options exist to resolve this problem:

- Delete the `fullName` attribute of the `Borrower` BOM class, and delete the `displayFullName` action rule.
- Implement the Getter method of the `fullName` attribute of the `Borrower` BOM class in its **BOM to XOM mapping** section to replace the missing XOM method.

However, these options are business decisions, so before you implement a solution, you must first consult with the business analysts.

**Requirements**

The decision is made to implement the first listed solution.

- 9. Delete the `fullName` attribute of the `Borrower` BOM class.
 - a. Open the `Borrower` BOM class in the BOM editor.
 - b. In the Members list, select **fullName** and click **Delete**.
 - c. When prompted to confirm removal of this BOM member, click **Yes**.
 - d. Save your work.
- 10. Delete the `displayFullName` action rule.
 - a. In the Rule Explorer, right-click the `displayFullName` action rule.
 - b. Click **Delete**.
 - c. When prompted to confirm deletion, click **Yes**.

All errors and warnings should now be removed from the Problems view.

End of exercise

Exercise review and wrap-up

The first part of the exercise looked at how a change in the vocabulary affects rule artifacts, and how the rule artifacts relate to the vocabulary in the BOM.

The second part of the exercise looked at how to keep the XOM and the BOM of the rule project consistent with each other upon changes in the XOM.

(Optional) To see the solution to this exercise, switch to a new workspace and import the project **Ex 04: Refactoring > 02-answer**.

Exercise 5. Working with ruleflows

What this exercise is about

In this exercise, you learn how to create a ruleflow.

What you should be able to do

After completing this exercise, you should be able to:

- Describe the parts of a ruleflow
- Create a ruleflow
- Orchestrate rule selection and execution through the ruleflow

Introduction

In this exercise, you explore an existing ruleflow to see how ruleflows are designed and how they orchestrate rule execution. You then create your own ruleflow in the Ruleflow editor.

The exercise involves these tasks:

- Section 1, "Exploring a ruleflow diagram"
- Section 2, "Creating a ruleflow"
- Section 3, "Defining a main ruleflow"

Requirements

This exercise uses the files that are installed in the
`<InstallDir>\studio\training\Dev 05 - Ruleflows\01-start`
directory.

Section 1. Exploring a ruleflow diagram

In this section, you explore the parts of an existing ruleflow in the Ruleflow editor.

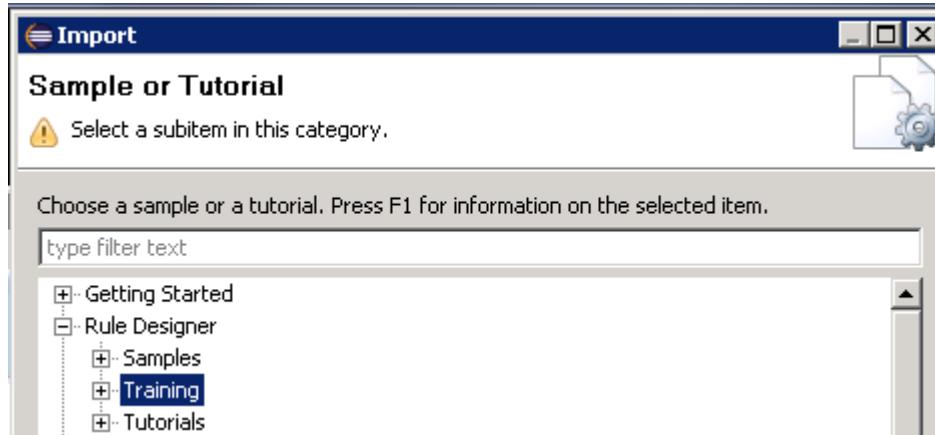
1.1. Setting up your environment for this exercise

- ___ 1. In Rule Designer, switch to a new workspace:
 - ___ a. From the **File** menu, click **Switch Workspace > Other**.
 - ___ b. In the Workspace Launcher window, enter the path:
`<LabfilesDir>\workspaces\ruleflow`
- ___ 2. Close the Welcome view.
- ___ 3. Use the **Import** menu to import the exercise start project.
 - ___ a. In Eclipse, click **File > Import** or right-click the Rule Explorer and click **Import**.
 - ___ b. In the Select window, expand **Operational Decision Manager**, select **Samples and Tutorials**, and click **Next**.
 - ___ c. On the “Sample or Tutorial” page, expand the **Rule Designer > Training** node.



Hint

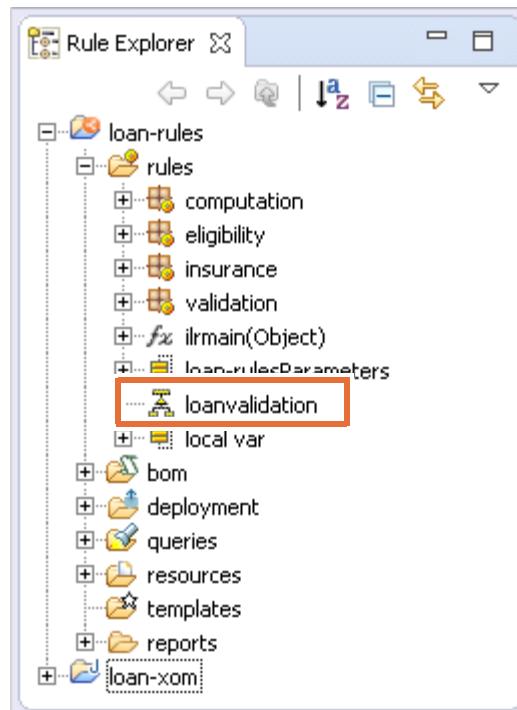
You can collapse the Rule Designer node and then expand it to simplify the list.



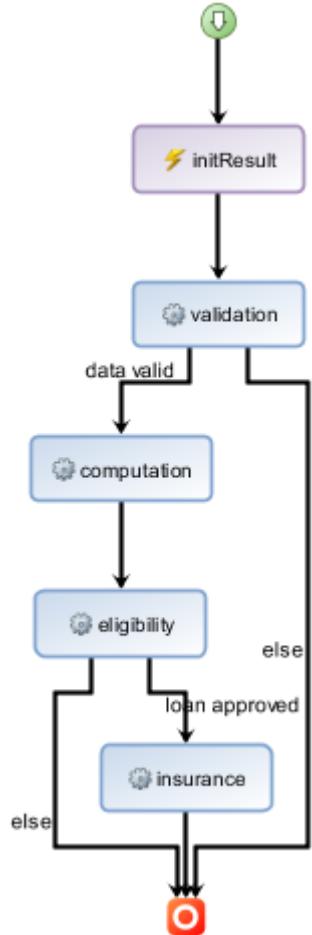
- ___ d. Expand the **Ex 05: Ruleflows** node, and select **01-start**.
- ___ e. Make sure that the **Import shared projects** check box is selected, and click **Finish**.
- ___ 4. When the workspace finishes building, close the Help view.

1.2. Exploring the ruleflow

1. In Rule Explorer, expand the `rules` folder of the `loan-rules` decision service and double-click the `loanvalidation` ruleflow to open it in the Ruleflow editor.



2. Notice the outline of tasks and the transitions between them.



Questions

Do you recognize the ruleflow tasks?

Do you understand the paths that can be taken through the ruleflow?

Can you determine what the ruleflow does by looking at how it is organized?

Answer

The ruleflow organizes the flow of rule execution in a ruleset to produce a decision.

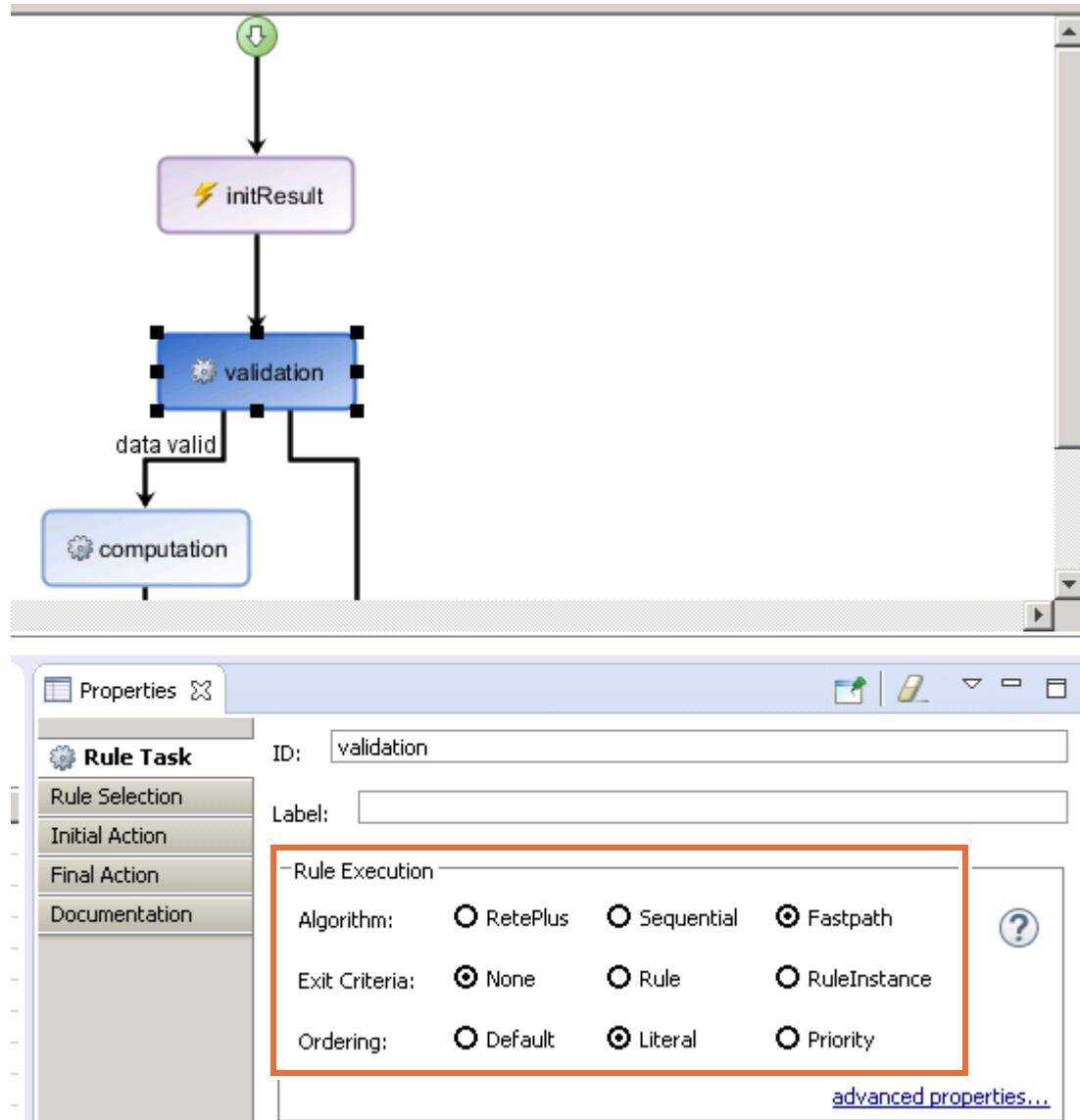
Within the ruleflow, rules are evaluated in groups or *rule tasks*. Each rule task is equivalent to a rule package.

Evaluation of each rule task produces a result or decision. For example, after executing the rules in the **validation** task, the result would be either *valid* or *not valid*. If the result is *valid*,

the ruleflow follows the path to the computation task. Otherwise, the ruleflow goes directly to the end and the rule execution terminates.

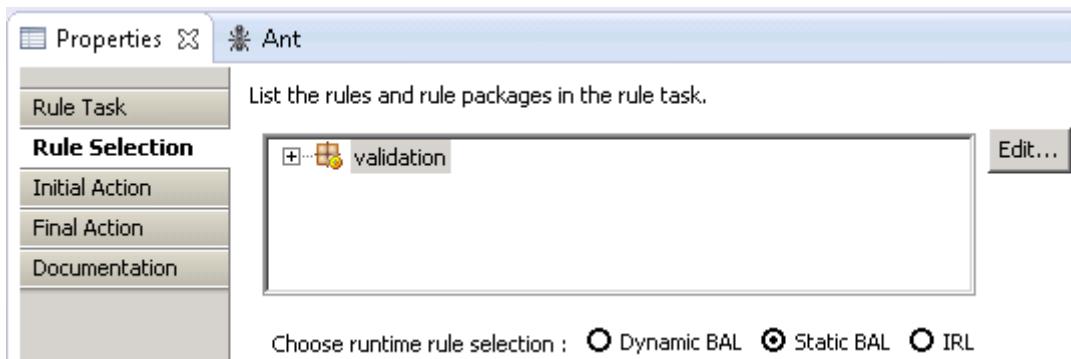
Transitions between tasks define the path through the ruleflow. You set conditions on transitions to define which path to take according to the results of evaluating a rule task.

- 3. Click the **validation** rule task in the diagram to see its properties in the Properties view.
- a. On the **Rule Task** tab of the Properties view, look at how you can set the **Algorithm**, **Exit criteria**, and **Ordering** rule execution properties of this rule task.

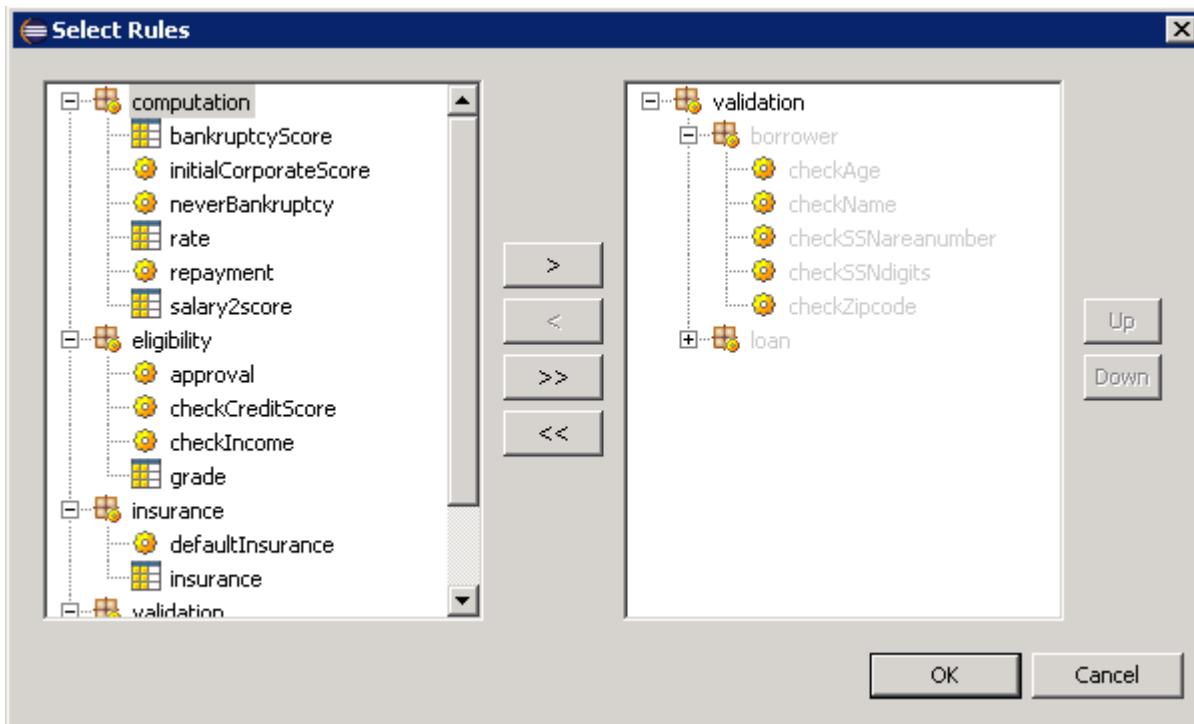


These execution properties dictate how instances of the rules in this rule task are executed.

- __ b. On the **Rule Selection** tab of the Properties view, expand the **validation** package and subpackages to see the contents.



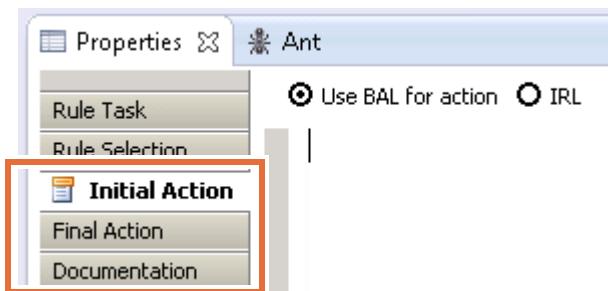
- __ c. On the **Rule Selection** tab of the Properties view, click **Edit** to open the Select Rules window.
 __ d. Take some time and experiment with selecting which rules to include in this task.



Notice that all the validation rules are included. However, this window provides the option of removing rules or changing the order in which they are evaluated by moving them up or down.

- __ e. Click **Cancel** to close the Select Rules window.

- ___ f. Take some time to explore the **Initial Action** tab, the **Final Action** tab, and the **Documentation** tab of the validation task properties.



- On the **Initial Action** tab, you define the actions to take before this task executes. You can write these actions either in Business Action Language (BAL) or in ILOG Rule Language (IRL).
 - On the **Final Action** tab, you define the actions to take after this task executes. You can write these actions either in BAL or in IRL.
 - On the **Documentation** tab, you can write comments to describe this task.
- ___ 4. Create a rule task.
- ___ a. Drag any rule package from the Rule Explorer into the Ruleflow editor.
 - ___ b. Delete the task from the diagram when you are finished.

1.3. Exploring action tasks and transitions

See now how action tasks and transition conditions are expressed in the ruleflow, and how they can use ruleset parameters and ruleset variables.



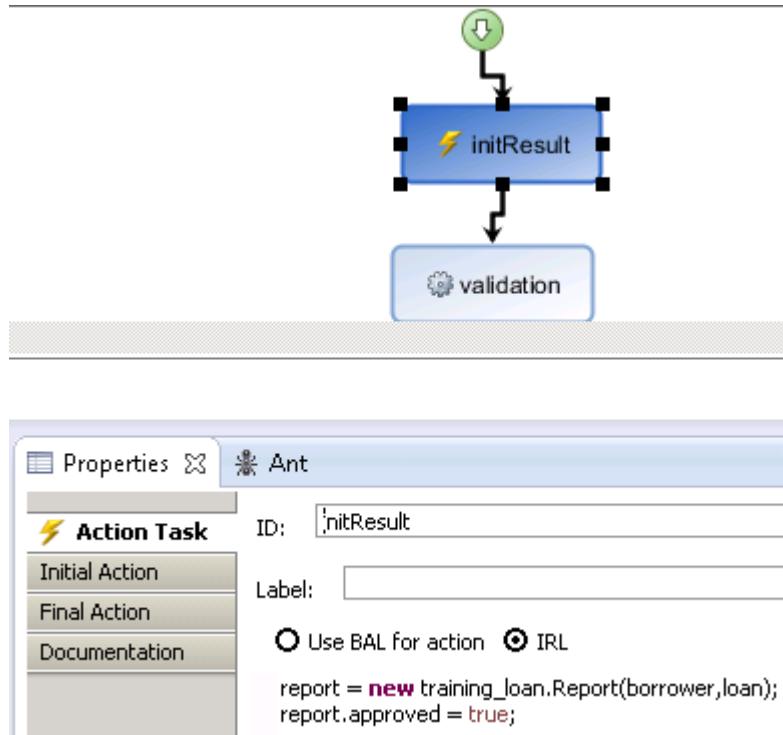
Questions

Can you identify an action task in the ruleflow?

Answer

The `initResult` task is an action task.

- 1. In the `loanvalidation` ruleflow, select the `initResult` task to see its properties in the Properties view.



- 2. On the **Action Task** tab of the Properties view, look at the code of the `initResult` task.

The function code of this `initResult` action task is written in ILOG Rule Language (IRL), which is similar to Java code.

In the present case, the `initResult` action task is used to create the `report` output ruleset parameter.

**Questions**

The function code does not set the initial value of the `loanApproved` ruleset variable. Why not?

Answer

You do not have to set the initial value of the `loanApproved` ruleset variable in the ruleflow. This initial value is already given as part of the ruleset variable definition, in the `local var` variable set.

- ___ 3. View the properties for the `data valid` transition.
 - ___ a. In the `loanvalidation` ruleflow diagram, click the `data valid` transition.
 - ___ b. In the Properties view, click the **Condition** tab to see the condition properties.

**Questions**

Do you see how to label a transition and how to write the condition of a transition?

Answer

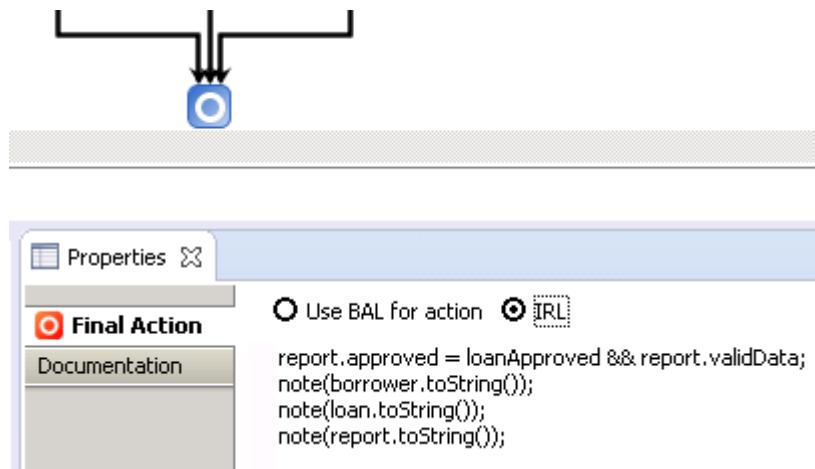
You set the label and the condition of a transition on its **Condition** tab. The condition might be written in either Business Action Language (BAL) or ILOG Rule Language (IRL). You learn what BAL and IRL are later in this course.

Often, this transition is expressed by using a Boolean ruleset parameter, a Boolean ruleset variable, one of their Boolean attributes, or a logical combination of them. In the present case, the transition is based on the value of the Boolean `validData` attribute of `report`, the output ruleset parameter of the `training_loan.Report` class.

This transition condition is written in BAL.

- ___ 4. Take some time to explore the relationship between the `validData` attribute of the `training_loan.Report` class, its verbalization that you can see in the BOM of the `loan-rules` decision service, and the way this transition condition is expressed.
- ___ 5. View the properties for the end node.
 - ___ a. In the `loanvalidation` ruleflow diagram, click the end node.
 - ___ b. In the Properties view, click the **Final Action** tab.

- __ c. View the **Final Action** properties.



The Final Action makes sure that the `approved` attribute of the `report` ruleset parameter is set according to the latest values of the `loanApproved` ruleset variable and the `validData` attribute of the `report` ruleset parameter.

This Final Action is written in IRL:

```
report.approved=loanApproved&&report.validData;
note(borrower.toString());
note(loan.toString());
note(report.toString());
```

1.4. Using ruleset parameters and variables in rules

Rule authoring is described in detail later, but for now, take some time to explore how the rules manipulate the ruleset parameters and the ruleset variable that are defined in the rule project.

- __ 1. In the `loan-rules` decision service, expand the `eligibility` package.
- __ 2. Double-click the `approval`, `checkCreditScore`, and `checkIncome` rules of the `eligibility` package to open them in the rule editor.



Questions

How do these action rules manipulate the `loanApproved` ruleset variable and the `report` ruleset parameter that are defined for this project?

Answer

The action rules of the eligibility package are expressed in BAL, and their expression is based on the BOM verbalization.

- These action rules use the `loanApproved` ruleset variable, which is verbalized as:
'the loan is approved'
- These action rules use the `report` ruleset parameter, which is verbalized as:
'the loan report'

- ___ 3. Close the rule editor windows for these rules.

Section 2. Creating a ruleflow

Now that you explored a ruleflow in its entirety, create a ruleflow with the Ruleflow editor.

- __ 1. In the `loan-rules` project, create a ruleflow:
 - __ a. Right-click the `rules` folder in the `loan-rules` project and click **New > Ruleflow**.
 - __ b. In the **Name** field of the New Ruleflow wizard, type `loanvalidation_2` and click **Finish**.

The `loanvalidation_2` ruleflow automatically opens in the Ruleflow editor.

In the next steps, you re-create parts of the `loanvalidation` ruleflow.



Hint

Keep the `loanvalidation` ruleflow open while you work on the `loanvalidation_2` ruleflow in a second Ruleflow editor. By switching between windows and tabs, you can see what you must update when you work through the next steps.

- __ 2. Add the start node of the `loanvalidation_2` ruleflow.
 - __ a. Click  **Create a start node**.
 - __ b. In the Ruleflow editor main area, click where you want create the start node.
- __ 3. Add the end node of the `loanvalidation_2` ruleflow.
 - __ a. Click  **Create an end node**.
 - __ b. In the Ruleflow editor main area, click where you want to create the end node.
- __ 4. Create the `initResult` action task of the `loanvalidation_2` ruleflow with the same IRL function code as in the `initResult` action task of the `loanvalidation` ruleflow.
 - __ a. Click  **Create an action task**.
 - __ b. In the Ruleflow editor main area, click where you want to create the action task.
 - __ c. Click the created action task, and in the Properties view, click the **Action task** tab.

__ d. On the **Action Task** tab:

- In the **ID** field, enter: initResult
- Select the **IRL** option.
- Enter the following IRL function code:

```
report = new training_loan.Report(borrower,loan);
report.approved = true;
```



__ e. Save your work.



You can ignore the errors that you see.

- __ 5. Create the validation, computation, eligibility, and insurance rule tasks by dragging each of those rule packages from Rule Explorer onto the Ruleflow editor.
- __ 6. Set the **Rule Task** properties of each rule task in the loanvalidation_2 ruleflow to match the corresponding **Rule Task** properties of the loanvalidation ruleflow.
 - __ a. Select one of the rule tasks.
 - __ b. In the Properties view, set the task properties on the **Rule Task** tab to match the properties in the loanvalidation ruleflow.
 - **validation**
 - Algorithm: **Fastpath**
 - Exit Criteria: **None**
 - Ordering: **Literal**
 - **computation**
 - Algorithm: **RetePlus**
 - Exit Criteria: **None**
 - Ordering: **Default**
 - **eligibility**
 - Algorithm: **Sequential**
 - Exit Criteria: **None**

- Ordering: **Priority**
- **insurance**
 - Algorithm: **RetePlus**
 - Exit Criteria: **None**
 - Ordering: **Default**

___ 7. Create the transitions between the tasks.

- ___ a. In the Ruleflow editor palette, click  **Create a transition**. You can now create multiple transitions in the diagram.
- ___ b. Click the start node and then click the `initResult` action task.
- ___ c. Click the `initResult` action task, and then click the `validation` rule task.
- ___ d. Click the `validation` rule task, and then click the `computation` rule task.
- ___ e. Click the `computation` rule task, and then click the `eligibility` rule task.
- ___ f. Click the `eligibility` rule task, and then click the `insurance` rule task.
- ___ g. Click the `insurance` rule task, and then click the end node.
- ___ h. Create another transition between the `validation` rule task and the end node.



Hint

To disable the transition creation mode in the Ruleflow editor palette, click  **Create a transition** again.

- ___ 8. Set the Condition properties of the transition between the validation and computation rule tasks to match the corresponding transition in the `loanvalidation` ruleflow.
 - ___ a. Click the transition between the `validation` and `computation` rule tasks.
 - ___ b. In the Properties view, click the **Condition** tab.
 - ___ c. In the **Label** field, enter: `data valid`
 - ___ d. Select **Use BAL for transition condition**.
 - ___ e. In the condition editor, enter: 'the loan report' is valid data
- ___ 9. To adjust the ruleflow diagram layout, go to the Ruleflow editor toolbar and click the  **Layout All Nodes** icon.
- ___ 10. Save your work.
- ___ 11. Compare your ruleflow to the `loanvalidation` ruleflow.



Questions

Are you missing any tasks or transitions?

What tasks must you do to complete the `loanvalidation_2` ruleflow diagram so that it matches the `loanvalidation` diagram?

Answer

To complete the `loanvalidation_2` diagram so that it matches the `loanvalidation` diagram, you must complete the following tasks.

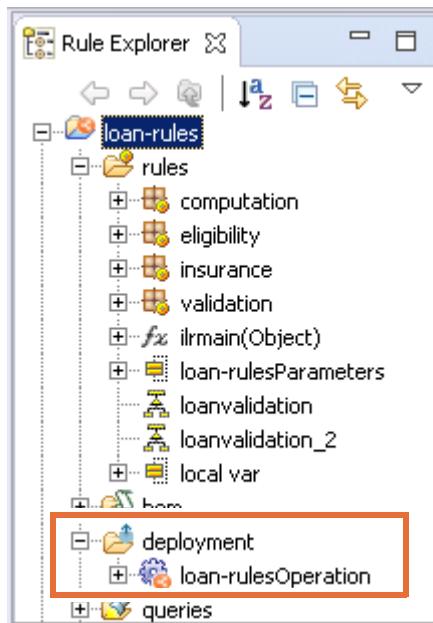
- ___ 1. Add a transition between the `eligibility` rule task and the end node.
- ___ 2. Add a label and a condition to the transition between the `eligibility` and `insurance` tasks.
 - ___ a. Select the transition between the `eligibility` and `insurance` rule tasks.
 - ___ b. Click the **Condition** tab.
 - ___ c. In the **Label** field, enter: `loan approved`
 - ___ d. Select **Use BAL for transition condition**.
 - ___ e. In the condition editor, enter: 'the loan is approved'
- ___ 3. Save your work.

Section 3. Defining a main ruleflow

An application can have several ruleflows, but one of them must be defined as the main ruleflow. The main ruleflow is defined in the decision operation for the decision service.

In this task, you learn how to define a main ruleflow.

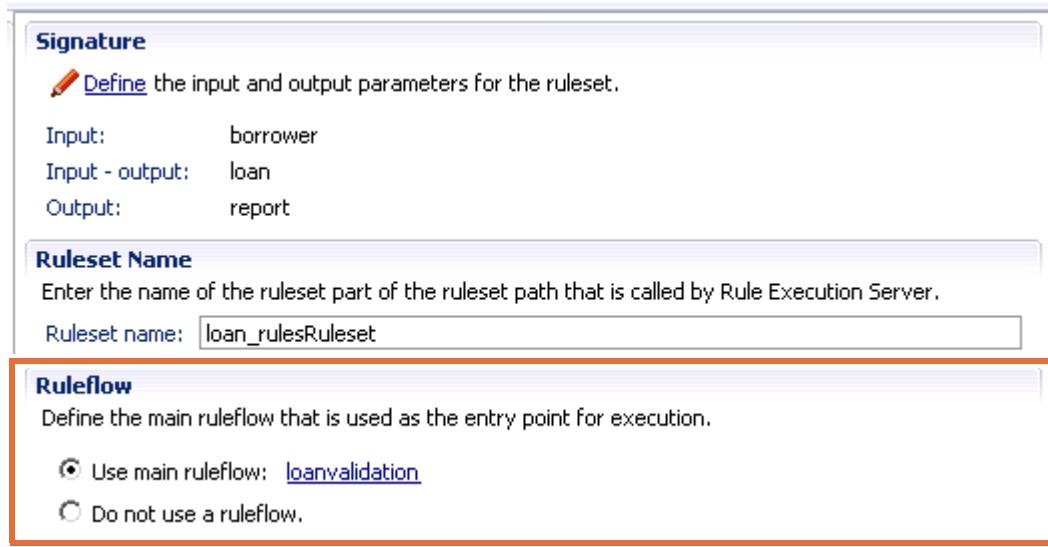
- __ 1. Open the decision operation for the loan-rules decision service.
- __ a. In the Rule Explorer, expand **loan-rules > deployment**.



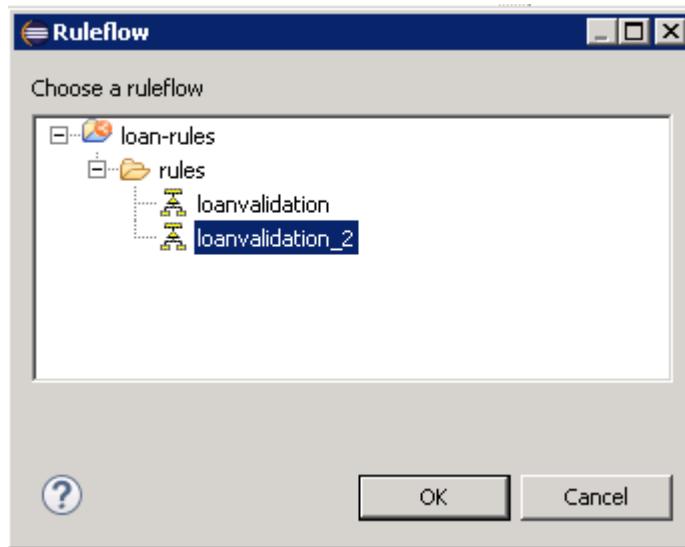
- __ b. Double-click **loan-validationOperation** to open it in the decision operation editor.

In the **Ruleflow** section of the decision operation editor, you see the following options:

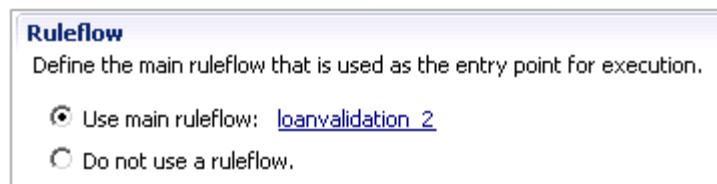
- **Use main ruleflow**
 - In this exercise, the loanvalidation ruleflow is already selected as the main ruleflow.
- **Do not use a ruleflow**



2. Change the main ruleflow to the loanvalidation_2.
- a. In the **Ruleflow** section, next to the **Use main ruleflow** option, click the **loanvalidation** link.
 - b. In the Ruleflow window, select **loanvalidation_2** and click **OK**.



The `loanvalidation_2` ruleflow is now set as the main ruleflow.



- 3. Change the main ruleflow back to the `loanvalidation` ruleflow.
- 4. Save your work (Ctrl+S).

End of exercise

Exercise review and wrap-up

This exercise looked at how to design a ruleflow.

Exercise 6. Exploring action rules

What this exercise is about

In this exercise, you learn how to write action rules.

What you should be able to do

After completing this exercise, you should be able to:

- Identify the parts of an action rule
- Explain the difference between using automatic variables or rule variables

Introduction

To learn how to author action rules, you review the Trucks and Drivers rule project, which is a complete project with rules that use various BAL constructs. You explore and run these rules to understand rule behavior during rule execution.

The exercise involves these tasks:

- Section 1, "Exploring rule structure"
- Section 2, "Exploring rule behavior"
- Section 3, "Working with automatic variables"

Requirements

This exercise uses the following files, which are installed in the `<InstallDir>\studio\training` directory:

- Start project: Dev 06 – `Exploring rules\01-start`
- Solution project: Dev 06 – `Exploring rules\02-answer`

You can use the solution project in "Exercise review and wrap-up" on page 6-12.

Section 1. Exploring rule structure

In this section, you explore the behavior of rules in the Trucks and Drivers rule project. This rule project is designed to demonstrate specific BAL constructs. You can open each rule, review the constructs that are used, and then view the results that are produced after rule execution.

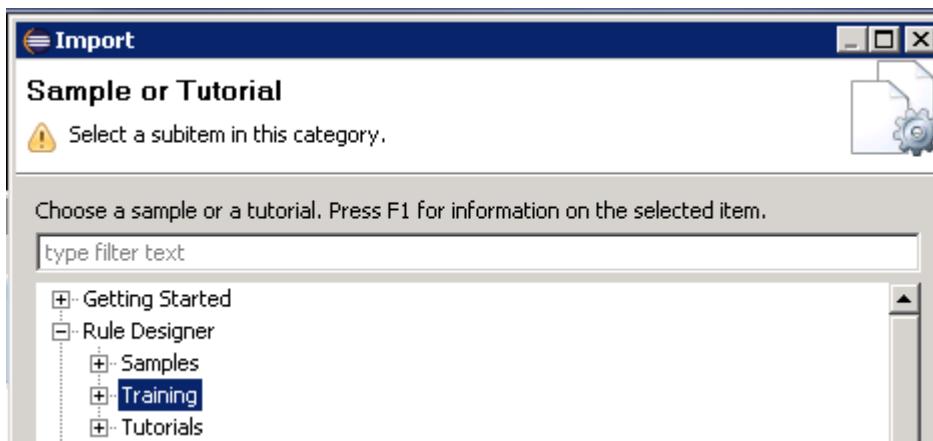
1.1. Setting up your environment for this exercise

- ___ 1. In Rule Designer, switch to a new workspace:
 - ___ a. From the **File** menu, click **Switch Workspace > Other**.
 - ___ b. In the Workspace Launcher window, enter the path:
`<LabfilesDir>\workspaces\explore_rules`
- ___ 2. Close the Welcome view.
- ___ 3. Use the **Import** menu to import the exercise start project.
 - ___ a. In Eclipse, click **File > Import** or right-click the Rule Explorer and click **Import**.
 - ___ b. In the Select window, expand **Operational Decision Manager**, select **Samples and Tutorials**, and click **Next**.
 - ___ c. On the “Sample or Tutorial” page, expand the **Rule Designer > Training** node.



Hint

You can collapse the Rule Designer node and then expand it to simplify the list.

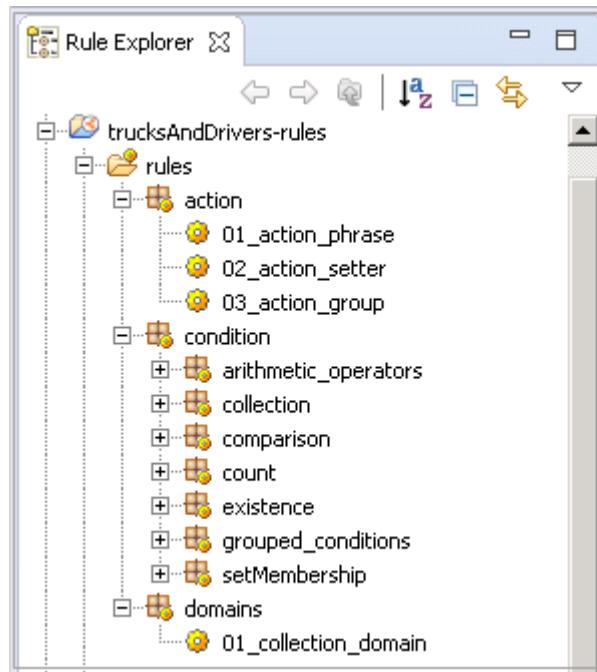


- ___ d. Expand the **Ex 06: Exploring rules** node, and select **01-start**.
- ___ e. Make sure that the **Import shared projects** check box is selected, and click **Finish**.
- ___ 4. When the workspace finishes building, close the Help view.

1.2. Exploring the rules

The action rules to explore are provided in different rule packages.

- 1. In Rule Explorer, expand: **trucksAndDrivers-rules > rules**.



- 2. Notice the **condition** package, which includes the following subpackages to illustrate these BAL constructs in condition statements:
 - **Arithmetic operators:** Action rules in this rule package show how to use mathematical operators to carry out calculations in the condition of an action rule.
 - **Collections:** Action rules in this rule package show how to define a variable to retrieve a list of objects (`java.util.Collection`).
 - **Comparisons:** Action rules in this rule package show how to use BAL operators to compare the values of strings or the number of objects in working memory.
 - **Count tests:** Action rules in this rule package show how to use operators that count the number of occurrences of an object.
 - **Existence tests:** Action rules in this rule package show how to use BAL operators to test the existence of objects with certain members in working memory.

Conditions are qualified with the `where` keyword.

 - **Grouped conditions:** Action rules in this rule package show how to use the `all`, `any`, and `none` BAL constructs to combine conditions, as an alternative to logical operators (`and`, `it is not true that`, `or`) between condition statements.
 - **Tests for membership in a set:** Action rules in this rule package show how to write conditions that test whether a member of an object in working memory is part of a set (`java.util.Collection`).
- 3. In the Rule Explorer, open the BOM by expanding **trucksAndDrivers-bom > bom > model**.

4. Expand **drivers** and review the objects and vocabulary.



Questions

Consider these questions as you review the **rules** and **vocabulary** in this rule project:

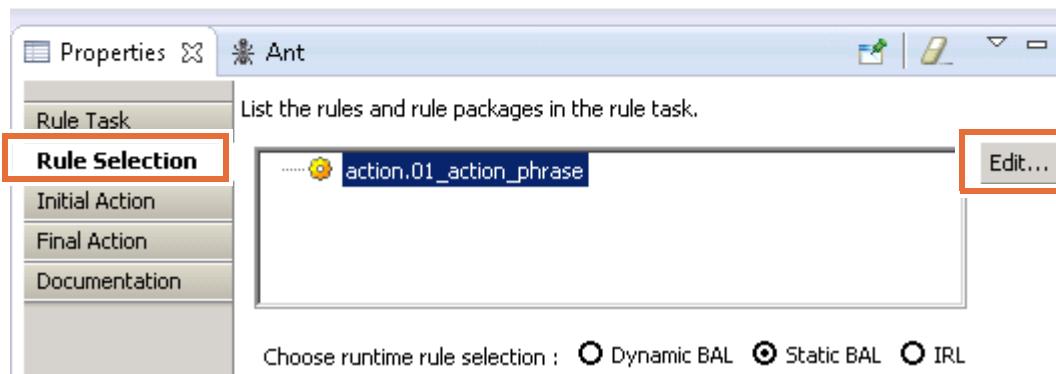
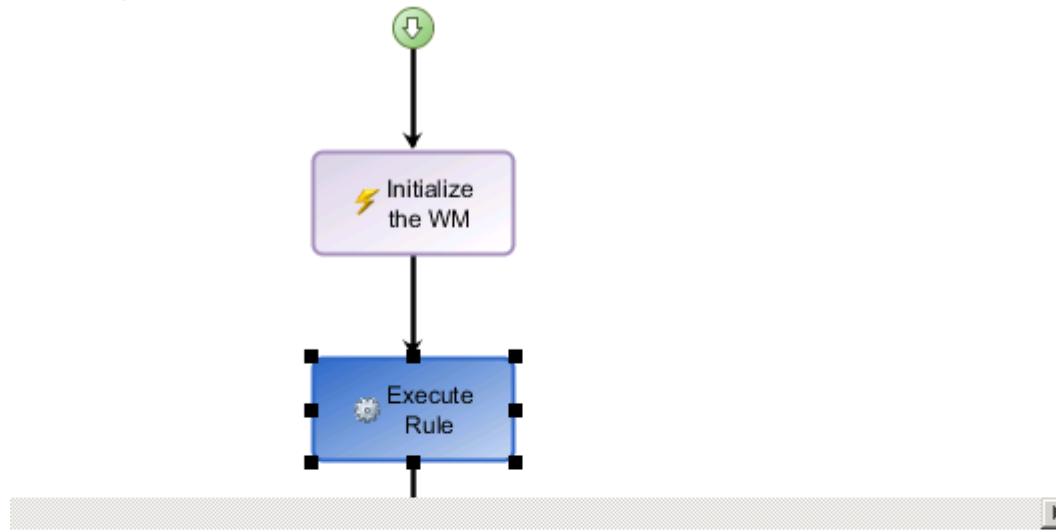
- Can you identify the objects on which the action rules work?
- Are there any objects for which you cannot identify the origin?
- Based on your experience with Java programming, can you determine what the terms *definitions*, *conditions*, *actions*, and *variables* mean in relation with the design of the rules?

Section 2. Exploring rule behavior

In this part of the exercise, you execute at least one action rule from each rule package in the trucksAndDrivers-rules project.

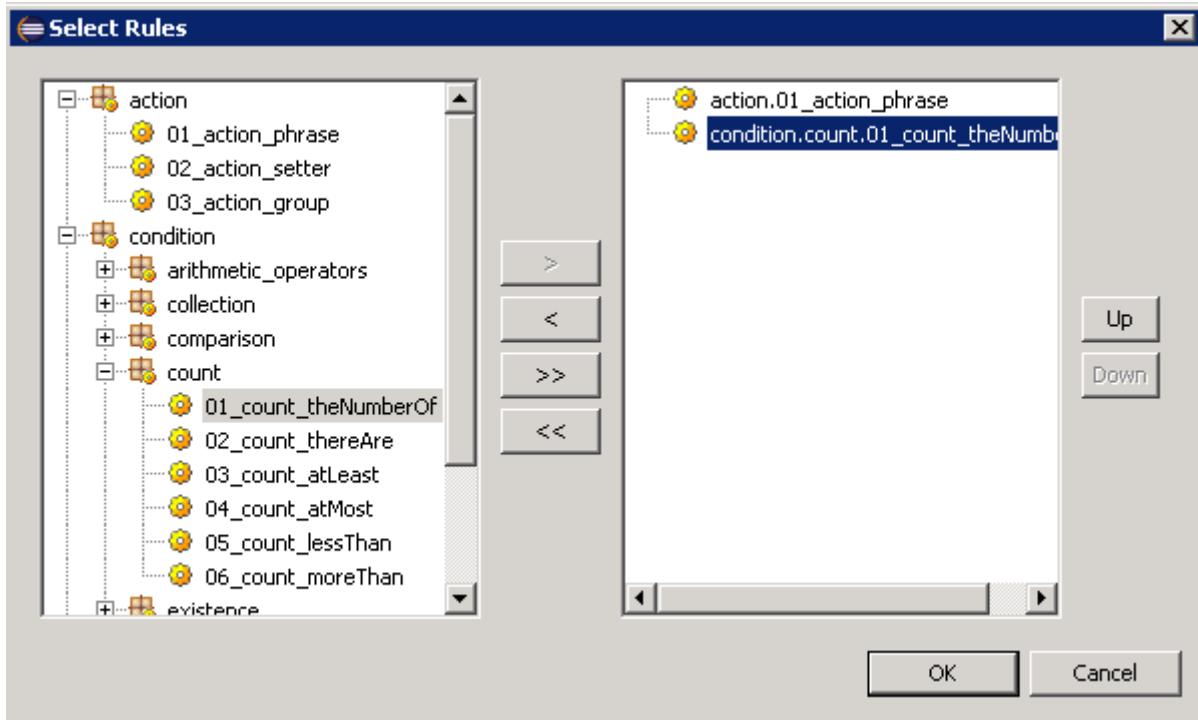
To execute a rule:

- ___ 1. In Rule Explorer, expand **trucksAndDrivers-tests > rules**.
- ___ 2. Double-click the **testFlow** ruleflow to open it in the Ruleflow editor.
- ___ 3. In the ruleflow diagram, select the **Execute Rule** rule task and open the Properties view.
- ___ 4. In the Properties view, click the **Rule Selection** tab, and click **Edit** to choose a rule to run.



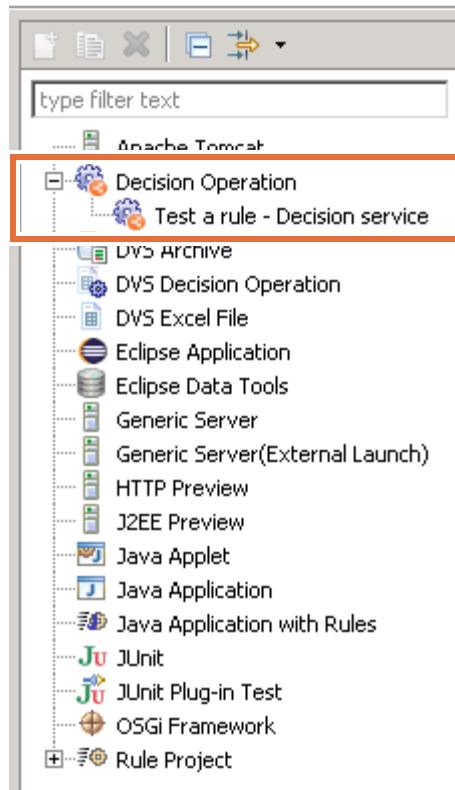
- ___ 5. In the Select Rules window, you can choose any rule to start with, and you can also repeat these steps to try other rules.
 - ___ a. Browse through the list of rules in the left section of the Select Rules window and select a rule to add.
 - ___ b. Click the move right (>) button to move the rule to the selected list.
 - ___ c. Click **OK**.

The example rule that is selected in this exercise is: **condition > count > 01_count_theNumberOf**



- ___ 6. Save your work (Ctrl+S).
- ___ 7. From the Run menu, click **Run Configurations** to set up rule execution.

8. In the Run Configurations window, expand **Decision Operation** and select **Test a rule - Decision service**.



9. In the Test a rule - Decision service launch configuration, click **Run** to start the execution of the trucksAndDrivers-tests ruleflow.

The console opens and you can see that the rule executed correctly by looking at the message, which should match the action statement in the rule.

```
<terminated> Test a rule - Decision service [Decision Operation] C:\Program Files\IBM\ODM88\jdk\bin\javaw.exe (Jan 13, 2016, 1:59:44)
The number of deliveries in the working memory is 10
the truck overloaded TRUCK-T2000 has its smallest delivery removed
```

Section 3. Working with automatic variables

In this part of the exercise, you create automatic variables in your BOM and use them in your rule artifacts. By doing so, you learn the differences in using an automatic variable instead of a rule variable in action rules.

- 1. In the `trucksAndDrivers-bom` project, expand **bom > model > drivers** and double-click the `Truck` BOM class to edit it in the BOM editor.
- 2. In the **Class Verbalization** section, select **Generate automatic variable**.

Class Verbalization

Generate automatic variable

Term: `truck` [Edit term.](#)

i the truck, a truck, the trucks....

A variable that is called `the truck` is automatically made available in the rule editors. You can use the automatic variable to author rules based on objects in the working memory that are instances of the `Truck` BOM class.

- 3. Save the BOM (Ctrl+S) and wait for the workspace to rebuild.
- An error message appears on the **trucksAndDrivers-rules** project.
- 4. In the Problems view, double-click the error to open the problem rule.

Content

```

definitions
  set !truck! to a truck ;
if
  the model of truck is one of { the F150 , the MACK TRUCK }
then
  display the message "The truck model of " + the serial number of

```

Intellirule IRL | 01_setMembership_isOneOf.brl

Rule Proj... Rule Exec... Problems Tasks BOM Up... DVS Proj... □ □

1 error, 0 warnings, 0 others

Description	Resource	Path	Location
Errors (1 item)	01_SetMem...	/trucksAndDrive...	line 2
<input checked="" type="checkbox"/> An automatic variable 'truck' is already declared.	01_SetMem...	/trucksAndDrive...	line 2

You can also open the rule directly in the `trucksAndDrivers-rules` project by expanding **rules > condition > setMembership** and double-clicking `01_SetMembership_isOneOf`.

**Questions**

Why does this rule no longer compile?

Answer

The `01_setMembership_isOneOf` action rule cannot compile because it declares a variable that is called `truck`, which conflicts with the name of the automatic variable that you created.

**Information**

In the following steps, you gain hands-on experience with the rule editor by trying to intuitively use its authoring functions.

**Questions**

How can you rewrite the rule so that it compiles?

Answer

- To avoid conflicting variables, you can delete the definitions part of the `01_setMembership_isOneOf` rule.
- As indicated in the BOM, the automatic variable for the `Truck` class is verbalized as `the truck`, meaning that everywhere you previously had `truck` only, you must now use: `the truck`

— 5. Delete the **definitions** statement.

- ___ 6. Modify the `01_setMembership_isOneOf` action rule to use the automatic variable as shown here:

```
if  
    the model of the truck is one of { the F150 , the MACK TRUCK }  
then  
    display the message "The truck model of " + the serial number of the truck  
    + " is F150 or MACK_TRUCK";
```

 **Hint**

You can click the error icons on the sides of the rule editor to see how to fix the error. Or you can right-click `truck`, click **Quick Fix** for a solution, and double-click the solution that is proposed.

You can also double-click `truck` and select the correct variable from the list, but if you use this option, be careful not to overwrite any other part of the rule statement.

- ___ 7. Save the rule.
- ___ 8. In the `trucksAndDrivers-rules` project, expand **rules > action** and double-click the `01_action_phrase` rule to open it.



Questions

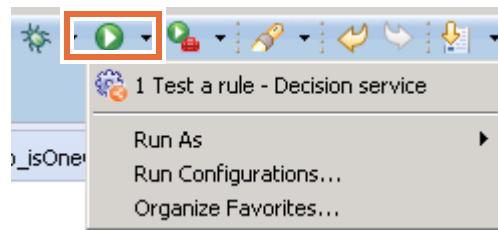
Can you figure out how you can simplify this rule with the new automatic variable?

Do not change it yet.

Answer

- The definitions in the `01_action_phrase` action rule are useless.
- Where the `01_action_phrase` action rule uses the rule variable '`the truck`' (with single quotation marks), it can use the automatic variable `the truck` (without quotation marks) instead.

9. To use the automatic variable, delete the definition statement and modify the `01_action_phrase` rule in the `action` rule package to match this content:
- ```
if
 the current load of the truck is more than the capacity of the model of the
 truck
then
 remove the smallest delivery of the truck;
 display the message "the truck overloaded " + the serial number of the
 truck + " has its smallest delivery removed";
```
10. Save the `01_action_phrase` action rule and execute it again to verify that the modification did not change the execution results.
11. You can rerun the Test a rule - Decision service configuration by clicking the **Run** icon on the toolbar, and clicking **Test a rule - Decision service** from the list.



The traces in the Console view are:

the truck overloaded TRUCK-T2000 has its smallest delivery removed

These traces should match the traces you had in Section 2, "Exploring rule behavior," on page 6-5.



### Optional

Optionally, you can make similar changes in the other action rules in the `trucksAndDrivers-rules` project.

- Delete all definitions of the rule variable called 'the truck' (with single quotation marks).
- Where you had the rule variable 'the truck' (with single quotation marks), you can use the automatic variable `the truck` instead (without quotation marks).

## End of exercise

## Exercise review and wrap-up

The first part of the exercise looked at how the action rules are structured and demonstrated their behavior during rule execution. You also saw the difference between using automatic variables or rule variables.

(Optional) To see the solution to this exercise, switch to a new workspace and import the project **Ex 06: Exploring rules > 02-answer**.

# Exercise 7. Authoring action rules

## What this exercise is about

In this exercise, you learn how to author action rules.

## What you should be able to do

After completing this exercise, you should be able to:

- Use the Intellirule editor and Guided editor to author action rules
- Use rule variables, automatic variables, and ruleset parameters in rule statements

## Introduction

In this exercise, you author the action rules that use rule variables, automatic variables, and ruleset parameters.

The exercise includes these tasks:

- Section 1, "Authoring action rules in the Intellirule editor"
- Section 2, "Authoring actions rules in the Guided editor"
- Section 3, "Authoring rules with ruleset parameters"
- Section 4, "Creating an action rule template"
- Section 5, "Authoring action rules from your action rule template"

## Requirements

You should complete these exercises before proceeding:

- Exercise 2, "Setting up decision services"
- Exercise 6, "Exploring action rules"

This exercise uses the following files, which are installed in the `<InstallDir>\studio\training` directory:

- Start project: Dev 07 – Authoring rules\01-start
- Solution project: Dev 07 – Authoring rules\02-answer
  - You can use the solution project in "Exercise review and wrap-up" on page 7-20.

## Section 1. Authoring action rules in the Intellirule editor

In this exercise, you author rules in the main Rule Designer rule editors:

- Intellirule editor
- Guided editor



### Note

You can also edit the action rules as pure text, in the Eclipse Text editor.

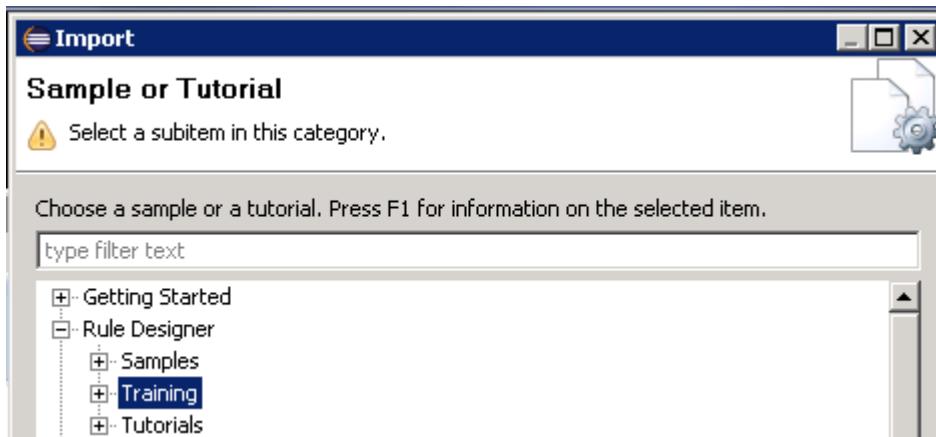
### 1.1. Setting up your environment for this exercise

1. In Rule Designer, switch to a new workspace:
  - a. From the **File** menu, click **Switch Workspace > Other**.
  - b. In the Workspace Launcher window, enter the path:  
`<LabfilesDir>\workspaces\author_rules`
2. Close the Welcome view.
3. Use the **Import** menu to import the exercise start project.
  - a. In Eclipse, click **File > Import** or right-click the Rule Explorer and click **Import**.
  - b. In the Select window, expand **Operational Decision Manager**, select **Samples and Tutorials**, and click **Next**.
  - c. On the “Sample or Tutorial” page, expand the **Rule Designer > Training** node.



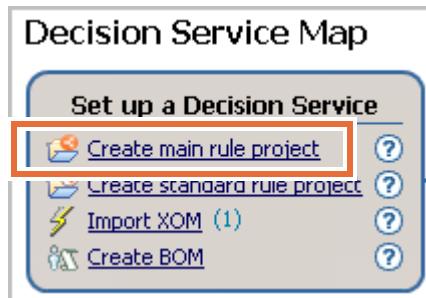
### Hint

You can collapse the Rule Designer node and then expand it to simplify the list.



- d. Expand the **Ex 07: Authoring rules** node, and select **01-start**.

- \_\_\_ e. Make sure that the **Import shared projects** check box is selected, and click **Finish**.
  - \_\_\_ 4. When the workspace finishes building, close the Help view.
- The workspace includes a predefined `loan-xom` XOM project and a separate `loan-bom` project.
- \_\_\_ 5. In Rule Designer, create a main rule project named: `loan-rules`
  - \_\_\_ a. In the “Set up a Decision Service” part of the Decision Service Map, click **Create main rule project**.



- \_\_\_ b. In the **Decision Service Rule Projects** section of the New Rule Project wizard, select **Main Rule Project** and click **Next**.
  - \_\_\_ c. In the **Project name** field, enter `loan-rules` and click **Next**.
  - \_\_\_ d. In the Main Rule Project References page, select **loan-bom**.  
This reference page indicates that the `loan-rules` project references the `loan-bom` project. The `loan-rules` project uses the vocabulary from this BOM project.
  - \_\_\_ e. Click **Finish**.
- \_\_\_ 6. Create a decision operation for the `loan-rules` decision service.
    - \_\_\_ a. In the “Define decision operation” part of the Decision Service Map, click **Add decision operation**.
    - \_\_\_ b. In the New Decision Operation window **Name** field, enter: `loan-rules` operation
    - \_\_\_ c. Click **Next**.
    - \_\_\_ d. In the Source Rule Project window, select **loan-rules** and click **Finish**.
  - \_\_\_ 7. Add a rule package named `loan` to the `loan-rules` decision service.
    - \_\_\_ a. In the “Define decision operation” part of the Decision Service Map, click **Go to operation map**.
    - \_\_\_ b. In the “Select an operation” window, select **loan-rules operation.dop**, and click **OK**.
    - \_\_\_ c. In the Orchestrate part of the Operation Map, click **Add rule package**.
    - \_\_\_ d. In the **Package** field of the New Rule Package window, enter: `loan`
    - \_\_\_ e. Click **Finish**.
  - \_\_\_ 8. Close the **loan-rules operation** tab.

**Information**

You can also add a rule package by expanding **loan-rules**, right-clicking the **rules** folder, and clicking **New > Rule Package**.

## 1.2. Authoring rules with the Intellirule editor

**Requirements****Scenario:**

A loan company must implement its loan policies, which evolve regularly, and must provide a loan application to its agents.

First, the application must verify input data from a form. Then, it must check customer eligibility, which is based on personal profile, score, and the type of loan requested. The application can also compute some insurance rates, as a function of the computed score.

Use the Intellirule editor to author some of the action rules that implement the Loan scenario.

- 1. In the **loan** rule package, add a rule that is called: `grade`
  - a. Right-click the **loan** rule package, and click **New > Action Rule**.  
The New Action Rule window opens.
  - b. In the **Name** field, enter: `grade`
  - c. Click **Finish** to close the New Action Rule window.  
The empty rule opens in the Intellirule editor.

**Note**

To verify whether you are using the Intellirule editor or the Guided editor, look at the tab of the editor window:

- **Guided** for the Guided editor



- **Intellirule** for the Intellirule editor



If the Intellirule editor is not the default editor that opens when you create an action rule, right-click this action rule in the Rule Explorer and click **Open With > Intellirule Editor**.

- \_\_ 2. Use the Intellirule editor to start authoring the grade rule.

The grade rule must define a rule variable to manipulate the Report object that is passed to the rule engine by the report ruleset parameter.

- \_\_ a. Press Ctrl+Spacebar to open the Content Assist menu and double-click **definitions**.



- \_\_ b. Content Assist continues to prompt you, so double-click **set <variable>**, and then double-click **variable1**.

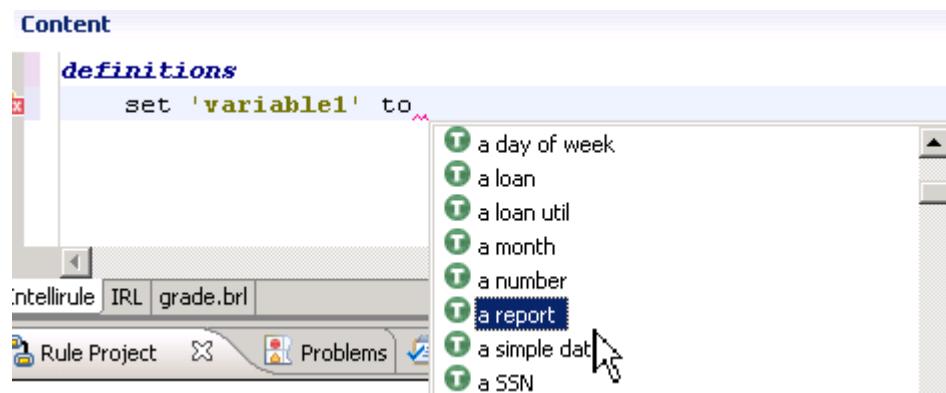


- \_\_ c. Put the cursor in the space after set 'variable1' and press Ctrl+Spacebar to open Content Assist.

- \_\_ d. Double-click **to <binding type>**.

The Content Assist menu now prompts you with a vocabulary list.

- \_\_ e. You want the variable to be of type Report, so scroll through the vocabulary list to find and double-click **a report**.



- \_\_ f. Finally, you are prompted to complete the definition statement for your variable with a semicolon (;), which is required in the Intellirule editor, or to add another clause to your statement. For this rule, the variable definition is complete, so you can double-click the semicolon.

```

Content
1 definitions
2 set 'variable1' to a report

```

Content Assist tooltip:

- ;
- from <relation>
- in <collection>
- where <test>

- \_\_ g. When prompted, select **if <condition>**.

```

Content
1 definitions
2 set 'variable1' to a report ;

```

Content Assist tooltip:

- if <condition>
- set <variable>
- then

The editor automatically formats the condition statement to start on a new line.

- \_\_ h. Before you continue with the condition statement, go back to the variable definition and type over 'variable1' to change it to: 'the report'



### Important

Make sure that you keep your variable name enclosed with single quotation marks ('').

- \_\_ 3. Now that you see how Content Assist can guide you to build the rule, complete the rule statement to match this content:

```

definitions
 set 'the report' to a report;
 if
 the amount of the loan of 'the report' is more than 1000000
 then
 set the grade of 'the report' to "GOLD";

```

- \_\_ 4. Save your work and close the editing window.  
\_\_ 5. In the `loan` rule package, create a second action rule called: `invalid corporate score`  
This rule must refuse a loan application when the corporate score is less than 1000.

6. For this rule, instead of using the Content Assist menu, type the rule content directly into the editor to match this content:

```
definitions
 set 'the report' to a report;
if
 the corporate score in 'the report' is less than 1000
then
 in 'the report', refuse the loan with the message "Corporate score less
 than 1000";
```

**Hint**

You can copy the rule content from the `author.txt` file and paste directly into the Intellirule editor. The `author.txt` file is in the `<LabfilesDir>\code` directory.

**Note**

Notice that you must pay attention to punctuation to avoid errors in the rule statement.

7. Save the rule (Ctrl+S).
8. Create another action rule in the `loan` rule package called: `too big loan`  
This rule should refuse a loan application when the loan is more than 10,000,000.

**Questions**

How do you write this action rule in the Intellirule editor?

**Answer**

The body of the action rule that is called `too big loan` must be:

```
definitions
 set 'the report' to a report;
if
 the amount of the loan of 'the report' is more than 10000000
then
 in 'the report', refuse the loan with the message "The loan amount is too
 big";
```



**Hint**

You can find the text of the too big loan rule in the `author.txt` file in the `<LabfilesDir>\code` directory.

- 9. Save your work and close the editing windows.

## Section 2. Authoring actions rules in the Guided editor

Use the Guided editor to author the remaining action rules to fully implement the Loan scenario. As in the previous section, you author these action rules by using rule variables and automatic variables.

- 1. Create an action rule in the `loan` rule package, named: `too long loan duration`.  
The rule must refuse a loan application when the loan duration is strictly greater than 200.  
By default, the rule opens in the Intellirule editor.
- 2. Close the editing window for the rule, and then in Rule Explorer, right-click the rule and click **Open With > Guided Editor**.



- 3. Define the rule content to match this text:

```
definitions
 set the report to a report
if
 the duration (in years) of the loan of the report is more than 200
then
 in the report, refuse the loan with the message The loan duration is too
 long
```

- a. Start with the variable definition statement:
  - Click **definitions**.
  - Click **variable1** and set the variable name by typing: `the report`
  - Click **select a choice** and click `<an object>` from the list.
  - Click **select an object** and select **a report** from the list.
- b. Continue editing the condition and action statements to complete the rule.



### Hint

To access the **is more than** BAL construct, click **is**.



### Questions

What differences did you identify between the Intellirule editor and the Guided editor?

## **Answer**

With the Intellirule editor, you must use punctuation, such as semicolons (;) at the end of definitions and actions, single quotation marks (') around variables, and double quotation marks ("") around String constants.

The Guided editor does not require these extra characters.

- 4. Save your work.
- 5. Close the editing window.

## Section 3. Authoring rules with ruleset parameters

For this part of the exercise, you use a ruleset parameter instead of the local rule variable. Recall that to define a ruleset parameter:

- You first create a variable set.
- Then, you bind the variables to ruleset parameters.



### Reminder

For detailed steps about defining ruleset parameters, see Section 4.2, "Creating ruleset variables for ruleset parameters" and Section 4.3, "Binding the variables to ruleset parameters" in Exercise 2, "Setting up decision services".

- 1. Expand **loan-bom > bom > model > training\_loan** to view BOM again.
- 2. Expand the `Report` class to see its members.

Notice the `Report.approved` member. This member can be used to record the decision that the rule engine returns.



### Questions

Which members of the `Report` class might be useful for recording output results from the rule engine?

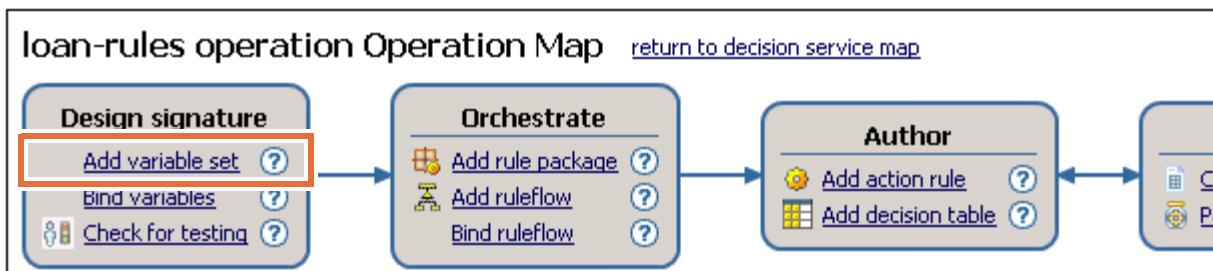
### Answer

This list includes some examples:

- `messages`
- `insuranceRequired`
- `insuranceRate`
- `validData`

Because objects of type `Report` are designed to carry output information, you can create an output ruleset parameter that is based on this object type.

- \_\_\_ 3. Add a variable set called `my parameters` to the loan-rules decision service.
- \_\_\_ a. In the “Design signature” part of the Operation Map, click **Add variable set**.



- \_\_\_ b. In the New Variable Set wizard, in the **Name** field, type: `my parameters`
- \_\_\_ c. Click **Finish**.

The Variable Set editor opens.

- \_\_\_ 4. Create a variable called `report`.
- \_\_\_ a. In the Variable Set editor, click **Add**.
- \_\_\_ b. Overwrite the default values in the **Name**, **Type**, and **Verbalization** columns by entering the following information:
- **Name:** report
  - **Type:** training\_loan.Report



### Reminder

To select the `training_loan.Report` type:

- In the **Type** column, click `java.lang.String`.
- Click the ellipses (...) to open the Types window.
- In the **Choose a type** field, enter: `report`
- In the **Matching types** field, select **Report**.
- Click **OK**.

- **Verbalization:** the loan report

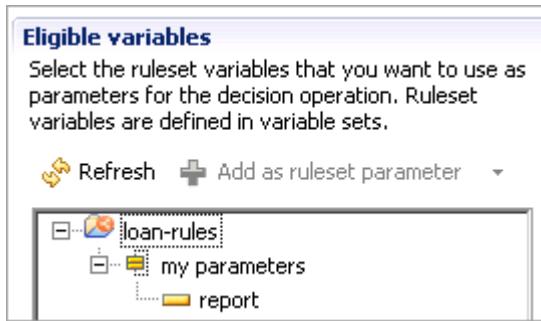
- \_\_\_ c. Save your work and close the **my parameters** tab.

- \_\_\_ 5. Bind the report variable to an output ruleset parameter.

- \_\_\_ a. In the “Design signature” part of the Operation Map, click **Bind variables**.

The Decision Operation Signature editor opens.

- \_\_\_ b. In the **Eligible variables** section, expand **my parameters**.



- \_\_\_ c. Select **report** and drag it to the Output Parameters table.

#### Output Parameters

Define the parameters that are initialized and returned by the execution.

| Parameter name | Verbalization   | Type                 |
|----------------|-----------------|----------------------|
| report         | the loan report | training_loan.Report |

- \_\_\_ d. Save your work and close the **Decision Operation Signature** tab.

- \_\_\_ 6. Rewrite the `grade` rule to use the `report` ruleset parameter.

You can use your existing local rule variable to manipulate the `report` ruleset parameter (which shows up in the vocabulary list as “the loan report”).

You can also delete the local rule variable and use the ruleset parameter directly.

```

Content

1 definitions
2 set 'the report' to a report;
3 if
4 the amount of the loa
5 then
6 in 'the report', ref

```

- \_\_\_ a. In Rule Explorer, expand **loan-rules > rules > loan** and double-click the `grade` rule.

- \_\_\_ b. Delete the **definitions** part of the rule, which includes these lines:

```

definitions
 set 'the report' to a report

```

You should now see errors in the rule.

```

Content

1
2 if
3 the amount of the loan of 'the report' is more than 10000000
4 then
5 in 'the report', refuse the loan with the message "The loan amount is

```

- \_\_\_ c. In the condition and action statements, double-click ‘the report’ and double-click ‘the loan report’ to select the new ruleset parameter.

```
Content
1
2 if
3 the amount of the loan of 'the report' is more than 10000000
4 then
5 in 'the report', refuse t...
```

The errors should now be gone.

- \_\_\_ 7. Save your work (Ctrl+Shift+S).  
\_\_\_ 8. When you are finished editing rules, close the rule editors.  
\_\_\_ 9. In the Problems view, verify that no messages are listed.

## Section 4. Creating an action rule template

The `loan-rules` decision service contains a series of action rules that share a similar structure. These action rules all check some data, accept or reject these data based on certain conditions, and generate a message that explains the rejection reason.

An efficient way to author all the similar rules of this project is to create an action rule template, and apply it to the creation of the rules.

In this section, you create the required business template.

### 4.1. Creating the template

- 1. In the `loan-rules` decision service, create an action rule template that is called: `checkData`
  - a. Right-click the `loan-rules` main rule project, and click **New > Action Rule Template**. The New Action Rule Template wizard opens.
  - b. Keep the default value in the **Template folder** field:  
`/loan-rules/templates`
  - c. Leave the **Folder** field empty.
  - d. Enter `checkData` in the **Name** field of your action rule template.
  - e. In the **Type** field, leave the default value as **ActionRule**.
  - f. Click **Finish**.

The Guided editor opens for you to edit the `checkData` action rule template, which is visible now in the `loan-rules/templates` folder.

- 2. With the Guided editor, edit the `checkData` action rule template so that the action statement matches this content:

```
if
<select a condition>
then
 in the loan report, reject the data with the message <enter a string>
```

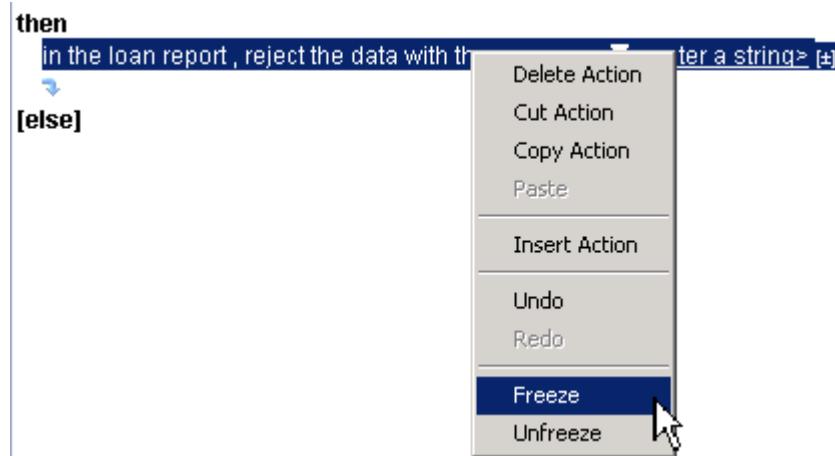


#### Important

Do not change the condition statement. This template is for the action statement only.

- 3. Follow these steps to freeze the actions **except** the `[±]` sign after `<enter a string>`, which you must leave unfrozen.
  - a. Select the rule action statement with your mouse.

- \_\_ b. Right-click the selected statement and click **Freeze** to freeze all actions.



- \_\_ c. Right-click the plus-minus sign (**[+]**) at the end of the actions and click **Unfreeze**.

If this plus-minus sign (**[+]**) is frozen, the message can contain only one `String` element. However, in many cases you must define a message by using multiple `String` elements.

You see an example with the `check SSN` action rule (in Section 5, "Authoring action rules from your action rule template," on page 7-17). You must unfreeze that sign before you proceed.

- \_\_ 4. Save the `checkData` action rule template.



#### Note

You can also freeze everything except the placeholders by right-clicking anywhere in the Guided editor and clicking **Freeze All**. Placeholders can still be editable (they are shown in blue).

To unfreeze everything, right-click anywhere in the Guided editor and click **Unfreeze All**.

- \_\_ 5. Close the template.

## Section 5. Authoring action rules from your action rule template

In this section, you see how the rule authors work with the templates you provide. This section illustrates how you can support rule authors during project development.

### 5.1. Publishing the BOM and the rules to Decision Center

- \_\_\_ 1. Make sure that the sample server is started.
- \_\_\_ 2. Publish `loan-rules` as an ungoverned decision service.
  - \_\_\_ a. In the Rule Explorer, right-click the `loan-rules` main rule project and click **Decision Center > Connect**.  
The Decision Center configuration wizard opens.
  - \_\_\_ b. Enter the connection entries:
    - **URL:** `http://localhost:9080/teamserver`
    - **User name:** `rtsAdmin`
    - **Password:** `rtsAdmin`
  - \_\_\_ c. Click **Connect**.
  - \_\_\_ d. When the connection is established, click **Next**.
  - \_\_\_ e. In the Synchronization Settings window, click **Next** (do not select the **Use Decision Governance** check box).
  - \_\_\_ f. In the Decision Service Dependent Projects page, make sure that **loan-bom** is selected, and click **Finish**.
- \_\_\_ 3. Click **No** when you are prompted to switch to the Team Synchronizing perspective.  
You do not have to open the Team Synchronizing perspective because it is empty.
- \_\_\_ 4. When synchronization completes, no changes are found, so you can click **OK** to close the window.

### 5.2. Authoring the rule in Decision Center

- \_\_\_ 1. Open the Decision Center Enterprise console at this URL:  
`http://localhost:9080/teamserver`  
If the console was already open, it might be necessary to refresh your browser and sign in again to see the newly published `loan-rules` and `loan-bom` projects.
- \_\_\_ 2. Sign in with `rtsAdmin` for the user name and password.
- \_\_\_ 3. On the **Home** tab, select **Work on a decision service** and choose **loan-rules** as the decision service to use.
- \_\_\_ 4. Click the **Compose** tab.

5. On the **Compose** tab, in the **Start from a template** section, you see your newly created template.

The screenshot shows the IBM ODM interface with the 'Compose' tab selected. On the left, a sidebar lists various types: Action Rule, Decision Table, Decision Tree, Folder, Smart Folder, Template, Variable Set, Function, Technical Rule, and Resource. A section titled 'Start from a template' contains a list of templates, with 'checkData' highlighted and enclosed in a red box. The main panel displays a dialog titled 'You are about to create an element base'. The 'Rule Type' section shows 'Content'. The 'if' condition is defined as '`if <condition>`' and the 'then' action is 'in 'the loan report'', followed by a message placeholder. The 'Initial Values' table lists: Name checkData, Active True, Locale en\_US, and Status New. The 'Documentation' section contains an 'OK' button.

6. Select the `checkData` template and click **OK**.
7. On the Properties page, define the rule properties.
- Change the name to: `check name`
  - Set the **Folder** to: `/loan`
  - Click **Next**.

The template opens in the rule editor on the Content page. Notice that this editor is like the Guided editor in Rule Designer and does not use special punctuation to distinguish parts of the rule.

8. On the Content page, edit the `check name` rule to state:

```
if
 the last name of the borrower of the loan report is empty
then
 in the loan report, reject the data with the message The borrower's name
 is missing
```

**Hint**

To change the condition phrase from `is <a string>` to `is empty`, click **is**.

**[definitions]****if**

the last name of the borrower of the report [±] **is** ▾ `<a string>` [±] **X**



- \_\_\_ 9. Notice that you cannot change the beginning of the action parts:  
in the loan report, reject the data with the message
- \_\_\_ 10. Click **Finish**.  
The completed rule opens in the Details view.
- \_\_\_ 11. Click the **Explore** tab and note that the new `check name` rule is now listed in the loan rule package.

### 5.3. Deleting projects

- \_\_\_ 1. While you are still in the `loan-rules` decision service, click the **Configure** tab.
- \_\_\_ 2. Go to the **Administration** section.
- \_\_\_ 3. Click **Erase Current Decision Service** and click **Yes** when prompted to confirm deletion.
- \_\_\_ 4. Go back to the **Home** tab and make sure that **Work on a decision service** is selected.
- \_\_\_ 5. Click the **Decision service in use** menu and confirm that `loan-rules` is no longer in the list of available decision services.
- \_\_\_ 6. Close the Enterprise console.

### End of exercise

## Exercise review and wrap-up

This exercise looked at how to author action rules in the Intellirule editor and in the Guided editor, by using ruleset parameters, rule variables, or automatic variables. You also saw how to create and work with rule templates to support the rule authors in Decision Center.

(Optional) To see the solution to this exercise, switch to a new workspace and import the project **Ex 07: Authoring rules > 02-answer**.

# Exercise 8. Authoring decision tables and decision trees

## What this exercise is about

In this exercise, you learn how to author decision tables and decision trees.

## What you should be able to do

After completing this exercise, you should be able to:

- Use the decision table editor to create a decision table
- Use the decision tree editor to create a decision tree

## Introduction

In this exercise, you work with patterns of rules to implement them as decision tables or decision trees.

This exercise includes these tasks:

- Section 1, "Authoring a decision table"
- Section 2, "Authoring a decision tree"

## Requirements

You should complete these exercises before proceeding:

- Exercise 7, "Authoring action rules"

This exercise uses the following files, which are installed in the *<InstallDir>\studio\training* directory:

- Start project: Dev 08 – Authoring decision tables and trees\01-start
- Solution project: Dev 08 – Authoring decision tables and trees\02-answer
  - You can use the solution project in "Exercise review and wrap-up" on page 8-24.

## Section 1. Authoring a decision table

In this part of the exercise, you author the following decision table, which was given as an example during the lectures.

The screenshot shows the IBM Rational Rules Designer interface with a decision table titled "rate". The table has two columns: "Loan duration (years)" and "Yearly rate (%)".

|   | Loan duration (years) | Yearly rate (%) |     |
|---|-----------------------|-----------------|-----|
|   | min                   | max             |     |
| 1 | < 5                   |                 | 5   |
| 2 | 5                     | 8               | 5.8 |
| 3 | [9                    | 12[             | 6.7 |
| 4 | 12                    | 16              | 7.4 |
| 5 | ≥ 17                  |                 | 7.9 |

The table includes a header row with "min" and "max" columns. Rule 1 covers durations less than 5 years at a rate of 5%. Rule 2 covers the range between 5 and 8 years at 5.8%. Rule 3 covers the range from 9 to 12 years at 6.7%. Rule 4 covers the range from 12 to 16 years at 7.4%. Rule 5 covers durations of 17 years or more at 7.9%.

This decision table defines the following five rules:

- If the duration of the loan is less than five years then set the yearly interest rate of the loan to 5.0
- If the duration of the loan is between five years and eight years then set the yearly interest rate of the loan to 5.8
- If the duration of the loan is at least nine years and less than 12 years then set the yearly interest rate of the loan to 6.7
- If the duration of the loan is between 12 years and 16 years then set the yearly interest rate of the loan to 7.4
- If the duration of the loan is at least 17 years then set the yearly interest rate of the loan to 7.9

By creating this table, you also learn how the operators that you can use in the decision table editor (<, [ . . ], and others) correspond to BAL operators (is less than, is between, and others).

### 1.1. Setting up your environment for this exercise

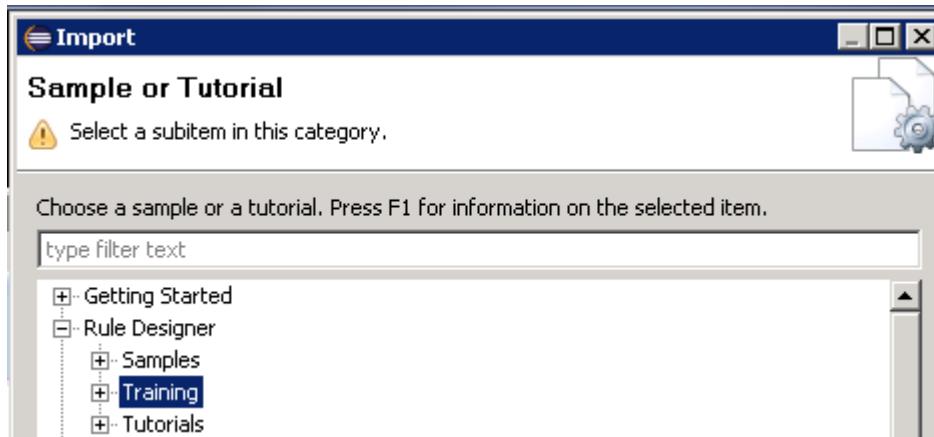
- \_\_\_ 1. In Rule Designer, switch to a new workspace:
  - \_\_\_ a. From the **File** menu, click **Switch Workspace > Other**.
  - \_\_\_ b. In the Workspace Launcher window, enter the path:  
`<LabfilesDir>\workspaces\dtabc_dtrees`
- \_\_\_ 2. Close the Welcome view.

- \_\_\_ 3. Use the **Import** menu to import the exercise start project.
  - \_\_\_ a. In Eclipse, click **File > Import** or right-click the Rule Explorer and click **Import**.
  - \_\_\_ b. In the Select window, expand **Operational Decision Manager**, select **Samples and Tutorials**, and click **Next**.
  - \_\_\_ c. On the “Sample or Tutorial” page, expand the **Rule Designer > Training** node.



### Hint

You can collapse the Rule Designer node and then expand it to simplify the list.

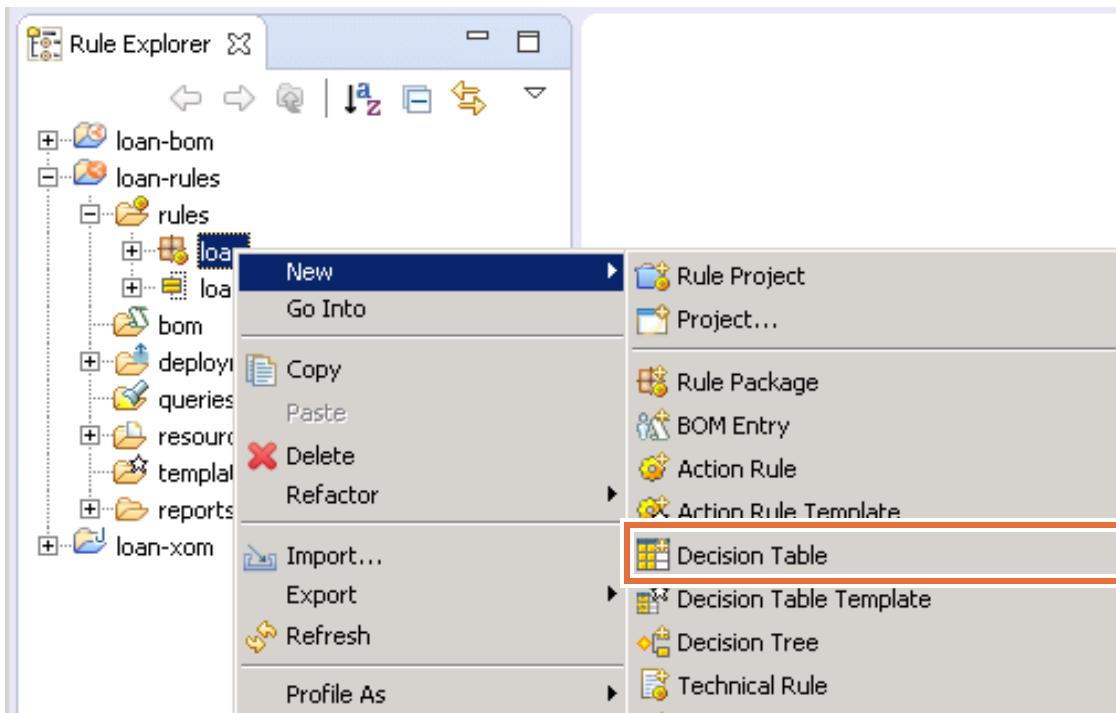


- \_\_\_ d. Expand the **Ex 08: Authoring decision tables and trees** node, and select **01-start**.
- \_\_\_ e. Make sure that the **Import shared projects** check box is selected, and click **Finish**.
- \_\_\_ 4. When the workspace finishes building, close the Help view.

## 1.2. Creating the table

- \_\_\_ 1. In the `loan` rule package, create a decision table named: `rate`
  - \_\_\_ a. Expand the `loan-rules > rules` project.

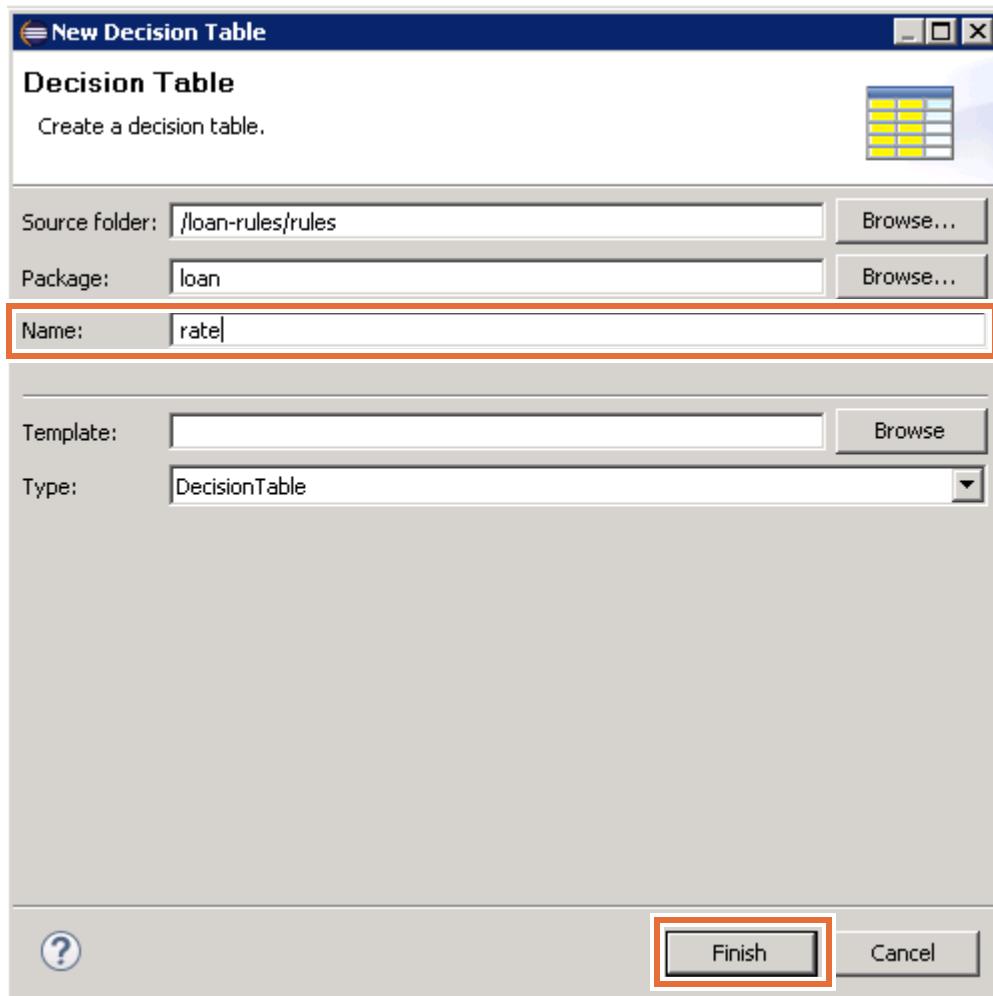
- \_\_ b. Right-click the `loan` rule package, and click **New > Decision Table**.



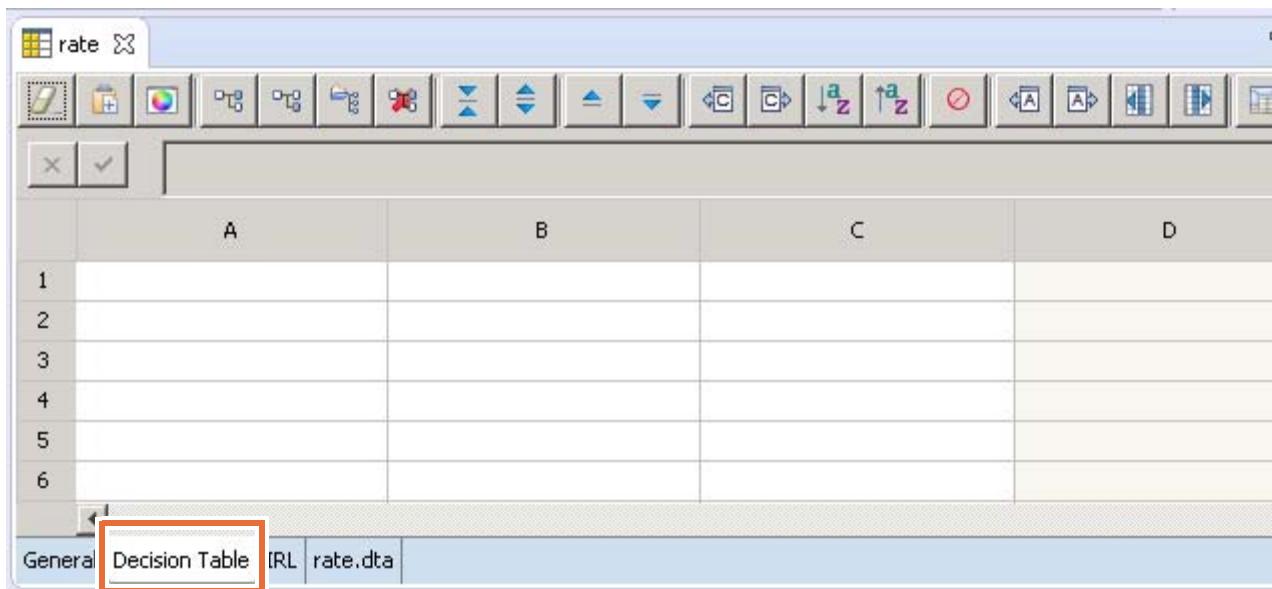
The New Decision Table wizard opens.

- \_\_ c. In the **Name** field, type: `rate`

- \_\_\_ d. Click **Finish**.



The new decision table opens in the decision table editor, with three condition columns and one action column.

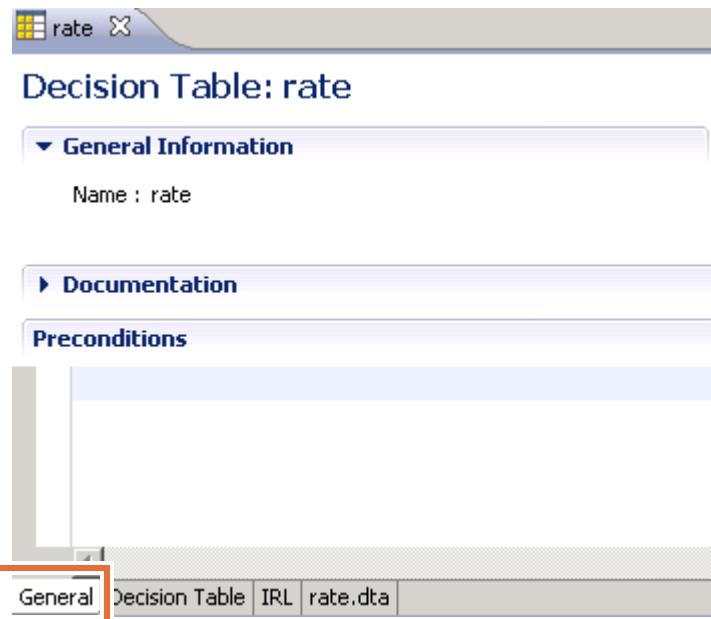


The editor tab is called **Decision Table**.

- 2. Open the **General** tab of the decision table editor, and enter the following definitions in the **Preconditions** section:

definitions

```
set 'the loan' to the loan of 'the loan report';
```



Later, when you define the condition tests and action statements, you use **the loan** variable instead of the longer term: **the loan of 'the loan report'**

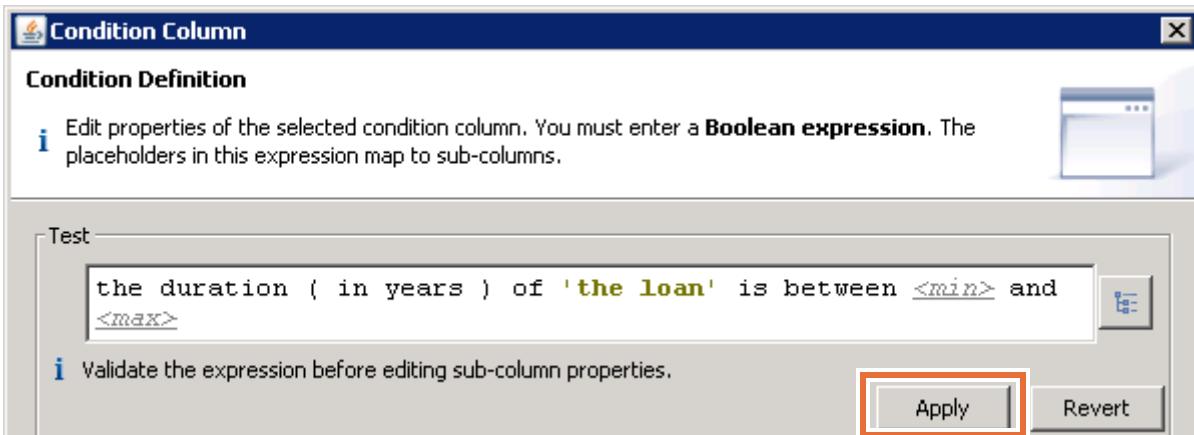


### Hint

You can press **Ctrl+Space** to open Content Assist to build your precondition, or you can type directly in the editor. If you type directly, you can use **Ctrl+Shift+F** to format the content.

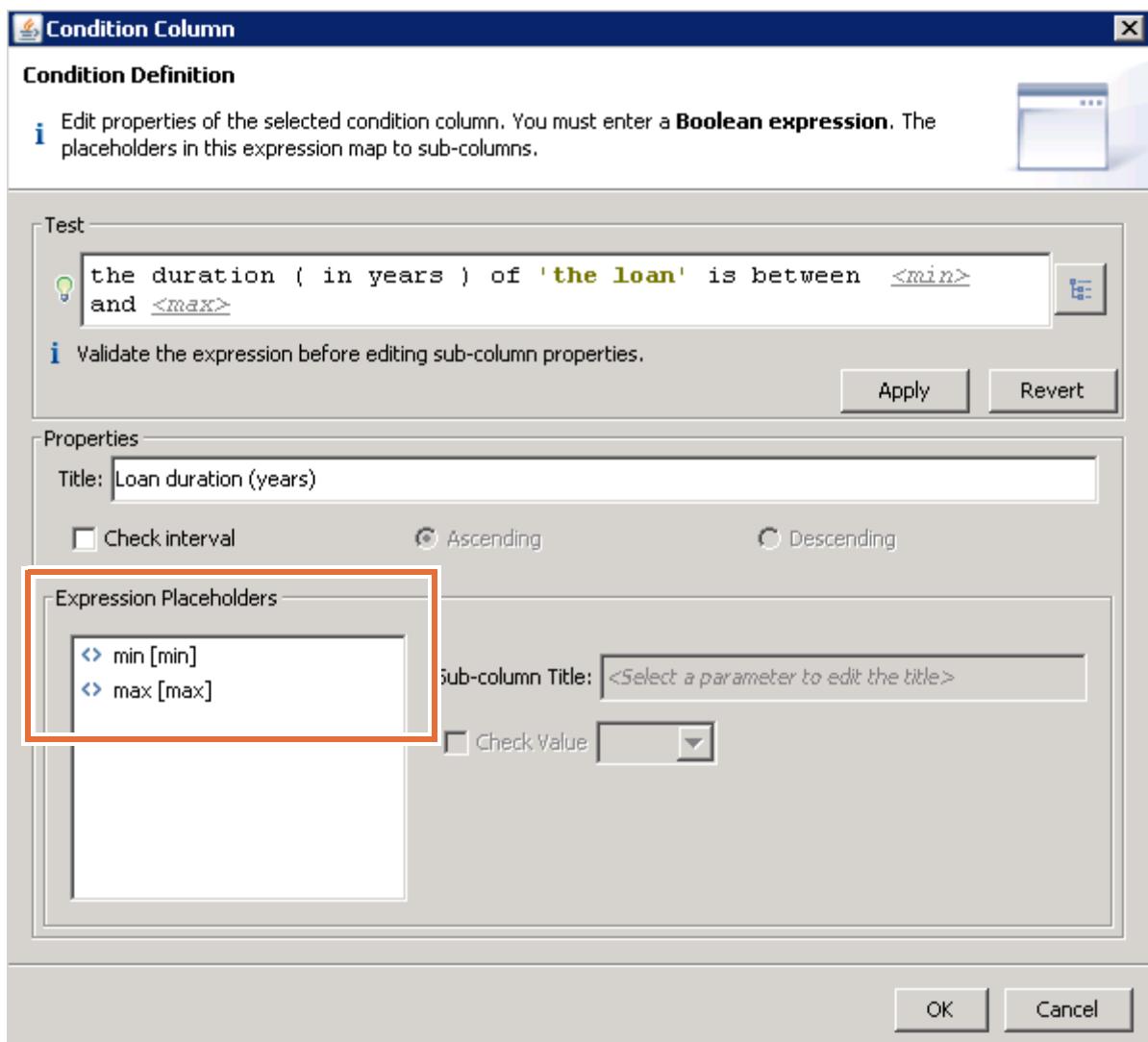
- 3. Save your work.
- 4. Click the **Decision Table** tab to return to the table editor and define the first condition column.
  - a. Double-click the header of the first condition column, which is labeled **A**, to open the Condition Column window.
  - b. In the **Test** section, click **<condition>** and select the appropriate vocabulary from the vocabulary list to build this condition:  
the duration (in years) of 'the loan' is between **<min>** and **<max>**

- \_\_\_ c. When you are finished editing the test, click **Apply**.



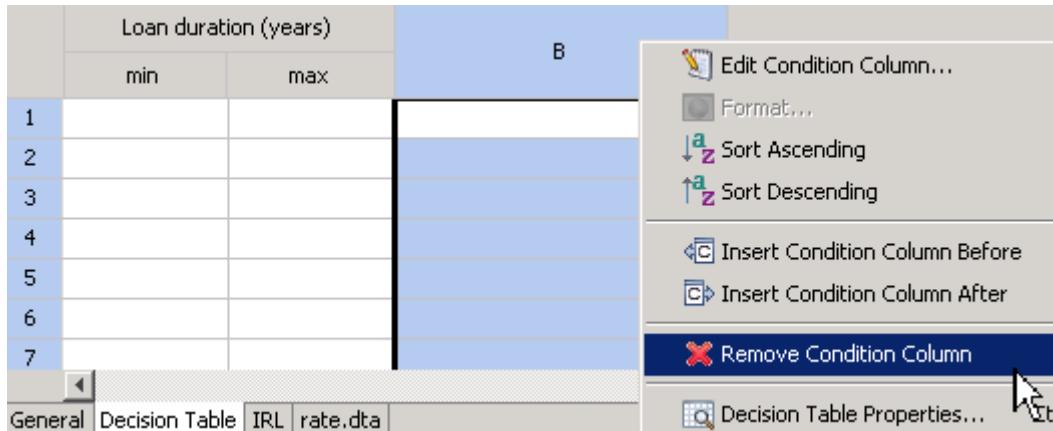
- \_\_\_ d. In the **Title** field, replace A with: Loan duration (years)

Notice that the placeholders in the condition test are now listed in the **Expression Placeholders** list.



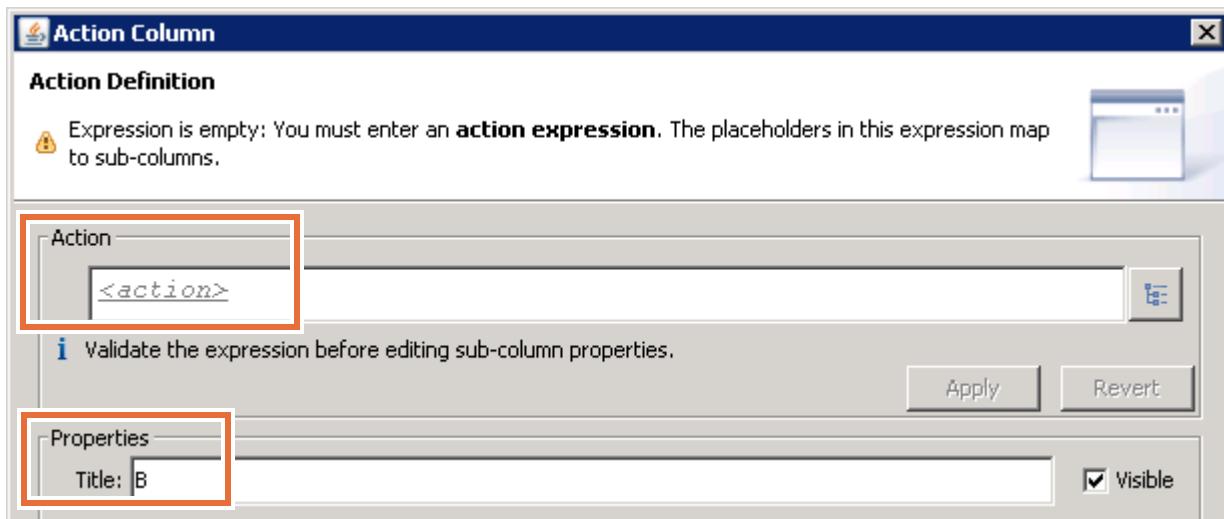
- \_\_\_ e. Click **OK**.

- \_\_\_ 5. Right-click each of the next two condition columns and remove them.



- \_\_\_ 6. Define the action column.

- \_\_\_ a. Double-click the action column header, which is now labeled **B**, to open the Action Column window.



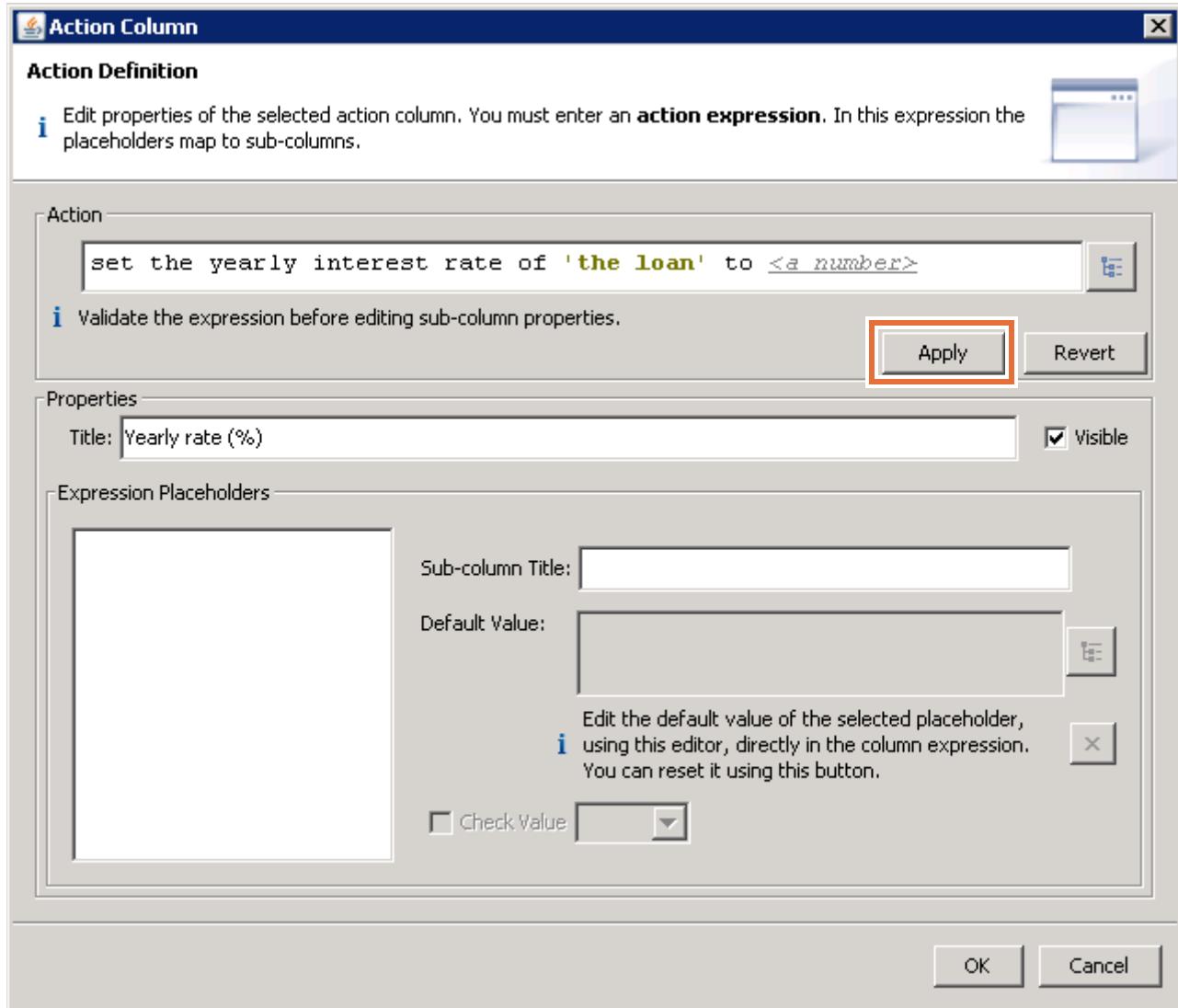
- \_\_\_ b. In the **Action** field, click **<action>**, and select the appropriate vocabulary from the vocabulary list to write this action statement:

set the yearly interest rate of 'the loan' to <a number>

- \_\_\_ c. In the **Action** section, click **Apply**.

- \_\_\_ d. In the **Title** field, replace **B** with: Yearly rate (%)

- \_\_\_ e. When you are finished editing the action, click **Apply**.



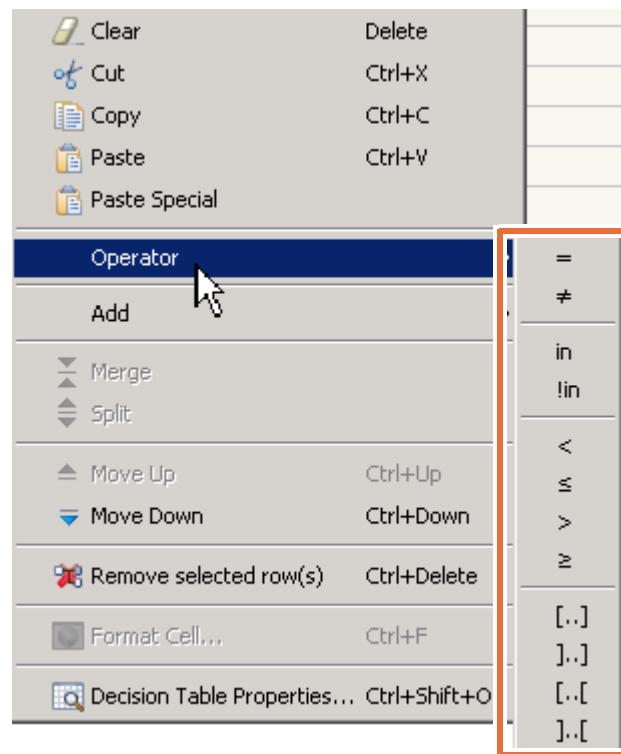
- \_\_\_ f. Click **OK** to close the window.

### 1.3. Completing the table cells with values

The cells in the `rate` decision table are currently empty. To complete the `rate` decision table, you work with operators in the next step.

**Hint**

To show the list of available operators in the decision table editor, right-click a decision table cell and click **Operator**. To enter an operator in a cell of the decision table, you can select the required operator from this list.

**Questions**

Can you relate the operators in the decision table editor (such as  $>$ ,  $<$ ) to BAL operators?

**Answer**

The operators in the decision table editor and the corresponding BAL constructs are as follows:

| Decision table editor operators | Corresponding BAL number operators |
|---------------------------------|------------------------------------|
| =                               | equals                             |
| !=                              | does not equal                     |
| in                              | is one of { ... }                  |
| !in                             | is not one of { ... }              |
| <                               | is less than                       |
| <=                              | is at most                         |
| >                               | is more than                       |
| >=                              | is at least                        |
| [...]                           | is between ... and ...             |
| [...[                           | is at least ... and less than ...  |
| ]...]                           | is more than ... and at most ...   |
| ]...[                           | is strictly between ... and ...    |

You are now ready to complete the `rate` decision table.

- 1. Review the five rules that you must write and the corresponding lines in the decision table, recalled at the beginning of this exercise. This table lists the values and operators that you use to complete the table.
- 2. In the decision table editor, first enter the following values in cells of the `rate` decision table:

| Loan duration (years) | Yearly rate (%) |
|-----------------------|-----------------|
| < 5                   | 5               |
| >= 5 AND <= 8         | 5.8             |
| >= 9 AND < 12         | 6.7             |
| >= 12 AND <= 16       | 7.4             |
| >=17                  | 7.9             |

For rows that have only a single value, enter that value in the first column only.

- 3. Next, you edit the operators that are used for each row in the condition.
- a. Right-click the first row of the condition column, and in the menu, click **Operator**.

- \_\_ b. From the **Operator** list, select the “less than” (<) operator.

|   | Loan duration (years) |     | Yearly rate (%) |
|---|-----------------------|-----|-----------------|
|   | min                   | max |                 |
| 1 | < 5                   |     |                 |
| 2 | 5                     | 8   |                 |
| 3 | 9                     | 12  |                 |
| 4 | 12                    | 16  |                 |
| 5 | 17                    |     |                 |

The subcolumns merge automatically to match the operator. Notice that the condition test changes from **is between** to **is less than**.

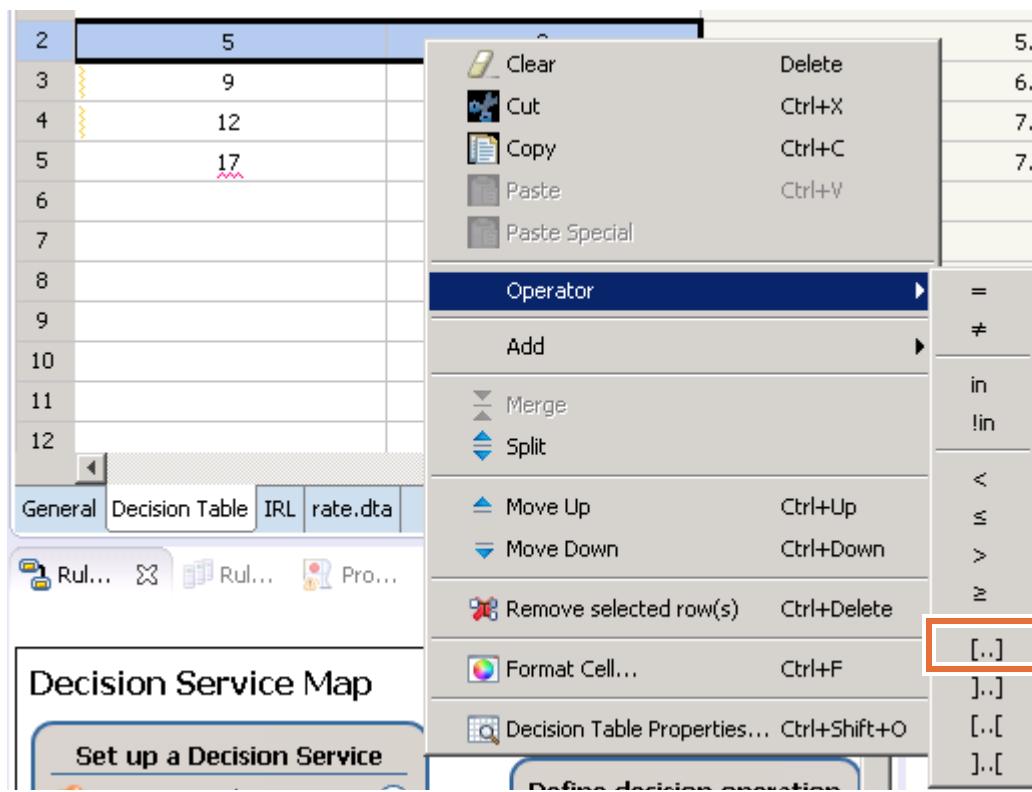
- \_\_ c. Click the second row of the condition column, and consider the condition test (**is between**), which is equivalent to the BAL operator: [...]

|   | Loan duration (years) |     | Yearly rate (%) |
|---|-----------------------|-----|-----------------|
|   | min                   | max |                 |
| 1 | < 5                   |     | 5               |
| 2 | 5                     | 8   | 5.8             |
| 3 | 9                     | 12  | 6.7             |

You want the row to state:  $\geq 5 \text{ AND } \leq 8$

In this case, these operators are equivalent to **is between** or [...].

- \_\_\_ d. Right-click the second row and click [...] from the Operator menu.



Notice that the cells do not change.

- \_\_\_ e. Right-click the condition column for row 3, and set the condition to **is at least and less than**, which is represented in BAL as: [...]
- \_\_\_ f. Complete the remaining rows of the condition column.
4. Delete the empty rows of the decision tables.
- \_\_\_ a. Select all of the empty rows.
- \_\_\_ b. Click **Remove selected row(s)** on the toolbar.

Your table should have five rows, and have the following values.

| Loan duration (years) |     | Yearly rate (%) |
|-----------------------|-----|-----------------|
| Min                   | Max |                 |
| < 5                   |     | 5               |
| 5                     | 8   | 5.8             |
| [ 9                   | 12[ | 6.7             |
| 12                    | 16  | 7.4             |
| >=17                  |     | 7.9             |

|   |           | the duration (in years) of ' <b>the loan</b> ' is between <a number> and <a number> |                 |  |
|---|-----------|-------------------------------------------------------------------------------------|-----------------|--|
|   |           | Loan duration (years)                                                               | Yearly rate (%) |  |
|   | min       | max                                                                                 |                 |  |
| 1 | < 5       |                                                                                     | 5               |  |
| 2 | 5         | 8                                                                                   | 5.8             |  |
| 3 | [9        | 12[                                                                                 | 6.7             |  |
| 4 | 12        | 16                                                                                  | 7.4             |  |
| 5 | $\geq 17$ |                                                                                     | 7.9             |  |



## Information

In the decision table editor, when you select a row of a decision table, you can see the corresponding rule at the top of the decision table. You can use this useful function to verify what you are writing.

- \_\_\_ 5. Verify the rule statement for each row in the decision table.

\_\_\_ a. Click 1 in row 1, and hover your mouse to see the full rule statement, including the precondition.

**definitions**  
set 'the loan' to the loan of 'the loan report';  
**if**  
  **all of the following conditions are true :**  
    - ( the duration (in years) of 'the loan' is less than 5 ),  
**then**  
  set the yearly interest rate of 'the loan' to 5 ;

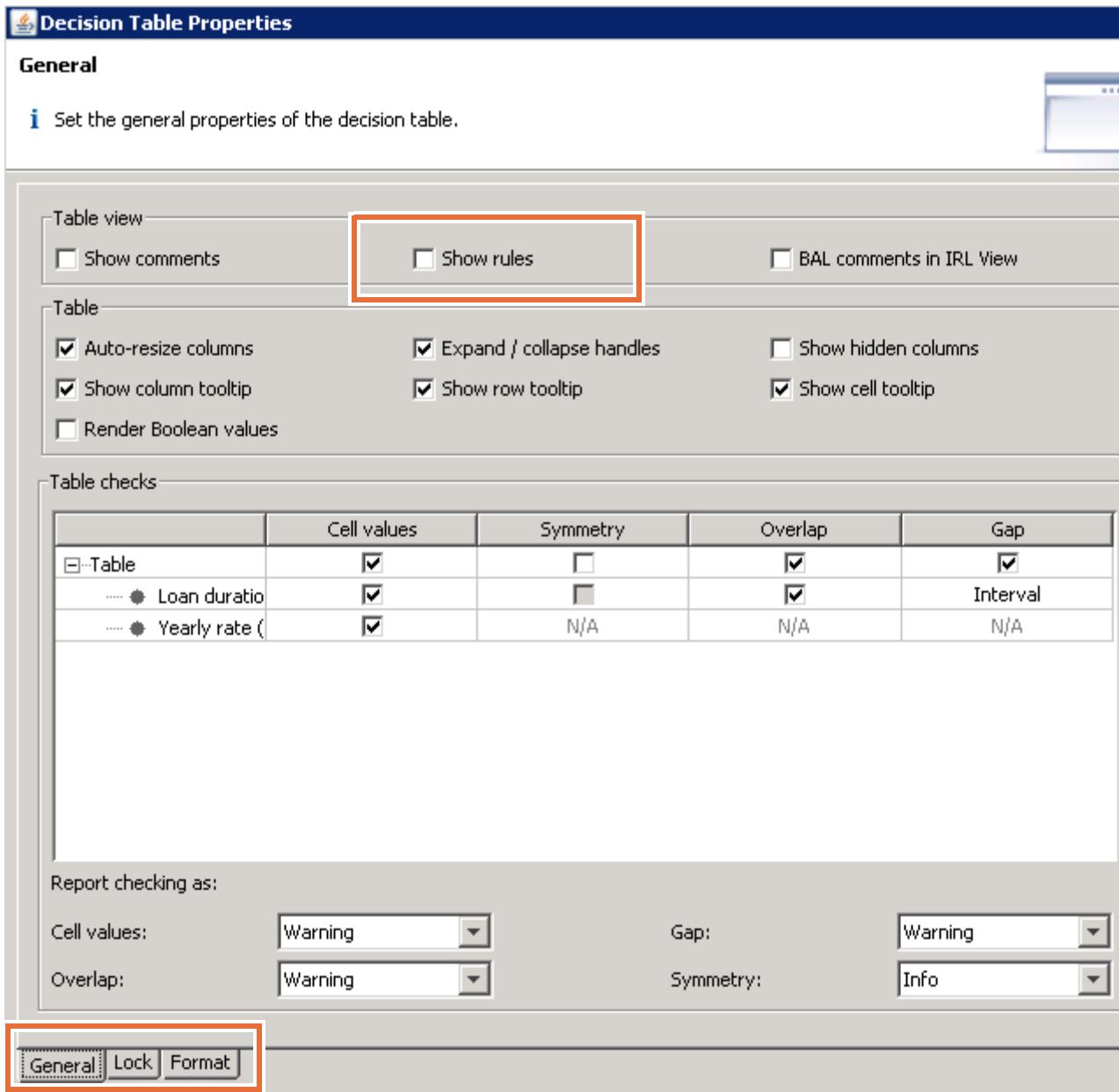
- \_\_ b. Click the remaining rows to verify that the rule conditions are correct for each row (or rule).
  - \_\_ c. Save your work.

\_\_ 6. Take some time to examine the Decision Table Properties window.

  - \_\_ a. Edit the decision table properties by clicking  **Open Decision Table Properties Editor** on the toolbar.



The Decision Table Properties window opens.



With this window, you can specify some general or lock properties of the decision table.

- \_\_\_ b. Open each of the **General**, **Lock**, and **Format** tabs to see the available options.
- \_\_\_ c. On the **General** tab, select **Show rules** and click **OK**.

- \_\_ d. Click 1 in row 1 again to see the rule in the rule pane that is now open In the decision table editor.

|   | < 5 | 5   | ≥ 17 |
|---|-----|-----|------|
| 2 | 5   | 8   |      |
| 3 | [9] | 12[ |      |
| 4 | 12  | 16  |      |
| 5 |     |     | ≥ 17 |

```

definitions
set 'the loan' to the loan of 'the loan report';
if
 all of the following conditions are true :
 - (the duration (in years) of 'the loan' is less than 5) ,
then
 set the yearly interest rate of 'the loan' to 5 ;

```

General Decision Table IRL rate.dta



### Questions

Can you determine how you can lock the preconditions of the decision table to prevent changes?

### Answer

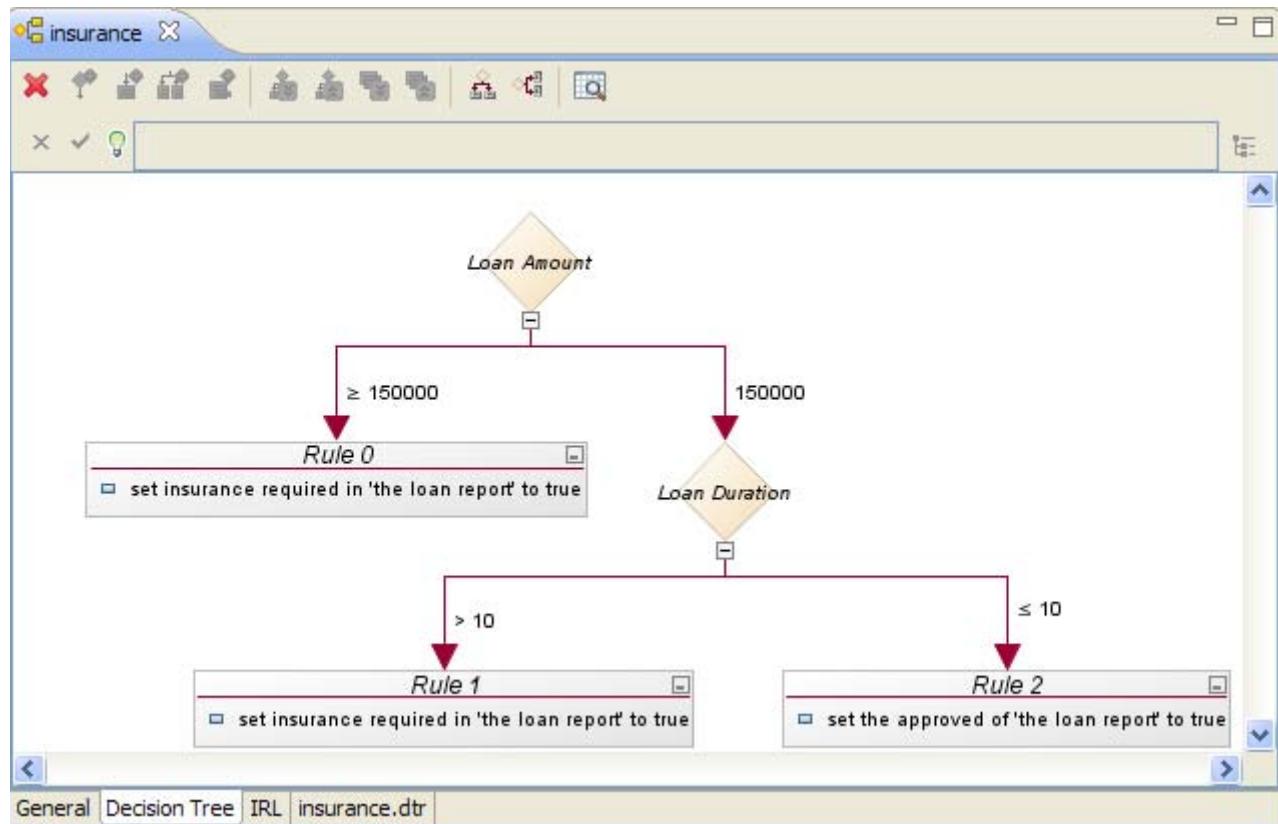
To lock the preconditions of the decision table, you must:

- Select the **Lock Preconditions** check box on the **Lock** tab of the Decision Table Properties window.
- Then, select the **Enforce locking rules on this table** check box on the **Lock** tab for locking to take effect.

- \_\_ 7. Save your work.  
\_\_ 8. Close the decision table editor.

## Section 2. Authoring a decision tree

In this part of the exercise, you author a decision tree that is called `insurance` in the `loan` rule package of the `loan-rules` project.



### Requirements

The decision tree implements rules to determine whether insurance is required to approve the loan. These rules are based on the amount and the duration of the loan, as follows:

- If the amount of the loan  $\geq 150000$ , then an insurance is required.
- If the amount of the loan  $< 150000$  then:
  - If the duration of the loan  $> 10$ , then an insurance is required.
  - If the duration of the loan  $\leq 10$ , then the loan is approved.

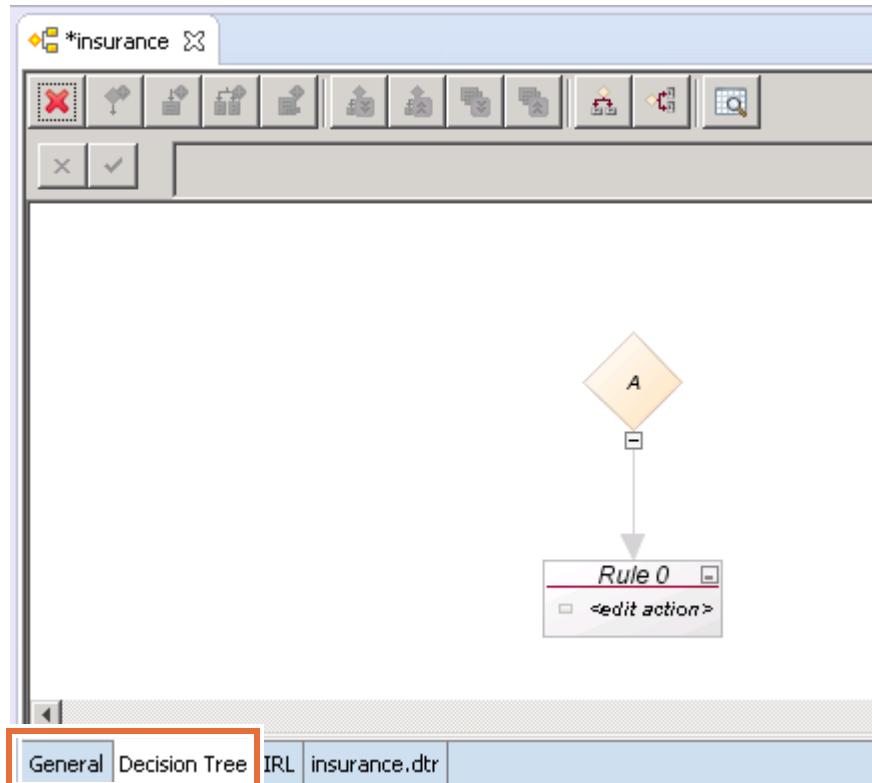
### 2.1. Creating the tree

- \_\_ 1. In the `loan` rule package of the `loan-rules` project, create a decision tree called: `insurance`
  - \_\_ a. Expand the `loan-rules` project.
  - \_\_ b. Right-click the `loan` rule package, and click **New > Decision Tree**.

The New Decision Tree wizard opens.

- \_\_\_ c. In the **Name** field, enter: insurance
- \_\_\_ d. Click **Finish**.

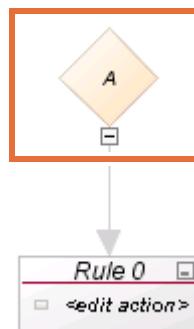
The new decision tree opens with one condition node, one branch, and one action. The tree diagram is on the **Decision Tree** tab.



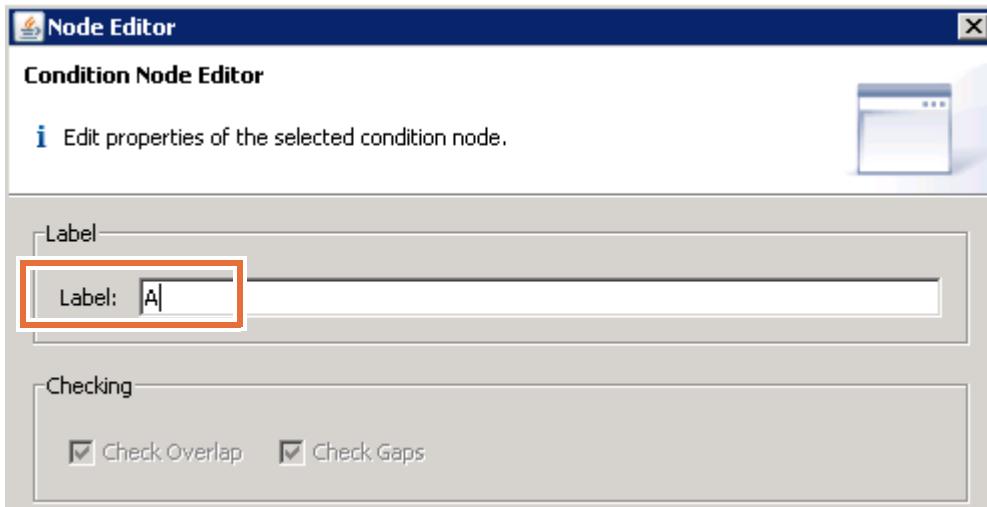
- \_\_\_ 2. On the **General** tab of the decision tree editor, enter the following definitions in the **Preconditions** section:

```
definitions
 set 'the loan' to the loan of 'the loan report';
```

- \_\_\_ 3. Save your work (Ctrl+S).
- \_\_\_ 4. Click the **Decision Tree** tab to return to the decision tree editor and define the condition.
- \_\_\_ a. Double-click the condition node.

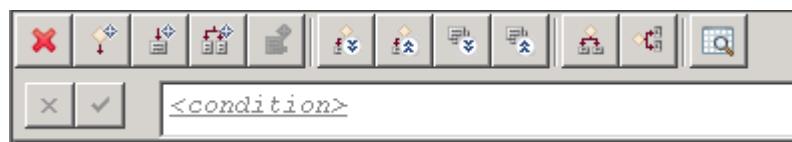


The Node editor opens.



- \_\_\_ b. In the **Label** field, replace **A** with **Loan Amount** and click **OK**.
- \_\_\_ c. In the edit bar, click **<condition>**, and select the appropriate vocabulary from the vocabulary list to write this condition:

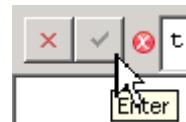
the amount of 'the loan' is less than <a number>



### Hint

You can also type directly or press Ctrl+Spacebar to open the Content Assist box when constructing your condition statement.

- \_\_\_ d. When you are finished editing the condition, click the check mark to save your edits.



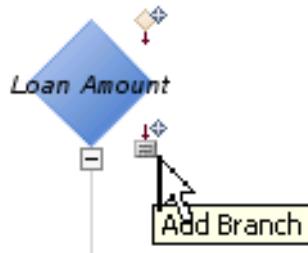
- \_\_\_ e. Click the **Loan Amount** condition node (if it is not already selected) and notice the **Insert a Condition Node** and **Add Branch** icons beside it, which you can use to insert new conditions or rules.



These icons are also available on the toolbar.

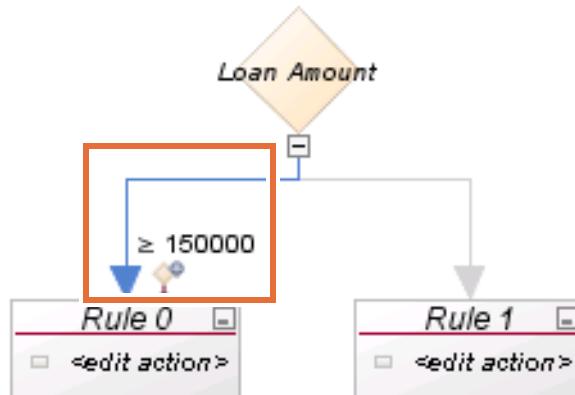


- \_\_\_ 6. Click **Add Branch** to create a branch.



- \_\_\_ 7. Define the condition for Rule 0.

- \_\_\_ a. Click the branch (on the line) to **Rule 0**.
- \_\_\_ b. In the edit bar, modify the condition to state:  
the amount of 'the loan' is at least 150000  
Click the **check mark** to save your change.



- \_\_\_ 8. Click the branch to **Rule 1**, and in the edit bar, modify the condition to state:

the amount of 'the loan' is less than 150000

**Important**

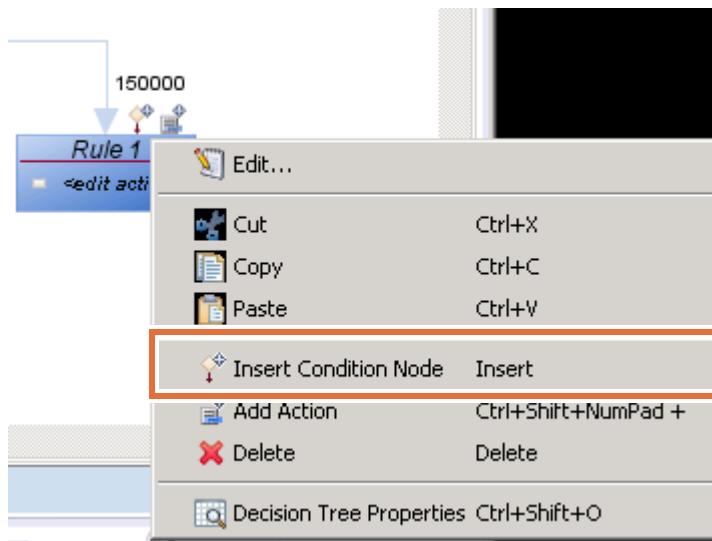
Remember to click the **check mark** to save your edit.

The branches are now defined and visible in dark red.

- \_\_\_ 9. In **Rule 0**, click **<edit action>**, and in the edit bar, modify the action to state:

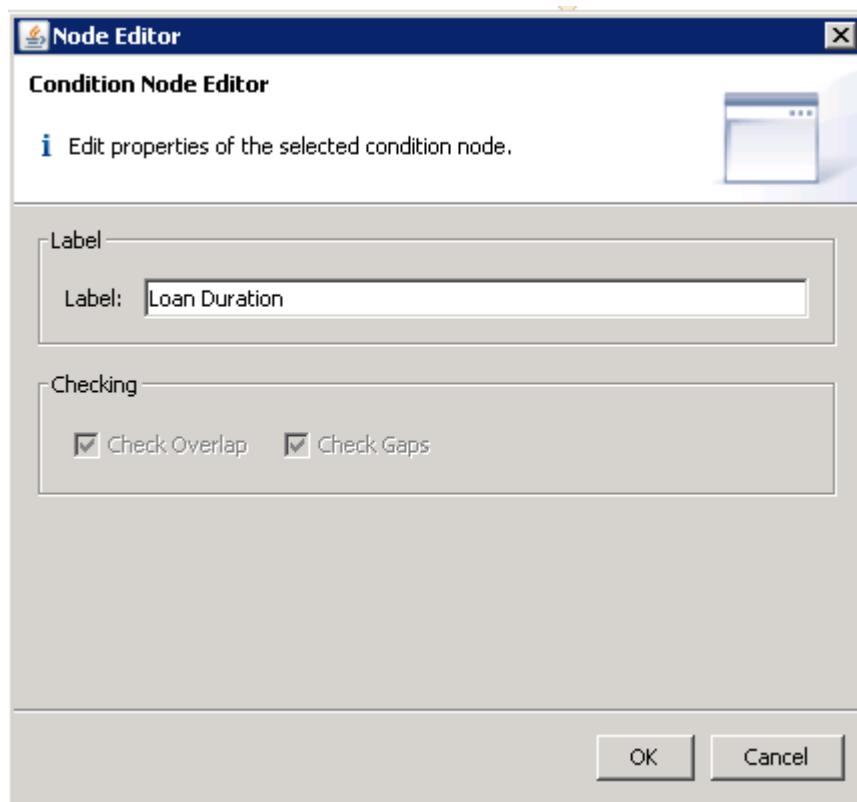
set insurance required in 'the loan report' to true

- \_\_\_ 10. Right-click **Rule 1**, and click **Insert Condition Node**.



- \_\_\_ 11. Rename the new condition node to: **Loan Duration**

- \_\_\_ a. Right-click the new condition node and click **Edit** to open the Node editor.
- \_\_\_ b. In the Node editor, in the **Label** field, enter: **Loan Duration**
- \_\_\_ c. Click **OK** to close the Node editor.



- \_\_\_ 12. Edit the condition to state:

the duration ( in years ) of 'the loan' is at least <a number>

**Important**

Remember to click the check mark to save your edit.

13. Click the **Loan Duration** node, and add a branch.

a. Modify the branch to Rule 1 to state:

the duration (in years) of 'the loan' is more than 10

**Reminder**

Remember to save each change for the branches and actions by clicking the check mark.

b. Modify the branch to Rule 2 to state:

the duration (in years) of 'the loan' is at most 10

c. Set the action for Rule 1 to:

set insurance required in 'the loan report' to true

d. Set the action for Rule 2 to:

set the approved of 'the loan report' to true

14. Save your work.

15. Select the **Show rules** property and view the rules in the decision tree editor.

a. Click **Decision Tree Properties**.

b. Select **Show rules** and then click **OK**.



- \_\_\_ c. In the tree diagram, click one of the rules to see the full rule statement in the rule pane that is now visible.

**Note**

Try clicking the rule header to see the rule. Clicking outside of the header might not show the rule.



- \_\_\_ 16. Save your work.

**End of exercise**

## Exercise review and wrap-up

The first part of the exercise looked at how to author a decision table.

The second part of the exercise looked at how to author a decision tree.

(Optional) To see the solution to this exercise, switch to a new workspace and import the project

**Ex 08: Authoring decision tables and trees > 02-answer.**

# Exercise 9. Working with static domains

## What this exercise is about

In this exercise, you learn how to simplify rule authoring by defining static domains in the BOM.

## What you should be able to do

After completing this exercise, you should be able to:

- Create various types of static domains
- Use domains in rules

## Introduction

In this exercise, you learn how to create and use static domains.

This exercise includes these sections:

- Section 1, "Exploring a collection domain"
- Section 2, "Working with an enumeration of literals"
- Section 3, "Defining an enumeration of static references"

## Requirements

This exercise uses the following files, which are installed in the `<InstallDir>\studio\training` directory:

- Start project: Dev 09 - Static domains\01-start
- Solution project: Dev 09 - Static domains\02-answer
  - You use the solution project in "Exercise review and wrap-up" on page 9-22.

## Section 1. Exploring a collection domain

In this section, you explore a collection domain to learn how you can use it in a business rule.



### Hint

In this exercise, you must write some code and some rule statements. You can find the code snippets and the rules in the `<LabfilesDir>\code\create_domains.txt` file, and you can copy and paste them in Rule Designer.

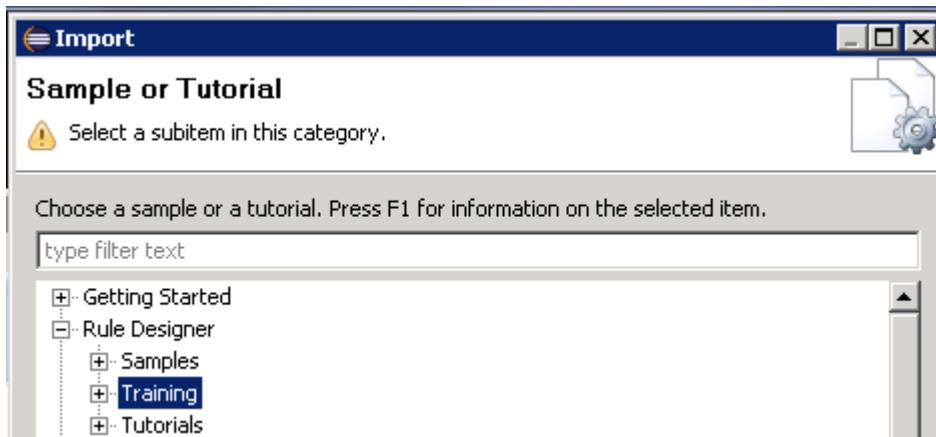
### 1.1. Setting up your environment for this exercise

- 1. In Rule Designer, switch to a new workspace:
  - a. From the **File** menu, click **Switch Workspace > Other**.
  - b. In the Workspace Launcher window, enter the path:  
`<LabfilesDir>\workspaces\static_domains`
- 2. Close the Welcome view.
- 3. Use the **Import** menu to import the exercise start project.
  - a. In Eclipse, click **File > Import** or right-click the Rule Explorer and click **Import**.
  - b. In the Select window, expand **Operational Decision Manager**, select **Samples and Tutorials**, and click **Next**.
  - c. On the “Sample or Tutorial” page, expand the **Rule Designer > Training** node.



### Hint

You can collapse the Rule Designer node and then expand it to simplify the list.



- d. Expand the **Ex 09: Static domains** node, and select **01-start**.

- \_\_\_ e. Make sure that the **Import shared projects** check box is selected, and click **Finish**.
- \_\_\_ 4. When the workspace finishes building, close the Help view.

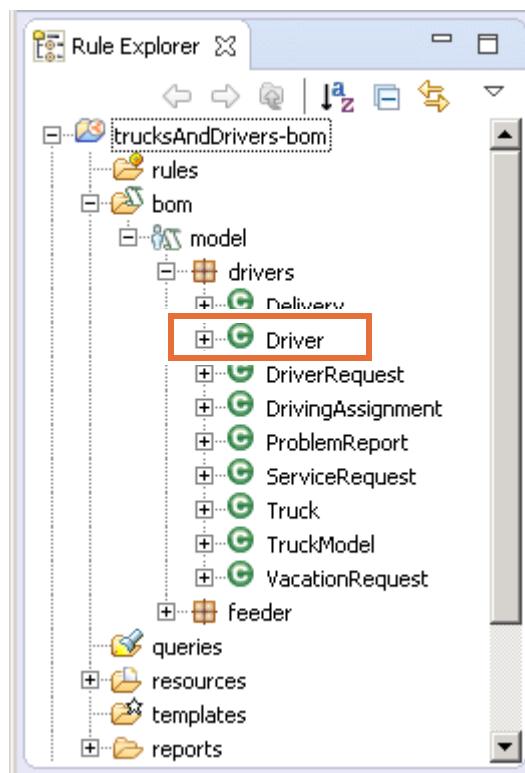
In your workspace, you now have the following projects in the `trucksAndDrivers-tests` decision service:

- `trucksAndDrivers-bom`
- `trucksAndDrivers-rules`
- `trucksAndDrivers-tests` (main rule project)
- `trucksAndDrivers-xom`

These projects define the business rule solution in which you explore and create domains.

## 1.2. Exploring the domain

- \_\_\_ 1. In the `trucksAndDrivers-bom` project, expand **bom > model > drivers** and double-click the `drivers.Driver` class to open it in the BOM editor.



2. In the **Members** section of the Class page, double-click the `drivingAssignments` member.

**Class Driver (package: drivers)**

**General Information**

|               |                  |           |
|---------------|------------------|-----------|
| Name:         | Driver           |           |
| Namespace:    | drivers          | Change... |
| Superclasses: | java.lang.Object | Change... |
| Interfaces:   |                  | Change... |

Deprecated

**Members**

Specify the members of this class.

- age
- drivingAssignments
- gender
- knownDrivers
- licenseClass
- name
- nbAccidents
- totalVacationTime
- vacationRequests

**Action Buttons:**

- New...
- Delete
- Edit

3. On the Member page, notice that the `drivingAssignments` member is defined as a vector.

**Member drivingAssignments (class: drivers.Driver)**

**General Information**

|        |                    |           |
|--------|--------------------|-----------|
| Name:  | drivingAssignments |           |
| Type:  | java.util.Vector   | Browse... |
| Class: | drivers.Driver     | Browse... |

4. The vector is defined with a collection domain, and each member of the collection is of type `drivers.DrivingAssignment`.

**Domain**

Create and edit a domain for this member.

[Edit the domain.](#)  
[Remove the domain.](#)

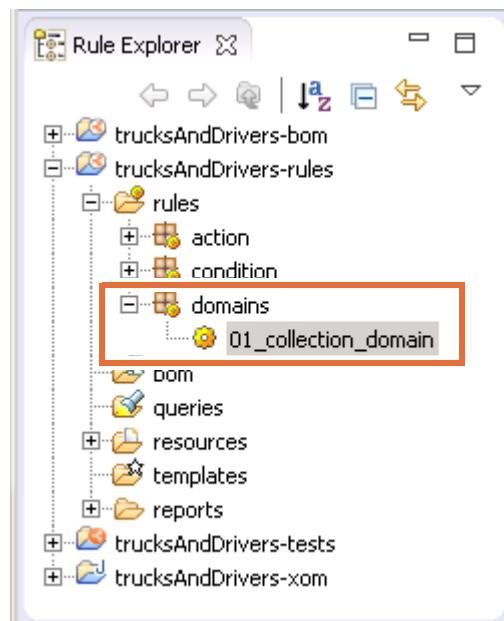
**Domain type: Collection**

Element type: `drivers.DrivingAssignment`

Cardinality: 0, \*

This collection domain was automatically defined when the BOM was generated from the XOM because this XOM attribute is defined as: `Vector<DrivingAssignment>`

- \_\_\_ 5. In Rule Explorer, expand **trucksAndDrivers-rules > rules > domains**.
- \_\_\_ 6. Double-click the `01_collection_domain` rule to open it and see how this rule uses the collection domain.



- \_\_\_ 7. When you are done, close the rule.

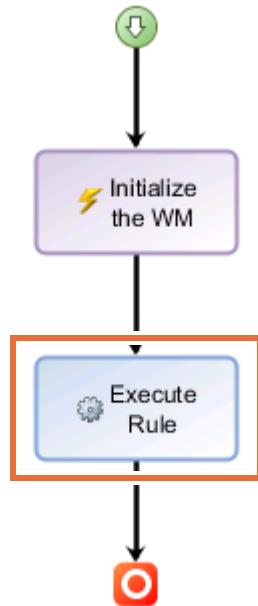
### 1.3. Testing the rule

With the next steps, you test the `domains\01_collection_domain` action rule.

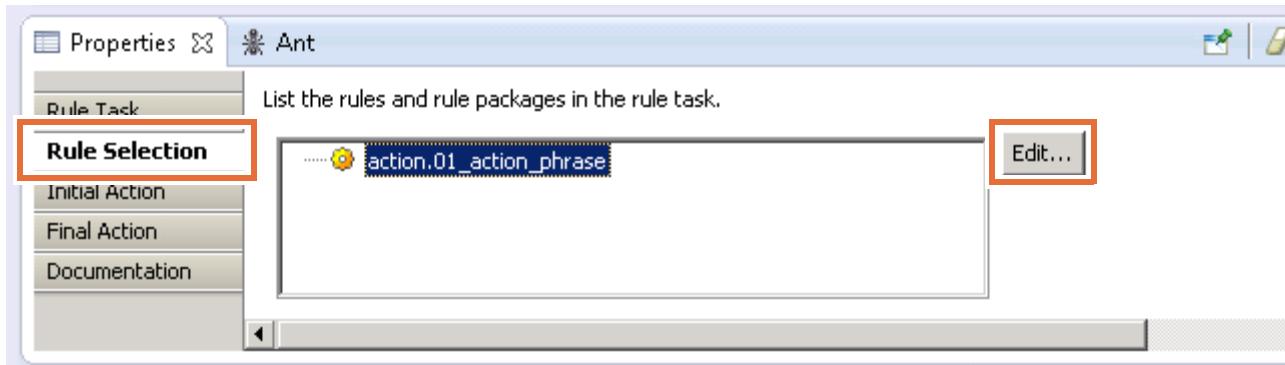
First, you verify that objects in the working memory exist that match the definitions of this rule. Then, you use the `trucksAndDrivers-tests` project to test this rule in the rule project.

- \_\_\_ 1. Expand the **trucksAndDrivers-xom > src > feeder** folder and double-click the `WMFeeder.java` file.
- \_\_\_ 2. Explore the `WMFeeder.java` file.
  - \_\_\_ a. Verify that a `DrivingAssignment` object is created for the `Driver` with the name: `John Jongle`
  - \_\_\_ b. Read through the code to see how the `DrivingAssignment` object is inserted into the working memory through the `IlrContext insert` method.
- \_\_\_ 3. Use the `trucksAndDrivers-tests` project to test the `01_collection_domain` rule.
  - \_\_\_ a. In the Rule Explorer, expand the **trucksAndDrivers-tests > rules** folder and double-click the `testFlow` ruleflow to open it.

- \_\_ b. Double-click the **Execute Rule** rule task to edit its properties.

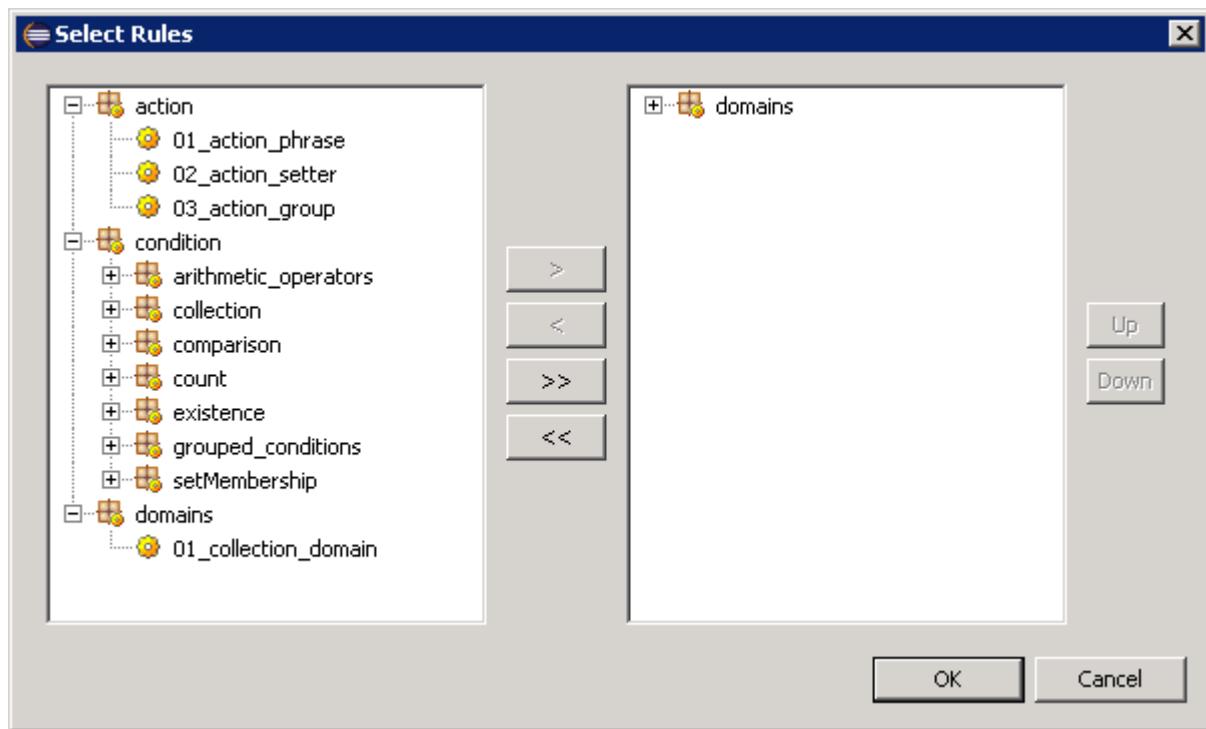


- \_\_ c. In the Rule Task properties, click **Rule Selection** and click **Edit**.



The Select Rules window opens.

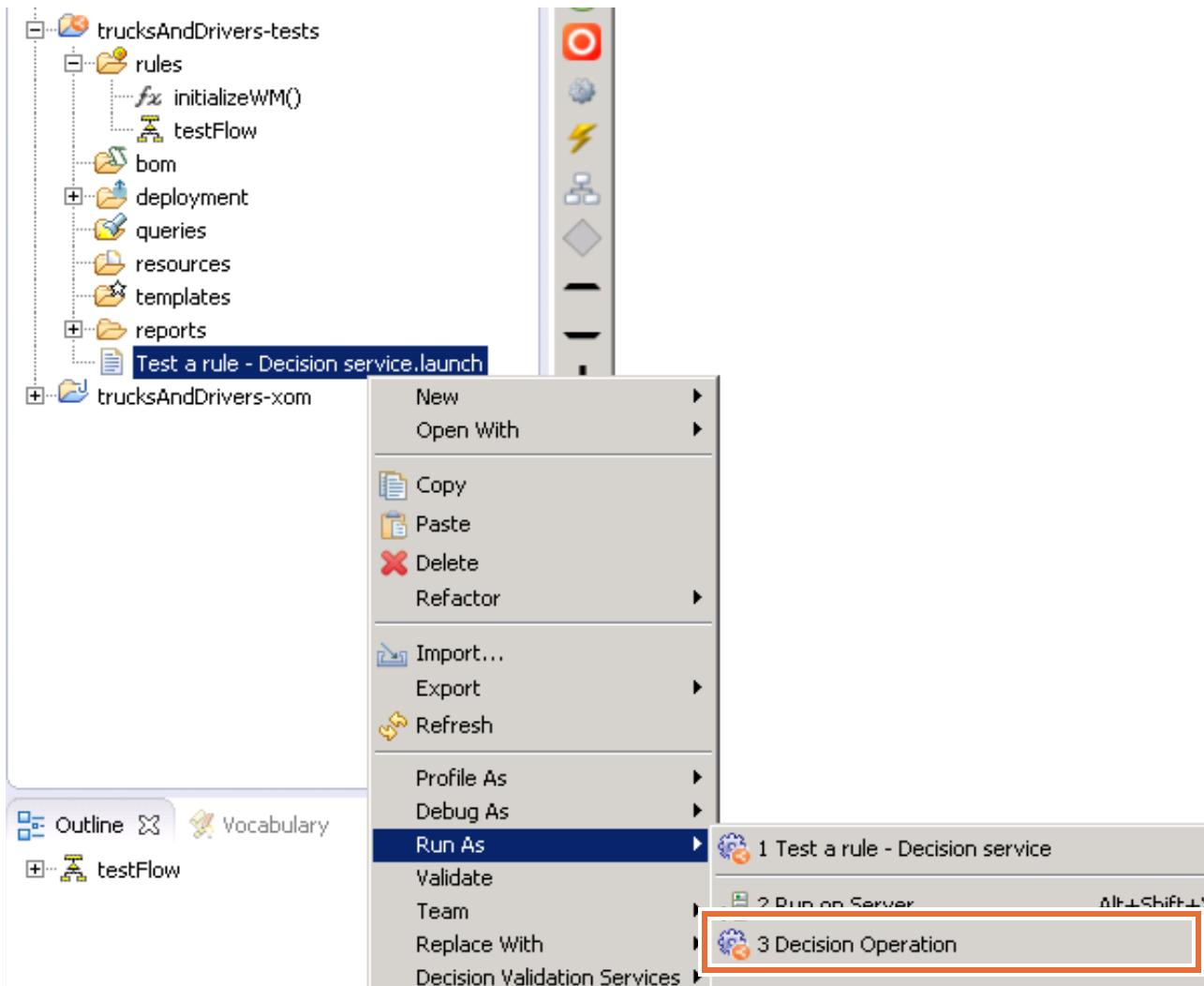
- \_\_\_ d. In the Select Rules window, replace the content on the right side with the `domains` package and click **OK**.



Now, only the rules of the `domains` package are executed in the `testFlow` ruleflow.

- \_\_\_ e. Save the ruleflow (Ctrl+S).

- \_\_ f. In the trucksAndDrivers-tests project, right-click the **Test a rule - Decision service.launch** configuration, and click **Run As > Decision Operation** to execute the testFlow ruleflow.



- \_\_ g. Verify that you have the following result in the Console view:

Added the pending driving assignment to the list of driving assignments of John Jongle



The following rule is another example of how you can use this collection domain.

```
definitions
 set 'the driver' to a driver ;
if
 there is at least one driving assignment in the driving assignments of 'the
 driver'
then
 ...

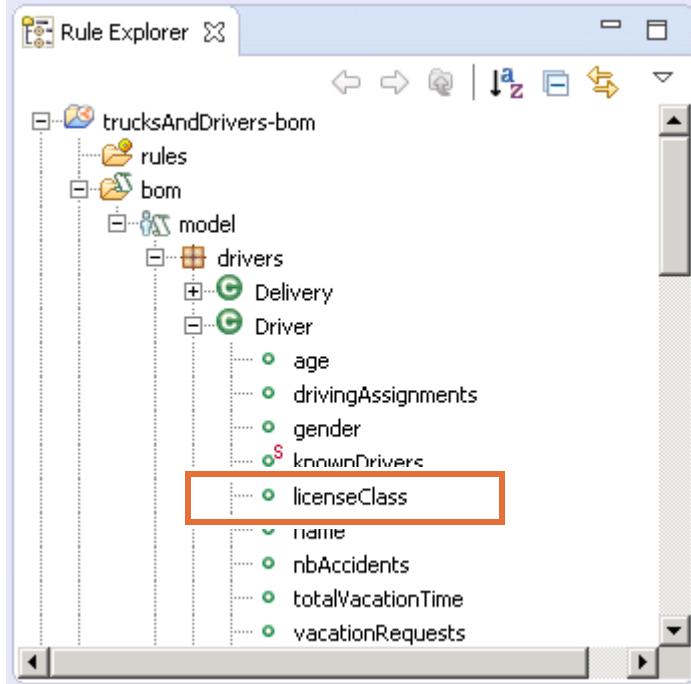
```

## Section 2. Working with an enumeration of literals

In this section, you define a domain as a list of literal values {A, B, C, D} for the `licenseClass` member of the `Driver` and `TruckModel` classes, and then use it to author an action rule.

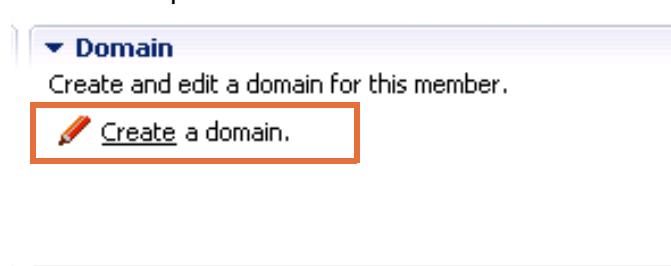
### 2.1. Creating the domain

- 1. In the `trucksAndDrivers-bom` project, expand the `drivers.Driver` class and double-click the `licenseClass` BOM member to open it in the BOM editor.

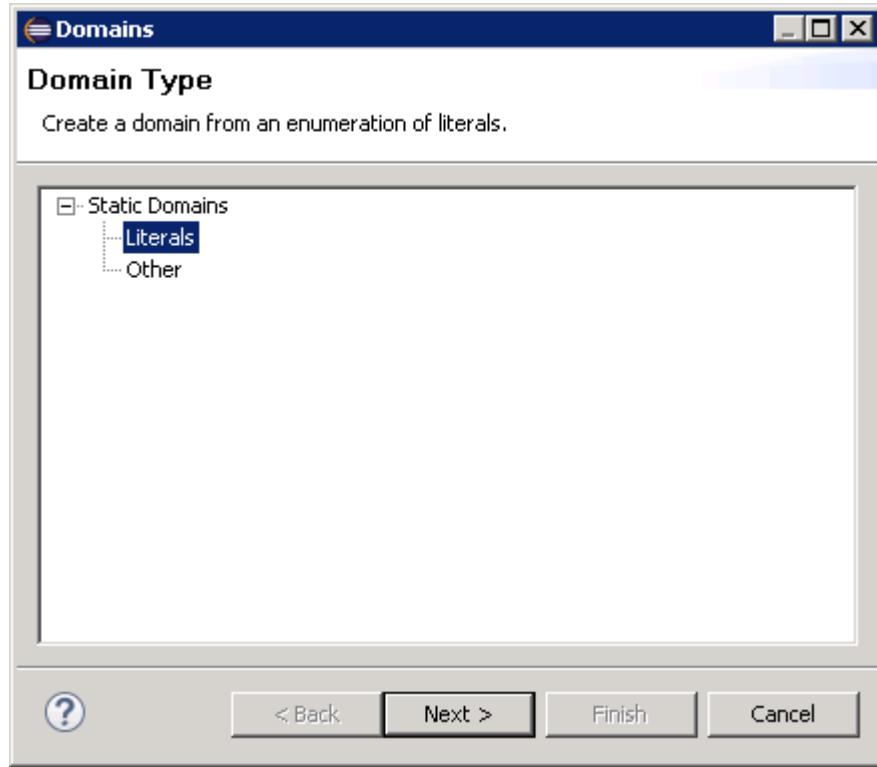


- 2. In the BOM editor, in the **Domain** section, create a domain of literal values:

- a. Click **Create a domain** to open the Domains window.



- \_\_ b. On the Domain Type page of the Domains windows, select **Literals** in the list and click **Next**.

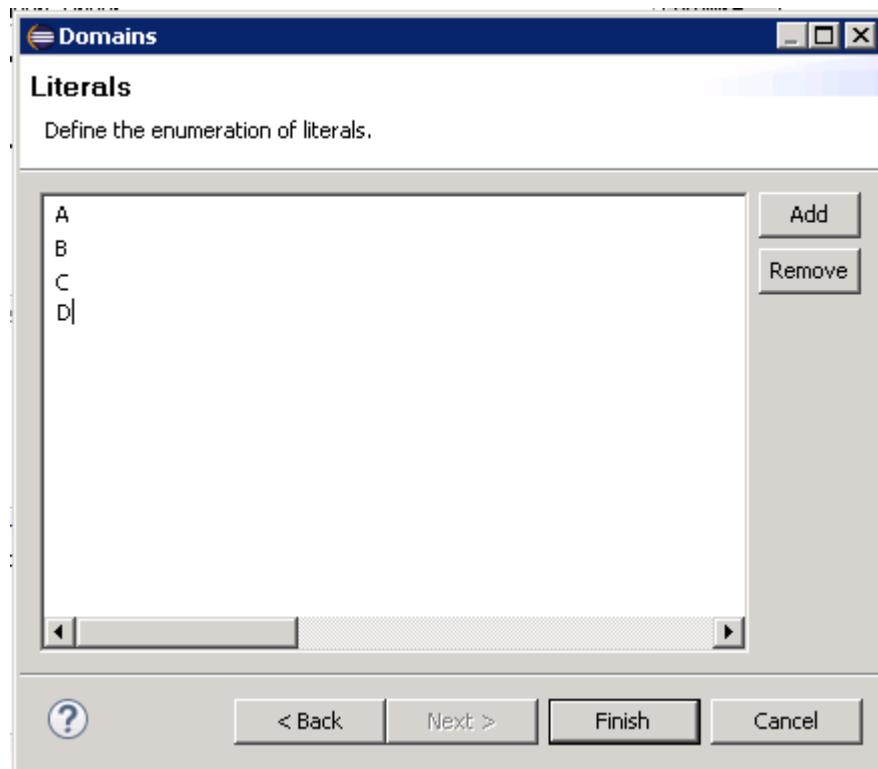


- \_\_ c. On the Literals page of the Domains window, click **Add** and type over the new domain value to rename it to: A

 **Hint**

If you want, you can click **Add** several times to add the placeholders for your domain values, and then type over the placeholder names later.

- \_\_\_ d. Add these values to the domain: B, C, and D



- \_\_\_ e. Click **Finish** to close the Domains window.

You created the domain of type Literals {A, B, C, D}.

**Domain type: Literals**

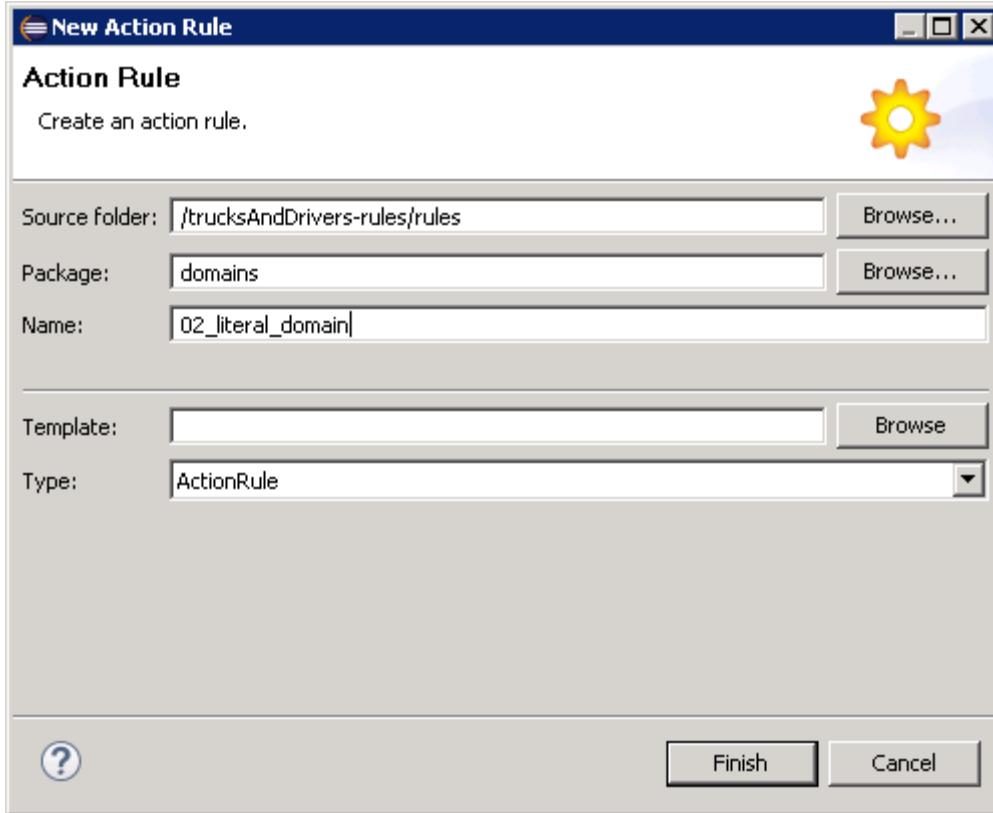
- A
- B
- C
- D

- \_\_\_ 3. Create the domain of type Literals {A, B, C, D} for the  
drivers.TruckModel.licenseClass BOM member as well.  
\_\_\_ 4. Save the BOM (Ctrl+S).

## 2.2. Testing the domain in a rule

In this section, you use your new domains to author a rule.

- 1. In the `domains` rule package of the `trucksAndDrivers-rules` project, create the action rule named: `02_literal_domain`



- 2. Enter this rule statement in the rule editor:

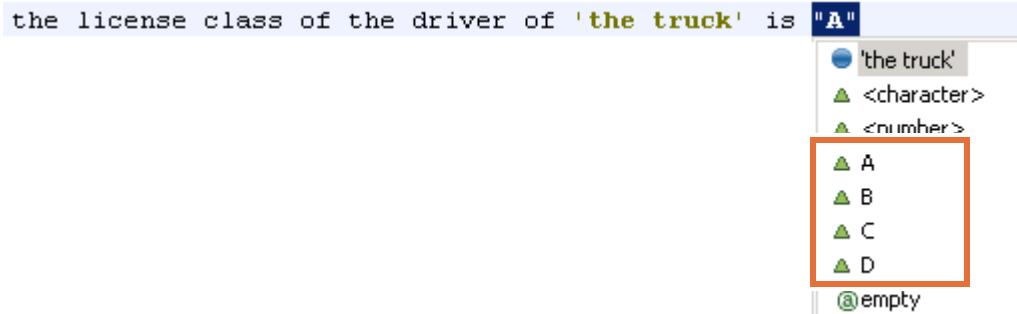
```
definitions
 set 'the truck' to a truck where the driver of this truck is not null;
if
 the license class of the driver of 'the truck' is "A" and the license
 class of the model of 'the truck' is one of {"B", "C", "D"}
then
 display the message "The driver (" + the name of the driver of 'the truck'
 + ") is not eligible to drive the truck " + the serial number of 'the
 truck' ;
```



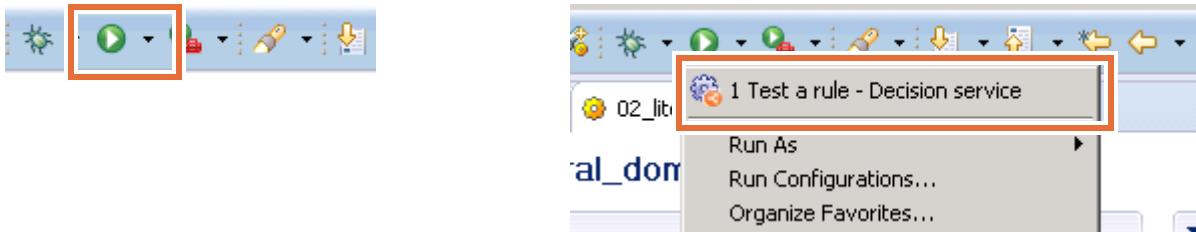
### Hint

Recall that you can type directly in the Intellirule editor or you can paste the code from the code snippet in the `<LabfilesDir>\code\create_domains.txt` file.

Because the licenseClass BOM member type is a domain of Literals, an enumeration of values is available to you when you author an action rule that uses this licenseClass member.



- \_\_\_ 3. After you finish editing, save the rule.
- \_\_\_ 4. Rerun the test project from the toolbar by clicking **Run > Test a rule - Decision service** on the toolbar.



### Information

The testFlow ruleflow is already correctly configured to execute all rules of the domains package. You do not have to explicitly add your new rule to the Execute task.

- \_\_\_ 5. Verify that, in addition to the result from the previous section, you also have the following result:

The driver (John Jongle) is not eligible to drive the truck TRUCK-F150

- \_\_\_ 6. Close the rule.

## Section 3. Defining an enumeration of static references

A domain that is set as an enumeration of static references specifies a list of references to constants, for example:

```
{static GroupA, static GroupB, static GroupC}
```

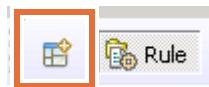
You can define attribute types, method return types, and arguments as follows:

- If you have an attribute of type A, you can define a domain of static references on it by using the static attributes of the class A (classic Java enumeration pattern)
- If you have an attribute of a primitive type, you can define a literal domain on it

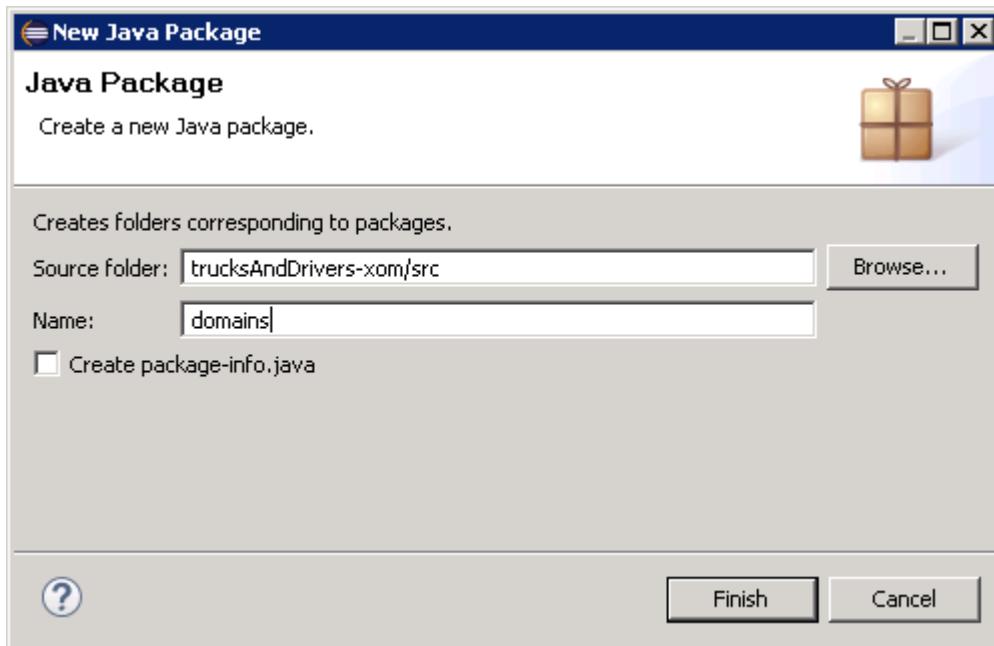
### 3.1. Creating the static references Java class

In this section, you change the attribute `gender` in the `Driver` class to use the static attributes of a new class called `GenderType`.

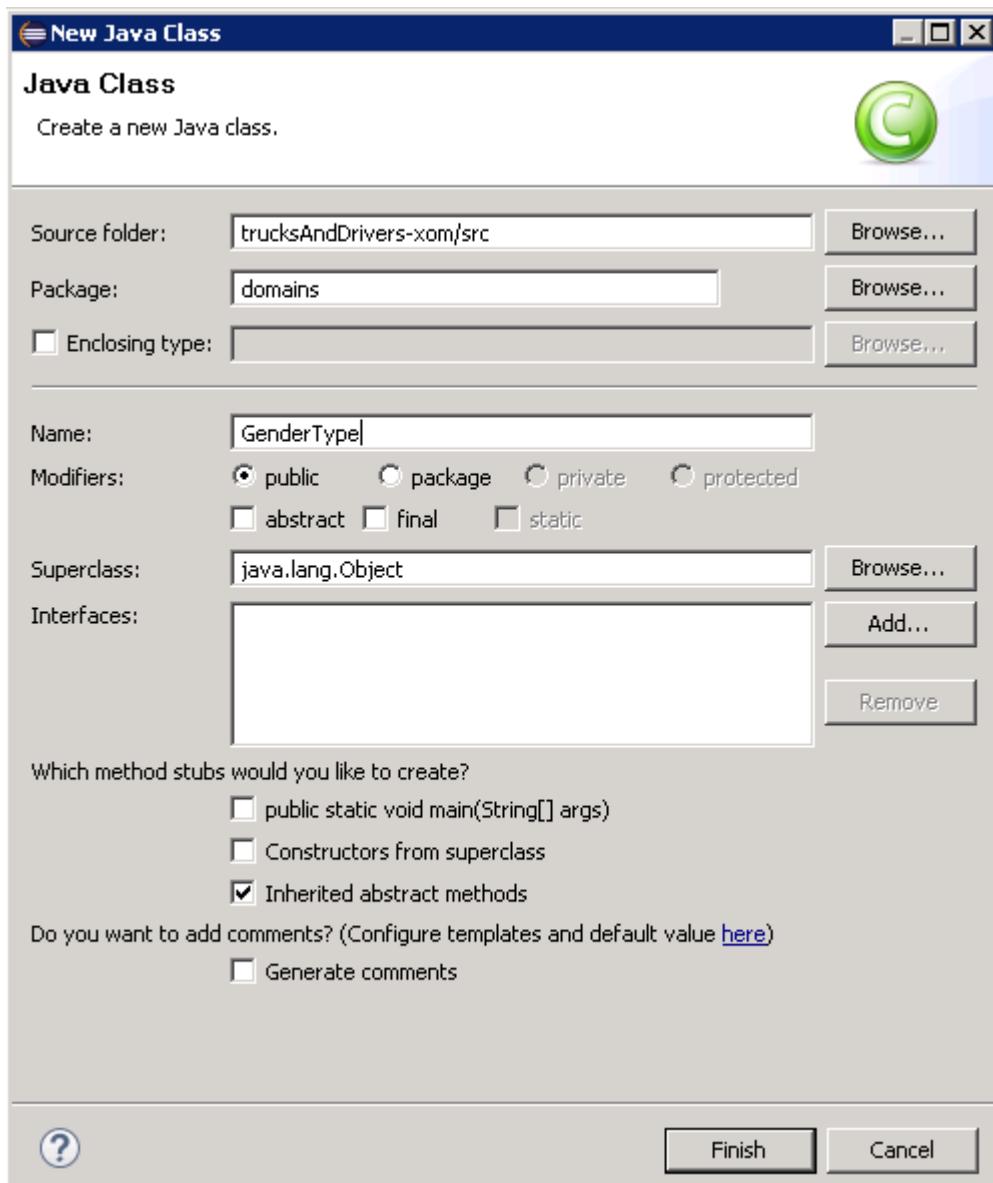
- 1. Switch to the Java perspective.
- a. Click the **Open Perspective** icon in the toolbar, which is in the upper-right corner of the Rule Designer window.



- b. Select **Java (default)** from the list of perspectives, and click **OK**.
- 2. Create a package and class in the `trucksAndDrivers-xom > src` project.
- a. Expand `trucksAndDrivers-xom > src`, right-click `src`, and click **New > Package**.
- b. Name the package: `domains`
- c. Click **Finish**.



- \_\_\_ d. Right-click the domains package, and click **New > Class**.
- \_\_\_ e. Name the class: GenderType
- \_\_\_ f. Leave the other default values and click **Finish**.



- \_\_\_ 3. When the `GenderType.java` file opens in the editor, define the static attributes in the Java file with this content:

```
package domains;
public class GenderType {
 private final String name;
 public static final GenderType MALE = new GenderType("male");
 public static final GenderType FEMALE = new GenderType("female");
 public static final GenderType UNKNOWN = new GenderType("unknown");
 private GenderType(String _name){
 this.name = _name;
 }
 public String toString(){
 return this.name;
 }
}
```

**Hint**

You can find this code snippet in the `<LabfilesDir>\code\create_domains.txt` file.

- \_\_\_ 4. Save your work (Ctrl+S).
- \_\_\_ 5. In the Package Explorer, find the `drivers.Driver` class in the **trucksAndDrivers-xom > src** folder, and change the type from `String` to `GenderType` for the following class members:

- gender attribute
- getGender method
- setGender method

- \_\_\_ a. Expand **trucksAndDrivers-xom > src > drivers.Driver**.
- \_\_\_ b. Double-click `drivers.Driver.gender`.
- \_\_\_ c. Find the following line:

```
private String gender;
```

```
public class Driver {
 private int age;
 private int nbAccidents;
 private String name; // name
 private String licenseClass; // driver's license class
 private Vector<DrivingAssignment> drivingAssignments; // current driving assignments
 private Vector<VacationRequest> vacationRequests; // vacation requests
 private String gender;
```

- \_\_\_ d. Edit the line by changing `String` to `GenderType`.

It should now read:

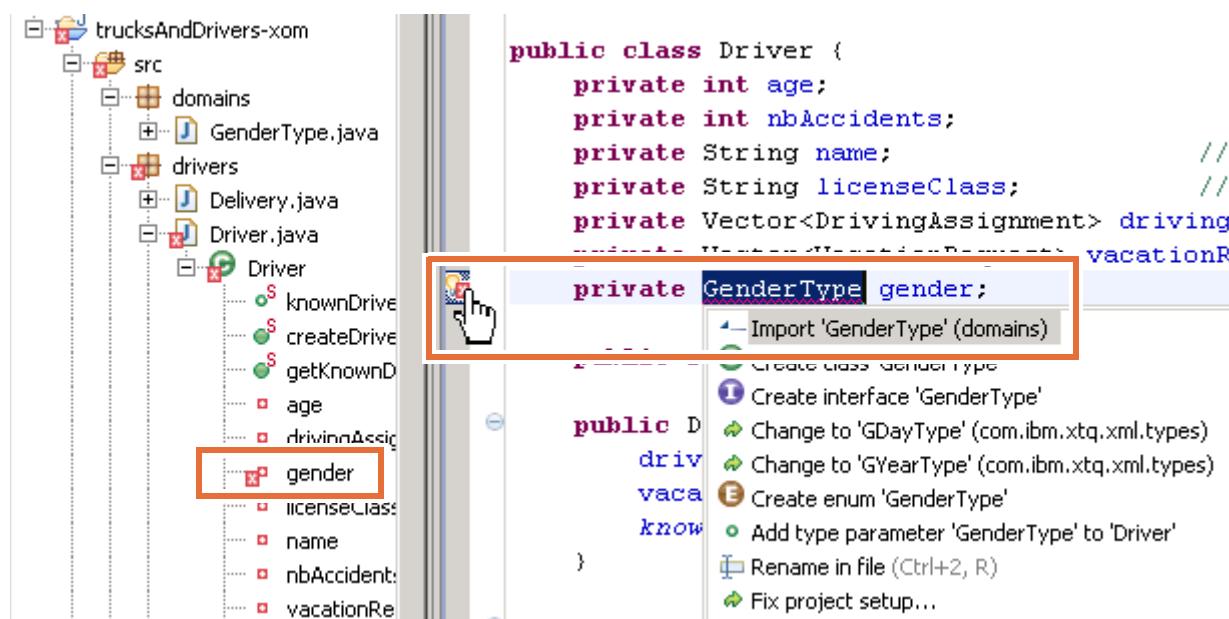
```
private GenderType gender;

public class Driver {
 private int age;
 private int nbAccidents;
 private String name; // name
 private String licenseClass; // driver's license class
 private Vector<DrivingAssignment> drivingAssignments; // current driving assignments
 private Vector<VacationRequest> vacationRequests; // vacation requests
 private GenderType gender;
```

- \_\_\_ e. Change the String attribute for drivers.Driver.getGender and drivers.Driver.setGender to GenderType.
- \_\_\_ f. Save your work (Ctrl+S).

Notice that you get an error on GenderType.

- \_\_\_ 6. To resolve the problem, click the error icon and double-click the **Import Gender Type (domains)** quick fix.



- \_\_\_ 7. Save your work (Ctrl+Shift+S).
- \_\_\_ 8. Update the `feeder.WMFeeder` class that uses the setter method.
  - \_\_\_ a. In the `WMFeeder` Java file, replace all occurrences of "MAN" with: `GenderType.MALE`

- \_\_\_ b. In the WMFeeder class, replace all occurrences of "UNKNOWN" with:  
GenderType.UNKNOWN

```
private void createModel() {
 // Definitions of the drivers
 Driver dA20 = new Driver("George Smith", TruckModel.MACK_TRUCK.getLicenseC
 dA20.setAge(20);
 dA20.setNbAccidents(3);
 dA20.setGender(GenderType.MALE);

 Driver dA18 = new Driver("John Jongle", TruckModel.MACK_TRUCK.getLicenseCl
 dA18.setAge(18);
 dA18.setNbAccidents(0);
 dA18.setGender(GenderType.MALE);

 Driver dB23 = new Driver("Marc Lansen", TruckModel.F150.getLicenseClass());
 dB23.setAge(23);
 dB23.setNbAccidents(0);
 dB23.setGender(GenderType.MALE);

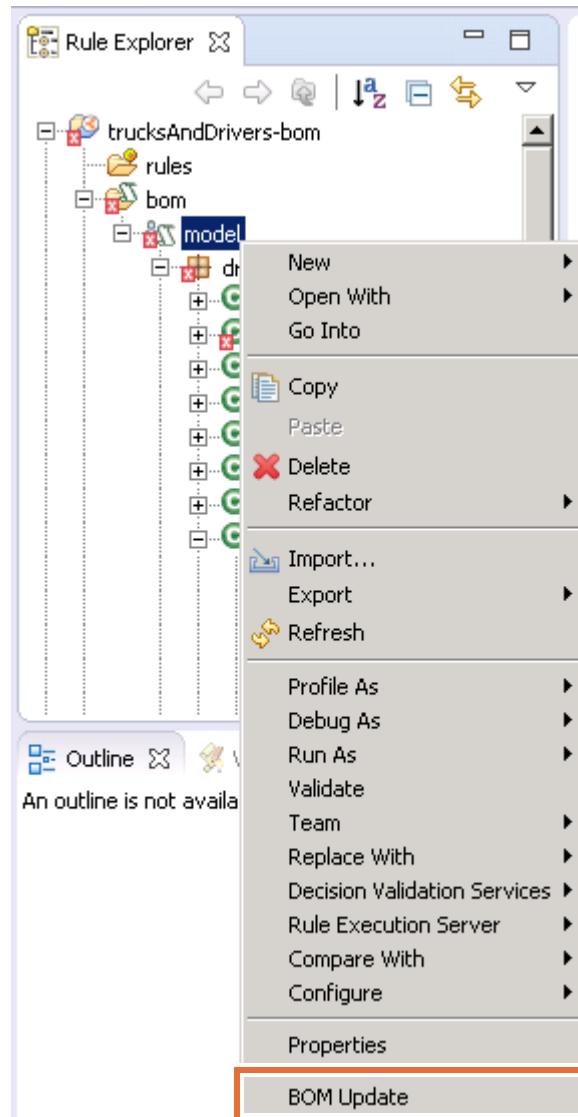
 Driver dB37 = new Driver("Jill Northwood", TruckModel.T2000.getLicenseClas
 dB37.setAge(37);
 dB37.setNbAccidents(1);
 dB37.setGender(GenderType.UNKNONW);
```

- \_\_\_ c. If you have errors on these values, use the **Import Gender Type (domains)** quick fix as you did in Step 4.

After you save the XOM, the Problems view lists an error on the BOM. You correct this error now by using the BOM Update view.

- \_\_\_ 9. Save your work and close the editing windows for the Java files.  
\_\_\_ 10. Switch back to the Rule perspective.

- \_\_\_ 11. In the trucksAndDrivers-bom project, right-click `bom.model` and click **BOM Update**.



You find some differences in the **Differences and Actions** section of the BOM Update view.

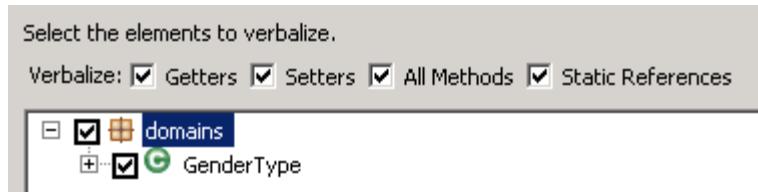
- The XOM class `domains.GenderType` is not found in the BOM.
- The definition of the attribute `drivers.Driver.gender` differs between the BOM and the XOM.

- \_\_\_ a. Select the **Import the XOM class** action and click **Perform and save**.

**Differences and Actions**

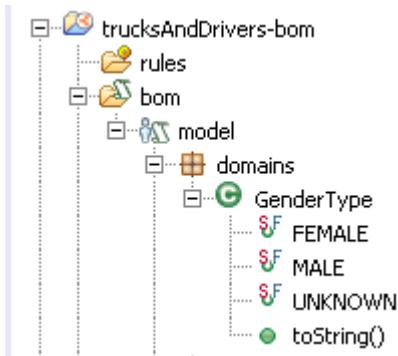
| Actions:                                                                                                                                                              |                    |                  |                                                         |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|------------------|---------------------------------------------------------|
| <input drivers.driver""="" type="text" value="Update the BOM class "/> <input type="button" value="Perform and save"/> <input type="button" value="Clear table"/>     |                    |                  |                                                         |
| Perform a                                                                                                                                                             |                    |                  |                                                         |
| <input domains.gendertype""="" type="text" value="Import the XOM class "/> <input type="button" value="Perform and save"/> <input type="button" value="Clear table"/> |                    |                  |                                                         |
| Origin                                                                                                                                                                |                    |                  |                                                         |
| XOM                                                                                                                                                                   | domains.GenderType | Missing from BOM | XOM class "domains.GenderType" not found in BOM.        |
| XOM                                                                                                                                                                   | drivers.Driver     | Modified         | Difference between attributes, XOM attribute "drivers.D |

- \_\_\_ b. In the Verbalize BOM window, make sure that you click **Select All** and that you select the **All Methods** check box to verbalize all members and methods of the BOM class `GenderType`.



- \_\_\_ c. Click **Finish**.
- \_\_\_ 12. Update the BOM class.
- \_\_\_ a. Go back to the BOM Update view.
- \_\_\_ b. Expand **Differences and Actions**.
- \_\_\_ c. Select the **Update the BOM class** action and click **Perform and save**.
- \_\_\_ 13. In the Configure Verbalization window, click **Finish**.
- \_\_\_ 14. After the BOM is updated, verify the following points:

- \_\_\_ a. A new `GenderType` BOM class exists under `domains` in the BOM and is defined as a domain of type Static References with these values: `FEMALE`, `MALE`, and `UNKNOWN`



- \_\_\_ b. The **Type** field of `drivers.Driver.gender` BOM member is changed to: `GenderType`

### Member gender (class: drivers.Driver)

| General Information |                                                                                          |
|---------------------|------------------------------------------------------------------------------------------|
| Name:               | <input type="text" value="gender"/>                                                      |
| Type:               | <input type="text" value="domains.GenderType"/> <input type="button" value="Browse..."/> |
| Class:              | <input type="text" value="drivers.Driver"/> <input type="button" value="Browse..."/>     |

- \_\_\_ 15. Close the BOM editor.

### 3.2. Authoring an action rule that uses a domain of static references

In this section, you use the `GenderType` static domain of type Static References to author an action rule.

- 1. In the `domains` rule package of the `trucksAndDrivers-rules` project, create the action rule `03_static_references_domain`:

```
definitions
 set 'the driver' to a driver;
if
 the gender of 'the driver' is UNKNOWN
then
 display the message "Update the personal data of " + the name of 'the
 driver';
```



#### Reminder

You can type directly in the Intellirule editor or you can paste the code from the code snippet in the `<LabfilesDir>\code\create_domains.txt` file.

- 2. Save and close the rule.
- 3. Rerun the `trucksAndDrivers-tests` project from the toolbar by clicking **Run > Test a rule - Decision service** on the toolbar.
- 4. Verify that the following line is now included in the result:

Update the personal data of Jill Northwood

### End of exercise

## Exercise review and wrap-up

This exercise looked at how to create static domains in the BOM, and how to work with them in rules. You also saw the relationship between domains and the XOM. You explored an existing domain of type Collection and learned how you can use it in an action rule.

(Optional) To see the solution to this exercise, switch to a new workspace and import the project **Ex 09: Static domains > 02-answer**.

# Exercise 10. Working with dynamic domains

## What this exercise is about

In this exercise, you learn how to define and use dynamic domains with Microsoft Excel spreadsheets.

## What you should be able to do

After completing this exercise, you should be able to:

- Create dynamic domains in Microsoft Excel spreadsheets
- Update and use dynamic domains in rules
- Access and update dynamic domains in Decision Center
- Synchronize dynamic domains between Rule Designer and Decision Center

## Introduction

In this exercise, you work in Rule Designer to create a dynamic domain with Microsoft Excel, update the values of a domain, and discover the effects of such updates.

You learn how to resolve inconsistencies across the BOM, XOM, and rules after modifying domain classes. You also learn how to synchronize updated domain values between Rule Designer and Decision Center.

The exercise involves these tasks:

- Section 1, "Creating a dynamic domain in Rule Designer"
- Section 2, "Using the dynamic domain in a rule"
- Section 3, "Updating the dynamic domain"
- Section 4, "Updating the XOM"
- Section 5, "Publishing the BOM and rule projects to Decision Center"
- Section 6, "Examining rules in Decision Center"
- Section 7, "Modifying the dynamic domain in Decision Center"
- Section 8, "Updating Rule Designer from Decision Center"

## Requirements

You should complete these exercises before proceeding:

- Exercise 3, "Working with the BOM"
- Exercise 6, "Exploring action rules"
- Exercise 9, "Working with static domains"

This exercise uses a series of Excel files that are stored in the `<LabfilesDir>\code` directory.

It also uses the following files, which are installed in the `<InstallDir>\studio\training` directory:

- Start project: Dev 10 – Dynamic domains\01-start
- Solution project: Dev 10 – Dynamic domains\02-answer
  - You use the solution project in "Exercise review and wrap-up" on page 10-39



### Important

---

For this exercise, you work with a series of Microsoft Excel spreadsheets to define the values of the dynamic domain, and their updates. The Excel spreadsheets are stored in the `<LabfilesDir>\code` directory.

During the exercise, you open and read these Excel spreadsheets, but you do not have to modify them.

You can read the Microsoft Excel spreadsheets by using either:

- Microsoft Excel Viewer, which is installed on the computer lab environment that is developed for this course
- Microsoft Excel, which is **not** installed on the computer lab environment that is developed for this course

If you have access to Microsoft Excel, you can *optionally* edit the Microsoft Excel spreadsheets.

---

## Section 1. Creating a dynamic domain in Rule Designer

In this section, you create a dynamic domain in the BOM to represent the values that are listed in an Excel spreadsheet.



### Hint

In this exercise, you must write some pieces of code. You can find the pieces of code to create in the `<LabfilesDir>\code\create_domains.txt` file, and you can copy and paste them in Rule Designer.

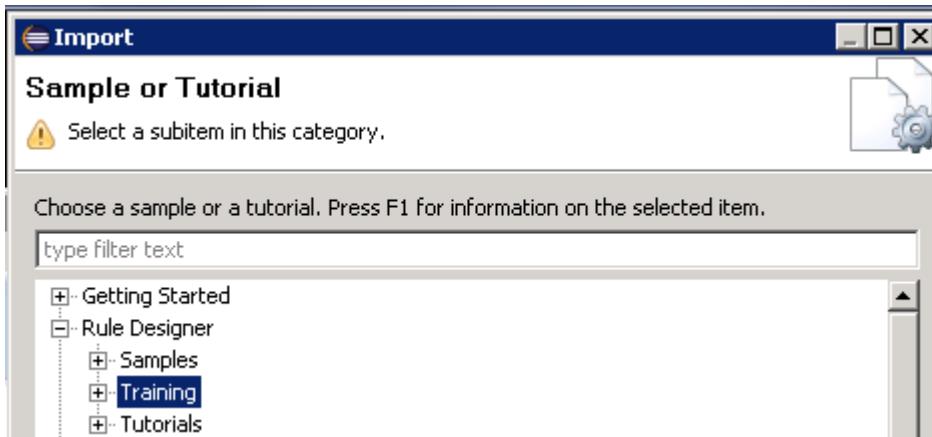
### 1.1. Setting up your environment for this exercise

- \_\_\_ 1. In Rule Designer, switch to a new workspace:
  - \_\_\_ a. From the **File** menu, click **Switch Workspace > Other**.
  - \_\_\_ b. In the Workspace Launcher window, enter the path:  
`<LabfilesDir>\workspaces\dynamic_domain`
- \_\_\_ 2. Close the Welcome view.
- \_\_\_ 3. Use the **Import** menu to import the exercise start project.
  - \_\_\_ a. In Eclipse, click **File > Import** or right-click the Rule Explorer and click **Import**.
  - \_\_\_ b. In the Select window, expand **Operational Decision Manager**, select **Samples and Tutorials**, and click **Next**.
  - \_\_\_ c. On the “Sample or Tutorial” page, expand the **Rule Designer > Training** node.



### Hint

You can collapse the Rule Designer node and then expand it to simplify the list.



- \_\_\_ d. Expand the **Ex 10: Dynamic domains** node, and select **01-start**.

- \_\_ e. Make sure that the **Import shared projects** check box is selected, and click **Finish**.
- \_\_ f. When the workspace finishes building, close the Help view.



### Information

In your workspace, you now have the `trucksAndDrivers-tests` decision service, which has the following projects:

- `trucksAndDrivers-bom`
- `trucksAndDrivers-rules`
- `trucksAndDrivers-tests`
- `trucksAndDrivers-xom`

These projects define the business rule solution to which you must add the dynamic domain.

#### **Scenario:**

In the Trucks and Drivers rule application, drivers might submit requests for vacations. In their current state, requests for vacations are not associated with any specific type.

After a discussion with the human resources department, business analysts tell you that the type of vacation request must be differentiated for statistical purposes.

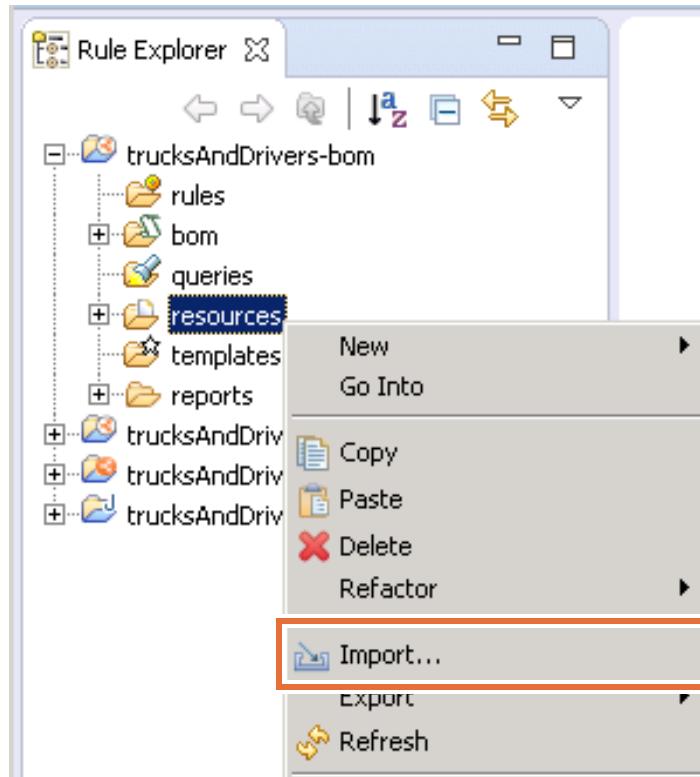
To start the work, the human resources department provides you with an Excel spreadsheet that lists their current list of possible types of vacation requests. They also indicate that this list might change regularly, and ask that it remain easy to update in case of changes.

Because of potential change, you cannot use a static domain to implement the requirement. Instead, you use a dynamic domain, which is an enumerated domain (or list of values) that can change dynamically.

## 1.2. Creating the domain

- \_\_ 1. Open the `<LabfilesDir>\code\vacationRequestTypes-1.xls` file and verify that each row of this file contains three columns, where:
  - The first column represents the name to the domain value.  
Example: `JuryDuty`
  - The second column represents the code in the BOM-to-XOM mapping.  
Example: `return "V78";`
  - The third column represents the verbalization of the domain value.  
Example: `Jury Duty`

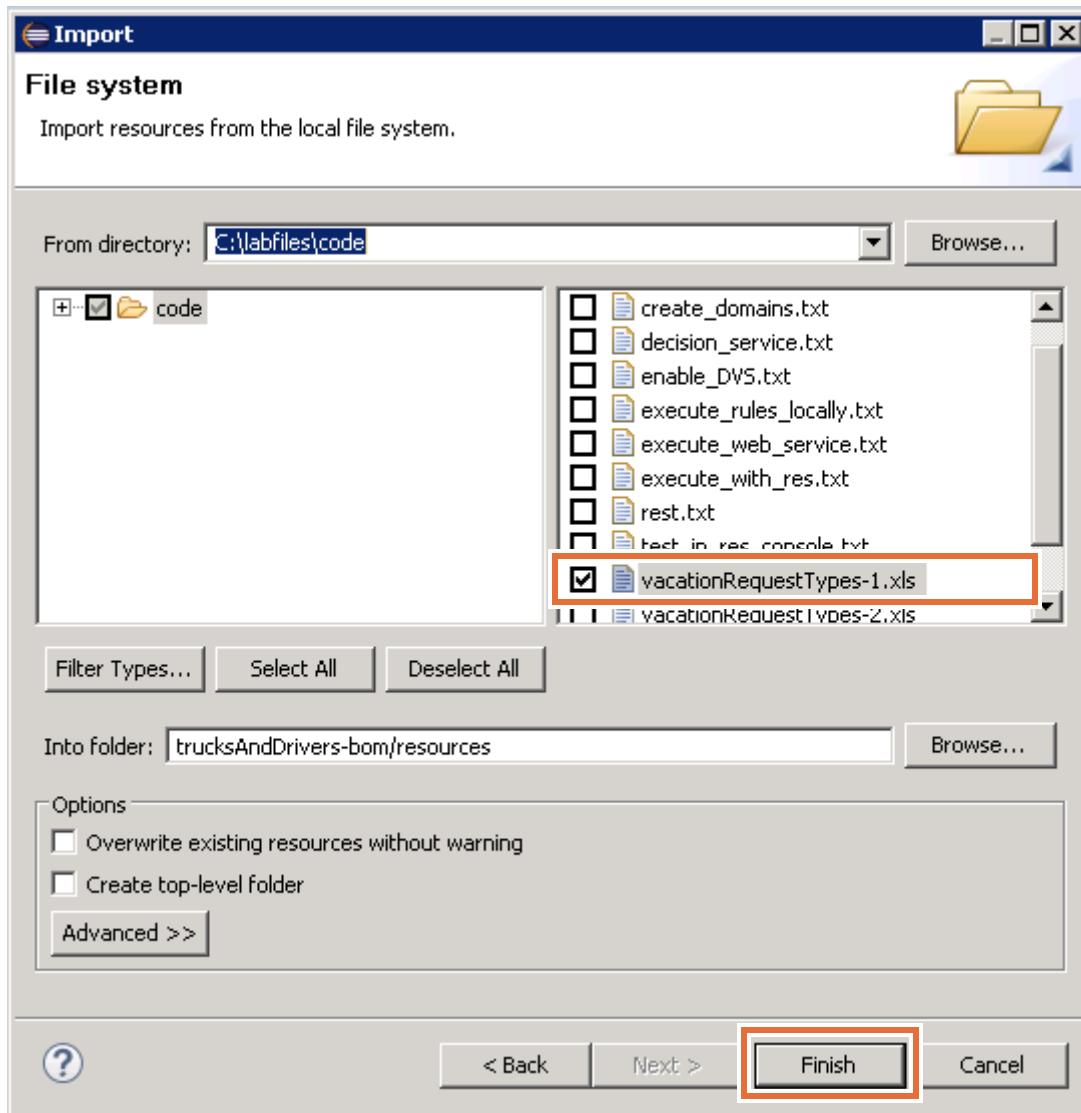
- \_\_\_ 2. In Rule Designer, import the Excel domain file into the `resources` folder of your BOM project.
- \_\_\_ a. Right-click the `resources` folder of the `trucksAndDrivers-bom` project, and click **Import**.



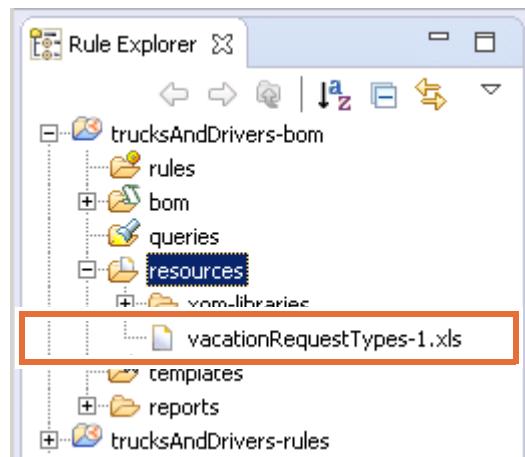
The Import window opens.

- \_\_\_ b. In the Select pane of the Import window, select **General > File System**, and click **Next**.
- \_\_\_ c. In the File System pane, click **Browse**, which is next to the **From directory** field, and select the `<LabfilesDir>\code` folder.

- \_\_ d. Select **vacationRequestTypes-1.xls** and click **Finish**.



The `vacationRequestTypes-1.xls` file is now visible under the `resources` folder of the `trucksAndDrivers-bom` project.



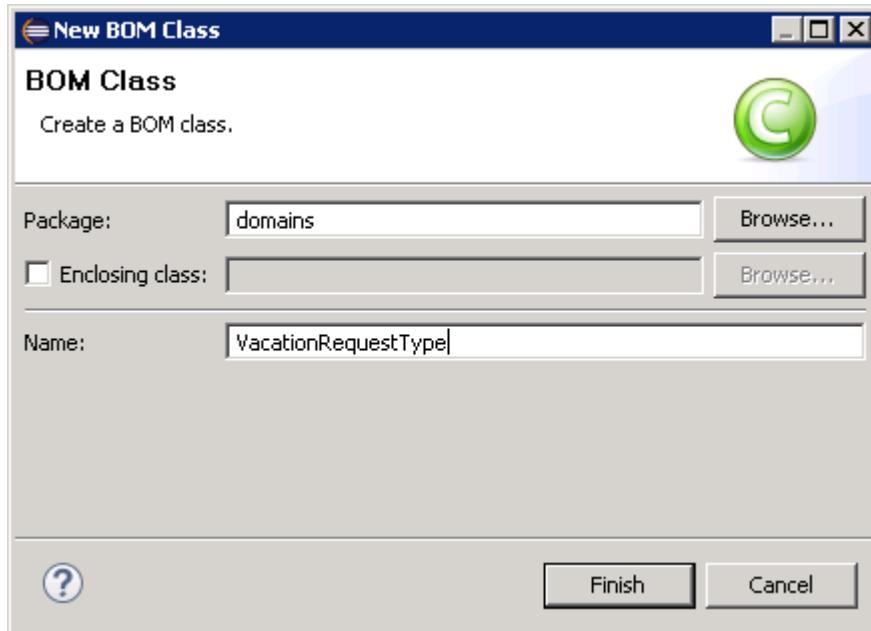
- \_\_\_ e. In your workspace, rename the `vacationRequestTypes-1.xls` file to: `vacationRequestTypes.xls`


**Hint**

To rename a file in Rule Designer, click the file and press F2 to open the Rename Resource window. Or, you can right-click the file, and click **Refactor > Rename**.

In the Rename Resource window, you type a value in the **New name** field and click **OK** to close the window and rename the file.

- \_\_\_ 3. Expand **trucksAndDrivers-bom > bom > model** and double-click **domains** to open the **domains** package in the BOM editor.
- \_\_\_ 4. Create a BOM class called `VacationRequestType` in the **domains** package.
- \_\_\_ a. In the **Business Object Model Entry: model** section, make sure that **domains** is selected.
  - \_\_\_ b. Click **New Class**.
  - \_\_\_ c. In the New BOM Class window, in the **Name** field, enter: `VacationRequestType`

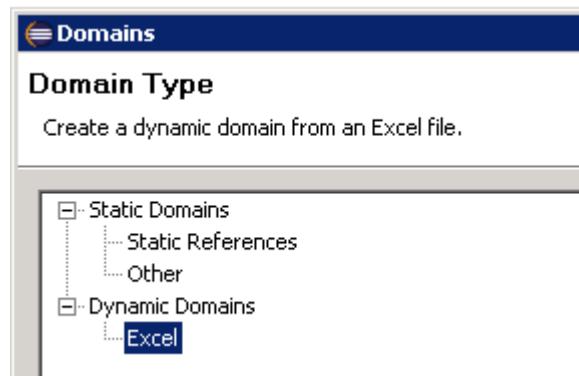


- \_\_\_ d. Click **Finish**.

- \_\_\_ 5. Edit the new `VacationRequestType` BOM class in the BOM editor to associate a dynamic domain with that class.
- \_\_\_ a. In the **Business Object Model Entry** section, make sure that **VacationRequestType** is selected, and click **Edit**.

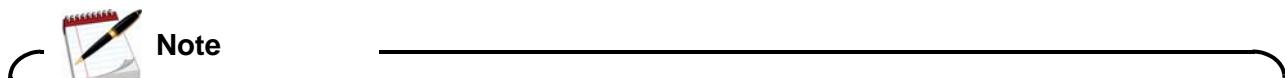


- \_\_\_ b. In the **Domain** section, click **Create a domain**.  
The Domains window opens.
- \_\_\_ c. In the Domains window, go to the **Dynamic Domains** section, select **Excel**, and click **Next**.



The Excel pane of the Domains window opens.

- \_\_\_ d. In the Excel pane, set the **Excel file** field to `vacationRequestTypes.xls`, which is the only available choice in the list.



This choice is available in this window because the `vacationRequestTypes.xls` file is present in the `resources` folder of the BOM project.

The `vacationRequestTypes.xls` file contains only the `vacationRequestTypes` sheet. Rule Designer automatically sets the value of the **Sheet** field to the name of that unique sheet.

- \_\_\_ e. Select **Table with header** to indicate that the first row in the `vacationRequestTypes` sheet is the header for the columns in that sheet.

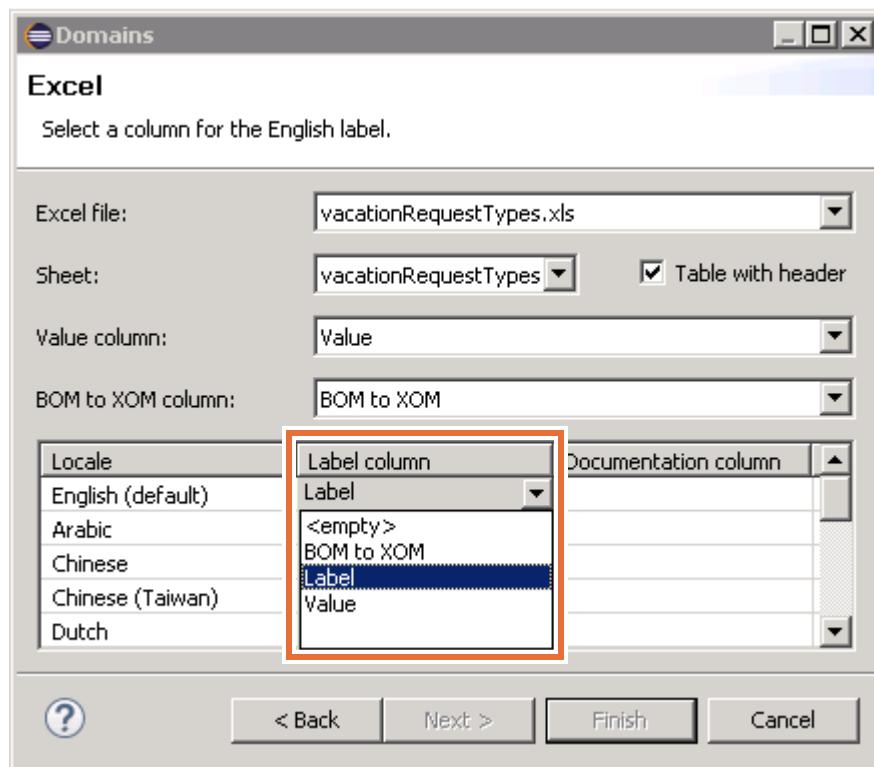


## Information

It is a good practice to set a header row in the Excel spreadsheet for dynamic domains. This header helps distinguish the purpose of each column.

Also, Rule Designer uses the header values as indications for you to select the appropriate columns when you configure the dynamic domain, as you see next.

- \_\_\_ f. In the **Value column** field, select **Value**, which is the header name for the column that gives the values of the dynamic domain in the spreadsheet.
- \_\_\_ g. In the **BOM to XOM column** field, select **BOM to XOM**, which is the header name for the column that provides the BOM-to-XOM mapping in the spreadsheet.
- \_\_\_ h. In the table, click the cell at the intersection of the row **English (default)** and the column **Label column**, and select **Label** in the list.

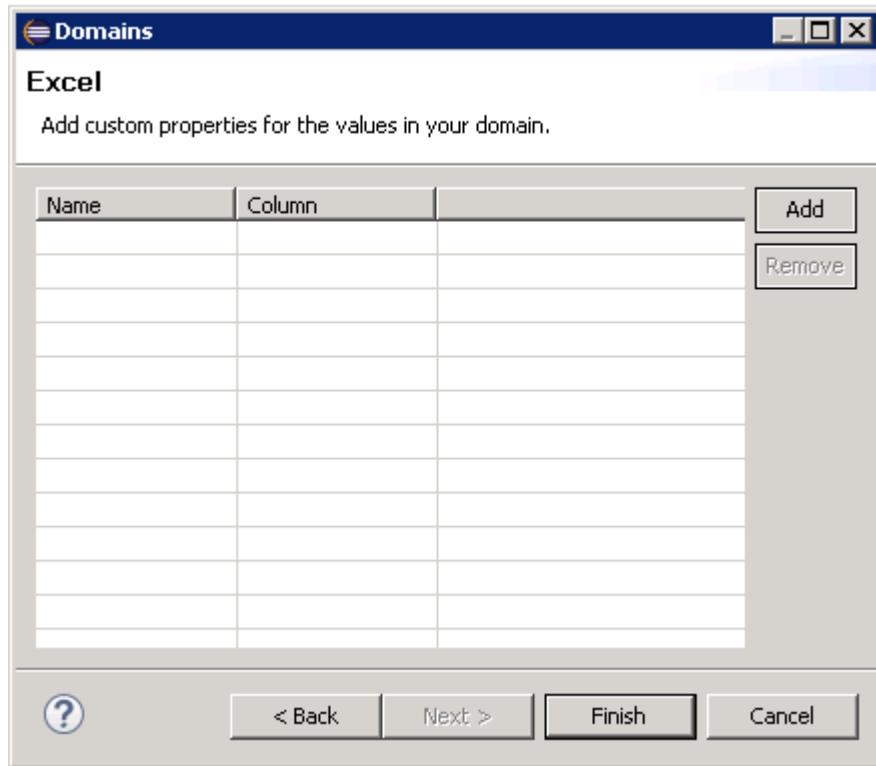


**Label** is the header name for the column that gives the labels of the dynamic domain in the spreadsheet.

The `vacationRequestTypes.xls` file does not define a documentation column, so you can leave the cells of the **Documentation column** empty.

- \_\_\_ i. Click **Next**.

The next pane opens, where you can add custom properties. For this domain, you have no properties.



- \_\_\_ j. Click **Finish** to close the Domains window.
- \_\_\_ 6. Save the BOM.  
The dynamic domain is created and its values are available in Rule Designer. A link that is named **Synchronize with dynamic values** is now visible in the **Domain** section of the Class page.

However, the Problems view indicates an error because the XOM does not include a

**▼ Domain**

Create and edit a domain for this class.

 [Edit](#) the domain.

 [Remove](#) the domain.

**Domain type: Excel**

 [Synchronize](#) with dynamic values.

- Administrative
- Compensatory
- Educational
- FamilyPersonal
- JuryDuty
- Military
- Overtime
- Pregnancy
- Recognition
- Sick
- Strike
- Vacation
- VolunteerFireAndRescue

`domains.VacationRequestType` class to translate your new BOM class.



**Questions**

How can you resolve this problem?

**Answer**

The `domains.VacationRequestType` class in the XOM does not exist. To resolve this problem, set the **Execution name** in the BOM-to-XOM mapping of the `domains.VacationRequestType` BOM class to an existing class.



**Questions**

Can you figure out which class you must use to set the **Execution name** in the BOM-to-XOM mapping?

**Hint**

Look at the content of the cells in the **BOM to XOM** column in the `vacationRequestTypes.xls` file.

**Answer**

The class that you require depends on the type for the values that the BOM-to-XOM mapping returns. In the `vacationRequestTypes.xls` file, the **BOM to XOM** column defines a BOM-to-XOM mapping. For example, the mapping for Jury Duty is:

```
return "V78";
```

The execution values associated with the values in the BOM dynamic domain are `String` objects, such as `"V78"`. The **Execution name** in the BOM-to-XOM mapping must be:

```
java.lang.String
```

- \_\_\_ 7. To correct the problem in the `VacationRequestType` BOM class, define the BOM-to-XOM mapping.
  - \_\_\_ a. Scroll down to the **BOM to XOM Mapping** section and expand this section.
  - \_\_\_ b. Next to the **Execution name** field, click **Browse**.
  - \_\_\_ c. In the **Choose a type** field, type `String`
  - \_\_\_ d. Select `String` from the list, and click **OK**. The type field is now set to:  
`java.lang.String`

**▼ BOM to XOM Mapping**  
Edit the mapping between this BOM class and the XOM.

|                 |                               |
|-----------------|-------------------------------|
| Execution name: | <code>java.lang.String</code> |
| Extender name:  |                               |

- \_\_\_ 8. Save the BOM.
- \_\_\_ 9. Verify that the Problems view no longer indicates any errors on this new class.

### 1.3. Examining the dynamic domain in Rule Designer

In this section, you continue working in the BOM editor to examine your new dynamic domain.

- \_\_\_ 1. In the **Custom Properties** section of the `VacationRequestType` BOM class, verify that two custom properties were defined.

**Custom Properties**

Define custom properties for this class.

| Name                    | Value                                    |        |
|-------------------------|------------------------------------------|--------|
| domainProviderResource  | vacationRequestTypes.xls                 | Add    |
| domainValueProviderName | com.ibm.rules.domainProvider.msexcel2003 | Remove |
|                         |                                          |        |
|                         |                                          |        |
|                         |                                          |        |

- The `domainProviderResource` property gives the name of the Excel spreadsheet that is used as the source for the values of the dynamic domain.

You must modify this value when you want to change the source Excel spreadsheet for your domain.

- The `domainValueProviderName` property gives the name of the classes that are used to manipulate this spreadsheet and transform its values into the dynamic domain.

- \_\_\_ 2. In the **Members** section, verify that you can see all the values that are defined in the `vacationRequestTypes.xls` file.

- \_\_\_ a. Double-click the **VolunteerFireAndRescue** BOM member to open it.
- \_\_\_ b. Verify that this BOM member is both static and final.

**General Information**

|                                                                                                              |                             |           |
|--------------------------------------------------------------------------------------------------------------|-----------------------------|-----------|
| Name:                                                                                                        | VolunteerFireAndRescue      |           |
| Type:                                                                                                        | domains.VacationRequestType | Browse... |
| Class:                                                                                                       | domains.VacationRequestType | Browse... |
| <input type="radio"/> Read/Write <input checked="" type="radio"/> Read Only <input type="radio"/> Write Only |                             |           |
| <input checked="" type="checkbox"/> Static <input checked="" type="checkbox"/> Final                         |                             |           |
| <input type="checkbox"/> Deprecated <input type="checkbox"/> Update object state                             |                             |           |
| <input type="checkbox"/> Ignore for DVS                                                                      |                             |           |

- \_\_\_ c. Verify that the **Type** of this BOM member is `domains.VacationRequestType`.
- \_\_\_ d. Verify that the verbalization of this BOM member is **Volunteer Fire and Rescue** as shown in the Microsoft Excel file.

**Member Verbalization**

~~Remove the verbalization.~~

|        |                           |  |
|--------|---------------------------|--|
| Label: | Volunteer Fire and Rescue |  |
|--------|---------------------------|--|

- \_\_\_ e. Expand the **BOM to XOM Mapping** section and verify that the getter method of this BOM member is as shown in the Microsoft Excel file:

```
return "K29";
```

▼ **BOM to XOM Mapping**

Edit the mapping between this BOM member and the XOM.

 [Edit the imports.](#)

▼ **Getter**

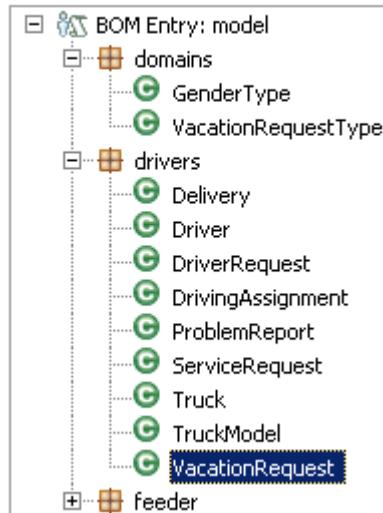
```
return "K29";
```

## Section 2. Using the dynamic domain in a rule

Now that the dynamic domain exists in the BOM of the rule project, use it to complement the `VacationRequest` BOM class per the request from the human resources department. You then author rules that are based on this type.

- 1. First, you update the `VacationRequest` BOM class by adding a `type` BOM attribute that business users can read and that is based on the dynamic domain.
  - a. Return to the Package page of the BOM editor and double-click the `drivers.VacationRequest` BOM class to open it on the Class page.

**Business Object Model Entry**

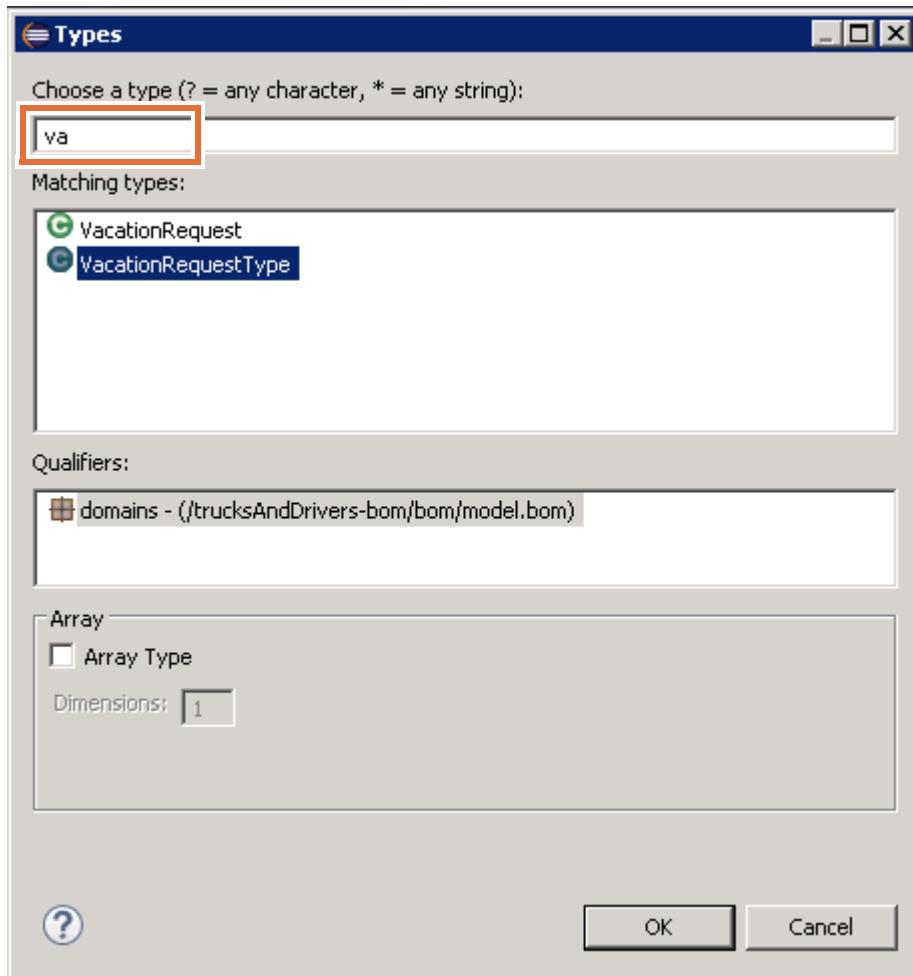


- b. In the **Members** section, add a BOM attribute named: `type`
- c. Set the **Type** field to: `domains.VacationRequestType`

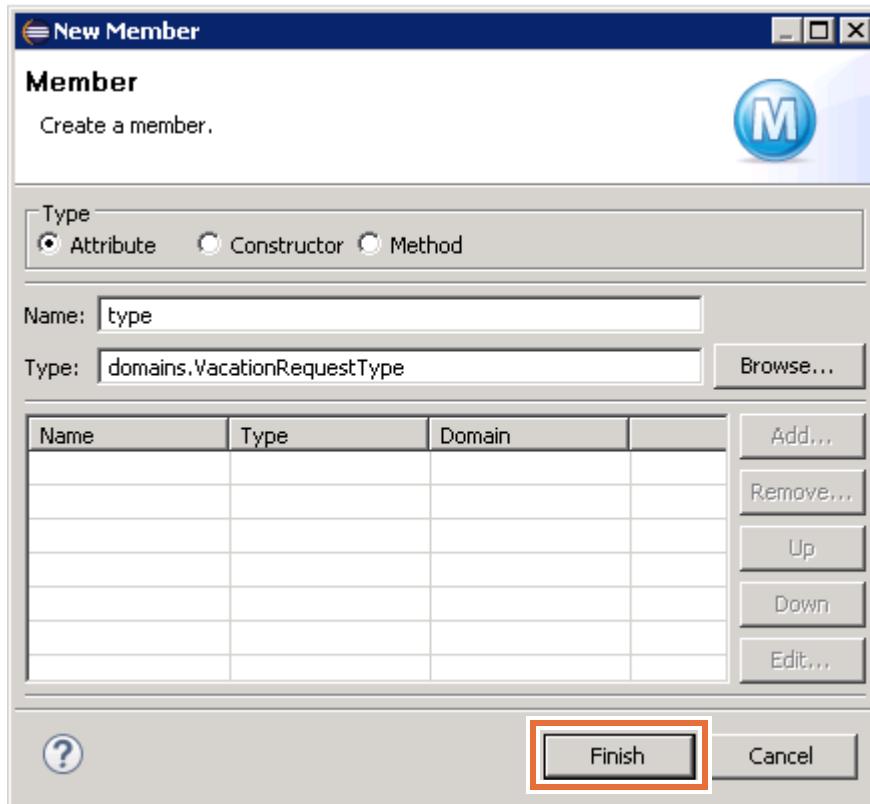


## Hint

If you click **Browse** to find the type, you can filter the list of options by starting to type the class name that you are looking for.



- \_\_ d. Click **Finish**.



- \_\_ 2. Set **type** to **Read Only** and create a default verbalization for it.
- From the Members list, double-click **type** to open the Member page.
  - In the **General Information** section, select **Read Only**.
  - Create a default verbalization for **type**.

#### Member type (class: drivers.VacationRequest)

|                                                                                                                     |                             |                                                                                             |
|---------------------------------------------------------------------------------------------------------------------|-----------------------------|---------------------------------------------------------------------------------------------|
| <b>General Information</b>                                                                                          |                             | <b>Member Verbalization</b>                                                                 |
| Name:                                                                                                               | type                        | <span style="color: orange;">⚠ This member is not verbalized. <a href="#">Create</a></span> |
| Type:                                                                                                               | domains.VacationRequestType | <input type="button" value="Browse..."/>                                                    |
| Class:                                                                                                              | drivers.VacationRequest     | <input type="button" value="Browse..."/>                                                    |
| <input type="radio"/> Read/Write <input checked="" type="radio"/> <b>Read Only</b> <input type="radio"/> Write Only |                             |                                                                                             |

- \_\_ 3. Save the BOM.



#### Questions

Are you done preparing this attribute?

## Answer

Your new `type` attribute for the `VacationRequest` BOM class is not associated with any attribute in the XOM. You cannot execute this business rule solution without an error at run time.

The Problems view shows this error:

B2X cannot find attribute "type" in execution class "drivers.VacationRequest"

You solve this problem later in this exercise, in "Updating the XOM" on page 10-24. For now, ignore the error and try to use the new dynamic domain in a rule.

4. Now, write an action rule that uses this new `type` BOM attribute in Rule Designer.
- a. Create an action rule that is named `05_dynamic_domain` in the `domains` rule package of the `trucksAndDrivers-rules` project, with the following code:
- ```
definitions
    set 'the driver' to a driver;
    set 'a vacation request' to a vacation request in the vacation requests
        of 'the driver';
if
    the type of 'a vacation request' is one of { Administrative,
        Compensatory, Educational }
then
    print "The driver " + the name of 'the driver' + " requested vacation
        for Administrative, Compensatory, or Educational reason";
```

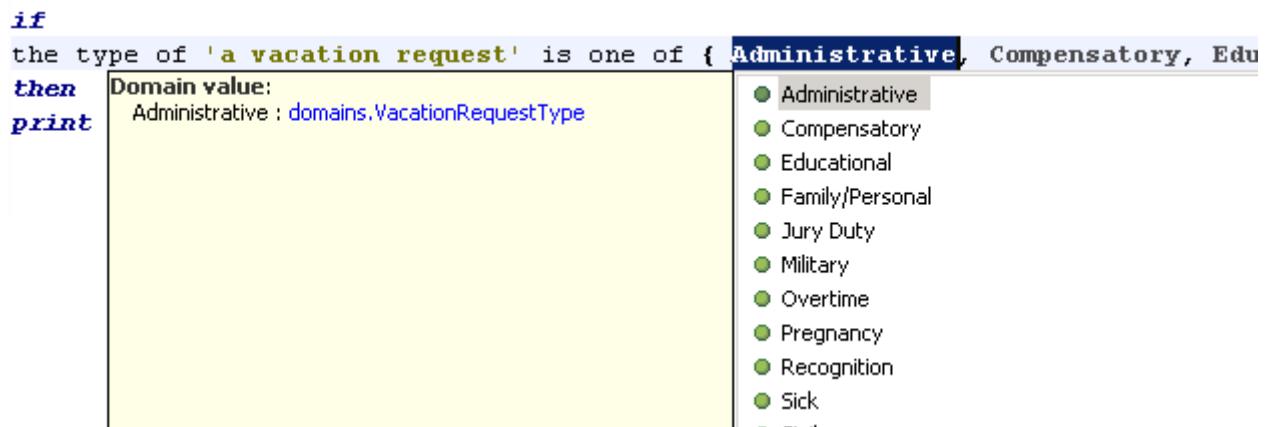


Hint

You can find the code for this rule in the `create_domains.txt` file in the `<LabfilesDir>\code` directory.

- ___ b. Notice how the rule editor shows the verbalization of the values in the dynamic domain, while you edit one of { }.

```
if  
the type of 'a vacation request' is one of { Administrative, Compensatory, Edu  
then Domain value:  
print Administrative : domains.VacationRequestType
```



A screenshot of a software interface showing a code editor and a dropdown menu. The code editor contains a simple rule definition:

```
if  
the type of 'a vacation request' is one of { Administrative, Compensatory, Edu  
then Domain value:  
print Administrative : domains.VacationRequestType
```

The word "Administrative" is selected in the dropdown menu, which lists ten categories: Administrative, Compensatory, Educational, Family/Personal, Jury Duty, Military, Overtime, Pregnancy, Recognition, and Sick. The "..." option is also visible at the bottom of the list.

- ___ 5. Save the rule.

Section 3. Updating the dynamic domain



Requirements

The human resources department indicates that the Compensatory value is no longer valid and must be replaced with the Other Vacation value. You must update the dynamic domain by modifying the Excel file and dynamically updating the dynamic domain of the BOM in Rule Designer. You also verify the consequences of the update.

Modifying the source Excel spreadsheet and update the dynamic domain

- 1. In Rule Designer, import the `<LabfilesDir>\code\vacationRequestTypes-2.xls` file into the resources folder of the trucksAndDrivers-bom project.
- 2. Open the `vacationRequestTypes-2.xls` file, and verify that its content is the same as the content of the `vacationRequestTypes.xls` file, except for the line:

Value: Compensatory

BOM to XOM: return "B34";

Label: Compensatory

This line is no longer present, and is replaced with a line that defines the Other value, with the following three cells:

Value: Other

BOM to XOM: return "O22";

Label: Other Vacation

- 3. Close the `vacationRequestTypes.xls` file, and in Rule Designer, rename this file back to: `vacationRequestTypes-1.xls`
- 4. Close the `vacationRequestTypes-2.xls` file, and in Rule Designer, rename this file to: `vacationRequestTypes.xls`

When you rename the new Excel spreadsheet, Rule Designer uses it as the source for the values in the dynamic domain.

5. In the BOM editor, open the `domains.VacationRequestType` BOM class, and click **Synchronize with dynamic values** in the **Domain** section.

Domain type: Excel

 [Synchronize](#) with dynamic values.

- Administrative
- Compensatory
- Educational
- FamilyPersonal
- JuryDuty
- Military
- Overtime
- Pregnancy
- Recognition
- Sick
- Strike
- Vacation
- VolunteerFireAndRescue

The removed `Compensatory` value is still visible, both under **Synchronize with dynamic values** and in the **Members** section. However, this value is marked as *deprecated* in the BOM editor (as you verify next).

The added `Other` value is also visible, both under **Synchronize with dynamic values** and in the **Members** section.



Important

The **Members** section lists all the static references that are, or were, part of this dynamic domain. The references that correspond to values that are no longer present in the dynamic domain are marked as *deprecated*.

Deprecated values are not removed and might still be used to author rule artifacts. However, because they are marked as deprecated, these values are underlined in the edited rule artifacts that use them, and associated warnings are added in the Problems view of Rule Designer.

6. Save the BOM.
7. Verify that the `Compensatory` value is deprecated.
- a. In the Problems view, verify that the following warning messages exist:
- Member '`domains.VacationRequestType.Compensatory`' is deprecated.
 - [BOM] GBRMO0011W: Member `domains.VacationRequestType.Compensatory` is deprecated.
 - '`Compensatory`' is deprecated.
- b. In the **Members** section of the `domains.VacationRequestType` BOM class, double-click the `Compensatory` member.

- ___ c. On the Member page for the `Compensatory` BOM member, verify that the **Deprecated** check box is selected.

After modifying the values of the dynamic domain, you must verify that the rule artifacts based on this dynamic domain are still correct. In the present case, you must verify only the `05_dynamic_domain` rule.

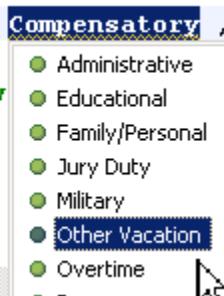
- ___ 8. Reopen the `05_dynamic_domain` rule in the `domains` rule package of the `trucksAndDrivers-rules` project.
- ___ a. Verify that the term `Compensatory` is underlined, and that the associated warning is the same as in the Problems view:
- 'Compensatory' is deprecated
- ___ b. In the **if** and **then** parts of the rule, replace `Compensatory` with `Other Vacation`.

if

```
the type of 'a vacation request' is one of { Administrative , Compensatory ,
```

then

```
print "The driver " + the name of 'the driver' + " requested v
```

**Note**

In the action statement, this term is part of a string.

You obtain the following rule:

```
definitions
  set 'the driver' to a driver ;
  set 'a vacation request' to a vacation request in the vacation requests of
  'the driver' ;
if
  the type of 'a vacation request' is one of { Administrative, Other
  Vacation, Educational }
then
  print "The driver " + the name of 'the driver' + " requested vacation for
  Administrative, Other Vacation, or Educational reason";
```

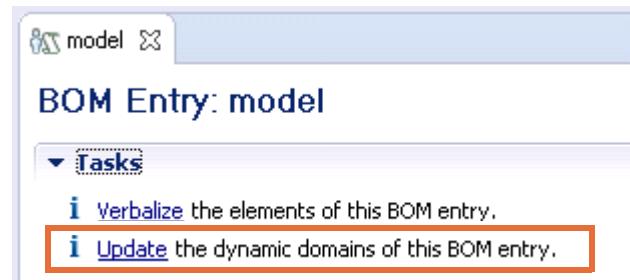
- ___ 9. Save the `05_dynamic_domain` rule.
- ___ 10. Verify that the '`Compensatory`' is deprecated warning is no longer present in the Problems view.
- ___ 11. Close the rule.



Information

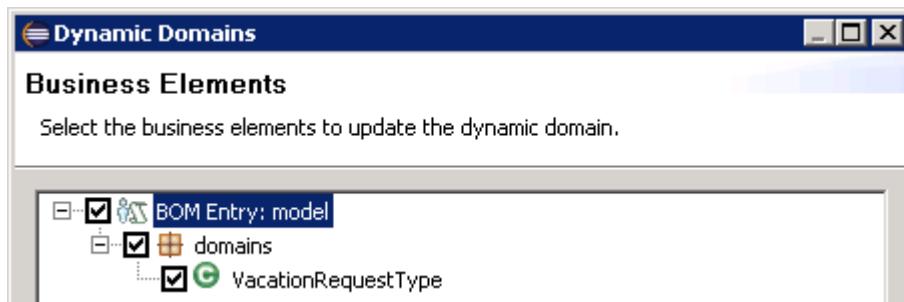
If you define multiple dynamic domains in a BOM entry, you can update or repopulate the values of all of them from the Package page of the BOM editor.

1. On the Package page of the BOM editor, click **Update the dynamic domains of this BOM entry** in the **Tasks** section.



The Dynamic Domains window opens.

2. In the Dynamic Domains window, select the check boxes that are associated with the dynamic domains you want to update.



3. Click **Finish** and save your work.

The Dynamic Domains window closes, and Rule Designer updates the selected dynamic domains.

Section 4. Updating the XOM

You must now make sure that the complete business rule solution is executable by updating its XOM to reflect the additions in the BOM.

First, you identify the inconsistency in the rule project.

- ___ 1. Reopen the Problems view (if it is closed) by clicking **Window > Show View > Other > Problems**.
- ___ 2. Verify that the Problems view contains the following message:

```
[B2X] Cannot find attribute "type" in execution class  
"drivers.VacationRequest"
```

This message indicates that the `type` attribute cannot be found in the XOM class that is associated with the `drivers.VacationRequest` BOM class.

- ___ 3. If you cannot see this error message in the Problems view, make sure that the BOM is selected against the XOM while you build the project, and rebuild the project.
 - ___ a. Go to **Window > Preferences**, and then go to **Rule Designer > Build**.
 - ___ b. On the Build page, make sure that the **Perform BOM to XOM checks during build** check box is selected, and then click **OK**.
 - ___ c. Click **Project > Clean** to clean and rebuild all projects in your workspace.

Correcting the inconsistency problem

- ___ 1. Expand `trucksAndDrivers-xom > src > drivers` and open the `VacationRequest` class to add a private attribute:

- **Name:** `type`
- **Type:** `String`

- ___ 2. Create the corresponding setter and getter methods.

```
public void setType(String type) {  
    this.type = type;  
}  
  
public String getType() {  
    return type;  
}
```

- ___ 3. Save the XOM.

The error that is related to this inconsistency in the BOM-to-XOM mapping is no longer visible in the Problems view.

- ___ 4. Close the `VacationRequest.java` file.

Section 5. Publishing the BOM and rule projects to Decision Center

In this section, you publish the `trucksAndDrivers-bom` project and the `trucksAndDrivers-rules` project to Decision Center to create the corresponding projects in Decision Center.

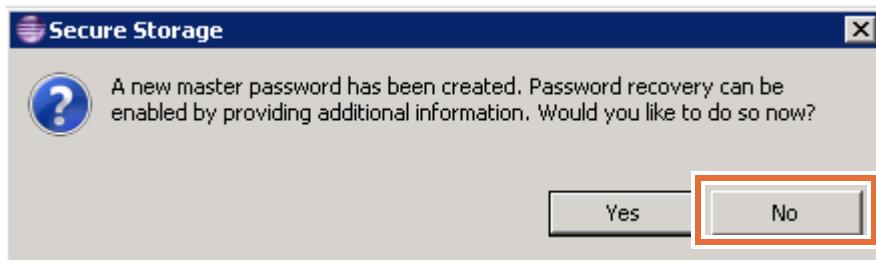
5.1. Publishing the decision service

- 1. If the sample server is not started, start it now by using the **Start server** desktop shortcut or by clicking **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Start server**.
Starting the server might take several minutes.
- 2. Publish `trucksAndDrivers-tests` as an ungoverned decision service.
 - a. In the Rule Explorer, right-click `trucksAndDrivers-tests` and click **Decision Center > Connect**.
The Decision Center configuration wizard opens.
 - b. Enter the connection entries:
 - **URL:** `http://localhost:9080/teamserver`
 - **User name:** `rtsAdmin`
 - **Password:** `rtsAdmin`
 - c. Click **Connect** and when the connection is established, click **Next**.
 - d. In the Synchronization Settings window, click **Next** (do not select the **Use Decision Governance** check box).
 - e. On the Decision Service Dependent Projects page, make sure that `trucksAndDrivers-bom` and `trucksAndDrivers-rules` are selected, and click **Finish**.



Information

If you see a Secure Storage window, click **No**.



- f. When synchronization completes, no changes are found, so you can click **OK** to close the Synchronize Complete window.
- g. Click **No** when prompted to switch to the Team Synchronizing perspective.

You do not have to open the Team Synchronizing perspective because it is empty.



Important

After you publish the `trucksAndDrivers-rules` project to Decision Center, do not disconnect. By keeping Rule Designer and Decision Center connected, you do not have to establish this connection again in the next steps.

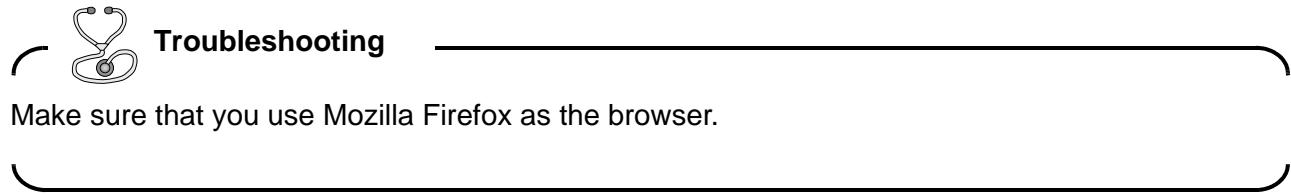
Section 6. Examining rules in Decision Center

In this section, acting as a business user in Decision Center, you work with the dynamic domain and the rule artifacts that you published from Rule Designer.

6.1. Opening the published projects in Decision Center

- __ 1. Sign in to the Decision Center Enterprise console as an administrator.
- __ a. Start Mozilla Firefox and type the following URL into a browser:

`http://localhost:9080/teamserver`



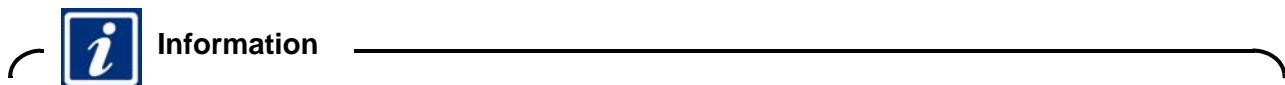
- __ b. Sign in by entering `rtsAdmin` in both the **Username** field and the **Password** field.
- __ 2. On the **Home** tab, switch to the `trucksAndDrivers-rules` decision service.
- __ a. Make sure that **Work on a decision service** is selected.
- __ b. In the **Decision service in use** field, select `trucksAndDrivers-tests`.

The screenshot shows the IBM Decision Center interface. At the top, there's a navigation bar with tabs: Home, Explore, Compose, Query, and Analyze. Below the navigation bar, a welcome message reads "Welcome to the Decision Center Home Page". There are two main sections of configuration controls:

- The first section is for a "rule project" and includes fields for "Project in use" (set to "loanvalidation-rules"), "Branch in use" (set to "<none>"), and "Current action" (set to "Work on branch").
- The second section, which is highlighted with a red box, is for a "decision service" and includes fields for "Decision service in use" (set to "trucksAndDrivers-tests"), "Branch in use" (set to "main"), and "Current action" (set to "Work on branch").

- __ 3. Click the **Explore** tab.

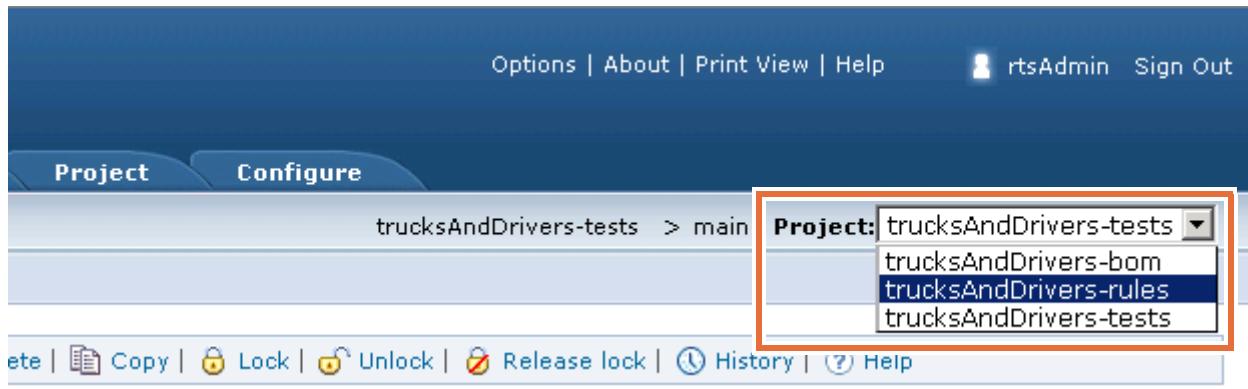
- ___ 4. Switch to the trucksAndDrivers-rules project.



When you select a decision service in the Enterprise console, the default view is the main rule project.

In this case, the trucksAndDrivers-test project is the main rule project, and it is the first project that you see on the **Explore** tab. To access the rule artifacts in the trucksAndDrivers-rules project, you must switch projects.

- ___ a. On the **Explore** tab, click the **Project** menu in the upper-right corner.

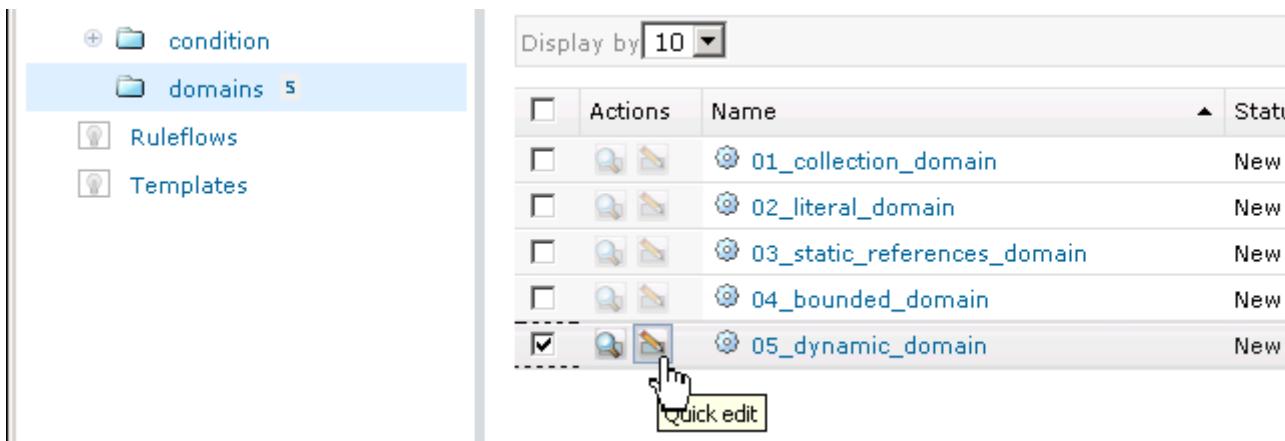


- ___ b. Select **trucksAndDrivers-rules**.

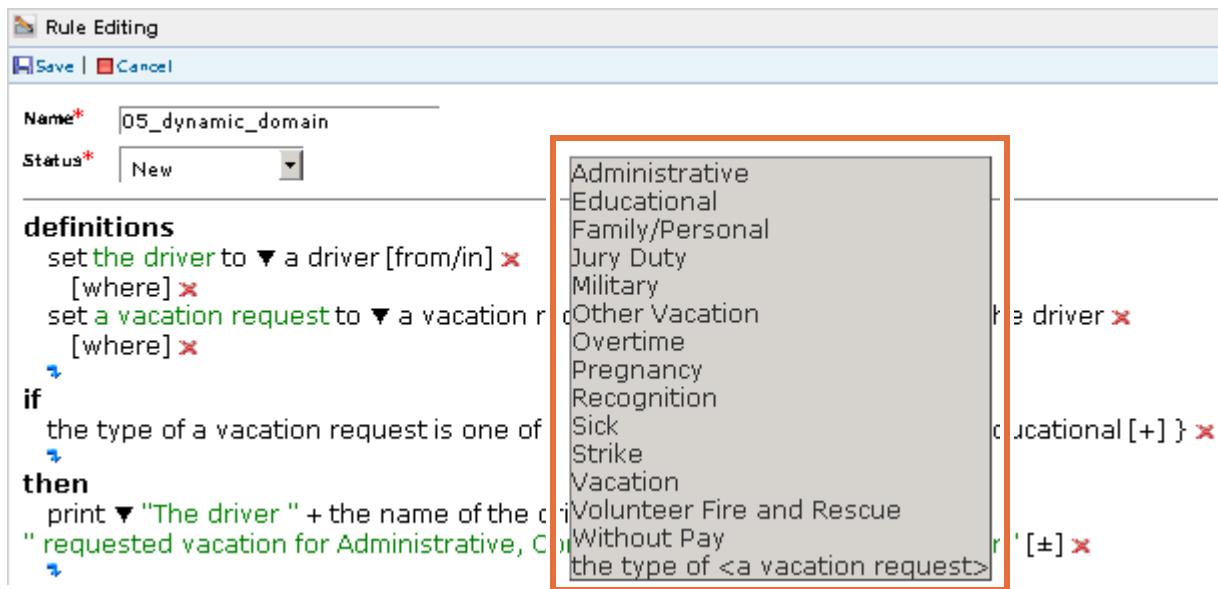
The Smart Folders pane on the left side of the window now shows the folders and rule artifacts in the trucksAndDrivers-rules project.

- ___ 5. Open the 05_dynamic_domain action rule in the Quick Edit editor.

- ___ a. In the Smart Folders pane, click **Business Rules > domains**.
 ___ b. Select the **05_dynamic_domain** action rule and click the **Quick Edit** icon to open the rule in the editor.



6. In the **if** part of the rule, click one of the vacation request values, such as **Administrative**, to see the list of domain values.



7. Click **Cancel** to exit the edit mode.
 8. On the **Explore** tab, click the **Project** menu and switch to the `trucksAndDrivers-bom` project.



Questions

Do you see the `vacationRequestType.xls` file that you added to the `trucksAndDrivers-bom` project in Rule Designer? If not, how can you access it?

Answer

The `vacationRequestType.xls` file is stored in the `resources` folder of the `trucksAndDrivers-bom` project.

However, Decision Center does not have a default smart folder that lists the content of the `resources` folder. To see the `vacationRequestType.xls` file, you must create a smart folder that lists the content of the `resources` folder.

**Information**

The resources folder of a rule project is not automatically accessible in Decision Center when you publish the project from Rule Designer. This folder is used mainly for technical data that must be shared between Decision Center and Rule Designer. Hiding this folder in Decision Center can help prevent non-technical business users from modifying the data by mistake.

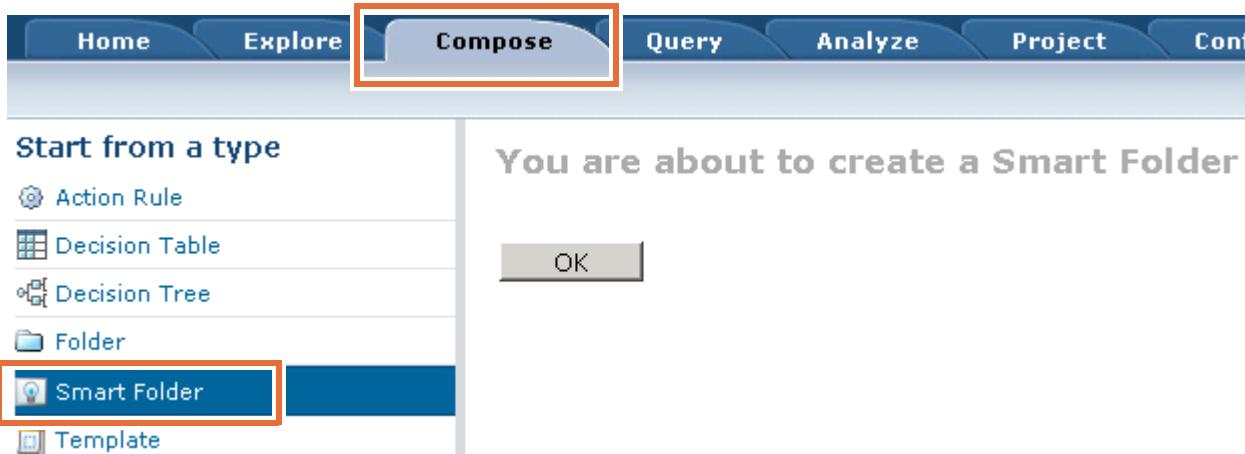
6.2. Creating a Resources smart folder

In this section, you learn how to create a smart folder that lists the content of the resources folder.

- 1. Check the **Projects** menu to make sure that you are in the trucksAndDrivers-bom project.

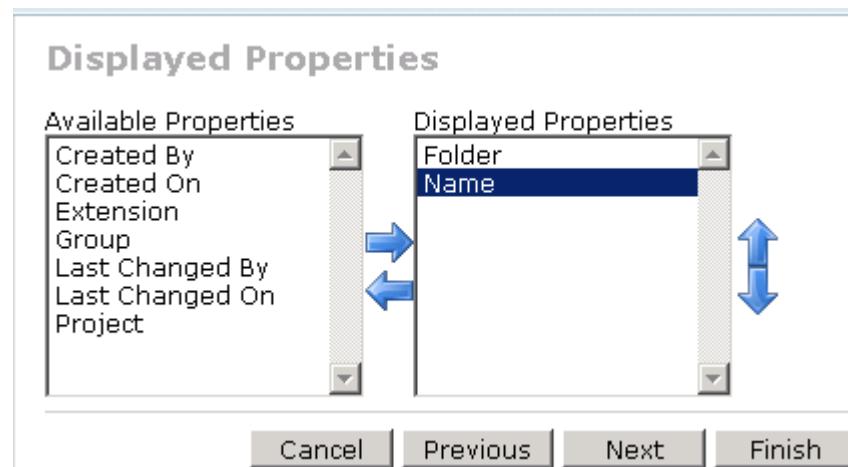


- 2. Click the **Compose** tab.
- 3. Select **Smart Folder** in the **Start from a type** list, and click **OK**.



- 4. On the Properties page, enter **Resources** in the **Name** field and click **Next**.
- 5. On the Query page, write the query to define this smart folder.
 - a. In the **Find** statement, click **all business rules** and select **all resources**.
 - b. Click **Next**.
- 6. On the Displayed Properties page, set up the list of the elements that you want the smart folder to display.
 - a. In the **Available Properties** section, select **Folder** and click the right-arrow that goes from the **Available Properties** list on the left to the **Displayed Properties** list on the right.
 - b. Repeat the previous step to add the **Name** property to the **Displayed Properties** list.
 - c. Make sure that **Folder** is listed before **Name** in the **Displayed Properties** section.

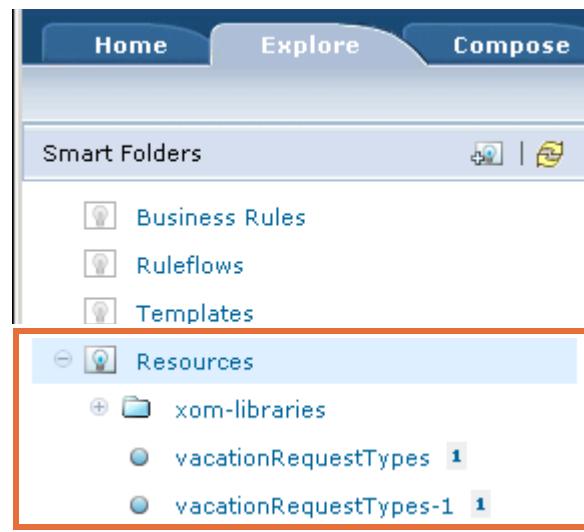
If required, use the  up-arrow to move the property.



- ___ d. Click **Next**.
- ___ e. In the Documentation pane, enter a description for this folder, such as:
Smart folder to list the contents of the resources folder
- ___ f. Click **Finish**.

The **Resources** smart folder is now listed on the **Explore** tab and should contain the two Excel spreadsheets that you imported in Rule Designer:

- vacationRequestTypes
- vacationRequestTypes-1



Section 7. Modifying the dynamic domain in Decision Center

You learned how to use Excel source files for dynamic domains in rules that are authored in Rule Designer. Next, you see how the domain can be modified and used in rules that are authored in Decision Center.

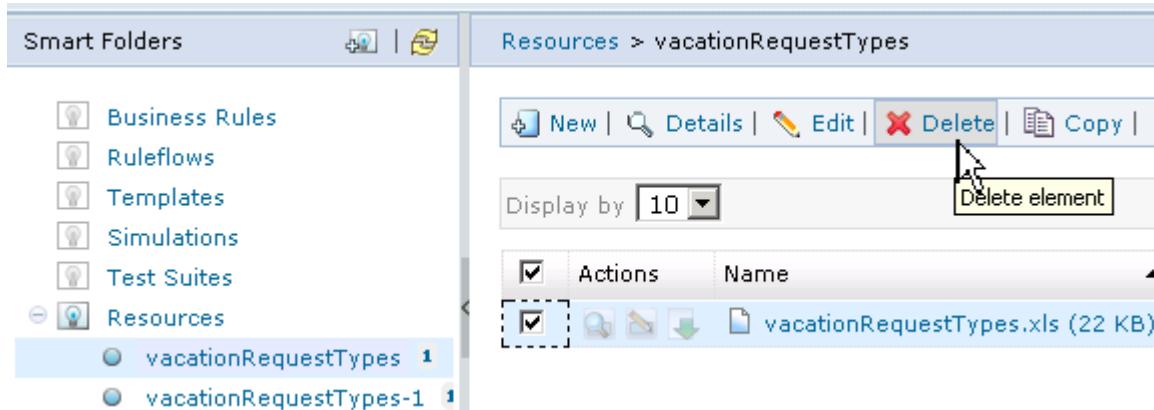


Note

If you are using the VMware image that was developed for this course, you cannot modify Excel files. Instead, the modified domain values are provided in the `vacationRequestTypes-3.xls` file in the `<LabfilesDir>\code` directory. You delete the Excel file that is in use and replace it with: `vacationRequestTypes-3.xls`

7.1. Modifying the values of the dynamic domain

- 1. In the `<LabfilesDir>\code` directory, open the `vacationRequestTypes-3.xls` Excel file to view the modified domain values.
The first line, which defined the `Administrative` value, is now replaced with the `Training` value.
- 2. Make a backup of this file by renaming it as: `vacationRequestTypes-3-copy.xls`
- 3. Rename the `vacationRequestTypes-3-copy.xls` file as: `vacationRequestTypes.xls`
- 4. In Decision Center, delete the existing `vacationRequestTypes.xls` file in the `trucksAndDrivers-bom` project.
 - a. On the **Explore** tab, expand the **Resources** smart folder and click the `vacationRequestTypes.xls` file.
 - b. In the table of artifacts, select **vacationRequestTypes.xls**, and click **Delete** on the toolbar.



- c. When prompted to confirm the deletion, click **Yes**.
- 5. Import the new file to Decision Center.
 - a. On the **Explore** tab, click the **Resources** smart folder and click **New** on the toolbar.

- ___ b. On the Properties page, click **Browse** next to the **File** field, select the <LabfilesDir>\code\vacationRequestTypes.xls Excel spreadsheet, and click **Open**.
 - ___ c. Click **Finish**.
 - ___ d. Click the **Explore** tab and verify that the vacationRequestTypes.xls file is now visible in the **Resources** smart folder.
- ___ 6. Click the **Project** tab.
- ___ 7. On the **Project** tab, under **Business object model**, click **Update Dynamic Domains**.
The Dynamic Domains page opens.
- ___ 8. Select **domains.VacationRequestType**.

You can expand `domains.VacationRequestType` to see the current list of domain values.

- ___ 9. Click **Update** on the toolbar.
- ___ 10. Click **OK** after the domains are successfully reloaded.
- ___ 11. Expand **domains.VacationRequestType** again to see the updated list.

7.2. Modifying the domain value in the rule

In this section, you examine the consequences of your changes in the `trucksAndDrivers-rules` project.

- ___ 1. Click the **Explore** tab and use the **Project** menu to switch to the `trucksAndDrivers-rules` project.
- ___ 2. In the Rule Explorer, click the **Business rules > domains** folder.

3. Click the **Preview** icon for the `05_dynamic_domain` rule in the **domains** folder to confirm that the rule has a warning.

 Rule Preview

 Edit

Name 05_dynamic_domain
Status New

 'Administrative' is deprecated.

```
definitions
    set 'the driver' to a driver;
    set 'a vacation request' to a vacation request in the vacation requests of 'the driver';
if
    the type of 'a vacation request' is one of { Administrative , Other Vacation , Educational }
then
    print "The driver " + the name of 'the driver' +
    " requested vacation for Administrative, Other Vacation, or Educational reason" ;
```



Questions

Why is there a warning?

Answer

The `Administrative` value no longer exists in the dynamic domain, so it is marked as deprecated (similar to what occurs in Rule Designer).

4. Click **Edit** to edit the **if** and **then** parts of the `05_dynamic_domain` rule.

 Rule Preview

 Edit

- ___ 5. Replace Administrative with: Training

if

the type of a vacation request is one of { **Administrative**, Other Vacation , Educational [+]

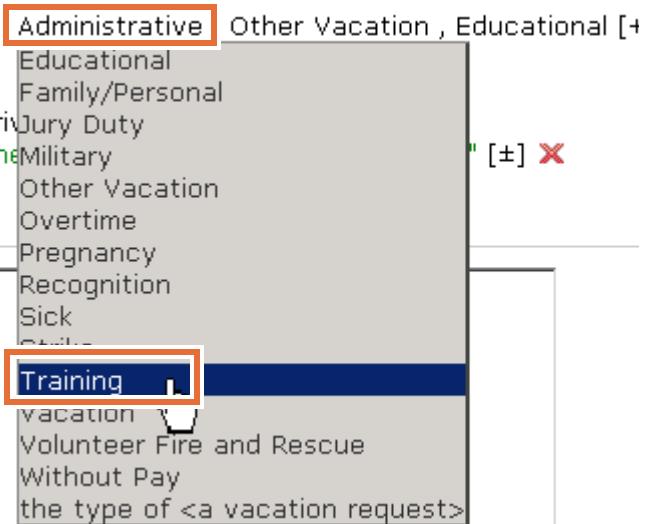
then

print "The driver " + the name of the driver

" requested vacation for **Administrative**, **Other Vacation** , **Educational**

[else]

Add a comment to this version



- ___ 6. The rule should state:

definitions

```
set 'the driver' to a driver ;
set 'a vacation request' to a vacation request in the vacation requests of
'the driver' ;
```

if

the type of 'a vacation request' is one of { Training, Other Vacation,
Educational }

then

print "The driver " + the name of 'the driver' + " requested vacation
for Training, Other Vacation or Educational reason";

- ___ 7. Save the rule 05_dynamic_domain and verify that it no longer has a warning.

- ___ 8. Sign out and close the Enterprise console.

Section 8. Updating Rule Designer from Decision Center

In Decision Center, you changed both the rule project and the BOM. You must now update the Rule Designer copy of the BOM and rule project by synchronizing with Decision Center.

8.1. Synchronizing the rule project

- 1. In Rule Explorer, synchronize the `trucksAndDrivers-tests` decision service with Decision Center.
 - a. Right-click the `trucksAndDrivers-tests` decision service, and click **Decision Center > Synchronize with Decision Center**.
 - b. In the Synchronization Settings window, click **Finish**.
- 2. When prompted to open the Team Synchronize perspective, click **Yes**.
- 3. In the Team Synchronize perspective, view the artifacts that must be synchronized.

Notice that both the **trucksAndDrivers-bom - main** and **trucksAndDrivers-rules - main** projects are listed.



Information

The **trucksAndDrivers-bom - main** project is listed because you added `Training` in the dynamic domain and used this value in the rule in Decision Center.

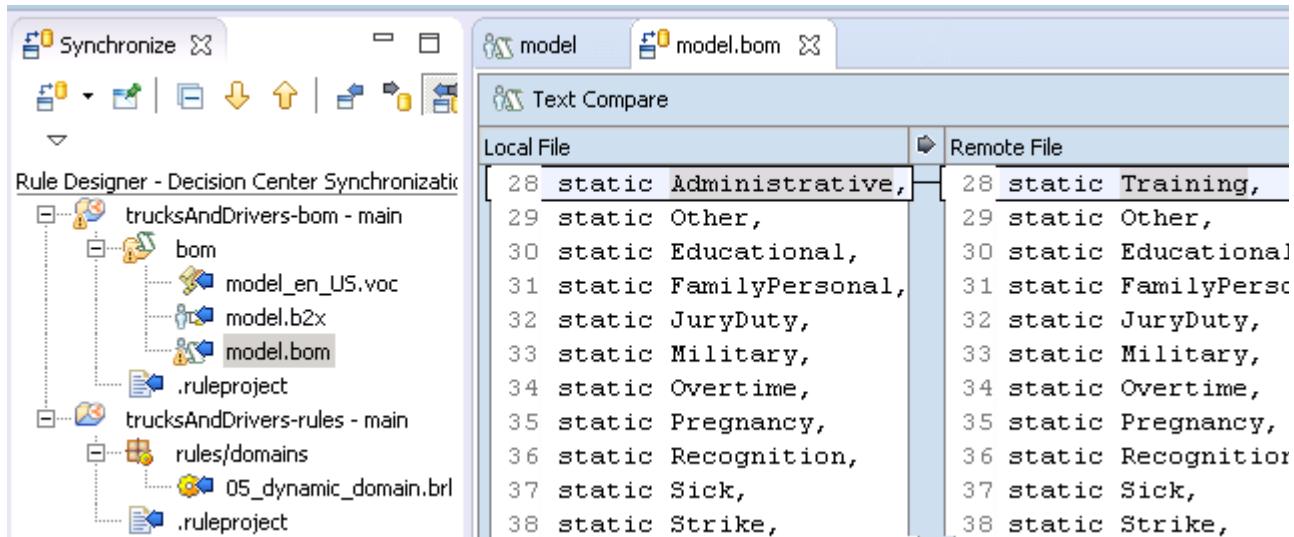
The **trucksAndDrivers-rule - main** project is listed because you added the `05_dynamic_domain` rule in the Enterprise console.

In both rule projects, the blue arrows that point left on the rule artifacts indicate that Decision Center has a more recent version than Rule Designer.

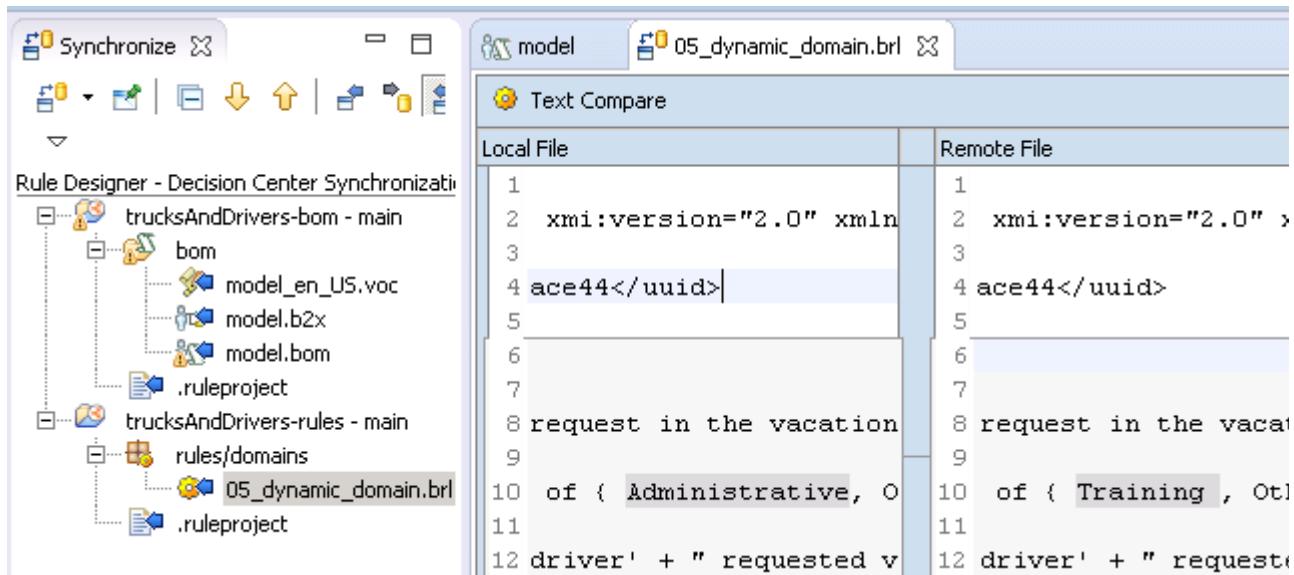
The suggested action for the changes to both of these rule projects is to *update* Rule Designer.

- 4. View the changes to the BOM and to the `05_dynamic_domain.brl` rule.
 - a. In the Synchronize pane, expand **trucksAndDrivers-bom - main > bom**.
 - b. Double-click **model.bom** to open the compare view.

- ___ c. Notice that Training in the Remote File (Decision Center) replaces Administrative in the Local File (Rule Designer).



- ___ d. Expand **trucksAndDrivers-rules - main > rules/domains**.
- ___ e. Double-click **05_dynamic_domain.brl** to open it in the Text Compare view, and notice that Training replaces Administrative.



- ___ 5. Update the Rule Designer projects with the latest changes from Decision Center.
- ___ a. In the Team Synchronize perspective, right-click **trucksAndDrivers-bom - main** and click **Update**.
- ___ b. Repeat this step to update **trucksAndDrivers-rules - main**.
After you update the Decision Center projects, the Synchronize pane no longer lists any changes.
- ___ c. Close the Text Compare window.

- ___ 6. Return to the Rule perspective, and view the rule `05_dynamic_domain` in the rule editor.
- ___ a. In the Rule Explorer, expand **trucksAndDrivers-rules > rules > domains**.
 - ___ b. Double-click `05_dynamic_domain` to open it in the rule editor.
- ___ 7. Verify that the value `Training` is in the rule.

Content

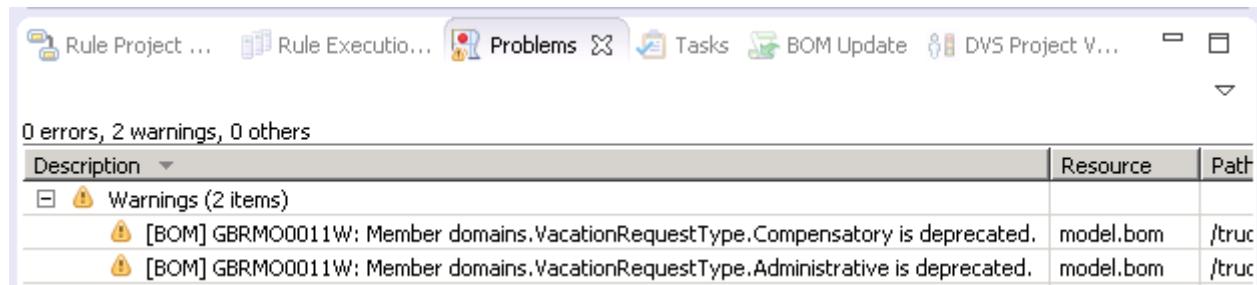
```

1 definitions
2   set 'the driver' to a driver ;
3   set 'a vacation request' to a vacation request in the vacation requ
4 if
5   the type of 'a vacation request' is one of { Training Other Vacat:
6 then
7   print "The driver " + the name of 'the driver' + " requested vacati

```

- ___ 8. Open the Problems view, and note the following warning message:

Member domains.VacationRequestType.Administrative is deprecated.



The `Administrative` domain value was deprecated when you updated the domain to include the `Training` value.

- ___ 9. Disconnect Rule Designer and Decision Center.
- ___ a. Right-click the `trucksAndDrivers-test` main rule project, click **Decision Center > Disconnect**.
 - ___ b. In the Disconnect from Decision Center window, select **Keep the Decision Center entries files**.
 - ___ c. Click **Yes**.

End of exercise

Exercise review and wrap-up

This exercise looked at how to define and use a dynamic domain, how to update dynamic domains in Decision Center, and how to synchronize them between Rule Designer and Decision Center. You also saw the effects of changing dynamic domain values in both Decision Center and Rule Designer.

(Optional) To see the solution to this exercise, switch to a new workspace and import the project **Ex 10: Dynamic domains > 02-answer**.



Information

As you learned, the values of the dynamic domains are shared between business users and developers, and can be updated independently. To ensure consistency across projects, you must establish governance guidelines:

- Clarify who can make these types of updates, and whether to enforce restrictions on when and how updates are made.
- Make sure that all roles that are involved in the business rule solution are aware of these changes, and update their working environment to reflect the changes.



Important

If you want to redo this exercise, you must delete the decision service that you published to Decision Center.

- 1. If the Decision Center console is closed, sign in again with `rtsAdmin` as the user name and password.
- 2. Delete the `trucksAndDrivers-tests` decision service.
 - a. On the **Home** tab, select **Work on a decision service**.
 - b. From the **Decision service in use** menu, select `trucksAndDrivers-tests`.
 - c. On the **Configure** tab, under Administration, click **Erase Current Decision Service**.
 - d. Click **Yes** when prompted to confirm the project deletion.

Exercise 11. Working with queries

What this exercise is about

This exercise teaches you how to define queries and rule extractors on rule projects.

What you should be able to do

After completing this exercise, you should be able to:

- Search for rule artifacts and find rules according to their dependencies
- Define and run queries and apply actions on query results

Introduction

In many cases, you must analyze your ruleset to find dependencies between your rules. In this exercise, you search your rule project for rule artifacts that have specific properties, or rules that can enable or disable some other rule artifacts. You also learn how to create and run queries on rule projects.

The exercise involves these tasks:

- Section 1, "Searching for rule artifacts"
- Section 2, "Querying rule projects"

Requirements

This exercise uses the following files, which are installed in the `<InstallDir>\studio\training` directory:

- Start project: Dev 11 - Queries\01-start
- Solution project: Dev 11 - Queries\02-answer
 - You use the solution project in "Exercise review and wrap-up" on page 11-14.

Section 1. Searching for rule artifacts

In this part of the exercise, you search for rule artifacts within a rule project. You also search for which rules are enabled or disabled by others, or which ruleflows might select them.

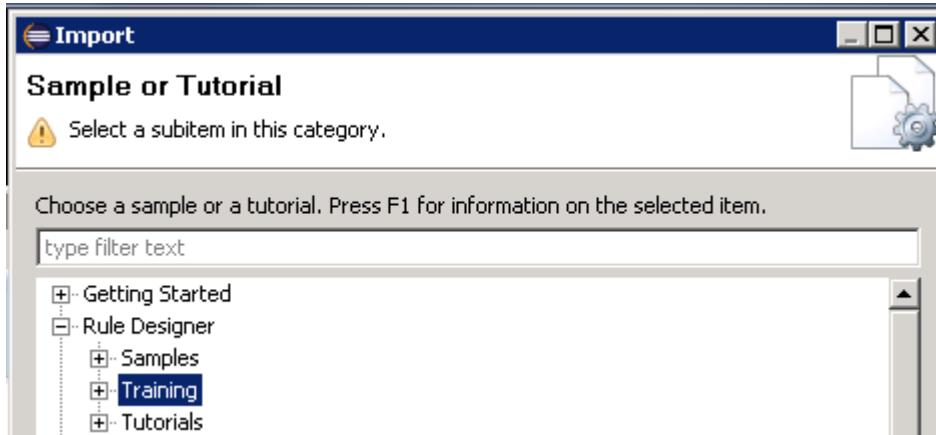
1.1. Setting up your environment for this exercise

- ___ 1. In Rule Designer, switch to a new workspace:
 - ___ a. From the **File** menu, click **Switch Workspace > Other**.
 - ___ b. In the Workspace Launcher window, enter the path:
`<LabfilesDir>\workspaces\queries`
- ___ 2. Close the Welcome view.
- ___ 3. Use the **Import** menu to import the exercise start project.
 - ___ a. In Eclipse, click **File > Import** or right-click the Rule Explorer and click **Import**.
 - ___ b. In the Select window, expand **Operational Decision Manager**, select **Samples and Tutorials**, and click **Next**.
 - ___ c. On the “Sample or Tutorial” page, expand the **Rule Designer > Training** node.



Hint

You can collapse the Rule Designer node and then expand it to simplify the list.

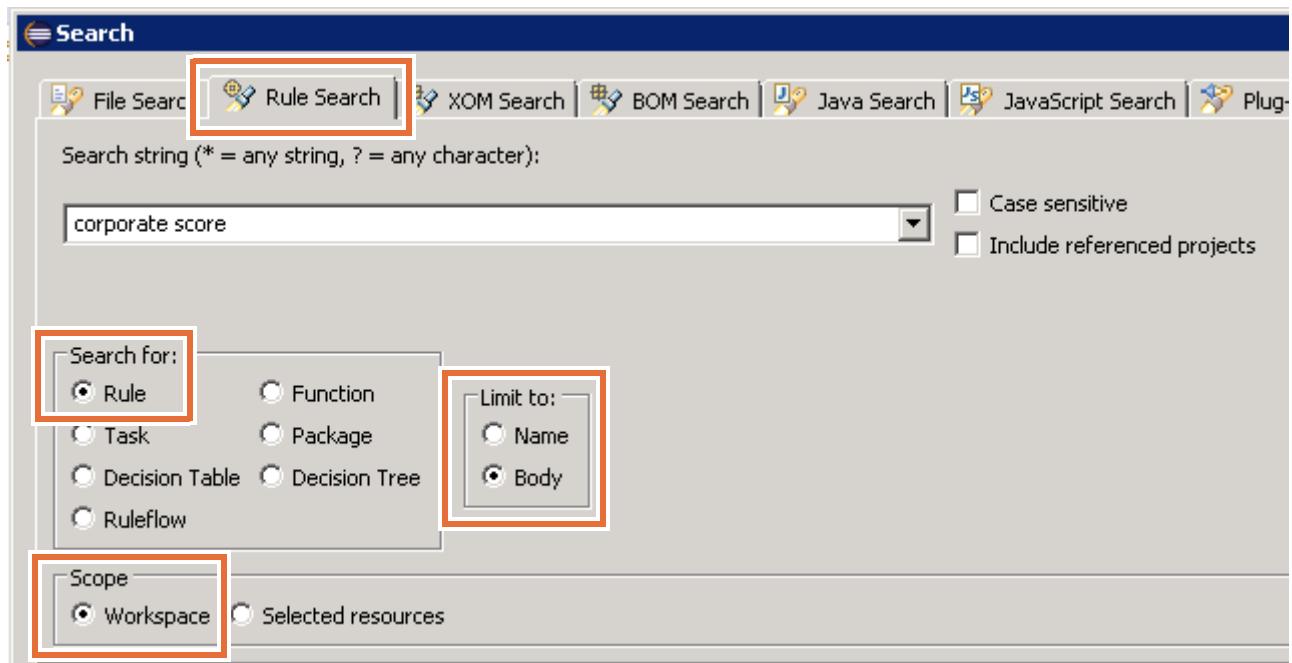


- ___ d. Expand the **Ex 11: Queries and ruleset extraction** node, and select **01-start**.
- ___ e. Make sure that the **Import shared projects** check box is selected, and click **Finish**.
- ___ 4. When the workspace finishes building, close the Help view.

1.2. Searching for specific criteria across the rules

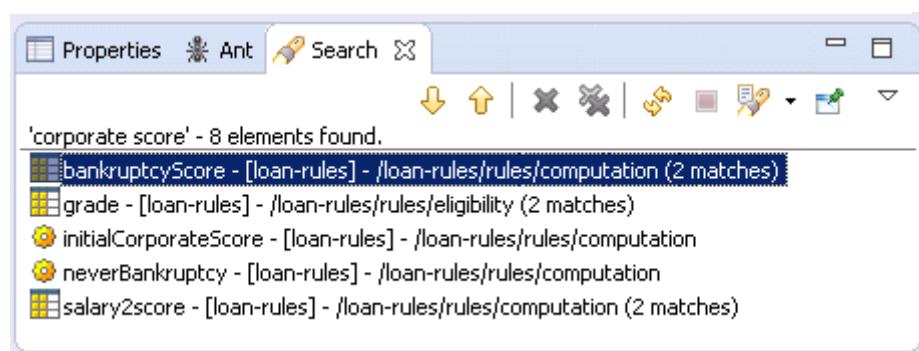
- ___ 1. From the **Search** menu, click **Search** to open the Search window.

- ___ 2. In the Search window, click the **Rule Search** tab. Take several minutes and figure out what the different elements on this tab are used for.
- ___ 3. Search all rules in the workspace that contain the `corporate score` string in their names or bodies.
 - ___ a. In the **Search string** field, type: `corporate score`
 - ___ b. Make sure that **Rule** is selected in the **Search for** section.
 - ___ c. Make sure that **Body** is selected in the **Limit to** section.
 - ___ d. Make sure that **Workspace** is selected in the **Scope** section.



- ___ e. Click **Search**.

All the results are listed in the Search view.

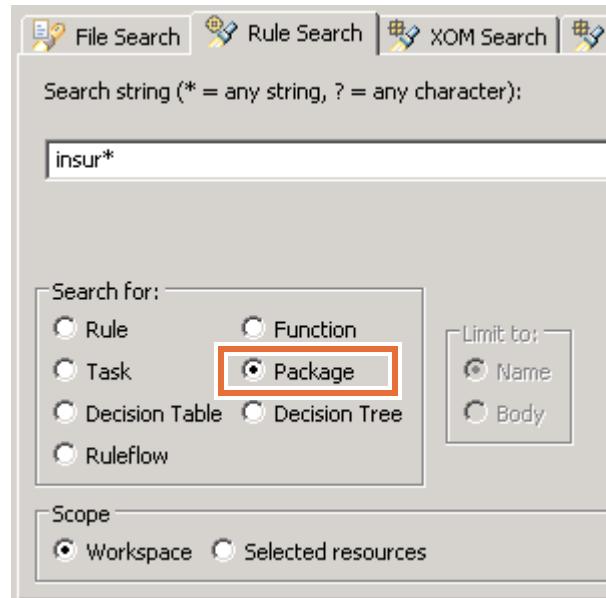


- ___ f. In the Search view results, double-click **initialCorporateScore**.

The `initialCorporateScore` action rule opens. The `corporate score` string that you searched for is highlighted in the rule artifact.

- ___ 4. Close the `initialCorporateScore` rule.

- ___ 5. From the **Search > Search** menu, do a new search for the `insur*` string in all packages in the workspace.

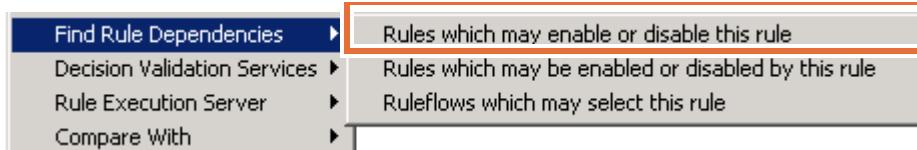


The `insurance` package is returned as a match.

- ___ 6. Open the Search window again and take some time to discover the purpose of the **XOM Search** tab and the **BOM Search** tab.

1.3. Searching for dependencies between rules

In this section, you search for dependencies between rules by using the **Find Rule Dependencies** menu.



1. In the `loan-rules` decision service, expand `rules > insurance`.
2. Right-click the `insurance` decision table, and click **Find Rule Dependencies > Rules which may enable or disable this rule**.

The Search view lists the `grade` decision table.



Questions

Why can the `grade` decision table enable or disable the `insurance` decision table that you selected?

Answer

The actions that are defined in the `grade` decision table set the grade of 'the loan report', which in turn determines which of the rows of the `insurance` decision table can be executed.

- 3. Expand the `eligibility` rule package, right-click the `grade` decision table, and click **Find Rule Dependencies > Rules which may be enabled or disabled by this rule**.

The search results show the `insurance` decision table, which you expected, along with the `approval` action rule.

**Questions**

Can you figure out why the `grade` decision table that you selected might enable or disable the `insurance` decision table and the `approval` action rule?

Answer

The `grade` decision table might enable or disable the `insurance` decision table because it sets the 'the loan report' grade, which is later used in the conditions of the `insurance` decision table.

The same mechanism applies to the `approval` action rule, where the condition is also based on this grade:

the loan grade in 'the loan report' is one of { "A" , "B" , "C" }

- 4. Expand the `computation` rule package, right-click the `initialCorporateScore` action rule in the rule project, and click **Find Rule Dependencies > Ruleflows which may select this rule**.

The Search view lists the `loanvalidation` ruleflow and the `computation` rule task.

**Questions**

Can you figure out why the `loanvalidation` ruleflow and its `computation` rule task are the answer?

Answer

The Search view shows the `loanvalidation` ruleflow because this ruleflow selects all the rule artifacts in the rule project.

The ruleflow includes rule tasks that correspond directly to all the rule packages that are defined in the project, and can thus select all the rule artifacts of this rule project. If the path through the `loanvalidation` ruleflow includes the `computation` rule task, then the `initialCorporateScore` rule is executed.

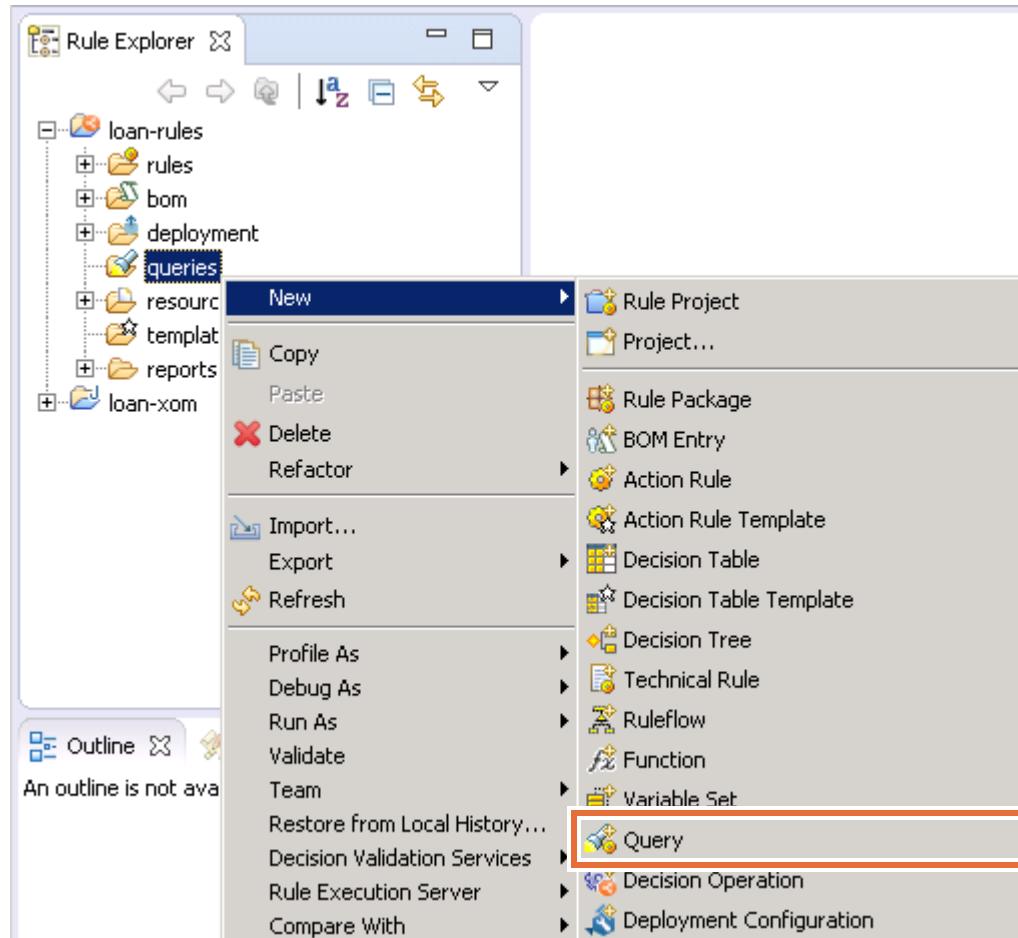
Section 2. Querying rule projects

In this part of the exercise, you create queries on rule projects to search for rules. You create different queries by using different criteria. You also run an action on the rules that the query returns.

2.1. Searching for rules that may become applicable

In this section, you search for rules that might become applicable if another rule is executed.

- 1. Create a query in the `loan-rules` decision service.
 - a. In the Rule Explorer, right-click the `queries` folder and click **New > Query**.



- b. In the **Name** field, type: `may become applicable`
- c. Click **Finish**.
- 2. In the new query, define the **such that** statement.
 - a. Click **enter a condition** and double-click **each business rule** to select it from the vocabulary list.

- __ b. Complete the statement to match this content:

Find all business rules
such that each business rule may become applicable
when ['a borrower' has filed a bankruptcy]

- __ c. Save the query.



Hint

To see the list of choices, press Ctrl+Spacebar at the location in the rule where you want to add text, or type directly in the Query editor.

- __ 3. In the **Run** section, click **Run query** to run the query on the rules.



The Search view indicates all rows of the `bankruptcyScore` decision table.



Questions

Do you know why you get this result?

Answer

The Search view indicates all rows of the `bankruptcyScore` decision table. All rows in this decision table have conditions that are based on the age of the latest bankruptcy of the borrower. The conditions are met only if the borrower filed a bankruptcy (which is the query condition).

The `neverBankruptcy` rule also relies on the bankruptcy status of the borrower for its conditions. However, if the borrower filed a bankruptcy, the `neverBankruptcy` action rule is not applicable because its conditions start with: it is not true that

No other rule artifact in the rule project relies on the bankruptcy status of the borrower.

- __ 4. Close the query.

2.2. Searching for rules that may lead to a state

In this section, you search for rules that can lead to a specific state of the application objects.

- 1. Create a query that is called `may lead to a state` with the following code:

```
Find all business rules
such that each business rule may lead to a state
where [ 'a report' is insurance required ]
```

- 2. Save the query and click **Run query**.

The Search view lists the `defaultInsurance` action rule and some of the rows in the insurance decision table.



Questions

Can you figure out why you obtain this result?

Answer

The Search view includes the `defaultInsurance` action rule because this action rule sets the `insuranceRequired` member of 'the loan report' to `true`.

The insurance decision table defines rules that set the `insuranceRequired` member of 'the loan report' to either `true` or `false`. All rows that are indicated in the Search view set this member to `true`.

The other rows (1 and 5) set this member to `false` and are not indicated in the Search view.

- 3. Verify that the results in the Search view are accurate and that all rows of the insurance decision table, except 1 and 5, set the `insuranceRequired` member of 'the loan report' to `true`.
 - a. In the Search view, right-click the insurance decision table and click **Show In > Rule Explorer**. Notice that the insurance decision table is highlighted in the Rule Explorer.
 - b. In the Rule Explorer, double-click the insurance decision table, which is highlighted, to open it in the decision table editor.
 - c. Read the values that are given in the `Insurance required` column and verify that all rows contain `true` except for rows 1 and 5, which contain `false`.

- ___ d. Select a row (for example, row 3), and read the complete action rule that corresponds to this row of the insurance decision table.

Grade	Amount of loan		Insurance required	Insurance rate
	Min	Max		
1	< 100,000		false	0
2	100,000	300,000	true	0.001
3	300,000	600,000	true	0.003
4	> 600,000		true	0.005
5	(the loan grade in 'the loan report' is not empty) and all of the following conditions are true : - (the loan grade in 'the loan report' is "A") - (the amount of 'the loan' is at least 300000 and less than 600000), then set insurance required in 'the loan report' to true ; set the insurance rate in 'the loan report' to 0.003 ;			
6	100,000	300,000	true	0.006
7	300,000	600,000	true	0.0085
8	> 600,000		true	0.0115

General Decision Table IRL insurance.dta

- ___ 4. Close the decision table and the query.

2.3. Searching for rules that use a BOM member

In this section, you search for rules that use a specific BOM member.

- ___ 1. Create a query that is called: using BOM member

The query should state:

Find all business rules

such that each business rule is using the BOM Member
"training_loan.Loan.amount"

- ___ 2. Save the query and run it.

The Search view indicates the checkAmount action rule, the insurance decision table, and the repayment action rule.



Questions

Can you figure out why you obtain this result?

Answer

The Search view indicates the `checkAmount` action rule because the conditions of this action rule are based on the value of the amount of 'the loan', that is, on the `training_loan.Loan.amount` BOM member.

The Search view also indicates the `insurance` decision table because the conditions for all the rows in this decision table are based on the same value.

Finally, the Search view indicates the `repayment` rule because the actions of this rule use the same value to compute the monthly repayment.

No other rule artifact in the rule project relies on the value of the amount of 'the loan'.

2.4. Applying actions with queries

In this section, you run a query and apply actions to the results. First, you search for rules that have a high priority, and modify their priority as an action that is associated with the query.

- ___ 1. Create a query that is called: the `priority of the rules`

The query should state:

Find all business rules

such that the priority of each business rule is "1000000"

Do set the priority of each business rule to "-1000000"



Important

The query action is introduced with the `Do` keyword and modifies the priority of the business rules that match the query criteria.

Before you run a query that potentially modifies your rule project, you first want to determine whether your query is correct, and verify which rule artifacts might be affected. For this reason, the **Run query actions** task is separate from the **Run query** task, which provides an opportunity to review the query results before applying actions.

- ___ 2. Save your work.

- ___ 3. Select the **Run query** option of the query "the priority of the rules".

You have two results:

- The `grade` decision table
- The `initialCorporateScore` action rule



Questions

Are these results correct?

Answer

In the loan-rules project, only the grade decision table and the initialCorporateScore action rule have their **priority** property that is defined with the value 1000000.

- 4. Now, apply the actions to the found rule artifacts by clicking **Run query actions** in the Run section.



- 5. Validate your results.
- a. In Rule Explorer, expand **loan-rules > rules > computation**.
 - b. Open the `initialCorporateScore` rule.
 - c. In the Properties view (which is in the lower-right corner), verify that the **priority** property of the `initialCorporateScore` action rule is now equal to -1000000.
 - d. Do the same verification for the `eligibility.grade` decision table.
 - e. Close the rule and decision table.
- 6. Run the `priority` of the rules query again.

No business rules match.



Questions

Why are there no matches?

Answer

The action of this query replaces the priority of 1000000 with a priority of -1000000. After this action is applied, the two rule artifacts found initially now have a priority of -1000000, and are thus not found if you run the query again.

-
- ___ 7. Close the query and the rules.

End of exercise

Exercise review and wrap-up

The first part of the exercise looked at how to search for rule artifacts within a rule project and how to find which rules are enabled or disabled by others, or which ruleflows might select them.

The second part of the exercise looked at how to create and run queries on rule projects. It also looked at how to create ruleset extractors that are based on queries, and how to use these ruleset extractors to create ruleset archives that match the queries.

(Optional) To see the solution to this exercise, switch to a new workspace and import the project **Ex 11: Queries > 02-answer**.

 **Warning**

In the answer project, the priorities for the `grade` decision table and for the `initialCorporateScore` action rule are still `1000000`. This value was left intentionally for you to be able to see the results of running the `priority` of the `rules` query actions on the answer project.

Exercise 12. Executing rules locally

What this exercise is about

This exercise teaches you how to run rule projects locally to ensure the correctness of rulesets.

What you should be able to do

After completing this exercise, you should be able to:

- Create launch configurations to run rulesets locally

Introduction

In this exercise, you create a launch configuration to execute the rule artifacts locally in Rule Designer, without writing code.

The exercise includes these tasks:

- Section 1, "Executing rules locally by using a launch configuration"

Requirements

This exercise uses the following file, which is installed in the *<InstallDir>\studio\training* directory:

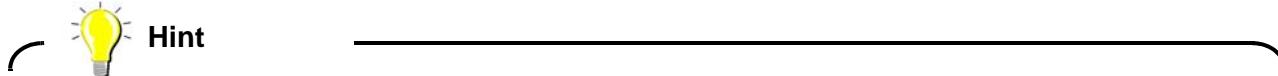
- Start project: Dev 12 - Executing rules locally\01-start

Section 1. Executing rules locally by using a launch configuration

In this part of the exercise, you create a launch configuration. You then use it to execute the rule artifacts in your rule project locally in Rule Designer.

1.1. Setting up your environment for this exercise

- ___ 1. In Rule Designer, switch to a new workspace:
 - ___ a. From the **File** menu, click **Switch Workspace > Other**.
 - ___ b. In the Workspace Launcher window, enter the path:
`<LabfilesDir>\workspaces\execute_locally`
- ___ 2. Close the Welcome view.
- ___ 3. Import the lab projects.
 - ___ a. From the **File** menu, click **Import**.
 - ___ b. On the Select page, select **Operational Decision Manager > Samples and Tutorials**, and click **Next**.
 - ___ c. In the **Rule Designer > Training** node, select **Ex 12: Executing rules locally > 01-start** and click **Finish**.
- ___ 4. After the workspace builds, close the Help view.

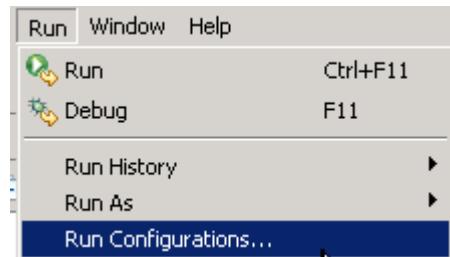


In this exercise, you write some code. To help you, you can copy and paste the code snippets from the `<LabfilesDir>\code\execute_rules_locally.txt` file into Rule Designer.

1.2. Creating a launch configuration

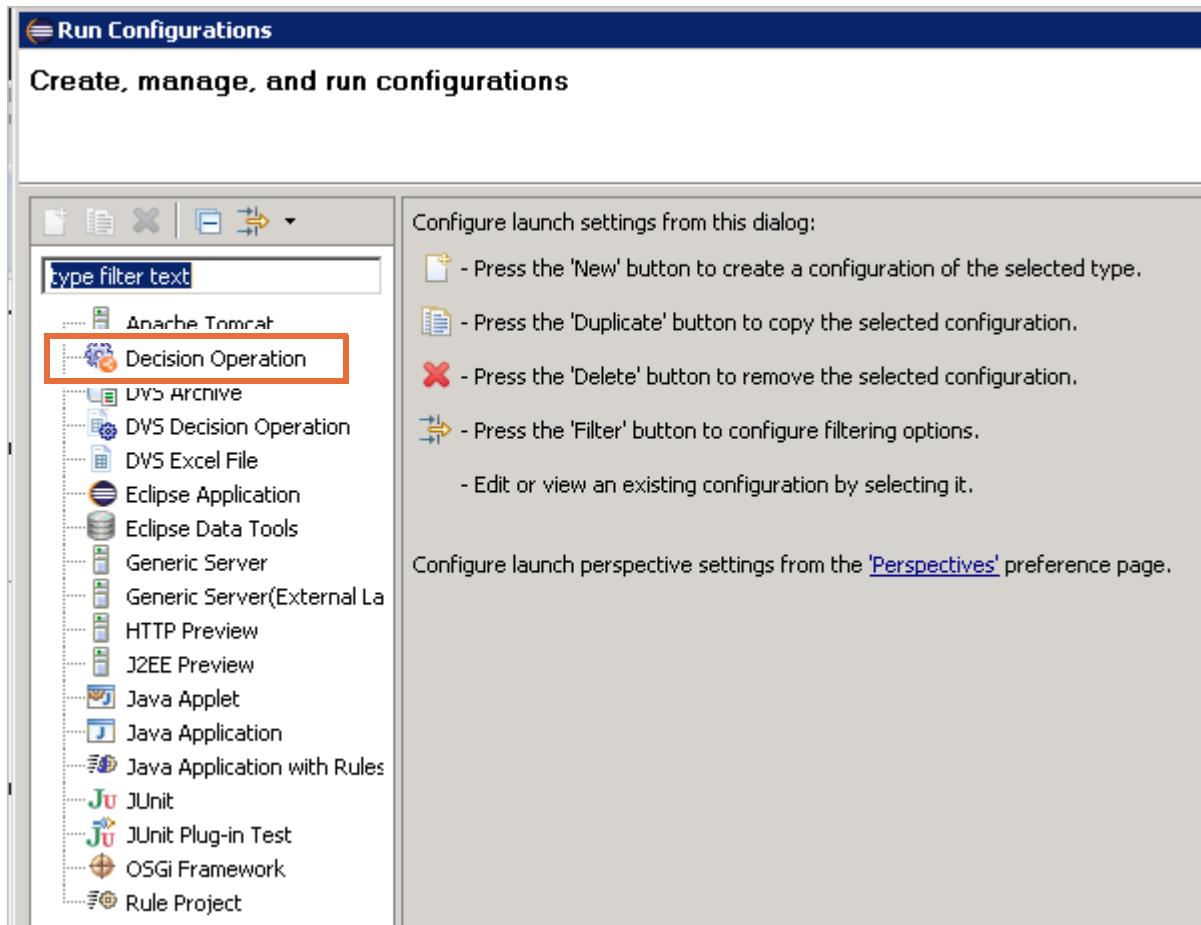
In this section, you create a launch configuration for the `loan-rules` project.

- ___ 1. From the **Run** menu, click **Run Configurations**.



The New Deployment Configuration window opens.

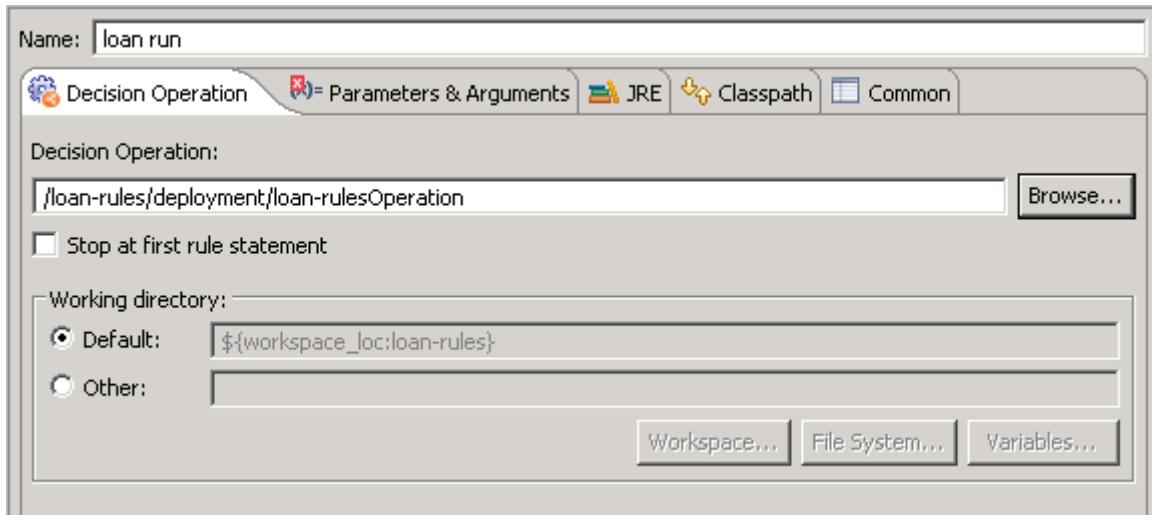
2. In the Run Configurations wizard, double-click **Decision Operation**.



3. In the **Name** field, enter: `loan run`
4. On the **Decision Operation** tab, configure the following properties.
- To set the **Decision Operation** field, click **Browse**, select **loan-rulesOperation**, and click **OK**.
 - Make sure that the **Stop at first rule statement** check box is *not* selected.

This check box is used only when you debug rules, which you do later.

- __ c. In the **Working directory** section, keep **Default** selected.



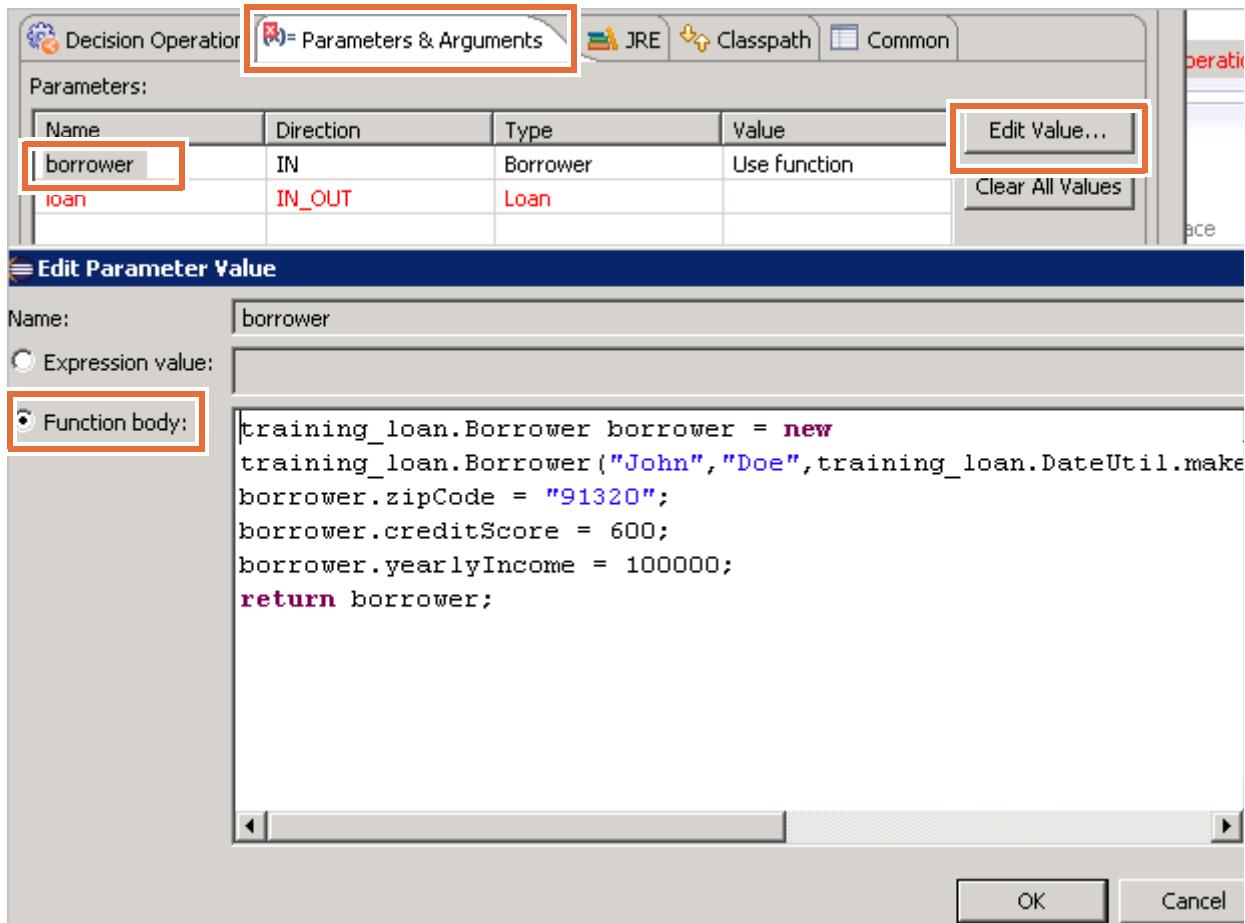
- __ 5. Click the **Parameters & Arguments** tab and configure the ruleset parameters that are required to execute your rule project.
- __ 6. Define the **borrower** ruleset parameter.
- __ a. Select the **borrower** ruleset parameter in the **Name** column, and click **Edit Value**.
 - __ b. Select the **Function body** option.
 - __ c. Overwrite the initial value with the following code to overwrite and click **OK**:

```
training_loan.Borrower borrower = new
training_loan.Borrower("John", "Doe", training_loan.DateUtil.makeDate(1968,
java.util.Calendar.MAY,12), "123456789");
borrower.zipCode = "91320";
borrower.creditScore = 600;
borrower.yearlyIncome = 100000;
return borrower;
```



Hint

You can find all code snippets for this exercise in the
`<LabfilesDir>\code\execute_rules_locally.txt` file.



7. Define the `loan` ruleset parameter.
- Select the `loan` ruleset parameter and click **Edit Value**.
 - Select **Function body** and overwrite the initial value with the following code and click **OK**:
- ```
training_loan.Loan loan = new
training_loan.Loan(training_loan.DateUtil.makeDate(2016,
java.util.Calendar.JANUARY,15),72,100000,0.7d);
return loan;
```
8. Click **Apply** to save the changes.
9. Click **Run**.

Rule Designer executes the rules in your rule project locally by using the ruleset parameters that are defined in the `loan run` launch configuration. The Console view shows the following execution traces:

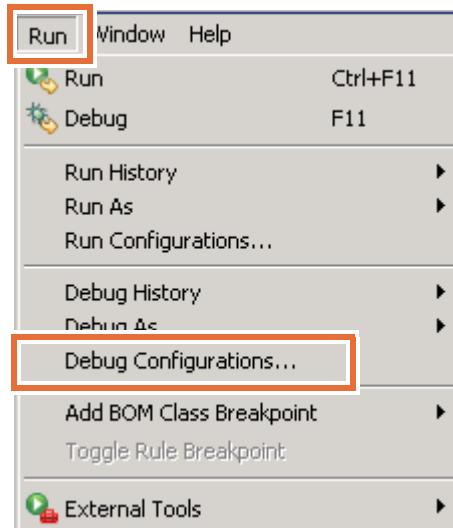
```
Borrower(firstName: John lastName: Doe born: May 12, 1968 SSN: 123-45-6789
zip code: 91320 income: 100000 credit score: 600)
Loan(amount: 100000 starting: Jun 1, 2015 duration: 72 month LTV: 70% rate:
5.7% monthly repayment: 1,643.16)
Report(validData: true approved: true score: 820 grade: B insuranceRequired:
true insuranceRate: 2% message: Low risk loan
Congratulations! Your loan has been approved
)
```

- \_\_\_ 10. Take some time to review the execution traces in the Console view and compare them to the defined ruleset parameters.

### 1.3. Debugging with a launch configuration

In the previous section, you created the `loan run.launch` configuration to execute your ruleset locally without the Rule Debugger. In this section, you modify this launch configuration to use the Rule Debugger.

- \_\_\_ 1. From the Run menu, click **Debug Configurations**.



The Debug Configurations window opens.



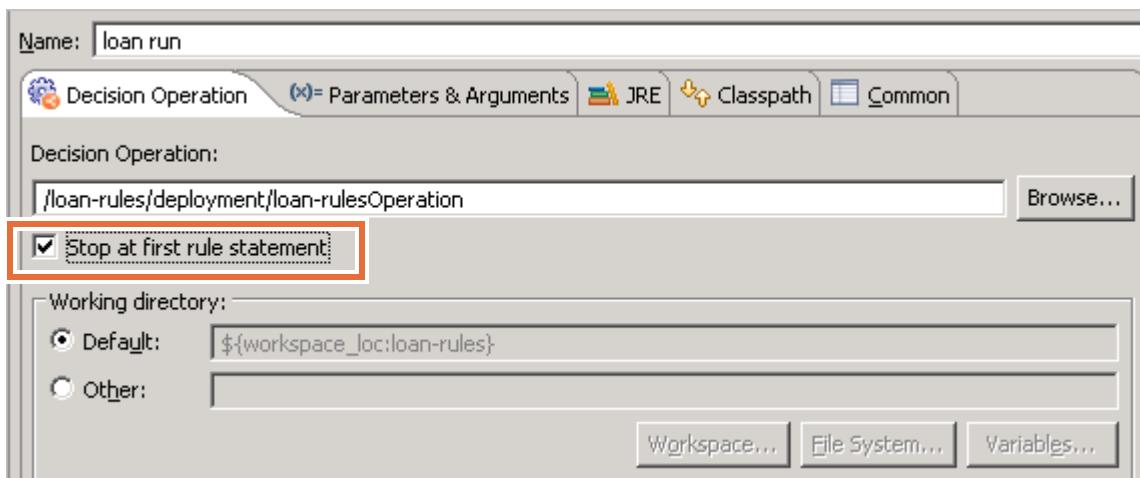
#### Questions

Do you see any differences between the Debug Configurations window and the Run Configurations window that you worked in earlier?

**Answer**

The Debug Configurations window and Run Configurations window share content except that **Debug** replaces **Run**.

2. On the **Decision Operation** tab, select **Stop at first rule statement**.



The **Stop at first rule statement** option is used when you run the debug configuration. If this check box is selected, the Rule Debugger stops before any business rule is executed. You can then complete any steps that are required for your debugging session (such as setting breakpoints or modifying ruleset parameters).

3. Click **Apply**.

**Warning**

Do not click **Debug**. You use the Rule Debugger in a later exercise.

4. Click **Close** to exit.

**Note**

On the **Overview** tab for the decision operation, you can click the **Run** or **Debug** icons to rerun the configuration.

Decision Operation Overview - loan-rulesOperation

**End of exercise**

## Exercise review and wrap-up

This exercise looked at how to execute rules locally by using a launch configuration in Rule Designer.

# Exercise 13.Debugging a ruleset

## What this exercise is about

This exercise teaches you how to debug a ruleset in Rule Designer.

## What you should be able to do

After completing this exercise, you should be able to:

- Debug a ruleset
- Set breakpoints on an action rule and on a task in a ruleflow
- Inspect objects in the working memory or rule instances in the agenda
- Use the various views of the Debug perspective

## Introduction

In many cases, executing rules locally in Rule Designer is not enough for you to find out why the ruleset does not create the expected decision. In this exercise, you debug a ruleset with the Rule Debugger.

The exercise includes these tasks:

- Section 1, "Running the debug launch configuration"
- Section 2, "Debugging with breakpoints"
- Section 3, "Debugging with breakpoints in the ruleflow"
- Section 4, "Resolving the problem"

The steps that are described here are extracted from the *Debugging a ruleset* tutorial, which is distributed with the product documentation. During the exercise, you do not fix the errors that you find. However, if you want instructions on how to fix the errors, you can consult the tutorial for more details.

## Requirements



### Note

#### For IBM ODM on Cloud users

This exercise requires that you use the on-premises version of ODM. ODM on Cloud does not include projects that are delivered as tutorials.

## Section 1. Running the debug launch configuration

In this section, you use a launch configuration to debug the ruleset execution.

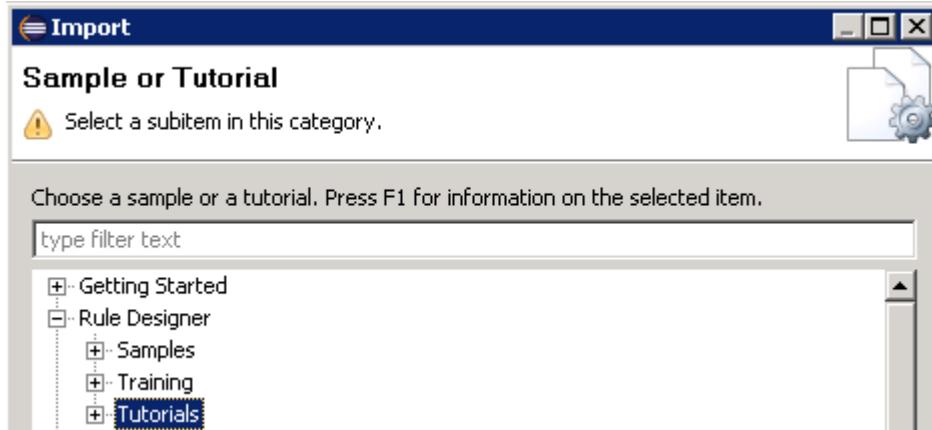
### 1.1. Setting up your environment for this exercise

- \_\_ 1. In Rule Designer, switch to a new workspace:
  - \_\_ a. From the **File** menu, click **Switch Workspace > Other**.
  - \_\_ b. In the Workspace Launcher window, enter the path:  
`<LabfilesDir>\workspaces\debug`
- \_\_ 2. Close the Welcome view.
  - \_\_ a. Click the **Open Perspective** icon in the upper-right part of the Eclipse window.
- \_\_ 3. Use the **Import** menu to import the exercise start project.
  - \_\_ a. In Eclipse, click **File > Import** or right-click the Rule Explorer and click **Import**.
  - \_\_ b. In the Select window, expand **Operational Decision Manager**, select **Samples and Tutorials**, and click **Next**.
  - \_\_ c. On the “Sample or Tutorial” page, expand the **Rule Designer > Tutorials > Debugging a ruleset** node.



#### Hint

You can collapse the Rule Designer node and then expand it to simplify the list.



- \_\_ d. In the **Tutorials > Debugging a ruleset** node, select **start**.
- \_\_ e. Make sure that the **Import shared projects** check box is selected, and click **Finish**.

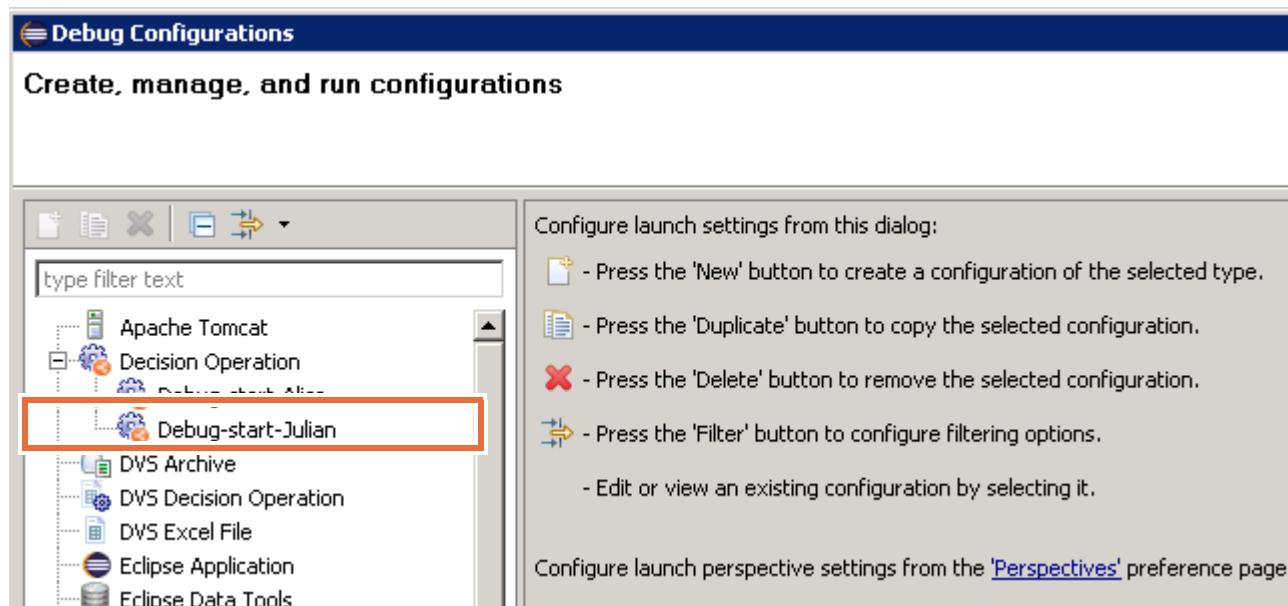
The Rule perspective opens with these projects:

- carrental-xom
- debug-rules-start

- \_\_\_ 4. When the workspace finishes building, close the Help view.

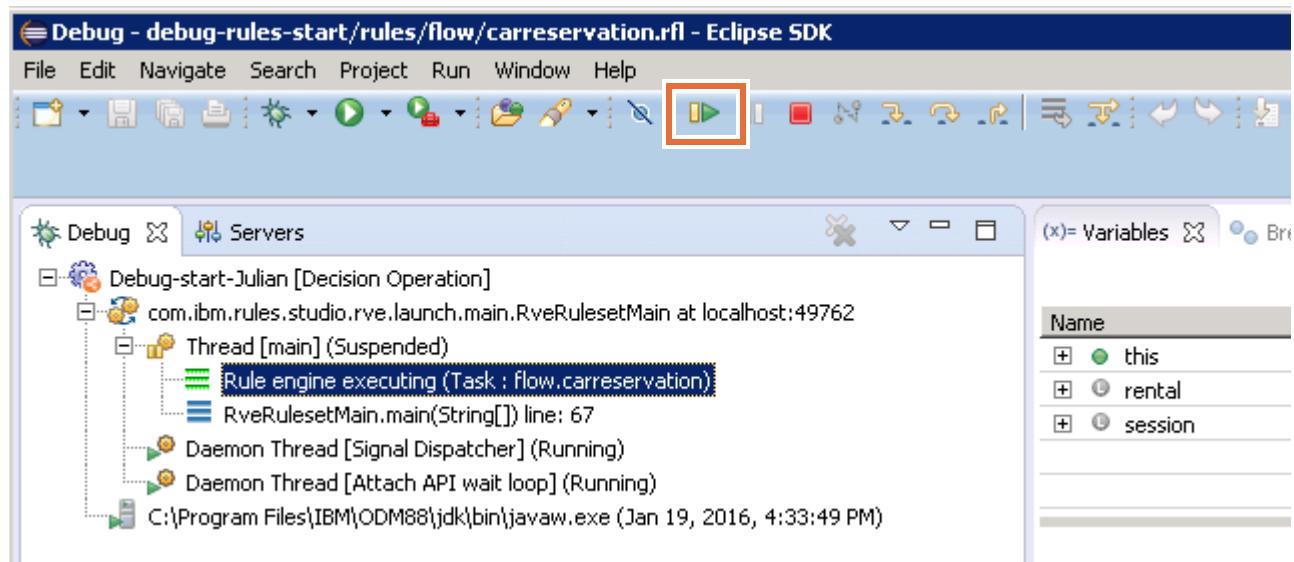
## 1.2. Running the debug configuration

- \_\_\_ 1. From the menu, click **Run > Debug Configurations**.
- \_\_\_ 2. In the Debug Configurations window, expand **Decision Operation** and select the **debug-start-Julian** launch configuration.



The `debug-start` is a launch configuration of type Rule Project that is configured to start the Rule Debugger.

- \_\_\_ 3. Click **Debug**.
- \_\_\_ 4. When prompted to switch to the Debug perspective, click **Yes**.
- \_\_\_ 5. On the toolbar In the Debug perspective, click **Resume** (or press F8) to execute the application.



- \_\_\_ 6. Look at the results in the Console view.

As you scroll through the results, notice that the prices for the special offers are set to 0.00.

### 1.3. Setting breakpoints in the rules

To identify the reason for these results, you must watch the rules that are related to special offers in the `pricing` package.

- \_\_\_ 1. Switch back to the Rule perspective.



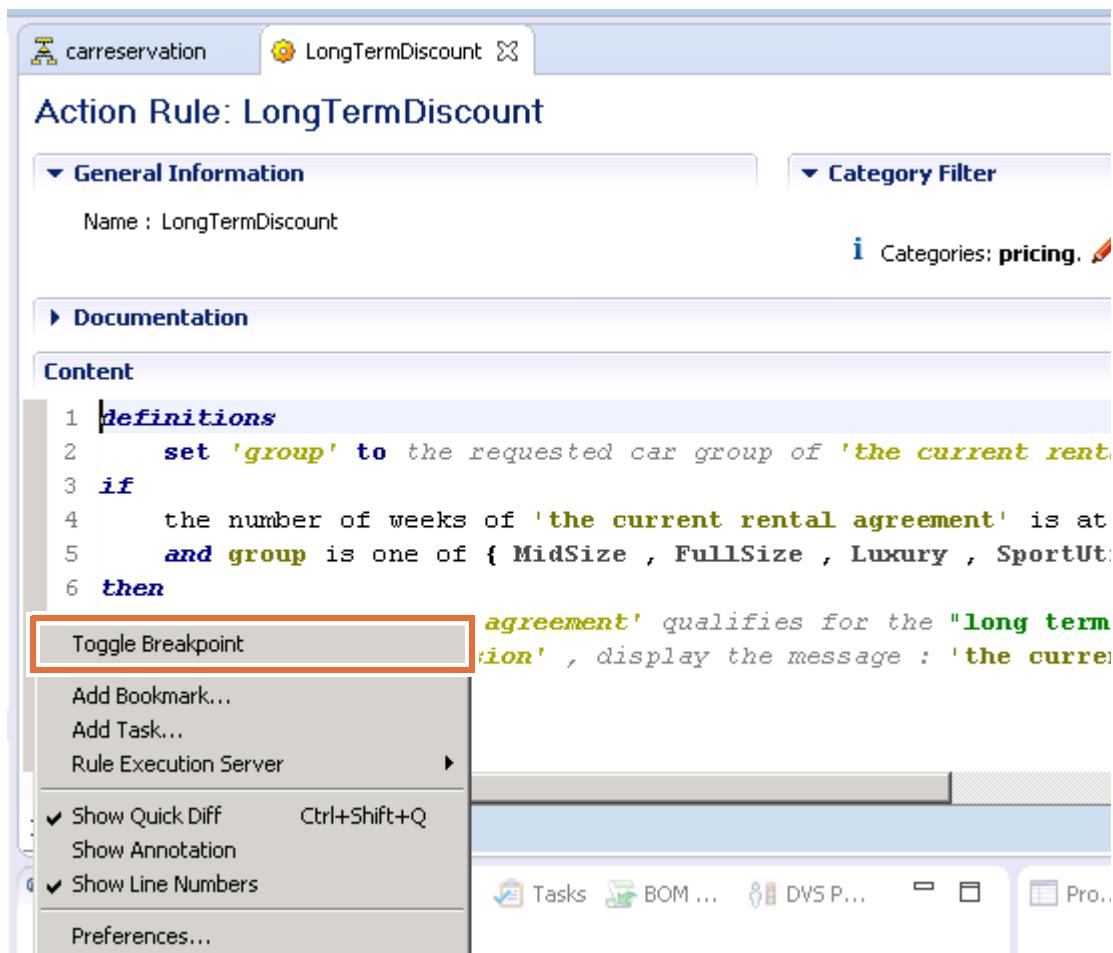
- \_\_\_ 2. In Rule Explorer, expand **debug-rules-start > rules > pricing.qualifyFor** and open the `LongTermDiscount` rule with the Intellirule editor.



#### Hint

Double-clicking an action rule opens it in the most recently used rule editor, either the Guided editor or the Intellirule editor. To make sure that you open an action rule with the Intellirule editor, right-click it and click **Open With > Intellirule Editor**.

3. In the Intellirule editor, right-click the shaded border in the **Content** section beside the first line of the `then` statement, and click **Toggle Breakpoint**.



A breakpoint is now visible on the shaded border in the **Content** section of this rule.

```

6 then
7 | 'the current rental ag
8 in 'the current sessio

```

4. Add a similar breakpoint on the first action statements in these rules:

- `pricing.qualifyFor.SpringSeason` rule
- `pricing.price.LongTermDiscount` rule

By adding these breakpoints, you ask execution to stop when it reaches these points so that you can check what is happening in the agenda.

## Section 2. Debugging with breakpoints

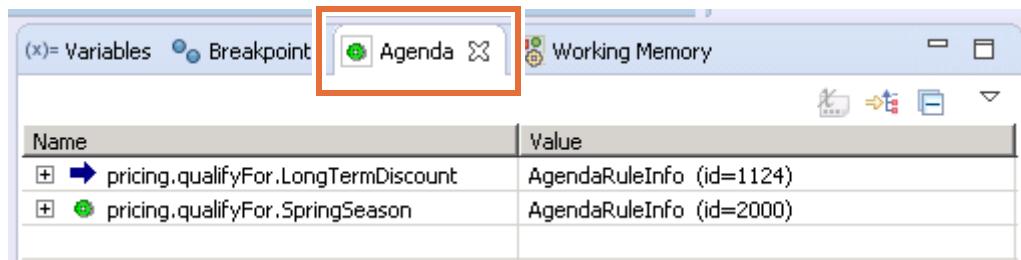
- 1. Restart the debugging session (as you did in Section 1, "Running the debug launch configuration," on page 13-2).
  - a. From the menu, click **Run > Debug Configurations**.
  - b. In the Debug Configurations window, expand **Decision Operation** and select the `debug-start-Julian` launch configuration.
  - c. Click **Debug**.
  - d. Click **Yes** when prompted to switch to the Debug perspective.

### 2.1. Inspecting the agenda

- 1. In the Debug perspective, click the **Agenda** tab so that the Agenda view is visible.



- 2. On the toolbar, click **Resume** (or press F8) to start the execution of the application.
- 3. When the execution stops, look at the Agenda view.



The Agenda view shows the rules that ran. The rules that are currently instantiated in the Agenda are:

- `pricing.qualifyFor.LongTermDiscount`
- `pricing.qualifyFor.SpringSeason`

The `qualifyFor.LongTermDiscount` is marked with an arrow, meaning that execution is stopped at the breakpoint in this rule.

However, the `pricing.price.LongTermDiscount` rule is not listed, although it is supposed to run every time the `pricing.qualifyFor.LongTermDiscount` rule is true.

- 4. Expand `pricing.qualifyFor.LongTermDiscount > matchedObjects` for `RentalAgreement`.

- \_\_\_ 5. Click offers and note the current value in the detail pane.

The screenshot shows the Eclipse IDE's Working Memory view during rule debugging. The 'Working Memory' tab is active. A table lists variables and their values. The 'offers' variable is highlighted in blue, indicating it is selected. In the detail pane at the bottom, the value of 'offers' is shown as a HashMap<K,V> (id=2004), with the specific entry '(standard=standard price: 579.90)' highlighted with a red box.

| Name                                | Value                     |
|-------------------------------------|---------------------------|
| pricing.qualifyFor.LongTermDiscount | AgendaRuleInfo (id=1124)  |
| matchedObjects                      | Object[] (id=1517)        |
| [0]                                 | RentalAgreement (id=1174) |
| actualCarGroup                      | null                      |
| assigned                            | false                     |
| coverages                           | String[4] (id=1854)       |
| customer                            | Customer (id=1730)        |
| <b>offers</b>                       | HashMap<K,V> (id=2004)    |
| pickupBranch                        | Branch (id=1387)          |
| pickupDate                          | Date (id=1502)            |
| rejected                            | false                     |
| requestedCarGroup                   | CarGroup (id=1807)        |
| returnBranch                        | Branch (id=1387)          |
| returnDate                          | Date (id=1994)            |

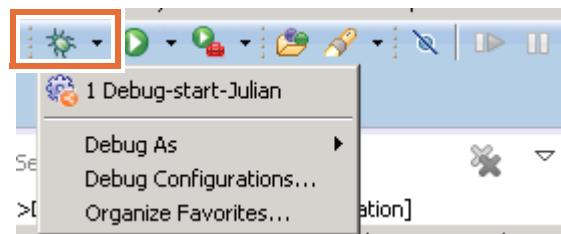
(standard=standard price: 579.90)

- \_\_\_ 6. Click **Resume** (or press F8) to resume debugging until execution stops at the next breakpoint, and again note which rules were put in the agenda, and which one is being fired. Execution stops in the pricing.qualifyFor.SpringSeason rule.  
Normally, the pricing.price.LongTermDiscount rule is supposed to run before the pricing.qualifyFor.SpringSeason rule.
- \_\_\_ 7. In the Agenda view, again check the value of pricing.qualifyFor.SpringSeason > matchedObjects > RentalAgreement > offers.  
The detail pane now displays the price for the long-term offer, which is 0.00. This price is incorrect.
- \_\_\_ 8. Click **Terminate** to stop debugging.

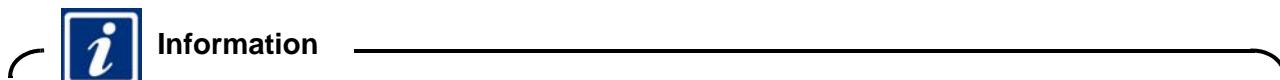
## Section 3. Debugging with breakpoints in the ruleflow

In this section, you add breakpoints in the ruleflow of the rule project.

- \_\_\_ 1. Switch to the Rule perspective.
- \_\_\_ 2. In Rule Explorer, expand **debug-rules-start > rules > flow** and double-click the carreservation ruleflow in the **flow** package.
- \_\_\_ 3. Set breakpoints on the `eligibility` and `pricing` tasks.
  - \_\_\_ a. Select the `eligibility` task, and right-click it to click **Toggle Breakpoint**.
  - \_\_\_ b. Add another breakpoint on the `pricing` task.
- \_\_\_ 4. Switch to the Debug perspective again, and click the **Debug** icon on the toolbar to restart debugging with the `debug-start-Julian` configuration.



- \_\_\_ 5. When the execution stops on the `eligibility` task, open the **Variables** view to find the `rental` variable.
- \_\_\_ 6. Click `rental` to see which customer's rental agreement is the current value of the `rental` variable.  
The rental agreement for Julian Bayles should be listed in the detail pane.
- \_\_\_ 7. Resume debugging (click **Resume** or press F8) until the execution stops.



With the current rule project, the special offers are always equal to 0. This situation happens because:

- The default value for each offer is set to 0.
- The `pricing.price.*` rules set the offers.

When the `qualifyFor` rules determine that the rental agreement qualifies, it adds that offer to the rental agreement.

The problem is that the rules do not update the `RentalAgreement` object, and the rule engine cannot see the offer to evaluate it. As a result, the application does not run the rules that compute the price.

In the BOM, the **Update object state** property of the `addOffer` method of the `RentalAgreement` class is not set. As a consequence, the rule engine is not notified of the update of the rental agreement. Although the RetePlus algorithm is used on the `Pricing` task, the rule engine does not reevaluate the rule instances to execute. The rule engine thus does not add the appropriate special offer price rule to the agenda.

To obtain special offers, you must make sure that the **Update object state** check box of the `addOffer` method of the `RentalAgreement` class is selected in the BOM editor.



## Section 4. Resolving the problem

In this section, you resolve the issue.

- \_\_\_ 1. Switch to the Rule perspective.
- \_\_\_ 2. In Rule Explorer, expand **debug-rules-start > bom > model > carrental > RentalAgreement**, and double-click the `addOffer` method to open it in the BOM editor.
- \_\_\_ 3. On the Member page for `addOffer`, select **Update object state**.

- \_\_\_ 4. Save your work.
- \_\_\_ 5. Return to the Debug perspective.
- \_\_\_ 6. Make sure that the Agenda view is open and run the debug tools again.
- \_\_\_ 7. After you stop at the first breakpoint, continue clicking **Resume** (or press F8) to run the application until you stop in the `pricing.qualifyFor.LongTermDiscount` rule.

In the Agenda view open, you see that:

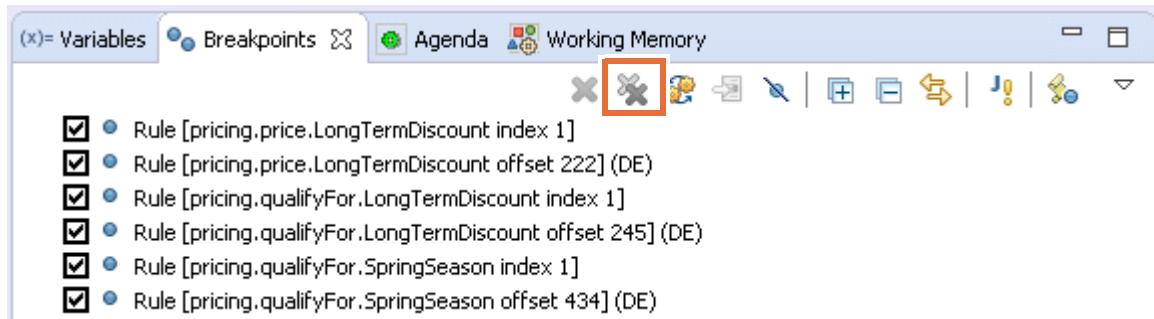
- In the `pricing.qualifyFor.LongTermDiscount` rule, the execution stopped at the phrase: 'the current rental agreement' qualifies for the "long term" offer
  - The `pricing.qualifyFor.SpringSeason` rule is listed.
- \_\_\_ 8. On the toolbar, click **Step Over** until you see in the Agenda view that the `pricing.price.LongTermDiscount` rule ran.

The `pricing.price.LongTermDiscount` rule now shows up in the Agenda as expected. The behavior is resolved from Activity 2.1, "Inspecting the agenda," on page 13-6.

| Name                            | value                    |
|---------------------------------|--------------------------|
| pricing.price.LongTermDiscount  | AgendaRuleInfo (id=1850) |
| pricing.qualifyFor.SpringSeason | AgendaRuleInfo (id=1665) |

- \_\_\_ 9. Continue clicking **Step Over** until the Console view shows that the special offers are now correctly included in the final price.

- \_\_\_ 10. After you finish debugging, open the Breakpoints view and clear the breakpoints by clicking the **Remove All Breakpoints** icon.



- \_\_\_ 11. Click **Yes** when prompted to confirm removal of the breakpoints.

**End of exercise**

## Exercise review and wrap-up

This exercise looked at how to debug a ruleset with the Rule Debugger. You saw how to set breakpoints on an action rule and on a task in a ruleflow, and you inspected objects in the working memory and rule instances in the agenda.

To see the solution projects for this exercise, switch to a new workspace and import the project **Rule Designer > Tutorials > Debugging a ruleset > answer > Import projects**.



### Note

You can use the Rule Debugger with a Java client application. You can also create a launch configuration for this purpose, which is then of type **Java Application with Rules**, and configure it to start the Rule Debugger.

# Exercise 14. Enabling tests and simulations

## What this exercise is about

This exercise teaches you how to set up testing and simulation functionality for business users.

## What you should be able to do

After completing this exercise, you should be able to:

- Validate the BOM and generate scenario file templates in Excel format
- Customize scenario file templates
- Create virtual attributes to test collections of complex objects
- Validate remote testing conditions for business users in the Business console

## Introduction

In this exercise, you prepare a decision service to make it ready for running tests and simulations. You also work with scenario file templates.

As developers, your role with testing is mainly to enable business users to run tests and simulations with minimal dependence on you. This exercise provides an overview of the tasks that are involved, including:

- Section 1, "Validating the rule project"
  - Choosing a constructor
  - Editing argument names for column headings in the template
- Section 2, "Generating the scenario file template"
- Section 3, "Populating the template and testing the scenario"
- Section 4, "Customizing input for the scenario file template"
- Section 5, "Enabling testing from Business console"
- Section 6, "Testing from Business console"

## Requirements

This exercise uses a series of Excel files that are stored in the `<LabfilesDir>\code` directory. It also uses the following files, which are installed in the `<InstallDir>\studio\training` directory:

- Start project: Dev 14 - DVS\01-start
- Solution project: Dev 14 - DVS\02-answer
  - You can use the solution project in "Exercise review and wrap-up" on page 14-23.



**Note**

**For IBM ODM on Cloud users**

This exercise uses some features that are not supported in IBM ODM on Cloud. ODM on Cloud does not support Decision Validation Services in Rule Designer.

## Section 1. Validating the rule project

To use the default Excel format for your scenarios, you must validate that the rule projects to be tested can correctly generate the appropriate columns in the Excel scenario file template.

You use the DVS Project Validation view when you validate your rule project. This view raises error and warning messages if the BOM or the input parameters of the rule project must be modified.

### 1.1. Setting up your environment for this exercise

- \_\_\_ 1. In Rule Designer, switch to a new workspace:
  - \_\_\_ a. From the **File** menu, click **Switch Workspace > Other**.
  - \_\_\_ b. In the Workspace Launcher window, enter the path:  
`<LabfilesDir>\workspaces\testing`
- \_\_\_ 2. Close the Welcome view.
- \_\_\_ 3. Import the lab projects.
  - \_\_\_ a. From the **File** menu, click **Import**.
  - \_\_\_ b. On the Select page, select **Operational Decision Manager > Samples and Tutorials**, and click **Next**.
  - \_\_\_ c. In the **Rule Designer > Training** node, select **Ex 14: Enabling tests and simulations > 01-start** and click **Finish**.
- \_\_\_ 4. After the workspace builds, close the Help view.



**Hint**

In this exercise, you must write some code. You can find the code snippets to create in the `<LabfilesDir>\code\enable_testing.txt` file, and you can copy and paste them in Rule Designer.

### 1.2. Validating the rule project

- \_\_\_ 1. In Rule Explorer, make sure the `loan-rules` project is selected so that the Rule Project Map opens.
- \_\_\_ 2. On the Decision Service Map, in the **Define decision operation** part of the map, click **Go to operation map**, select `loan-rulesOperation.dop`, and click **OK**.
- \_\_\_ 3. In the **Design signature** part of the map, click **Check for testing**.



- \_\_\_ 4. When prompted by the Decision Operation Selection window, click **Finish**.

An error is reported.

- \_\_\_ 5. Click **OK** to close the error message.

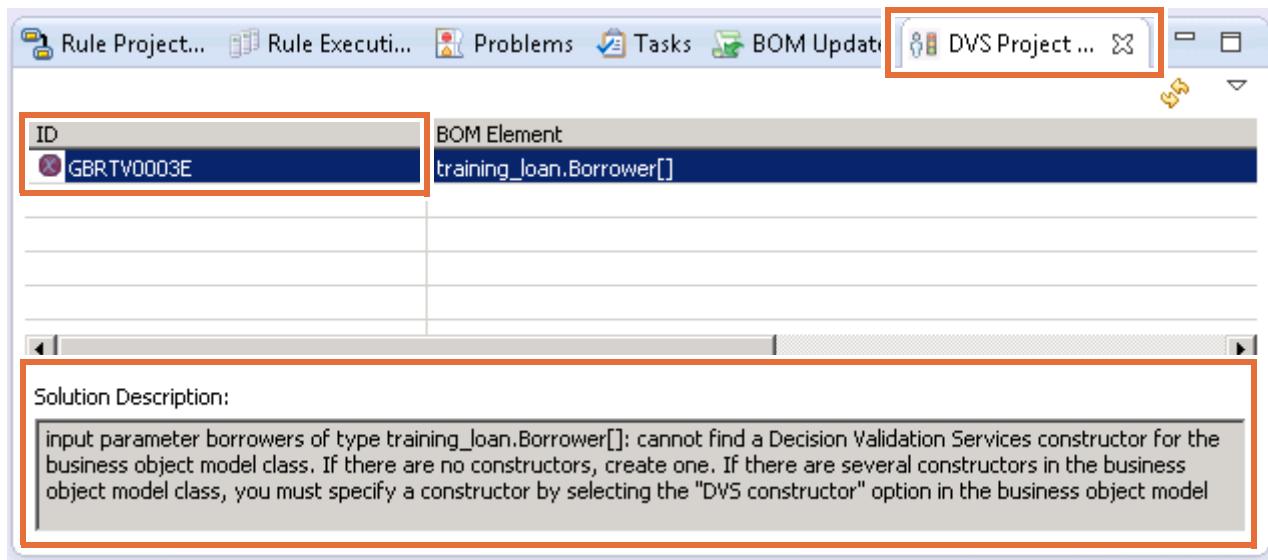


### Note

You can also check the project by right-clicking the `loan-rules` project in Rule Explorer, and clicking **Decision validation service > Check Project**.

- \_\_\_ 6. In the DVS Project Validation view, select the error to open and read the Solution Description, which reads:

`input parameter borrowers of type training_loan.Borrower[]: cannot find a Decision Validation Services constructor for the business object model class. If there are no constructors, create one. If there are several constructors in the business object model class, you must specify a constructor by selecting the "DVS constructor" option in the business object model`



An error is raised because no class constructor was defined or specified as the Decision Validation Services constructor.

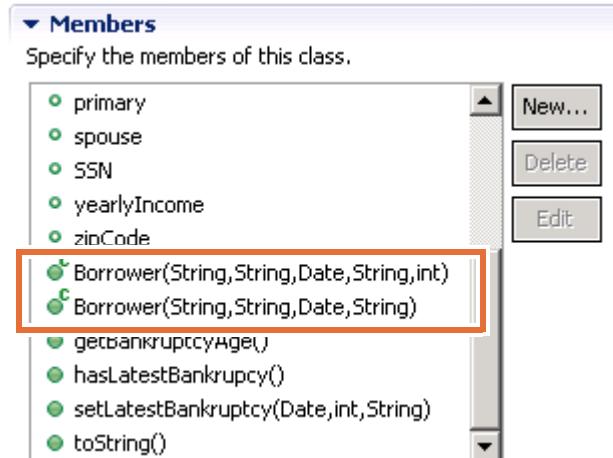
### 1.3. Choosing a constructor

BOM classes that are used to define ruleset input parameters must have at least one constructor. One of these constructors must be specified as the constructor for Decision Validation Services. The columns of the Microsoft Excel scenario file template then correspond to the argument of this BOM class constructor.

## Choosing the correct constructor

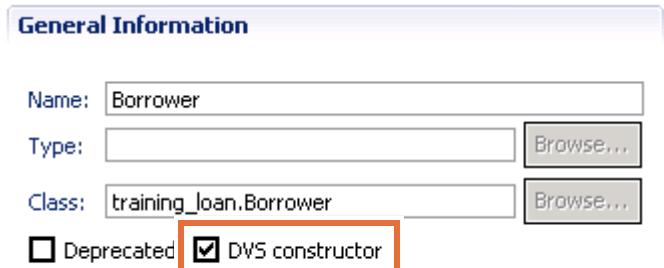
1. In the Rule Explorer, expand **loan-rules > bom > model > training\_loan** and double-click **Borrower** to open the BOM editor.

The Class page opens for the **Borrower** class. Scroll through the Members list and notice that **Borrower** has two constructors.



When a class has multiple constructors, you must specify which one to use for the template.

2. In the **Members** section of the Borrower class page, double-click the first constructor, **Borrower(String, String, Date, String, int)**.
3. On the Member page, select the **DVS constructor** check box.

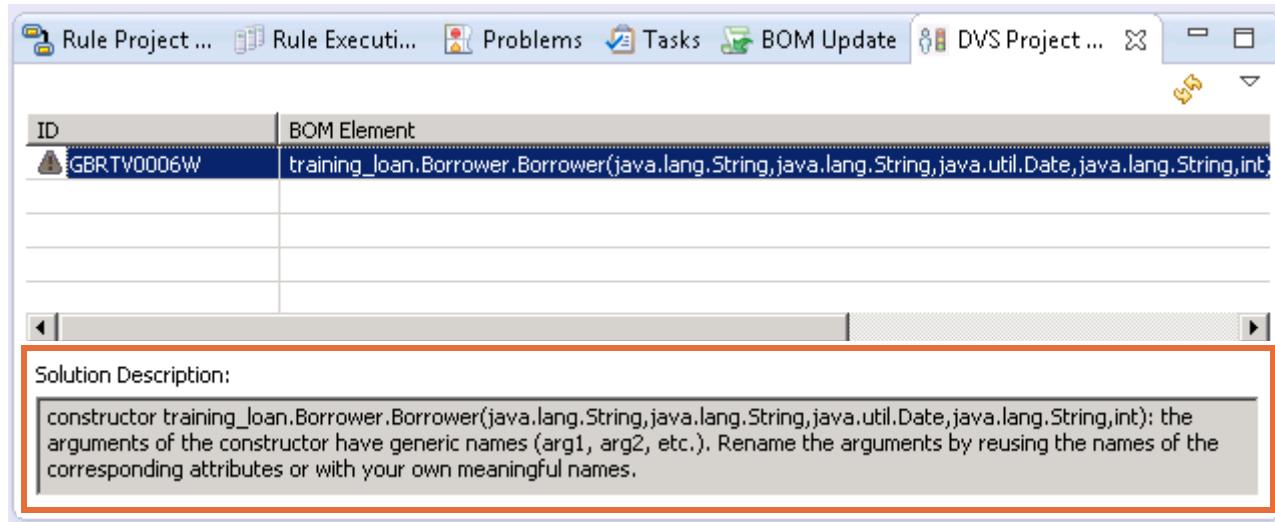


4. Save your work.
5. Check the project again by right-clicking **loan-rules** and clicking **Decision Validation Services > Check Project**, and click **Finish** when prompted to select the **loan-rulesOperation**.

This time, you see a warning message.

6. Click **OK** to close the warning.

In the DVS Project Validation view, when you select the warning to see that the constructor arguments use generic names that you must rename with meaningful names.



The arguments (arg1, arg2, arg3, arg4, and arg5) correspond to the arguments of the selected constructor:

Borrower (String, String, Date, String, int)

- \_\_\_ 7. Double-click the warning message to open the Borrower page and edit the constructor argument names.

**Arguments**

Edit the arguments of this member.

| Name | Type             | Domain |
|------|------------------|--------|
| arg1 | java.lang.String |        |
| arg2 | java.lang.String |        |
| arg3 | java.util.Date   |        |
| arg4 | java.lang.String |        |
| arg5 | int              |        |

## 1.4. Editing argument names for column headings in the template

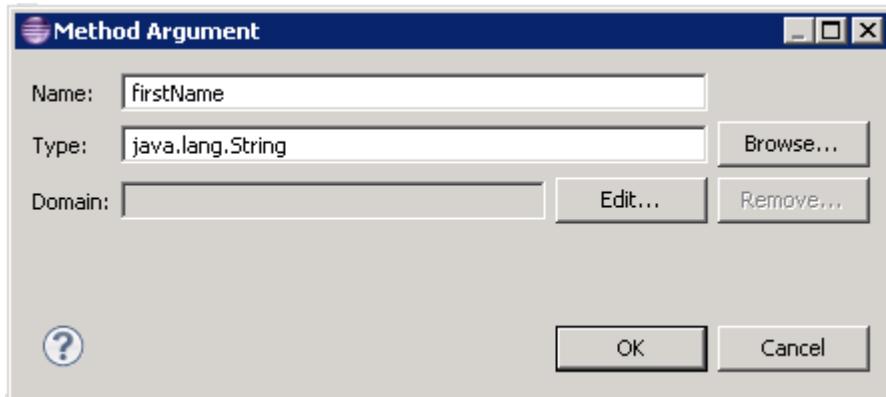
The argument names become the column headings in the scenario file template. You should edit the argument names to make them meaningful for the business users who use the scenario file.

When you rename the generic arguments, make sure that the names match the corresponding class attribute names. For example, if arg1 sets the value of the firstName attribute, then rename arg1 as: firstName

### Renaming the constructor arguments for column headings

- \_\_\_ 1. In the **Arguments** section of the Borrower class page, double-click arg1.

- 2. In the **Name** field, change `arg1` to `firstName` and click **OK**.



The name must match the corresponding attribute name in the `Borrower` class.

When the template is generated, the column headings display the verbalized name of the corresponding attribute or argument. For example, the generated template column for the `firstName` attribute uses the heading: first name

- 3. Edit the names of the remaining arguments to these attribute names:

- `secondName`
- `birthDate`
- `ssn`
- `creditScore`

- 4. Save your work.

**Arguments**  
Edit the arguments of this member.

| Name        | Type             | Domain |
|-------------|------------------|--------|
| firstName   | java.lang.String |        |
| secondName  | java.lang.String |        |
| birthDate   | java.util.Date   |        |
| ssn         | java.lang.String |        |
| creditScore | int              |        |
|             |                  |        |
|             |                  |        |
|             |                  |        |

- 5. Check the project again (by right-clicking `loan-rules`, clicking **Decision Validation Services > Check Project**, and clicking **Finish** when prompted to select a decision operation).

You see a message that the project is now compliant.

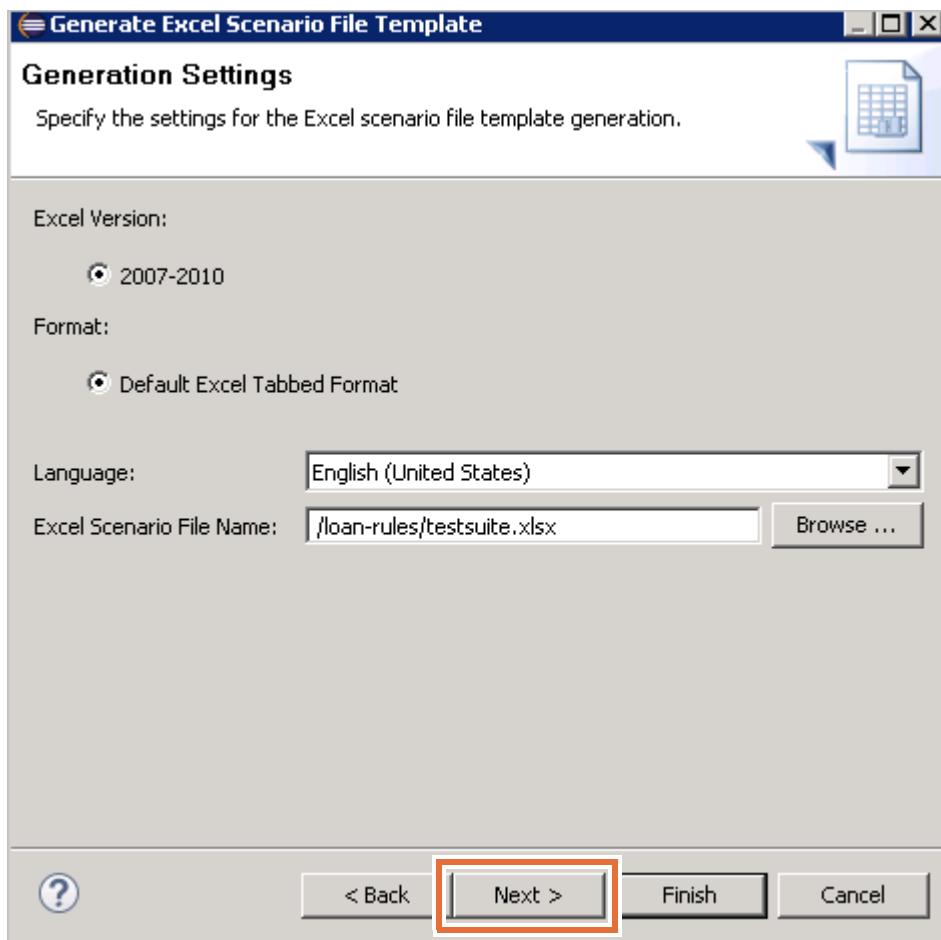
- a. Click **OK** to close the DVS BOM Validation window.

## Section 2. Generating the scenario file template

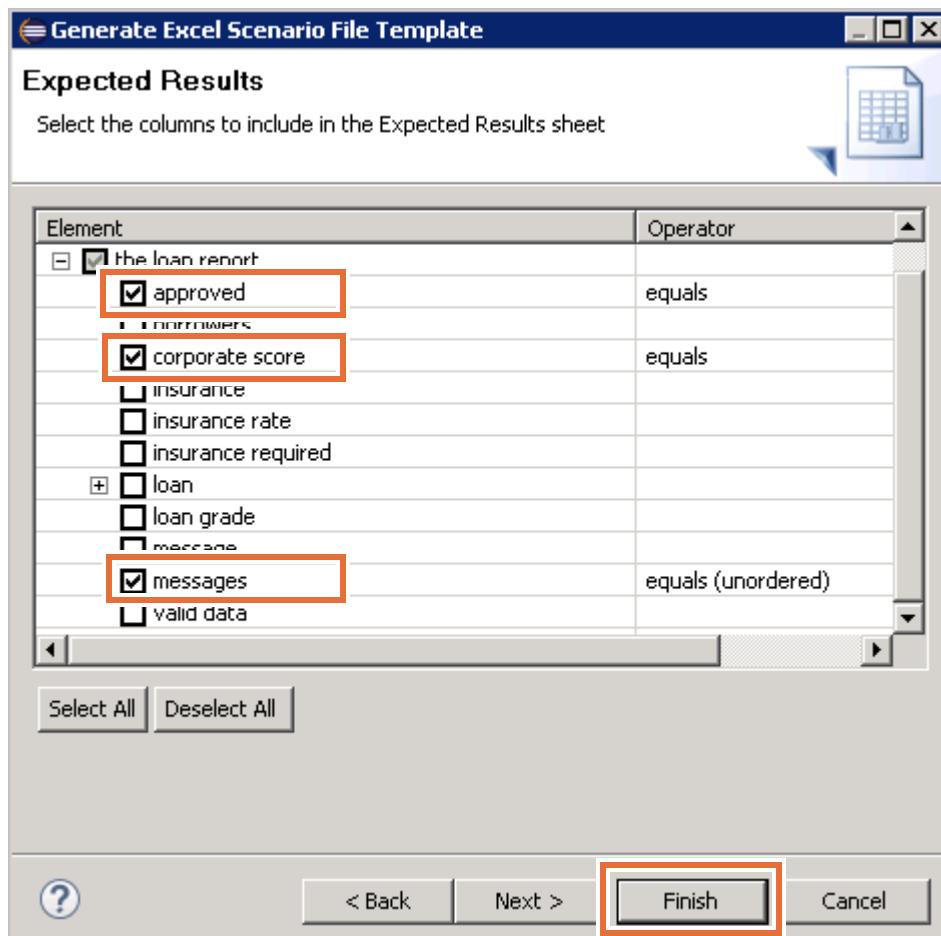
Decision Validation Services uses Microsoft Excel 2007 or Microsoft Excel 2010 as the default format for scenario file templates. You can populate the template manually or by using data from a database.

### Generating the Excel scenario file template

- \_\_\_ 1. In the Rule Explorer, right-click `loan-rules` and click **Decision Validation Services > Generate Excel Scenario File Template**.
- \_\_\_ 2. In the Generate Excel Scenario File Template window, select the following values.
  - \_\_\_ a. On the Rule Project page, make sure that `loan-rules` is selected and click **Next**.
  - \_\_\_ b. On the Decision Operation Selection page, make sure that the **loan-rulesOperation** is selected and click **Next**.
  - \_\_\_ c. On the Generation Settings page, make sure that:
    - **2007-2010** is selected as the **Excel version** option.
    - **Default Excel Tabbed Format** is selected as the **Format** option.
  - \_\_\_ d. Leave the remaining fields on the Generation Settings page unchanged, and click **Next**.



- \_\_\_ e. On the Expected Results page, expand **the loan report**, which is one of the output ruleset parameters, and select the attributes that you want to test:
- approved
  - corporate score
  - messages



### Warning

Make sure that you select the **messages** attribute, with the final “s”, not the **message** field.

These attributes are transformed into columns on the **Expected Results** sheet of the generated scenario file template. You must then complete these columns with the values of these attributes that you expect to be produced at run time. The values that you enter are then compared with the actual execution results at test time to give the test results.



### Information

For each attribute, you can select the operator that is used for the test. The possible operators depend on the type of the attribute. The default operator is the equality operator, meaning that the test succeeds when the value entered in the `testsuite.xlsx` equals the value that is obtained at run time. You might use other operators such as does not equal, is greater than, is lower than, or contains.

- \_\_\_ f. On the Expected Results page, click **Next**.
  - \_\_\_ g. On the Expected Execution Details page, click **Finish**.  
The `testsuite.xlsx` file becomes available in the `loan-rules` project in the Rule Explorer.
- \_\_\_ 3. In the Rule Explorer, in the `loan-rules` project, right-click `testsuite.xlsx` and click **Open With > System Editor** to open it with Microsoft Excel Viewer.
- \_\_\_ 4. Explore the `testsuite.xlsx` scenario file template:
- \_\_\_ a. Note the sheets that are generated: the **Scenarios** sheet, the **Expected Results** sheet, and a separate sheet with columns for the `Borrower` class attributes.
  - \_\_\_ b. Notice that the column headings on each of the sheets are the verbalized names of the classes and attributes.
  - \_\_\_ c. Close the test suite when you are finished.

## Section 3. Populating the template and testing the scenario

The scenario file template can be used to store up to thousands of scenarios. For tests and simulations that require a large number of scenarios, you can import the data from a database.



### Note

For this exercise, you use a prepopulated scenario file.

If you have access to Microsoft Excel on your computer, you can modify the scenario file template directly instead of using the prepopulated file.

### 3.1. Viewing the prepopulated scenario file

- \_\_\_ 1. In the Rule Explorer, expand the `data` folder in the `loan-rules` project.
  - \_\_\_ 2. Right-click the `testsuite-populated1.xlsx` file, and click **Open With > System Editor** to open it with Microsoft Excel Viewer.
- A set of values and expected results are provided for one scenario. In the **Scenarios** sheet, notice that the `monthlyRepayment` and the `yearlyInterestRate` columns are empty. The values for these attributes are calculated as a result of rule execution, so the columns remain empty.
- \_\_\_ 3. Close the `testsuite-populated1.xlsx` file.

### 3.2. Running the tests with the populated scenario file

- \_\_\_ 1. From the menu, click **Run > Run Configurations**.
- \_\_\_ 2. In the Run Configurations wizard, double-click **DVS Decision Operation** to create a configuration.
- \_\_\_ 3. In the **Name** field of the configuration, enter: `loan local test`
- \_\_\_ 4. On the **Excel File** tab, select the following values.
  - \_\_\_ a. For the **Excel File** field, click **Browse** to select `loan-rules > data > testsuite-populated1.xlsx`, and click **OK**.
  - \_\_\_ b. For the **Decision Operation** field, click **Browse** to select `loan-rules > loan-rulesOperation`, and click **OK**.
  - \_\_\_ c. For the **HTML Report** field, enter:  
`\loan-rules\data\report1`
- \_\_\_ 5. Click **Apply** and click **Run**.

The Console view shows the execution progress, and you should see the following traces after execution ends:

```
--- Output for scenario 'Scenario 1' :
Borrower(firstName: John lastName: Doe born: Dec 1, 1965 SSN: 111-11-1111
isPrimary: true zip code: 93210 income: 60000 credit score: 700)
Loan(amount: 100000 starting: July 1, 2014 duration: 72 month LTV: 70% rate:
5.7% monthly repayment: 1,643.16)
Report(validData: true approved: true score: 870 grade: B insuranceRequired:
true insuranceRate: 2% message: Low risk loan
Congratulations! Your loan has been approved
)
Execution finished
Starting generation of DVS HTML report now
Finished generating DVS HTML report
```

- \_\_\_ 6. In the Rule Explorer, right-click **loan-rules > data > report1.html** and click **Open With > Web Browser** to see the successful test results.



### Troubleshooting

---

If you do not see the `report1.html` file, refresh the **loan-rules > data** folder in Rule Explorer by selecting the **data** folder and pressing F5.

- \_\_\_ 7. When you are finished, close the report window.

## Section 4. Customizing input for the scenario file template

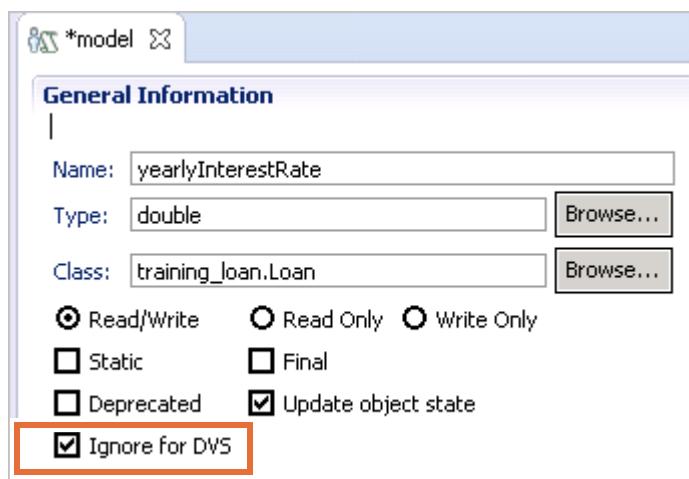
Because of the relationship between the BOM and the scenario file, customizing the scenario file template requires modifying the BOM.

### 4.1. Removing columns from the template

When preparing the BOM to generate a template, you can exclude BOM members that are not useful as input to tests or simulations. For example, attributes that are based on calculated values should not be included as input columns because they do not have input values.

#### Removing columns

- 1. In the Rule Explorer, expand **loan-rules > bom > model > training\_loan**, and double-click **Loan** to open it in the BOM editor.
  - 2. In the **Members** section of the Class page, double-click **yearlyInterestRate** to open it on the Member page.
- On the Member page for the `yearlyInterestRate` attribute, you can see that **Ignore for DVS** is not selected, meaning that this attribute is included in the scenario file template.
- 3. Select **Ignore for DVS**.



- 4. Repeat these steps to ignore the `monthlyRepayment` attribute of the `Loan` class.
- 5. Save your work.

No input columns are created for these attributes in the **Scenarios** sheet of the template. However, when you generate the template, you can still choose to include these attributes in the **Expected Results** sheet to test that their values are calculated correctly as a result of rule execution.

## 4.2. Creating a virtual attribute to test complex objects



### Requirements

The business users want test results to indicate whether the borrower that is included in the loan report is the primary borrower. The `training_loan.Borrower` BOM member has a Boolean attribute called: `primary`

However, this `primary` attribute is not directly accessible from the `training_loan.Report` BOM member, which is returned as an output ruleset parameter.

To handle this request, you create a virtual attribute in the `Report` class that makes the `Borrower.primary` attribute visible in the test results.

### Viewing the Borrower.primary attribute

- 1. In the Rule Explorer, expand `loan-rules > bom > model > training_loan` and double-click **Borrower** to open it in the BOM editor.
- 2. Notice that the `primary` attribute is listed in the **Members** section of the Class page.

### Viewing the Report.borrowers attribute

- 1. In the Rule Explorer, expand `loan-rules > bom > model > training_loan` and double-click **Report** to open it in the BOM editor.
- 2. In the **Members** section of the Class page, double-click the `borrowers` attribute to open it.

#### Member borrowers (class: `training_loan.Report`)

| General Information                                                                                                                                                                                                                                                                                           |                                                                                                                                                                        |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name:                                                                                                                                                                                                                                                                                                         | <input type="text" value="borrowers"/>                                                                                                                                 |
| Type:                                                                                                                                                                                                                                                                                                         | <input type="text" value="training_loan.Borrower[]"/> <span style="border: 2px solid red; padding: 2px;">(highlighted)</span> <input type="button" value="Browse..."/> |
| Class:                                                                                                                                                                                                                                                                                                        | <input type="text" value="training_loan.Report"/> <input type="button" value="Browse..."/>                                                                             |
| <input type="radio"/> Read/Write <input checked="" type="radio"/> Read Only <input type="radio"/> Write Only<br><input type="checkbox"/> Static <input type="checkbox"/> Final<br><input type="checkbox"/> Deprecated <input type="checkbox"/> Update object state<br><input type="checkbox"/> Ignore for DVS |                                                                                                                                                                        |

- 3. On the Member page, click **Browse** next to the **Type** field, and notice that `borrowers` is an array of `Borrower` objects.

To access the `primary` attribute for each `Borrower` class in the `training_loan.Report.borrowers` collection, you must create a virtual attribute.

## Creating the virtual attribute in the Report class

- \_\_\_ 1. Go back to the **Class** tab for the `Report` class in the BOM editor.
- \_\_\_ 2. Create a virtual read-only attribute for the `Report` class:
  - \_\_\_ a. In the **Members** section, click **New**.  
The New Member window opens.
  - \_\_\_ b. Make sure that the **Attribute** option is selected.
  - \_\_\_ c. In the **Name** field, enter: `borrowerPrimary`
  - \_\_\_ d. In the **Type** field, enter: `string`
  - \_\_\_ e. Click **Finish**.

The `borrowerPrimary` attribute is now listed in the **Members** section.
- \_\_\_ 3. Double-click **borrowerPrimary** to open it on the Member page and define the BOM-to-XOM mapping for the new virtual attribute.
  - \_\_\_ a. On the Member page, select **Read Only**.
  - \_\_\_ b. In the **Member Verbalization** section, click **Create a default verbalization**.
  - \_\_\_ c. Click **Edit the subject used in phrases**.
  - \_\_\_ d. Change the verbalization in the **Singular** field from `borrower primary` to `primary borrower` and click **OK**.
  - \_\_\_ e. In the **BOM to XOM Mapping** section, type the following code in the **Getter** field:

```
StringBuilder builder = new StringBuilder();
for(int i = 0; i < this.borrowers.length; i++) {
 builder.append(this.borrowers[i].primary + ",");
}
return builder.toString();
```



### Hint

You can find this code snippet in the `<LabfilesDir>\code\enable_testing.txt` file.



### Note

Notice the comma “,” that is added within the loop to separate the values for the different elements in the array. For each `borrower`, the appended text is either “true,” or “false.”. The returned String is thus a series of Boolean values that are followed with a comma.

- \_\_\_ 4. Save your work.

### 4.3. Regenerating the scenario file template

Now that you customized the BOM, regenerate the scenario file template.

- 1. In the Rule Explorer, right-click **loan-rules** and click **Decision Validation Services > Generate Excel Scenario File Template**.
- 2. On the Rule Project page, make sure that **loan-rules** is selected and click **Next**.
- 3. Make sure that **loan-rulesOperation** is selected and click **Next**.
- 4. On the Generation Settings page, change the value in the **Excel Scenario File Name** field to **/loan-rules/testsuite2.xlsx** and click **Next**.
- 5. On the Expected Results page, expand **the loan report**, and select the following criteria.
  - **approved**
  - **corporate score**
  - **messages**
  - **primary borrower**

#### Expected Results

Select the columns to include in the Expected Results sheet



| Element                                             | Operator           |
|-----------------------------------------------------|--------------------|
| <input type="checkbox"/> the loan report            |                    |
| <input checked="" type="checkbox"/> approved        | equals             |
| + <input type="checkbox"/> borrowers                |                    |
| <input checked="" type="checkbox"/> corporate score | equals             |
| <input type="checkbox"/> insurance                  |                    |
| <input type="checkbox"/> insurance rate             |                    |
| <input type="checkbox"/> insurance required         |                    |
| + <input type="checkbox"/> loan                     |                    |
| <input type="checkbox"/> loan grade                 |                    |
| <input type="checkbox"/> messages                   | equals (unordered) |
| <input checked="" type="checkbox"/> messages        | equals             |
| + <input type="checkbox"/> primary borrower         |                    |
| + <input type="checkbox"/> yearly data              |                    |



#### Warning

Make sure that you select the **messages** attribute, with the final “s”, not the **message** field.

- f. Click **Finish**.
- 6. In Rule Explorer, in the **loan-rules** project, right-click the **testsuite2.xlsx** file, and click **Open With > System Editor** to open it with Microsoft Excel Viewer.
- 7. Notice that the columns **monthly repayment** and **yearly interest rate** are no longer visible in the Scenarios spreadsheet.

Your new template is ready to be populated with scenario values and expected results.



### Information

Again, for this exercise, you use the prepopulated scenario file that is called `testsuite-populated2.xlsx`, which is in the `data` folder. A set of values and expected results are provided for one scenario.

If you open the populated file, notice that the value given for the column `the primary borrower of the loan report equals` is “true,” with the final comma.

- 8. Close the `testsuite2.xlsx` file.

#### 4.4. Testing the customized scenario file

- 1. Run the DVS Excel file launch configuration with the prepopulated scenario:
  - a. From the **Run** menu, click **Run Configurations**.
  - b. In the **Run Configurations** wizard, select **DVS Decision Operation > loan local DVS**.
  - c. In the **Excel file** field, browse to select **loan-rules > data > testsuite-populated2.xlsx** and click **OK**.
  - d. Change the report name in the **HTML Report** field to:  
`\loan-rules\data\report2`
  - e. Click **Apply** to save your changes, and then click **Run**.
  - f. After execution, refresh the `data` folder in Rule Explorer (by pressing F5).

- \_\_\_ g. Open the updated `report2.html` file by right-clicking it and clicking **Open With > Web Browser**.

| Name                       | Success Rate | Tests | Failures | Errors | Message |
|----------------------------|--------------|-------|----------|--------|---------|
| <a href="#">Scenario 1</a> | 100%         | 4     | 0        | 0      |         |

### Details by Scenario

#### Scenario 1

| Name                                               | Success | Failure | Error | Message                                                                                                                                                                              |
|----------------------------------------------------|---------|---------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| the loan report is approved equals ✓               |         |         |       | The observed value true is the expected value.                                                                                                                                       |
| the corporate score in the loan report equals      | ✓       |         |       | The observed value 870 is the expected value.                                                                                                                                        |
| the messages of the loan report equals (unordered) | ✓       |         |       | The set of values [Low risk loan, Congratulations! Your loan has been approved] equals (unordered) the expected values [Low risk loan, Congratulations! Your loan has been approved] |
| the primary borrower of the loan report equals     | ✓       |         |       | The observed value true, is the expected value.                                                                                                                                      |

The report shows 100% success and proves that the value for the virtual `borrowerPrimary` attribute was accessed successfully.

- \_\_\_ 2. When you are finished viewing the report, you can close the report window and any remaining windows that are open in Rule Designer.

## Section 5. Enabling testing from Business console

In this section, you deploy the XOM to the sample server so that business users run tests and simulations from Business console by accessing the sample server.

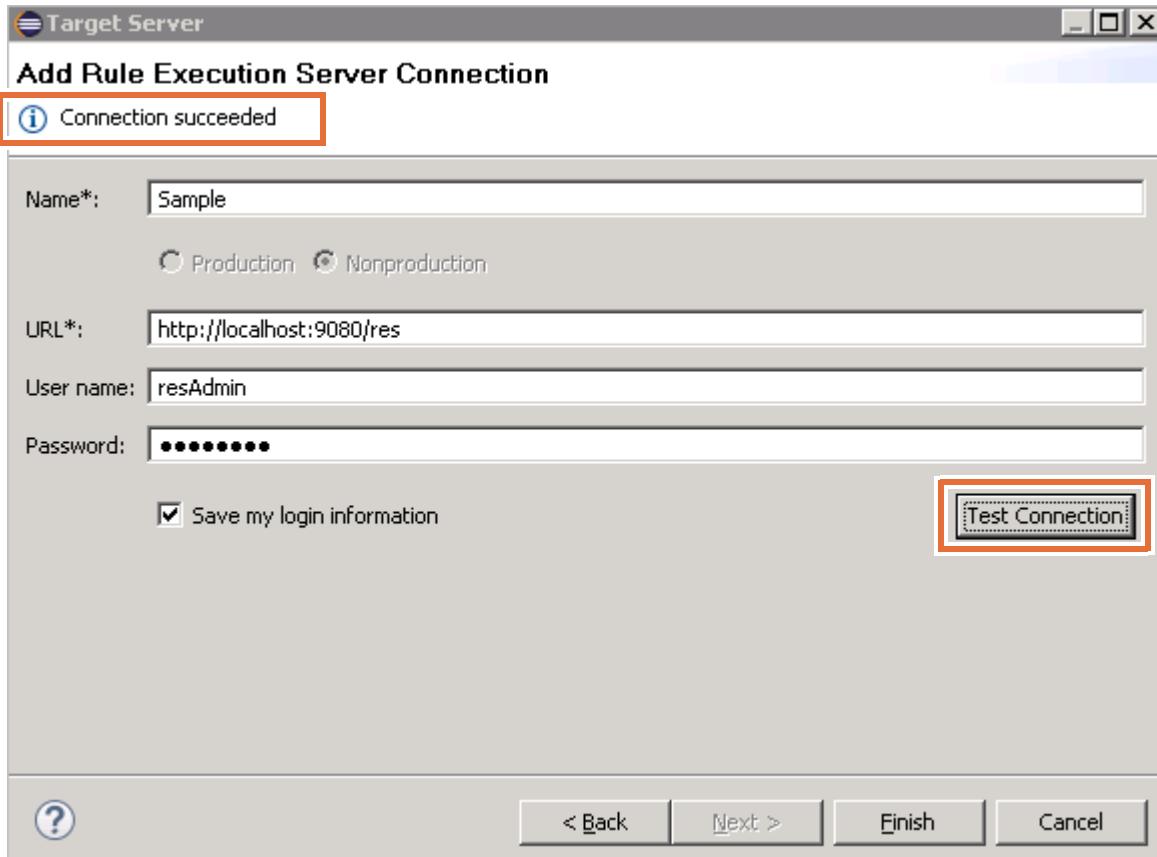
- \_\_\_ 1. In Rule Explorer, expand **loan-rules > deployment** and double-click **loan-test**, which is the deployment configuration that defines which server to deploy to.
- \_\_\_ 2. Click the **Target Servers** tab and define the target server.
  - \_\_\_ a. Click the plus sign to add a server.
  - \_\_\_ b. Leave **Create a Rule Execution Server connection** selected and click **Next**.
  - \_\_\_ c. Define these fields:
    - **Name:** Sample
    - **URL:** `http://localhost:9080/res`
    - **User name:** resAdmin
    - **Password:** resAdmin

### For IBM ODM on Cloud users

Use the name and URL of the Rule Execution Server for your IBM ODM on Cloud instance in the **URL** field instead of `Sample` and `localhost`.

- \_\_\_ d. Select **Save my login information**.

- \_\_ e. Click **Test Connection**.



You should see a Connection succeeded message.

- \_\_ f. Click **Finish**.

- \_\_ 3. Save your work.  
\_\_ 4. Close the Target Servers window.  
\_\_ 5. In Rule Explorer, right-click **loan-rules** and select **Decision Validation Services > Deploy XOM for testing from the Business Console**.  
\_\_ 6. In the XOM Deployment window, click **Finish**.

In the Deployment Report window, you see a message that the XOM deployment is successful.

- \_\_ 7. Close the Deployment Report window.  
\_\_ 8. Publish the decision service to Decision Center.  
\_\_ a. In Rule Explorer, right-click **loan-rules** and click **Decision Center > Connect**.  
\_\_ b. Enter the following values in the Decision Center Configuration window fields.
  - **URL:** http://localhost:9080/teamserver
  - **User name:** rtsAdmin
  - **Password:** rtsAdmin
  - **Data source:** (Leave the field empty)

- \_\_\_ c. Click **Connect**.
  - \_\_\_ d. Click **Finish**.
9. When prompted to switch to the Synchronizing view, click **No** (because no conflicts exist).
10. Click **OK** to close the Synchronize Complete window.

## Section 6. Testing from Business console

- \_\_\_ 1. Open Business console by double-clicking the shortcut on your desktop and sign in with the rtsAdmin for the user name and password.
- \_\_\_ 2. Make sure that you are on the **Library** tab.
- \_\_\_ 3. Click the loan-rules decision service and click **main** to open the main branch.
- \_\_\_ 4. Click the **Tests** tab and click the plus sign (+) to create a test suite.
- \_\_\_ 5. In the **File to use** section, click **Choose** and browse to the testsuite-populated2.xlsx file in your workspace.  
By default, the path is: <LabfilesDir>\workspaces\testing\loan-rules\data
- \_\_\_ 6. Click **Save and Run**.
- \_\_\_ 7. When prompted, click **Create New Version**, and then click **OK**.
- \_\_\_ 8. Click the report link to see that the tests ran successfully.
- \_\_\_ 9. Log out of the Business console and close the web browser window.

### End of exercise

## Exercise review and wrap-up

To see the solution to this exercise, switch to a new workspace and import the project **Ex 14: Enabling tests and simulations > 02-answer**.

The first part of the exercise looked at how to set up the testing environment for business users by preparing the BOM and creating a customized scenario file template.

The second part of the exercise looked at how you can set up the test environment for business users to run tests from Decision Center.



# Exercise 15. Managing deployment

## What this exercise is about

This exercise teaches you how to deploy rules and XOMs for managed execution with Rule Execution Server.

## What you should be able to do

After completing this exercise, you should be able to:

- Define a RuleApp and ruleset properties
- Use deployment configurations to deploy decision services
- Deploy the XOM for its management in Rule Execution Server

## Introduction

After finalizing the content of the rule project, you are ready to deploy it to the execution environment in Rule Execution Server.

In this exercise, you create a RuleApp to package your ruleset for deployment to Rule Execution Server. You learn how to deploy from Rule Designer and how to manage the XOM in Rule Execution Server.

This exercise is divided into these sections:

- Section 1, "Creating deployment configurations"
- Section 2, "Deploying a RuleApp from Rule Designer"
- Section 3, "Adding a ruleset property to a deployment configuration"
- Section 4, "Deploying the managed XOM from Rule Designer"
- Section 5, "Exporting and importing deployment server definitions"

## Requirements

This exercise uses the following files, which are installed in the `<InstallDir>\studio\training` directory.

- Start project: Dev 15 - Deployment\01-start
- Solution project: Dev 15 - Deployment\02-answer
  - You can use the solution project in "Exercise review and wrap-up" on page 15-18.

## Section 1. Creating deployment configurations

In this part of the exercise, you create a deployment configuration that can be used to deploy the decision service to Rule Execution Server or for a Java SE environment.

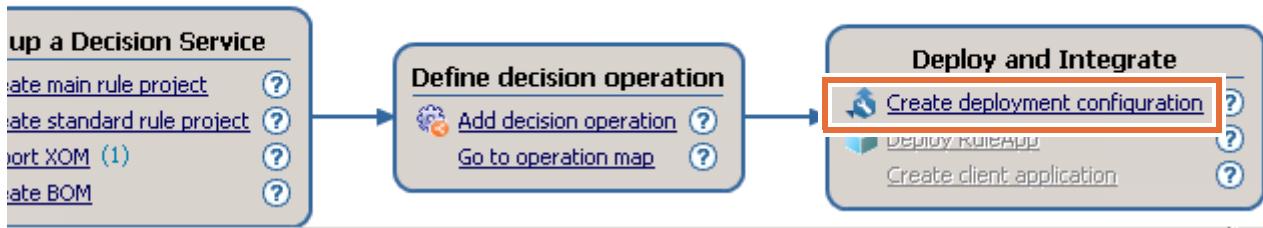
### 1.1. Setting up your environment for this exercise

- 1. If the sample server is not started, start it now by double-clicking the **Start server** shortcut on the desktop.  
Starting the server might take several minutes.
- 2. In Rule Designer, switch to a new workspace:
  - a. From the **File** menu, click **Switch Workspace > Other**.
  - b. In the Workspace Launcher window, enter the path:  
*<LabfilesDir>\workspaces\deployment*
- 3. Close the Welcome view.
- 4. Import the lab projects.
  - a. From the **File** menu, click **Import**.
  - b. On the Select page, select **Operational Decision Manager > Samples and Tutorials**, and click **Next**.
  - c. In the **Rule Designer > Training** node, select **Ex 15: Managing deployment > 01-start** and click **Finish**.
- 5. After the workspace builds, close the Help view.

### 1.2. Creating a deployment configuration

- 1. In the **Rule Project Map** view, in the **Deploy and Integrate** part of the Decision Service Map, click **Create deployment configuration**.

on Service Map

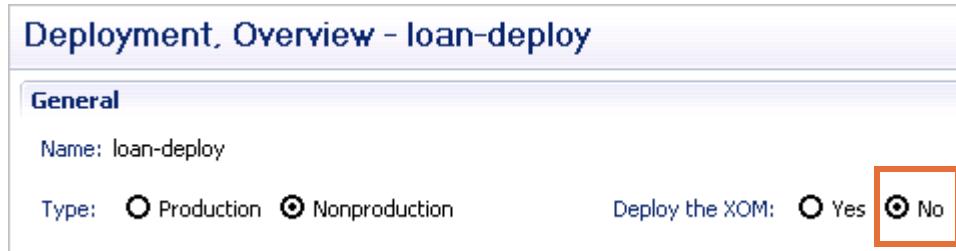


- 2. In the **Deployment folder** field, click **Browse** and select the **loan-rules > deployment** folder and click **OK**.
- 3. In the **Name** field, type `loan-deploy` and click **Finish**.

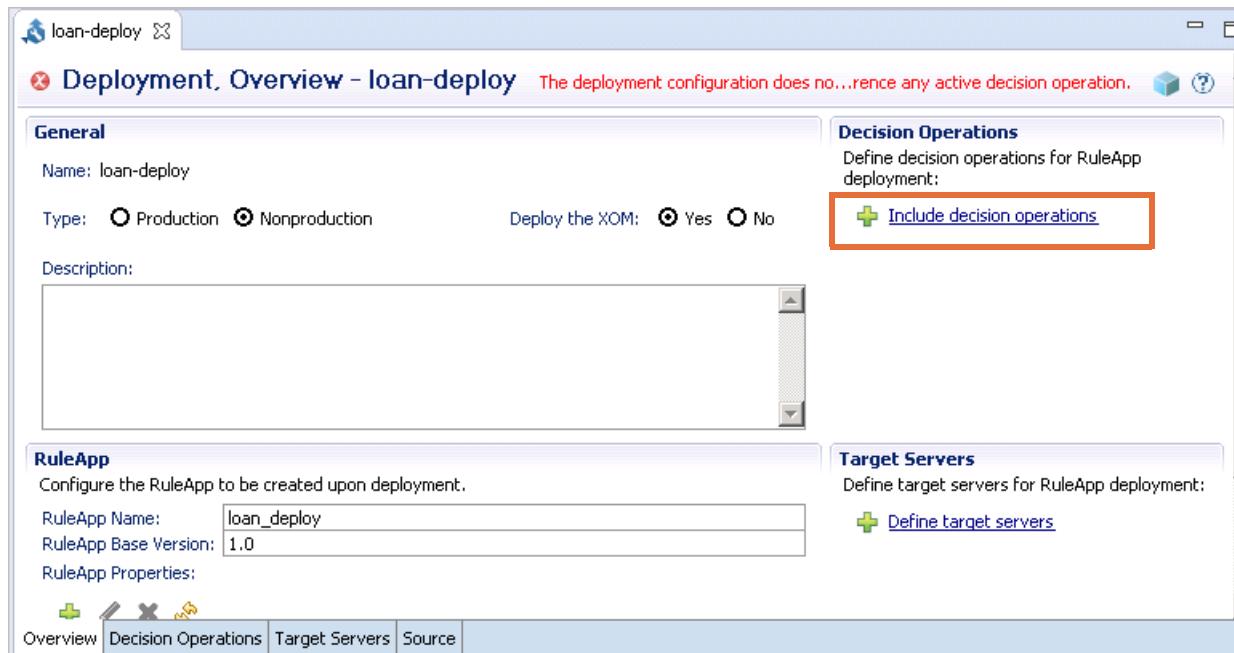
The deployment configuration is now listed in Rule Explorer with errors because the configuration settings are not yet defined. The configuration editor opens to the **Overview**

tab. The editor also includes the **Decision Operations** tab and the **Target Servers** tab, which you use next.

- 4. On the **Overview** tab, in the **General** section, in the **Deploy the XOM** option, choose **No**.



- 5. In the **Decision Operations** section, click **Include decision operations**.



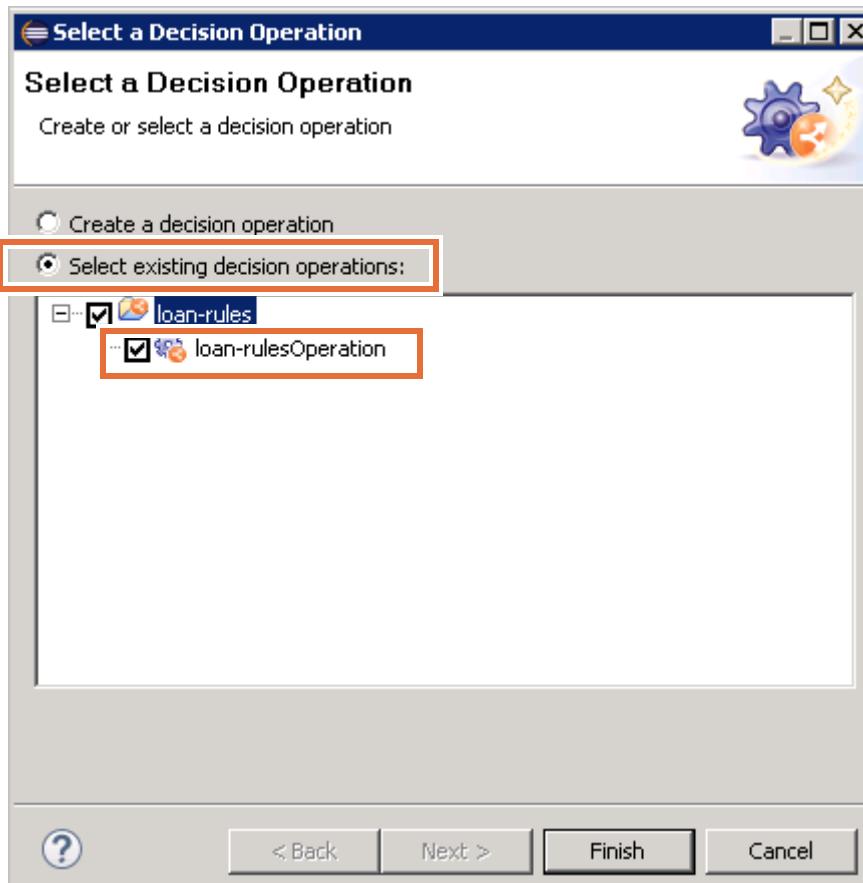
The editor switches to the **Decision Operations** tab.

- 6. Add the `loan-rulesOperation` decision operation to the configuration.

- a. In the **Configured Decision Operations** section, click the plus sign (+).



- \_\_ b. Choose **Select existing decision operations** and expand **loan-rules** to choose **loan-rulesOperation**.



- \_\_ c. Click **Finish**.

- \_\_ 7. Click the **Target Servers** tab and define the target server.

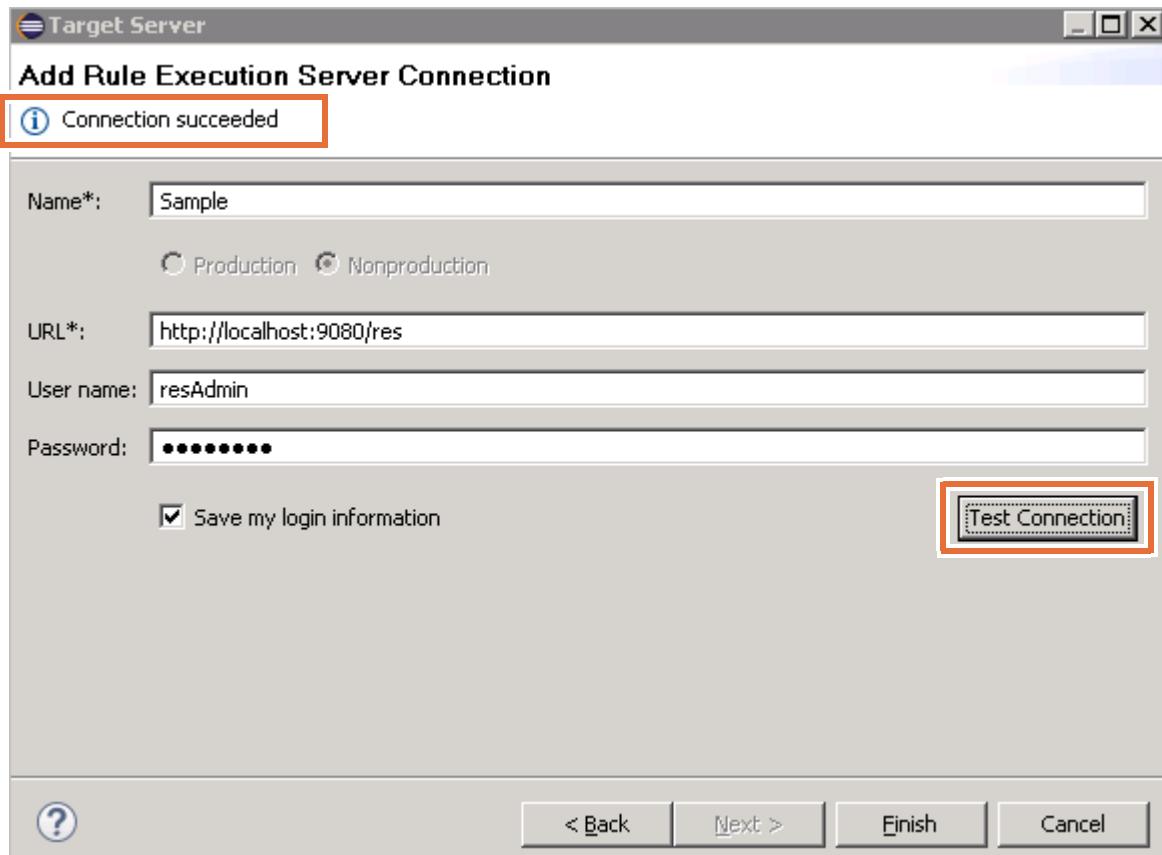
- \_\_ a. Click the plus sign (+) to add a server.
- \_\_ b. Leave **Create a Rule Execution Server connection** selected and click **Next**.
- \_\_ c. Define these fields:
  - **Name:** Sample
  - **URL:** http://localhost:9080/res
  - **User name:** resAdmin
  - **Password:** resAdmin

#### For IBM ODM on Cloud users

Use the name and URL of the Rule Execution Server for your IBM ODM on Cloud instance in the **URL** field instead of **Sample** and **localhost**.

- \_\_ d. Select **Save my login information**.

- \_\_ e. Click **Test Connection**.



You should see a Connection succeeded message.

- \_\_ f. Click **Finish**.
- \_\_ 8. Save your work by pressing Ctrl+S.

## Section 2. Deploying a RuleApp from Rule Designer

In this section, you deploy a RuleApp for its managed execution in Rule Execution Server.

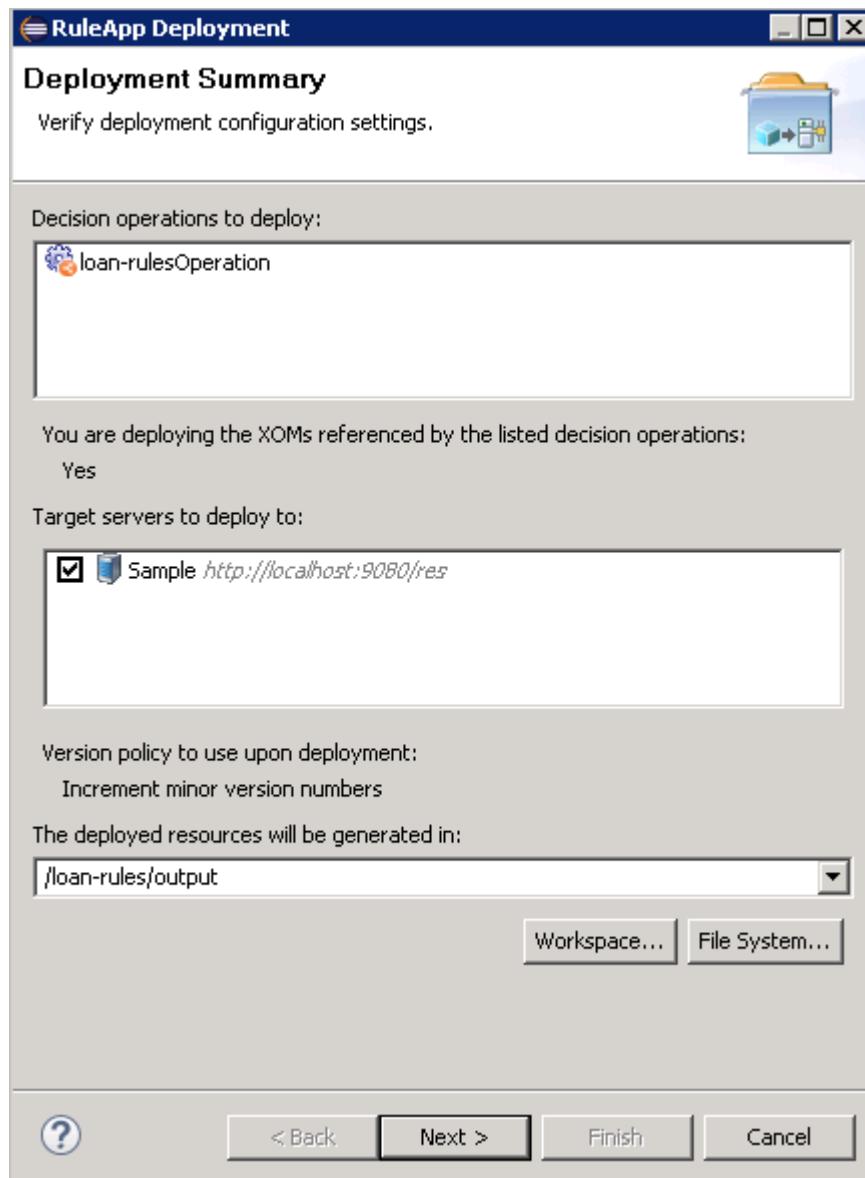
### 2.1. Deploying the RuleApp

In this section, you deploy the RuleApp.

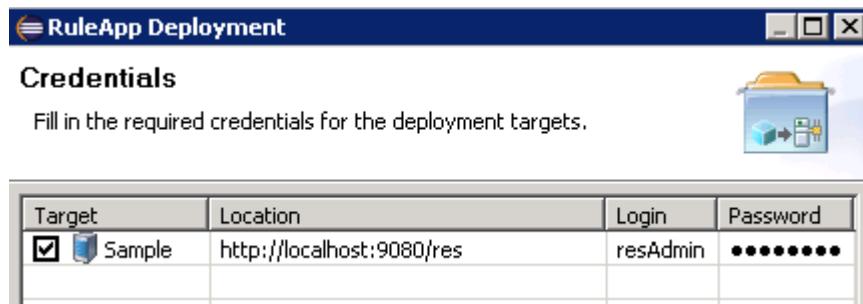
- 1. Return to the **Overview** tab of the `loan-deploy` deployment configuration editor.
- 2. In the **Deployment** section, click **Proceed to RuleApp deployment**.



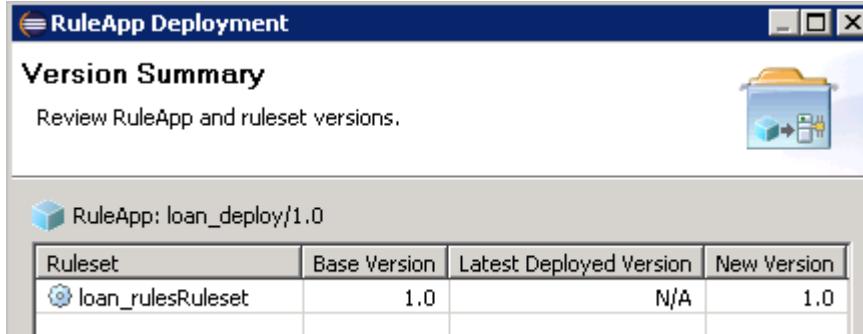
The RuleApp Deployment window opens.



- \_\_\_ 3. After reviewing the deployment summary, click **Next**.



- \_\_\_ 4. After verifying the login credentials for the server, click **Next**.



- \_\_\_ 5. Click **Finish**.

The Deployment Report opens and lists the deployed artifacts.

Deployment Report - loan-deploy

**RuleApp Information - loan-deploy**

- File name: loan\_deploy.jar
- Version: 1.0

**Ruleset Information - loan-rulesOperation**

- File name: loan\_rulesRuleset.jar
- Version: 1.0

**RuleApp Build Status**

- RuleApp file written to: C:\labfiles\workspaces\ex15-jan11\loan-rules\output\loan\_deploy.jar
- RuleApp build finished with no errors.

**RuleApp Deployment**

- The RuleApp was successfully deployed to Sample (http://localhost:9080/res).

**Deployment successful**



### Questions

Do you notice a difference in the name of the deployment configuration? What is the name of the ruleset that was deployed?

### Answer

The hyphen (-) character is not an authorized character in the name of a RuleApp or of a ruleset.

The RuleApp defined with the `loan-deploy` project is thus deployed as a `loan_deploy` file to Rule Execution Server. The same principle applies to the name of the deployed ruleset, which is modified as `loan_rulesRuleset`.



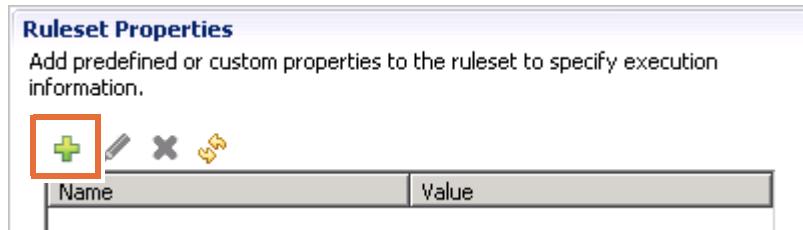
### Important

This `/loan_deploy/1.0/loan_rulesRuleset/1.0` ruleset path, without the hyphen character, is what client applications must use to request the execution of the `loan_rulesRuleset` ruleset.

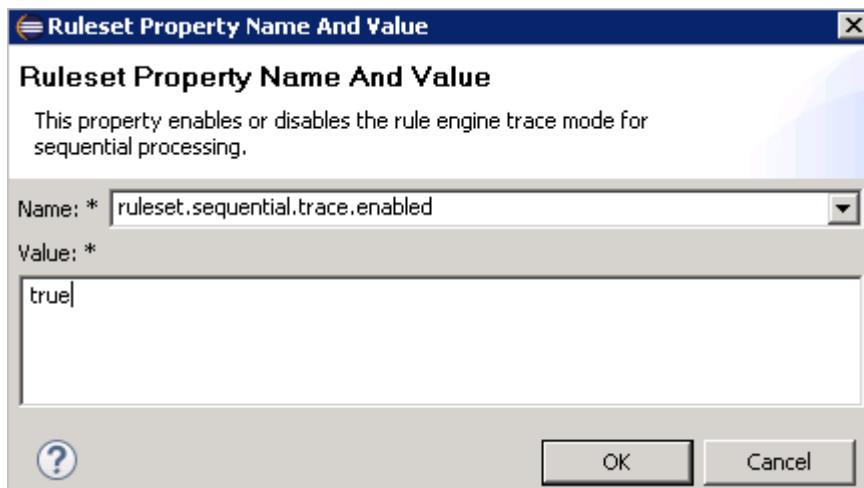
## Section 3. Adding a ruleset property to a deployment configuration

In this section, you add the `rulesetSEQUENTIALTRACEENABLED` ruleset property to the `loan-deploy` deployment configuration.

- \_\_\_ 1. Reopen the `loan-deploy` deployment configuration.
- \_\_\_ 2. Open the **Decision Operations** tab, and in the **Configured Decision Operations** section, select **loan-rulesOperation** with your mouse.
- \_\_\_ 3. In the **Ruleset Properties** section, click the plus sign (+) to add a ruleset property.



- \_\_\_ 4. Define the ruleset property.
  - \_\_\_ a. In the Ruleset Property Name And Value window, in the **Name** field list, scroll through the property list and select the `rulesetSEQUENTIALTRACEENABLED` property. The names in the list are names of predefined ruleset properties. Each ruleset property has a specific role, which is described in the product documentation.
  - \_\_\_ b. In the **Value** field, type: `true`



- \_\_\_ c. Click **OK** to close the Ruleset Property Name And Value window.
- \_\_\_ d. Save your work.
- \_\_\_ 5. Open the **Source** tab to find the added XML code, and verify that the XML definition of the ruleset property is as follows:

```
<properties key="rulesetSEQUENTIALTRACEENABLED">
 <value><![CDATA[true]]></value>
</properties>
```

- \_\_\_ 6. Return to the **Decision Operations** tab and delete the rulesetSEQUENTIALTRACEenabled property by selecting it and clicking the X.



You do not need this property for the rest of this exercise.

- \_\_\_ 7. Save your work (Ctrl+S).

The XML definition of the ruleset property is no longer visible in the XML code of the **Source** tab.

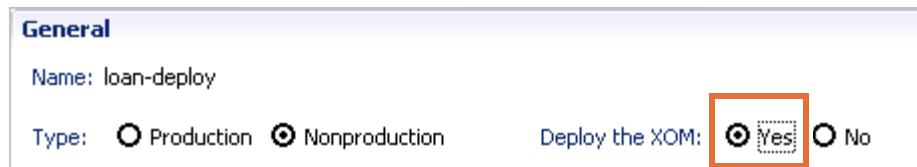
## Section 4. Deploying the managed XOM from Rule Designer

In this part of the exercise, you deploy the XOM that the `loan-rules` project uses. When you deploy the XOM to Rule Execution Server, you can access and manage the XOM through the Rule Execution Server console.

### 4.1. Deploying the managed XOM from the rule project

In this section, you deploy the XOM directly from the rule project.

- \_\_\_ 1. Back in your Rule Designer workspace, in Rule Explorer, expand **loan-rules > resources > xom-libraries** to see the `loan-xom.zip` file.
- \_\_\_ 2. Deploy the XOM that the `loan-rules` project uses.
  - \_\_\_ a. If the `loan-deploy` deployment configuration is closed, reopen it to the **Overview** tab.
  - \_\_\_ b. In the **General** section, in the **Deploy the XOM** option, choose **Yes**.



- \_\_\_ c. Save your work (by pressing Ctrl+S).
- \_\_\_ d. In the **Deployment** section, click **Proceed to RuleApp deployment**.
- \_\_\_ e. Click **Next** until you reach the Version Summary page, and then click **Finish**.
- \_\_\_ 3. Look at the Deployment Report and notice the **XOM Deployment** section, which provides details on the deployment of the XOM.

#### XOM Deployment

- i** The decision operation 'loan-rulesOperation' references the following target Rule project: [loan-rules].
- i** The following XOM resources were found for Rule project 'loan-rules': [loan-xom].
- i** No previous library matching this deployment was found on Sample (<http://localhost:9080/res>): a new library will be created.
- i** The resource 'loan-xom.zip' was successfully deployed with version '1.0' on Sample (<http://localhost:9080/res>).
- i** The library 'loan\_deploy\_1.0' was successfully deployed with version '1.0' on Sample (<http://localhost:9080/res>).



#### Note

Depending on the order in which you completed the exercises, the version number for this deployment might be different.

- \_\_\_ 4. In Rule Explorer, expand **loan-rules > resources > xom-libraries** and double-click the `loan-xom.zip` file to open it.

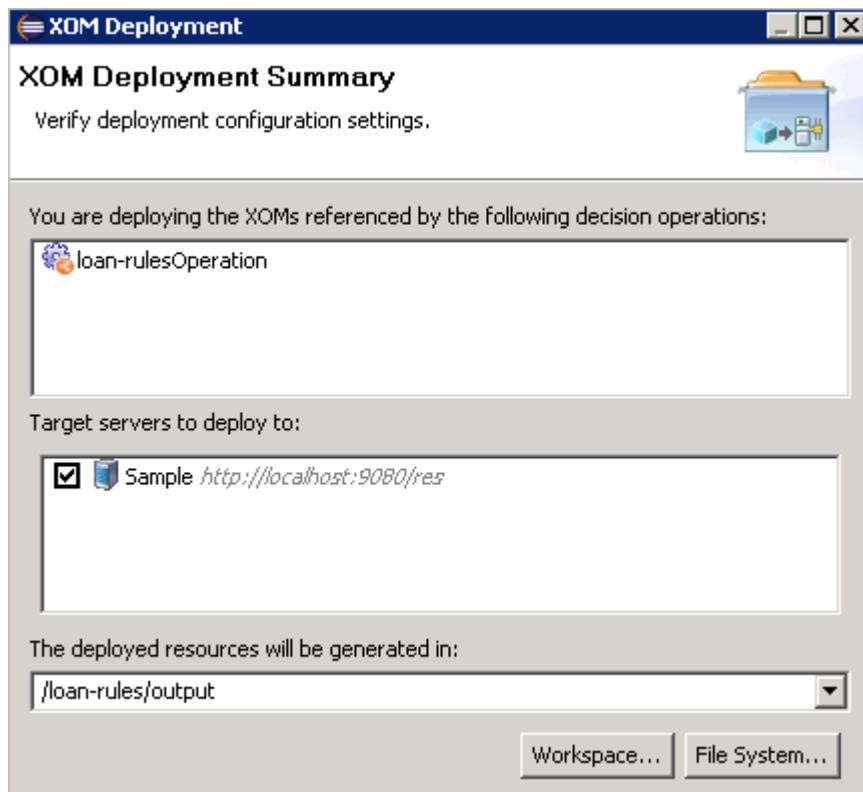
The `loan-xom.zip` file opens in Windows Explorer to the **training\_loan** folder, which contains the classes.

**Note**

Depending on the number of times you try deploying the XOM, you might see a different version number in your results. The main point of this exercise is to recognize how you can use version numbers to manage deployments.

- \_\_\_ 5. In the next steps, you change the XOM, redeploy it, and verify that the version of the deployed XOM is incremented.
  - \_\_\_ a. In the Rule Explorer, expand **loan-xom > src > training\_loan** and double-click the **Test.java** class.
  - \_\_\_ b. Edit the **Test.java** class by commenting the final line of the **main** method:  
`// System.out.println(r1);`
  - \_\_\_ c. Save the **Test** class and wait for the workspace to rebuild.

If **Build Automatically** is selected on the **Project** menu, your workspace should build automatically after saving.
- \_\_\_ 6. Redeploy the XOM from the **loan-rules** project.
  - \_\_\_ a. Right-click the **loan-rules** project and click **Rule Execution Server > Deploy XOM**.  
The XOM Deployment wizard opens.



- \_\_\_ b. Click **Next** and click **Finish**.

The **XOM Deployment** section of the Deployment Report shows the `loan-xom` version is now incremented.

### XOM Deployment

- i** The decision operation 'loan-rulesOperation' references the following target Rule project: [loan-rules].
- i** The following XOM resources were found for Rule project 'loan-rules': [loan-xom].
- i** The resource 'loan-xom.zip' was successfully deployed with version '2.0' on Sample (<http://localhost:9080/res>).
- i** A new resource has been deployed to Sample (<http://localhost:9080/res>): a new version of library 'loan\_deploy\_1.0' is thus required.
- i** The library 'loan\_deploy\_1.0' was successfully deployed with version '1.1' on Sample (<http://localhost:9080/res>).

### Deployment successful

Because the XOM changed, its checksum also changed, as did the version of the managed XOM in Rule Execution Server.



### Note

Depending on whether the XOM was deployed earlier during exploration of the Rule Execution Server, you might have a different version number. The main point of this exercise is to recognize how you can use version numbers to manage deployments.

## 4.2. Deploying a RuleApp associated with a managed XOM

In this section, you deploy the RuleApp again. But this time, you also indicate the managed XOM that is associated with this RuleApp.

- 1. Undo the changes in the XOM that you made in Step 5 on page 15-12.
  - a. In the `loan-xom` project, open the `Test` class and uncomment the final line of the `main` method:

```
System.out.println(r1);
```

  - b. Save the `Test` class and close the file and wait for your workspace to rebuild.
- 2. Redeploy the XOM.
  - a. In Rule Explorer, right-click the `loan-rules` project and click **Rule Execution Server > Deploy XOM**.
  - b. In the XOM Deployment window, click **Finish**.
  - c. Look at the Deployment Report and look for this line:

The resource 'loan-xom-zip' is already deployed with version '1.0' on Sample (<http://localhost:9080/res>). Skipping it.

The XOM is not redeployed because the XOM now matches the originally deployed XOM.



### Information

The version in the URI of the managed XOM depends on the checksum that Rule Execution Server computes for this XOM. Because you restored the XOM to its initial state, the deployed XOM has the same checksum as the earlier version of the XOM.

As a consequence, Rule Execution Server considers that you deployed that XOM version again; so that URI is used.



### Note

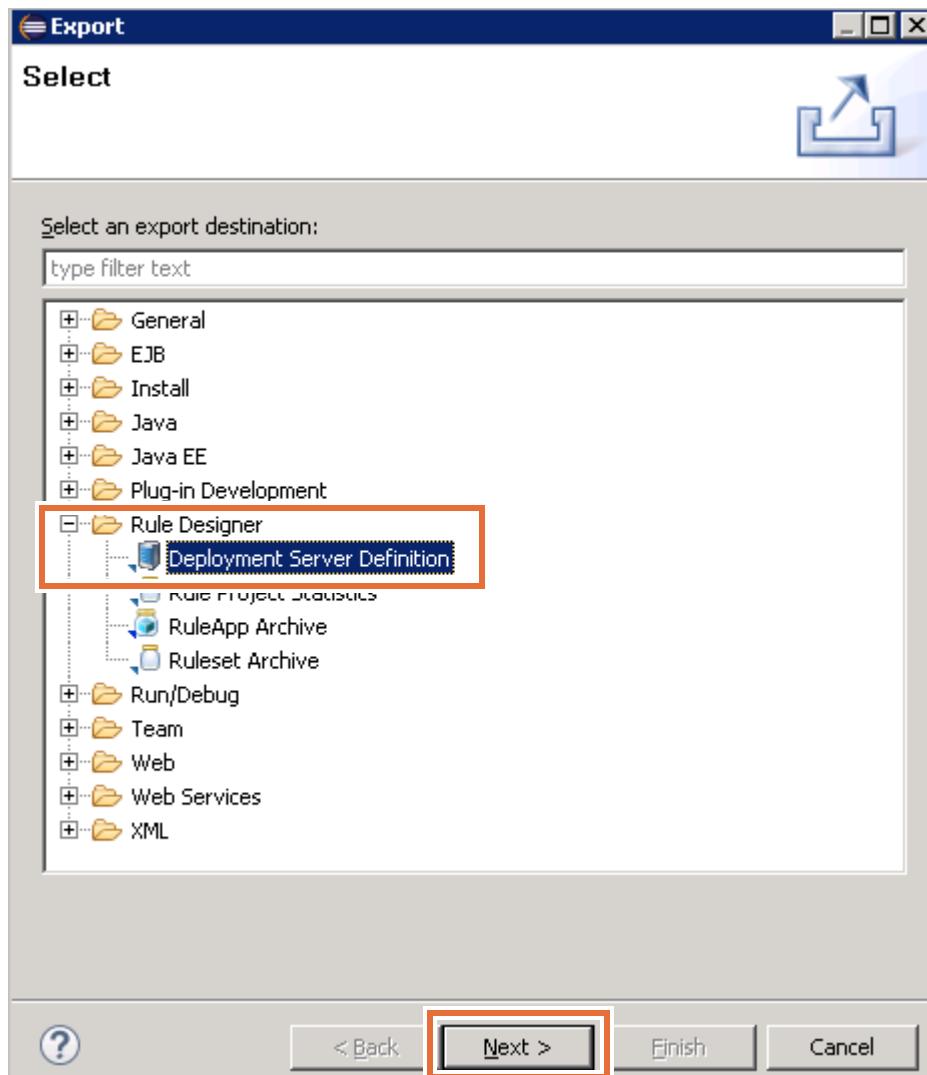
Depending on the number of times you tried deploying the RuleApp during this exercise, you might see a different version number in your results. The main point of this exercise is to recognize how you can use version numbers to manage deployments.

## Section 5. Exporting and importing deployment server definitions

When you define a target server in a deployment configuration, that definition is stored at the workspace level. If you want to reuse the target server definition in another workspace, you can export the definition and save it. You then import it to the next workspace.

### 5.1. Exporting deployment configurations

- \_\_\_ 1. From the **File** menu, click **Export**.
- \_\_\_ 2. In the Export wizard, expand **Rule Designer**, select **Deployment Server Definition**, and click **Next**.



- \_\_\_ 3. Save the `server-list.xml` file to the `<LabfilesDir>/code` folder.

The file is now available for import to any workspace where you want to define a deployment configuration that uses the sample server.

## 5.2. Importing deployment configurations

To see how to import a deployment server definition, you can switch to a new (empty) workspace.

- \_\_\_ 1. In Rule Designer, switch to a new workspace with the path:

`<LabfilesDir>\workspaces\deployment-answer`

- \_\_\_ 2. Import the solution to this exercise.

- \_\_\_ a. From the **File** menu, click **Import**.
- \_\_\_ b. On the Select page, select **Operational Decision Manager > Samples and Tutorials**, and click **Next**.
- \_\_\_ c. In the **Rule Designer > Training** node, select **Ex 15: Managing deployment > 02-answer** and click **Finish**.

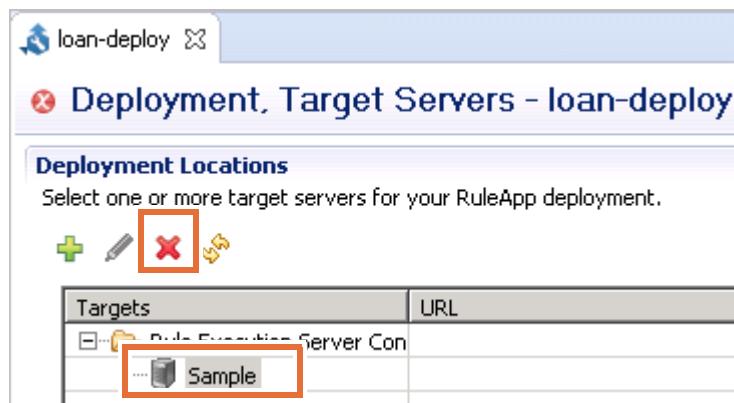


### Note

The project imports with an error on the `loan-deploy` deployment configuration file.

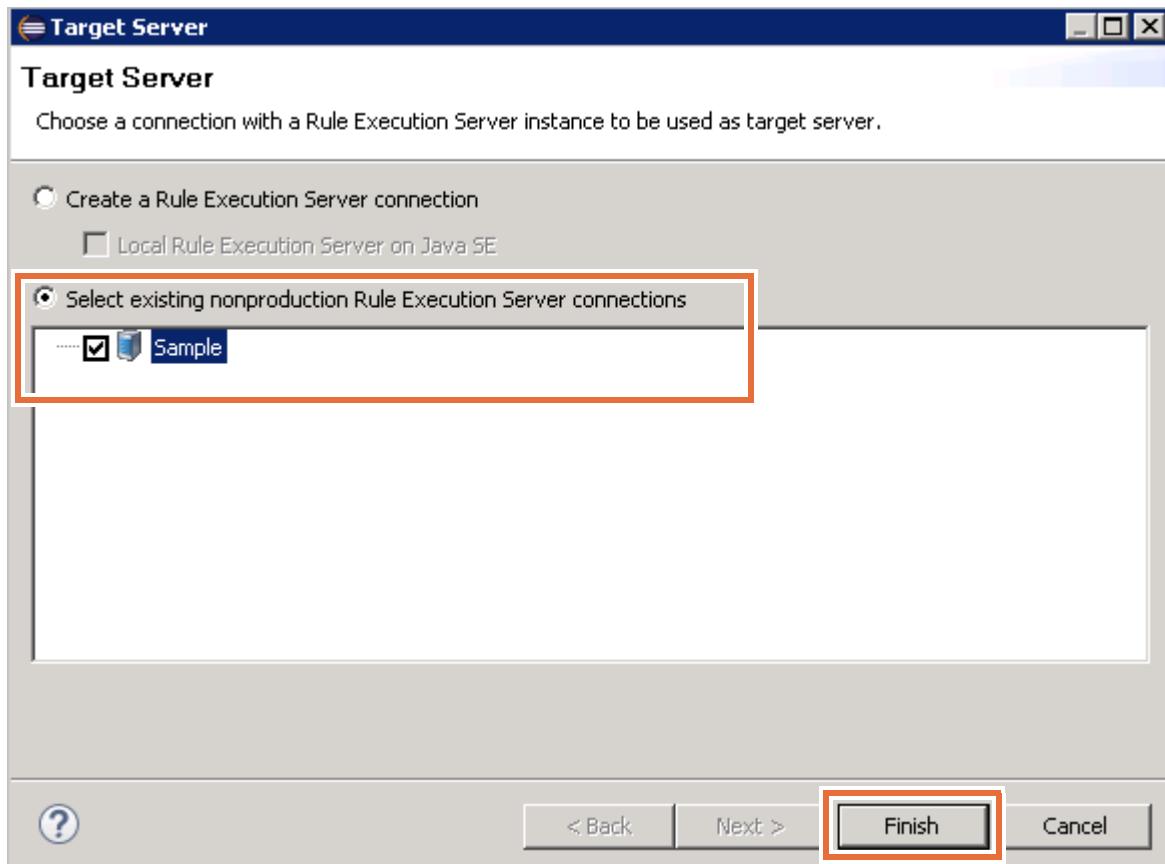
- \_\_\_ 3. Delete the target server definition.

- \_\_\_ a. In Rule Explorer, expand **loan-rules > deployment** and double-click **loan-deploy**.
- \_\_\_ b. Open the **Target Servers** tab, select **Sample**, and delete it by clicking the X.



- \_\_\_ c. Click **OK** when prompted to confirm deletion.
- \_\_\_ 4. Import the previously exported deployment server definition.
  - \_\_\_ a. From the **File** menu, click **Import**.
  - \_\_\_ b. Expand **Rule Designer**, select **Deployment Server Definition**, and click **Next**.
- \_\_\_ 5. In the **File** field, click **Browse** to go to the `<LabfilesDir>/code` folder and select the `server-list.xml` file, and click **Finish**.
- \_\_\_ 6. Redefine the target server.
  - \_\_\_ a. On the **Target Server** tab of the deployment configuration, click the plus sign (+).

- \_\_\_ b. In the Target Server wizard, choose **Select existing nonproduction Rule Execution Server connections**, select **Sample**, and click **Finish**.



- \_\_\_ 7. Save your work.

**End of exercise**

## Exercise review and wrap-up

The first part of the exercise looked at how to create a deployment configuration. You also learned how to add and remove a ruleset property. The next parts of the exercise showed you how to deploy a RuleApp and how to deploy a managed XOM.

# Exercise 16.Exploring the Rule Execution Server console

## What this exercise is about

This exercise teaches you how to work with the Rule Execution Server console.

## What you should be able to do

After completing this exercise, you should be able to:

- Work with Rule Execution Server console tools
- Manage RuleApps and rulesets through the Rule Execution Server console

## Introduction

After you deploy RuleApps and XOMs to Rule Execution Server, you can manage them through the Rule Execution Server console.

In this exercise, you explore the Rule Execution Server console environment and features.

The exercise is divided into these sections:

- Section 1, "Exploring the Rule Execution Server console"
- Section 2, "Exploring the deployed RuleApps"
- Section 3, "Working with deployed resources"
- Section 4, "Exploring the Diagnostics and Server Info tabs"
- Section 5, "Exploring the REST API tab"
- Section 6, "Managing RuleApps and rulesets"
- Section 7, "Testing a deployed ruleset"

## Requirements

You should complete Exercise 15, "Managing deployment" before starting this exercise.

This exercise uses the following file, which is installed in the  
`<InstallDir>\studio\training` directory: Dev 16 –  
Deployment\01-start

## Section 1. Exploring the Rule Execution Server console

### 1.1. Setting up your environment for this exercise

- 1. If the sample server is not started, start it now by double-clicking the **Start server** shortcut on the desktop. You can also start the server by clicking **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Start server**.

Starting the sample server might take several minutes.

#### For IBM ODM on Cloud users

You do not need to do this step. Your Rule Execution Server instance is already running in your IBM ODM on Cloud environment.

- 2. In Rule Designer, switch to a new workspace:
  - a. From the **File** menu, click **Switch Workspace > Other**.
  - b. In the Workspace Launcher window, enter the path:  
`<LabfilesDir>\workspaces\explore_RES`
- 3. Close the Welcome view.
- 4. Import the lab projects.
  - a. From the **File** menu, click **Import**.
  - b. On the Select page, select **Operational Decision Manager > Samples and Tutorials**, and click **Next**.
  - c. In the **Rule Designer > Training** node, select **Ex 16: Exploring Rule Execution Server console > 01-start** and click **Finish**.
- 5. After the workspace builds, close the Help view.



#### Troubleshooting

The project imports with an error on the `loan-deploy` deployment configuration file. If the RuleApp was already deployed in Exercise 15, "Managing deployment", you can ignore the error.

If you need to redeploy, you can fix the error by importing the deployment server definition, as described in Section 5, "Exporting and importing deployment server definitions" of Exercise 15, "Managing deployment".

## 1.2. Exploring the Rule Execution Server console pages

- 1. Open the Rule Execution Server console.
  - a. Double-click the **Rule Execution Server console** shortcut on the desktop or click **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Rule Execution Server console**.
  - b. In both the **User Name** and **Password** fields, enter: `resAdmin`

### For IBM ODM on Cloud users

Instead of going through your operating system, you start the Rule Execution Server console by logging in to the User Portal, then clicking **Launch** under **Rule Execution Server console**.

The Rule Execution Server console is configured to open to the URL of your ODM on Cloud instance, so you do not need to enter a URL or port number.

- 2. Note the tabs: **Home**, **Explorer**, **Decision Warehouse**, **Diagnostics**, **Server Info**, and **REST API**.



#### Note

You work with the **Decision Warehouse** tab in Exercise 19, "Auditing ruleset execution through Decision Warehouse".

- 3. Click the **Explorer** tab.

The Navigator pane gives access to the elements that you can manage in the Rule Execution Server console.



- The **RuleApps** link gives access to the RuleApps View pane, where you can manage the deployed RuleApps.

- The **Resources** link gives access to the Resources View pane, where you can manage resources.
- The **Libraries** link gives access to the Libraries View pane, where you can manage libraries.
- The **Service Information** link gives access to the Transparent Decision Services View pane, where you can manage the created decision services.

By default, the Explorer page opens with the RuleApps View pane visible, and you can see the RuleApps that you deployed in Exercise 15, "Managing deployment".

## Section 2. Exploring the deployed RuleApps

In this section, you explore the RuleApps deployed in Rule Execution Server, and relate them to what you did in Exercise 15, "Managing deployment".



### Note

Depending on the number of deployments that you tried before doing this exercise, the version numbers that you see in the Rule Execution Server console might differ from the versions in these instructions.

#### To explore the RuleApp deployed from Rule Designer:

- 1. In the Navigator pane, expand **RuleApps > /loan\_deploy/1.0**.  
These artifacts are the ones that you deployed in Exercise 15, "Managing deployment".
- 2. In the Navigator pane, click **/loan\_deploy/1.0** to open the deployed RuleApp on the RuleApp View page.

The screenshot shows the 'RuleApp View' interface. At the top, there is a toolbar with buttons for 'Add Ruleset', 'Add Property', 'Download Archive', and 'Edit'. Below the toolbar, the title 'RuleApp View' is displayed above a sub-section titled '/loan\_deploy/1.0'. This sub-section contains the following information:

Name	loan_deploy
Version	1.0
Creation Date	Jan 11, 2016 11:59:54 AM GMT-08:00
Display Name	
Description	

- \_\_\_ 3. In the Navigator pane, click **loan\_rulesRuleset/1.0** to open the ruleset on the Ruleset View page.

The screenshot shows the 'Ruleset View' page for the ruleset **/loan\_deploy/1.0/loan\_rulesRuleset/1.0**. The page header includes links for 'Test Ruleset', 'View Statistics', 'View Execution Units', and 'Upload Ruleset Archive'. Below the header, the ruleset details are listed:

<b>Name</b>	loan_rulesRuleset
<b>Version</b>	1.0
<b>Creation Date</b>	Jan 11, 2016 11:59:55 AM GMT-08:00
<b>Display Name</b>	loan-rulesOperation
<b>Description</b>	
<b>Rule engine</b>	Classic Rule Engine
<b>Status</b>	enabled
<b>Debug</b>	disabled

A 'Permanent link' button is located at the bottom left of the details section.

The Ruleset View pane opens, where you can see the `borrower`, `loan`, and `report` ruleset parameters that are defined for this ruleset.

- \_\_\_ 4. Scroll down the Ruleset View page to see the **Ruleset Parameters** section and other links on that page.
- \_\_\_ 5. In the Navigator page, click **loan\_rulesRuleset/1.1** to open the other deployed ruleset, and in the Ruleset View page, click **Show Managed URIs**.

You can see that this ruleset is associated with a deployed XOM. The URI, `reslib://loan_deploy_1.0/1.0`, corresponds to the XOM that you deployed in Exercise 15, "Managing deployment".

## Section 3. Working with deployed resources

In this section, you explore the XOM resources that are deployed in Rule Execution Server, and relate them to what you did in Exercise 15, "Managing deployment".

- 1. In the Navigator pane, click **Resources**.
- 2. On the Resources View page, click the **loan-xom.zip** link for the first XOM that you deployed in Exercise 15, "Managing deployment".
- 3. Click **Show references from Rulesets** to see which rulesets reference this XOM.
- 4. Click **Show references from Libraries** to see which XOM libraries reference this XOM.  
You see which versions of the ruleset and XOM use this resource.

## Section 4. Exploring the Diagnostics and Server Info tabs

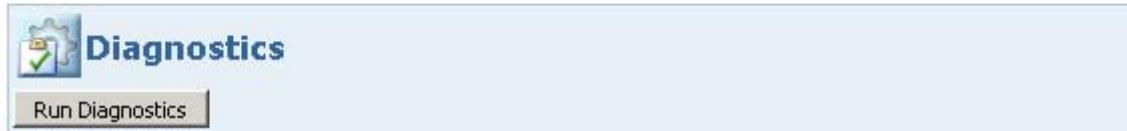
In this section, you explore the Diagnostics and Server Info pages.

\_\_ 1. Explore the Diagnostics page.

\_\_\_ a. Click the **Diagnostics** tab.



### Diagnostics View



\_\_\_ b. Click **Run Diagnostics**.

\_\_\_ c. Click **Expand All** to view the results.

\_\_ 2. Explore the Server Info page.

\_\_\_ a. Click the **Server Info** tab.



\_\_\_ b. Notice the location of the log file in the **Log File** field, the logging level, and the server that is being used for the execution unit. You can view errors that occur on that server by clicking the server name link.

## Section 5. Exploring the REST API tab

The REST API tool is a web application that you can use to test the management capabilities of REST resources and their associated parameters. REST resources include RuleApps, rulesets, XOM libraries, and XOM resources.

- 1. In the Rule Execution Server console, click the **REST API** tab to display the test tool.



- 2. Explore the **REST API** tab by clicking the tab for each resource and noting which methods are available:

- /ruleapps
- /rulesets
- /libraries
- /xoms

**REST API tool**

You can use this test tool to test the management REST API. To test the ruleset execution REST API, REST API WADL file: </res/apiauth/v1/DecisionServer.wadl>

**/ruleapps** **/rulesets** **/libraries** **/xoms**

**GET /ruleapps**  
getRuleApps Returns all the RuleApps contained in the repository.

**GET /ruleapps?count=true**  
getCountOfRuleApps Counts the number of elements in this list.

**POST /ruleapps**  
deployRuleAppArchive Deploys a RuleApp archive in the repository, based on the merge request body.

**POST /ruleapps**  
addRuleApp Adds a new RuleApp in the repository. The RuleApp representation is passed in the request body, the response body contains a specific error description and the HTTP status 202 is returned.



### Note

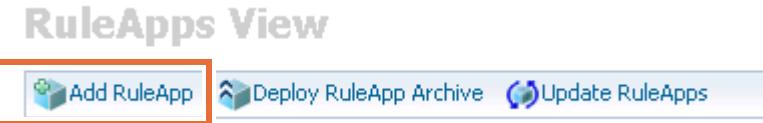
You work with the REST API in Exercise 20, "Working with the REST API".

## Section 6. Managing RuleApps and rulesets

In this section, you learn the basics of RuleApp and ruleset management with the Rule Execution Server console. You can use the Rule Execution Server console to create, edit, and delete RuleApps and rulesets.

### 6.1. Creating a RuleApp

- \_\_ 1. Click the **Explorer** tab.
- \_\_ 2. In the RuleApps View, click **Add RuleApp**.



- \_\_ 3. In New RuleApp pane, leave the **Name** field set to: NewRuleApp
- \_\_ 4. In the **Version** field, keep: 1.0
- \_\_ 5. In the **Display Name** field, enter: Test RuleApp
- \_\_ 6. Click **Add**.



The RuleApp View now shows the properties of the NewRuleApp/1.0 RuleApp.

### 6.2. Adding a ruleset to your RuleApp

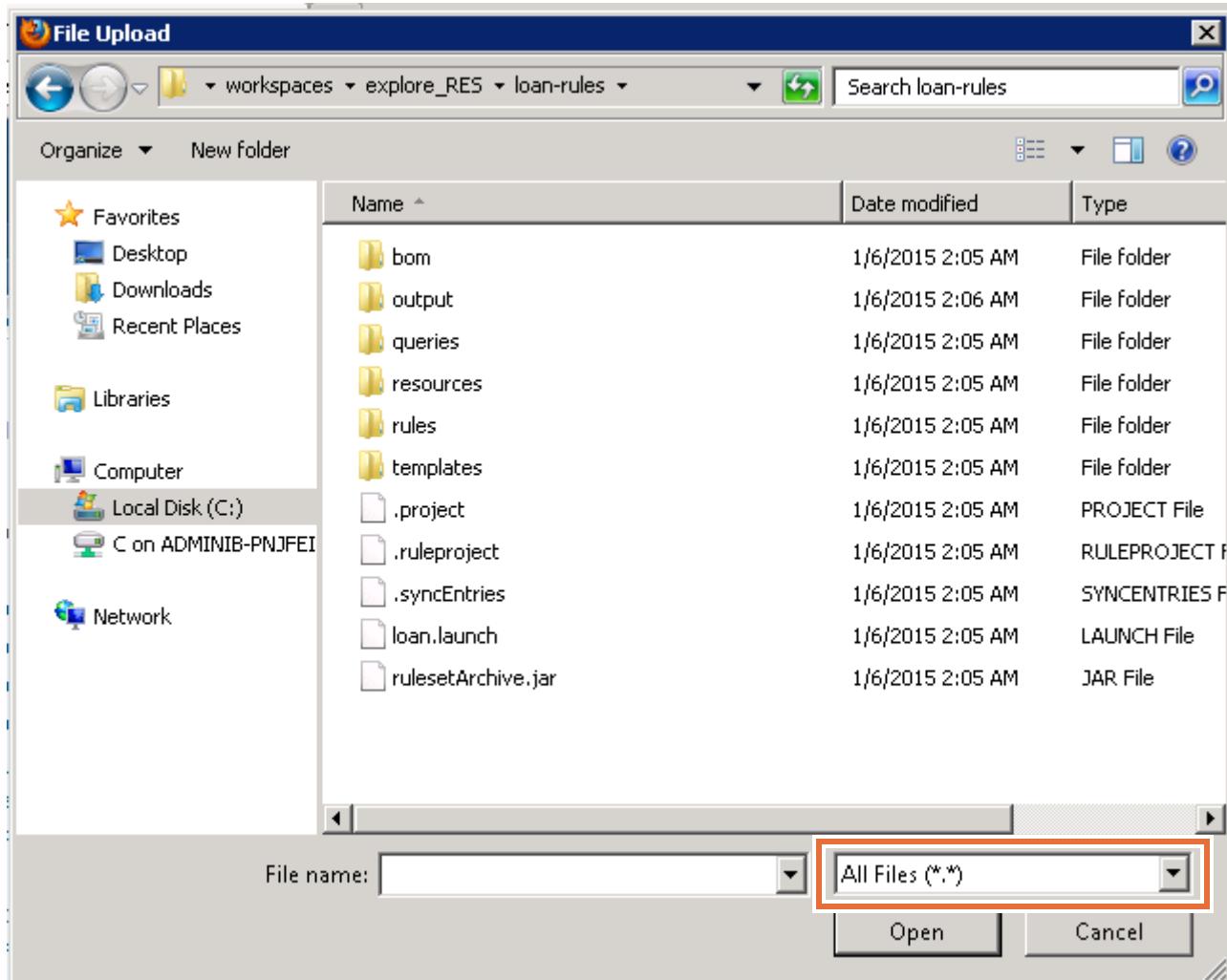
- \_\_ 1. Add a ruleset to the new NewRuleApp.
  - \_\_ a. In the RuleApp View pane, click **Add Ruleset**.



The New Ruleset pane opens.

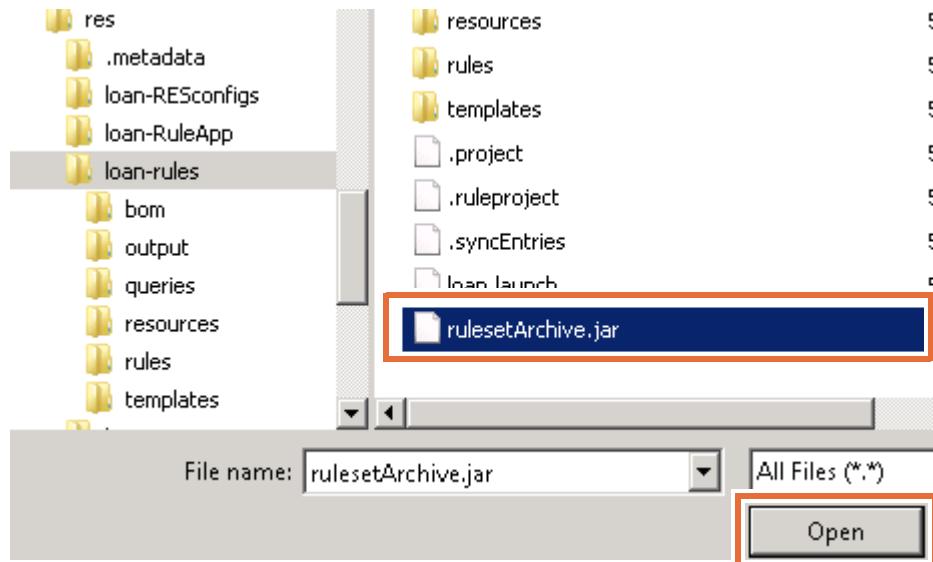
- \_\_ b. In the **Name** field, keep: NewRuleset

- \_\_\_ c. In the **Version** field, keep: 1.0
- \_\_\_ d. In the **Display Name** field, enter: Test Ruleset
- \_\_\_ e. Click **Browse** next to **Local path of ruleset archive** and go to the following folder in your current Rule Designer workspace directory:  
`<LabfilesDir>\workspaces\workspace_name\loan-rules\`
- \_\_\_ f. Make sure that the file type to search for is set to **All Files (\*.\*)**.



The Ruleset Archive is a .jar file, and is not visible with the default file type setting.

- \_\_ g. Select the `rulesetArchive.jar` file, and click **Open**.



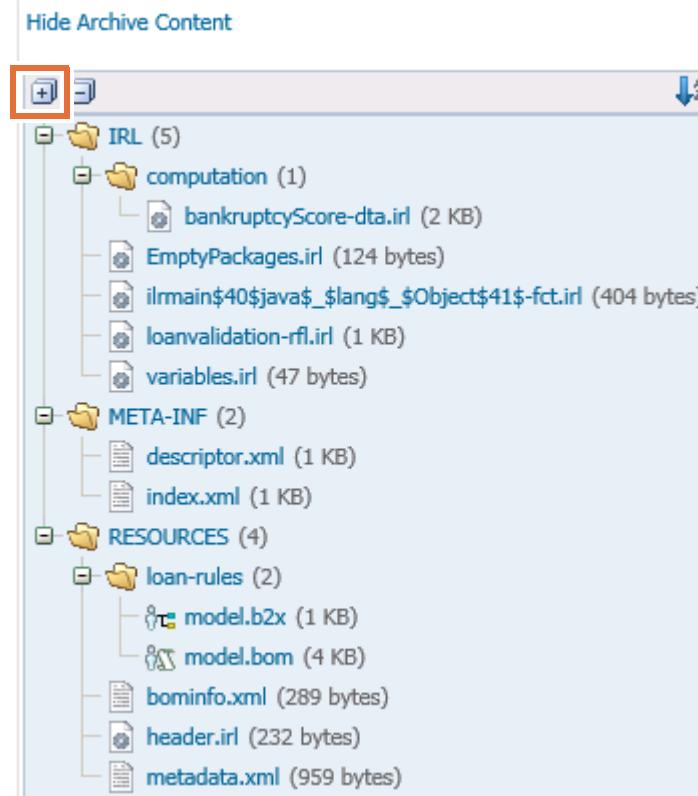
- \_\_ h. On the New Ruleset page, leave the other fields unchanged, and click **Add**.

The `NewRuleApp/1.0/NewRuleset/1.0` ruleset is now created.

- \_\_ 2. On the Ruleset View page for the `NewRuleApp/1.0/NewRuleset/1.0` ruleset, scroll down the page and click **Show Archive Content**.



- \_\_ 3. Expand the folders to explore the content of the ruleset archive.



- The **IRL** folder contains the IRL version of all the rule artifacts in the ruleset.
- The **META-INF** folder contains the `descriptor.xml` and the `index.xml` files.
  - The `descriptor.xml` file contains the properties of the ruleset (name, ruleset signature, version).
  - The `index.xml` file contains the indexes of the ruleset, that is, the names of the different files that constitute this ruleset.
- The **RESOURCES** folder contains the definition of the business object model, and the definition of the rule artifact properties.

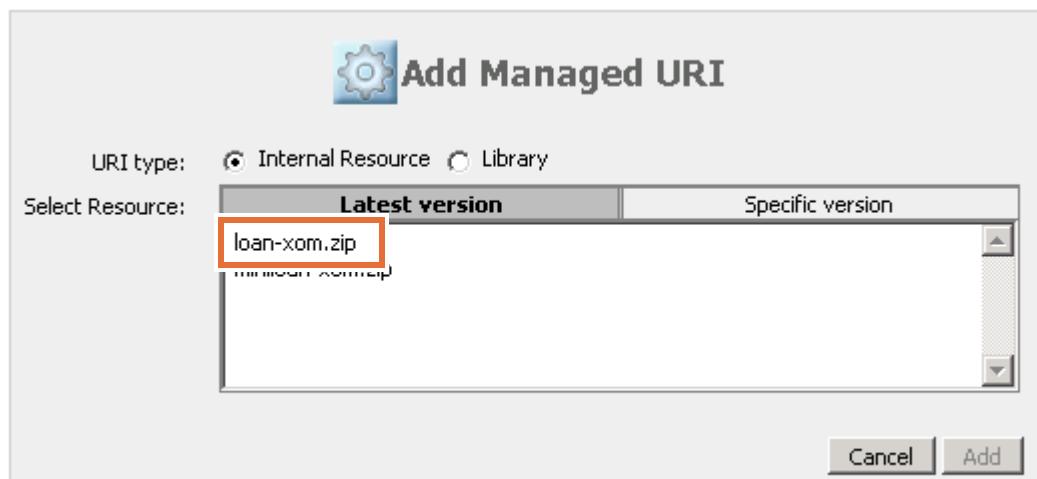
### 6.3. Adding a managed XOM to your ruleset

- 1. On the toolbar of the Ruleset View page, click **Add Managed URI**.



The Add Managed URI window opens with two tabs: **Latest version** and **Specific version**.

- 2. On the **Latest version** tab of the Add Managed URI window, select **loan-xom.zip** and click **Add**.



- 3. On the Ruleset View page, click **Show Managed URIs**.

The URI `resuri://loan-xom.zip` is now visible in the list of managed XOMs associated with the ruleset.

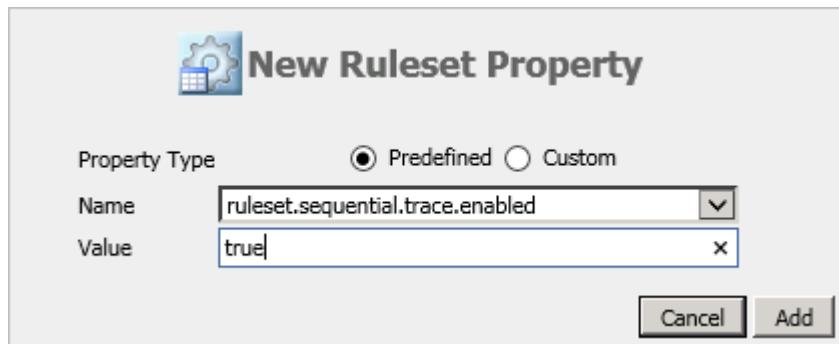
- 4. View the references, as you did in Section 3, "Working with deployed resources".

- a. On the Navigator page, click **Resources**.
- b. In the Resource View, click the most recent **loan-xom.zip**.
- c. Click **Show references from Rulesets** to see the rulesets that depend on this XOM.

The reference points to your new test ruleset.

## 6.4. Adding a ruleset property to the ruleset

- \_\_\_ 1. Return to Ruleset View page for your NewRuleset ruleset.
- \_\_\_ a. In the Navigator pane, expand **RuleApps > /NewRuleApp/1.0**.
- \_\_\_ b. Click **/NewRuleset/1.0**.
- \_\_\_ 2. On the toolbar, click **Add Property**.
- \_\_\_ 3. In the New Ruleset Property dialog box, in the **Name** field, select **rulesetSEQUENTIAL.trace.enabled** in the list.
- \_\_\_ 4. In the **Value** field, type: **true**
- \_\_\_ 5. Click **Add**.



- \_\_\_ 6. On the Ruleset View page, click **Show Properties** to view the ruleset properties.

Hide Properties

**1 properties**

<input type="checkbox"/> Select All	Name
<input type="checkbox"/>	rulesetSEQUENTIAL.trace.enabled

properties 1 - 1 of 1



You learn about the monitoring options in Exercise 19, "Auditing ruleset execution through Decision Warehouse".

## 6.5. Deleting the RuleApp

- \_\_\_ 1. In the Navigator pane, click **RuleApps** to open the list of RuleApps in the Ruleapps View.

2. In the list of RuleApps, select **NewRuleApp** and click **Remove**.

Total Number of RuleApps 2

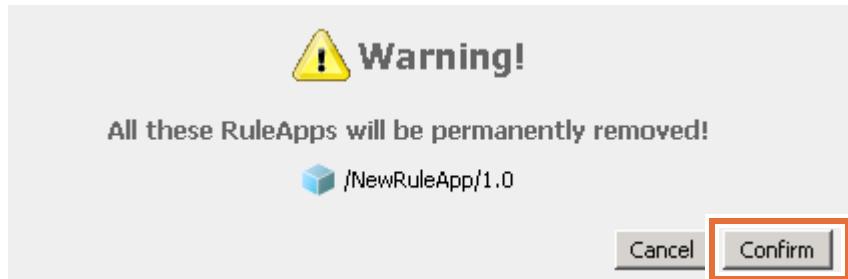
2 RuleApp(s)

<input type="checkbox"/> Select All	Name	Version	Creation Date	Number of rulesets	
<input type="checkbox"/>	loan_deploy	1.0	Jan 11, 2016 11:59:54 AM GMT-08:00	2	Download Archive
<input checked="" type="checkbox"/>	NewRuleApp	1.0	Jan 11, 2016 7:48:17 PM GMT-08:00	1	Download Archive

RuleApp 1 - 2 of 2      prev 10 next 10

**Remove**

3. When prompted to confirm the removal of the RuleApp, click **Confirm**.



This RuleApp is not required for the rest of the course.

## Section 7. Testing a deployed ruleset

In this section, you test the `loan_rulesRuleset` ruleset directly in the Rule Execution Server console.



### Hint

In this exercise, you must write some code. You can use the code snippets in the `<LabfilesDir>\code\test_in_res_console.txt` file, and copy and paste them in Rule Designer.

### 7.1. Testing the ruleset

- 1. In the Navigator pane, expand **RuleApps > /loan\_deploy/1.0**.
- 2. Click the most recent version of the `loan_rulesRuleset` ruleset.

The Ruleset View page opens for the selected `loan_rulesRuleset` ruleset.

- 3. On the toolbar, click **Test Ruleset**.

#### Ruleset View

Direction	Name	Kind	XOM Type	Value
<input type="checkbox"/>		<b>borrower</b>	native	<code>training_loan.Borrower</code>
<pre>import training_loan.*; Borrower borrower = null;</pre>				
<input type="checkbox"/>		<b>loan</b>	native	<code>training_loan.Loan</code>
<pre>import training_loan.*; Loan loan = null;</pre>				

**Execute Task**   
**Task Name**

To test your ruleset, you must provide values to these input ruleset parameters. You can either edit the associated Java code or provide a Java class that defines the ruleset parameter.

- 4. In the Test Ruleset View pane, define the values of the input ruleset parameters.
  - a. Click **Edit mode** next to the `borrower` ruleset parameter.

You can now write code in the **Value** section of the `borrower` ruleset parameter.

The  **Visualization mode** is also available for you to save your changes.

- b. In the **Value** field of the `borrower` ruleset parameter, enter the following code:

```
import training_loan.*;
import java.util.Calendar;
```

```
Borrower borrower = new Borrower("John", "Doe", DateUtil.makeDate(1968,
 Calendar.MAY, 12), "123456789");
borrower.zipCode = "91320";
borrower.creditScore = 600;
borrower.setYearlyIncome(100000);
borrower.setLatestBankruptcy(DateUtil.makeDate(1990, Calendar.JANUARY, 01),
 7, "Unemployment");
```



### Hint

You can find this code snippet in the `<LabfilesDir>\code\test_in_res_console.txt` file.



### Note

The code that you require to define an input ruleset parameter is like Java code. It is similar to the code that you can find in the `Test` class of the `loan-xom` project to define the `Borrower` `b1` object. However, in the Rule Execution Server console:

- The name of the instance must be the ruleset parameter name. In the present case, it must be: `Borrower borrower`
- You must write only the lines of code that create and initialize the instance of the appropriate class, and they must be preceded by the required import statements.

- c. Click  **Visualization mode** to save your change.
- d. Click  **Edit mode** next to the `loan` ruleset parameter.
- e. In the **Value** field of the `loan` input ruleset parameter, enter the following code:

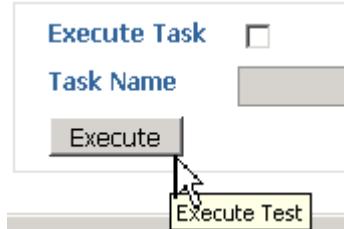
```
import training_loan.*;
import java.util.Calendar;
```

```
Loan loan = new Loan(DateUtil.makeDate(2016, Calendar.JUNE, 1), 60,
 100000, 0.70);
```

**Hint**

You can find this code snippet in the `<LabfilesDir>\code\test_in_res_console.txt` file.

- \_\_ f. Click **Visualization mode** to save your change.  
 \_\_ 5. Click **Execute**.



The Execution result pane opens and displays the following items:

- The executed canonical ruleset path
- The execution duration
- The number of rules (total, fired, not fired)
- The rules that were fired (for example, validation.borrower.checkZipcode)
- The number of tasks (total, executed, not executed)
- The tasks that were executed, for example:

```
loanvalidation
loanvalidation>initResult
loanvalidation>validation
```

- The output traces, for example:

```
Borrower(firstName: John lastName: Doe born: May 12, 1968 SSN:
123-45-6789 zip code: 91320 income: 100000 credit score: 600 (bankruptcy
on: Jan 1, 1990 for: Unemployment filed under chapter: 7))
Loan(amount: 100000 starting: Jun 1, 2016 duration: 60 month LTV: 70%
rate: 5.7% monthly repayment: 1,919.36)
Report(validData: true approved: true score: 790 grade: B
insuranceRequired: true insuranceRate: 2% message: Low risk loan
Congratulations! Your loan has been approved
)
```

The Output Parameters pane is also open and shows the values of the output ruleset parameters.

- \_\_ 6. Sign out of the Rule Execution Server console and close the web browser window.

## End of exercise

## Exercise review and wrap-up

This exercise looked at how you can work with the Rule Execution Server console. You explored the different pages of the Rule Execution Server console. You also learned how to manage the RuleApps, the rulesets, and the managed XOMs.



# Exercise 17. Executing rules with Rule Execution Server in Java EE

## What this exercise is about

This exercise teaches you how to execute rules with Rule Execution Server in Java EE.

## What you should be able to do

After completing this exercise, you should be able to:

- Use the basic Rule Execution Server API to request ruleset execution with Rule Execution Server in Java EE

## Introduction

You deployed the rule project as a RuleApp for managed execution with Rule Execution Server deployed in WebSphere Application Server. Next, you create a web client application that uses your ruleset.

In this exercise, you create the web client application that is based on the API for Rule Execution Server in Java EE.

This exercise includes these sections:

- Section 1, "Building the web application"
- Section 2, "Deploying the web application to WebSphere Application Server"
- Section 3, "Executing the web application with the RuleApp deployed"

## Requirements

This exercise uses the following files, which are installed in the `<InstallDir>\studio\training` directory.

- Start project: Dev 17 - Java EE\01-start
- Solution project: Dev 17 - Java EE\02-answer
  - You can use the solution project in "Exercise review and wrap-up" on page 17-10.

This exercise uses the RuleApp that you deployed in the earlier exercise, Exercise 15, "Managing deployment".



**Stop**

If you deleted the RuleApp from Rule Execution Server for any reason, you must redeploy it to Rule Execution Server. To redeploy, follow the instructions in "Deploying a RuleApp associated with a managed XOM" on page 15-13.

## Section 1. Building the web application



### Requirements

Business analysts ask you to create a web application that requests the execution of the `loanrules` ruleset that is packaged in the `loanRuleApp` RuleApp. They give you the `loan-webapp` project, which is a skeleton of a client web application that requests the execution of business rules execution from a JSP page. You must complete, deploy, and execute this client application.

In this section, you examine and finalize the `loan-webapp` project.

### 1.1. Setting up your environment for this exercise

- 1. If the sample server is not started, start it now by clicking **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Start server**.  
Starting the server might take several minutes.
- 2. In Rule Designer, switch to a new workspace:
  - a. From the **File** menu, click **Switch Workspace > Other**.
  - b. In the Workspace Launcher window, enter the path:  
`<LabfilesDir>\workspaces\java_EE`
- 3. Close the Welcome view.
- 4. Import the lab projects.
  - a. From the **File** menu, click **Import**.
  - b. On the Select page, select **Operational Decision Manager > Samples and Tutorials**, and click **Next**.
  - c. In the **Rule Designer > Training** node, select **Ex 17: Executing rules in Java EE > 01-start** and click **Finish**.
- 5. After the workspace builds, close the Help view.



### Troubleshooting

You might see errors on the `loan-rules` project after the workspace finishes building. You do not deploy during this exercise, so to resolve the error, you can delete the Sample target server from the deployment configuration and save your work.

If you need to redeploy, you can fix the error by importing the deployment server definition, as described in Section 5, "Exporting and importing deployment server definitions" of Exercise 15, "Managing deployment".

You have these projects available in the Rule Explorer:

- loan-webapp
- This project is a client web application with a minimal graphical user interface, from which you can request business rule execution.
- loan-xom
  - loan-rules

**Hint**

In this exercise, you must write some code. You can use the code snippets in the `<LabfilesDir>\code\execute_with_res.txt` file, and copy and paste them in Rule Designer.

## 1.2. Finalizing the client application code to call your rules for execution

As you learned during the lectures, the Rule Execution Server API follows a uniform pattern. In a Java EE-based client application, the Java EE rule session factory uses the `IlrPOJOSessionFactory`.

- 1. In the Rule Explorer, expand **loan-webapp > src > WebContent** and open the `executerules.jsp` file with the JSP editor in Rule Designer.

**Hint**

To edit the JSP page, either double-click it, or right-click it and click **Open With > JSP editor**.

- 2. In the `executerules.jsp` file, search for the lines with dots ... and replace these dots with the appropriate calls.

- a. Create an `IlrPOJOSessionFactory` factory by replacing dots ... after the following line:

```
// Create session factory
```

With:

```
factory = new IlrPOJOSessionFactory();
```

- b. Create a stateless rule session from the factory by replacing dots ... after the following line:

```
// Create rule session
```

With:

```
ruleSession = factory.createStatelessSession();
```



## Questions

Find the following line:

```
IlrPath path = new IlrPath("...", "...");
```

How should you replace the dots to set the `IlrPath` correctly?

## Answer

The `IlrPath` class defines the ruleset path that is used to execute the ruleset. You might use a canonical ruleset path, that is, use the RuleApp version and the ruleset version. However, in most cases and in this exercise, you use the non-canonical ruleset path to make sure that Rule Execution Server uses the most recently deployed versions of the RuleApp and ruleset at run time.

- \_\_\_ c. Define the non-canonical ruleset path by replacing the following line:

```
IlrPath path = new IlrPath("...", "...");
```

With:

```
IlrPath path = new IlrPath("loan_deploy", "loan_rulesRuleset");
```



## Important

The rule project is called `loan-rules`, and its ruleset is deployed by using a RuleApp project that is called `loan-deploy`. However, the deployed RuleApp is called `loan_deploy` (with an underscore instead of a hyphen), and its ruleset is called `loan_rulesRuleset` (with an underscore). To define the ruleset path, you must use these later names, which are known in Rule Execution Server.

- \_\_\_ 3. Save your work (Ctrl+Shift+S) and close the JSP editor.

### 1.3. Building the application for deployment

- \_\_\_ 1. Create the web application in the `loan-webapp` project by using the `build.xml` Ant file.
  - \_\_\_ a. In the `loan-webapp` project, double-click **common.xml** to open the file.
  - \_\_\_ b. Find the `wodm.home` property and make sure that it is equal to `<InstallDir>` (for example, "C:/Program Files/IBM/ODM88") and if required, save the `common.xml` file.
  - \_\_\_ c. Close the `common.xml` file.

- \_\_ d. If you see warnings on the project, click **Build All** on the **Project** menu to rebuild the workspace.
- \_\_ e. Right-click **build.xml** and click **Run As > Ant Build...**



**Warning**

Make sure that you choose the **Ant Build...** option with the ellipsis (...).

The Edit Configuration window opens.

- \_\_ f. In the Edit Configuration window, make sure that **loanvalidation.ear [default]** is selected and click **Run**.

The Edit Configuration window closes.

The `loanvalidation.ear` target of the `build.xml` file runs and creates the `loanvalidation.ear` application file in the `loan-webapp\build` folder.

- \_\_ 2. Verify that the web application is created.

- \_\_ a. In the Rule Explorer, select the `loan-webapp` project and press F5 to refresh the content.
- \_\_ b. Expand the `loan-webapp` project and verify that the `build` folder is now visible.
- \_\_ c. Verify that the `build` folder contains the `loan-xom.jar` file and the `loanvalidation.ear` file.
  - The `loan-xom.jar` file is the Java archive file for the XOM. This file is an intermediate artifact that is required to build the `loanvalidation.ear` file.
  - The `loanvalidation.ear` file is the file that you must deploy to WebSphere Application Server for its execution.

## Section 2. Deploying the web application to WebSphere Application Server

In this section, you deploy the `loanvalidation.ear` file to WebSphere Application Server for its execution. Then, you verify the deployment of the `LoanValidation` application in the WebSphere Application Server administrative console.

### 2.1. Deploying the web application

- 1. Right-click the `build.xml` file and click **Run As > Ant Build...**

Make sure to choose the Ant Build option with the ellipsis (...). The Edit Configuration window opens.

- 2. In the Edit Configuration window:

- Clear **loanvalidation.ear [default]**.
- Select **deploy**.
- Click **Apply** and click **Run**.

The Edit Configuration window closes.

The `deploy` target of the `build.xml` file runs and deploys the `loanvalidation.ear` application file into WebSphere Application Server.

Deploying the application might take some time, and successfully completes when you can see the following information in the Console view of Rule Designer:

```
LoanValidation was successfully installed
BUILD SUCCESSFUL
```

### 2.2. Verifying application deployment to the application server

- 1. Open and sign in to the WebSphere Application Server administrative console.
  - a. Click **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Administrative console**.



#### Warning

When you start the WebSphere Application Server administrative console, you might have a message that the browser detects a problem with the security certificate of this website. Ignore this message, and click **Continue to this website** (Internet Explorer) or **I Understand the Risks > Add Exception > Confirm Security Exception** (Firefox) to proceed.

You might also have a JSP error. You can ignore the error for this exercise.

- b. In the **User ID** and **Password** fields, enter: `odm`
- c. Click **Log in**.



## Troubleshooting

If you start the WebSphere Application Server administrative console and are logged in as another user (such as `rtsAdmin` or `resAdmin`), log out and log back in with `odm` in the **User ID** and **Password** fields.

- 2. In the left pane, expand **Applications > Application Types**, and click **WebSphere enterprise applications**.

- 3. In the “Enterprise application” pane, verify that the `LoanValidation` application is present, and that its application status is started, which is marked with a green arrow.

- 4. Click **Logout** to exit the WebSphere Application Server console.

## Section 3. Executing the web application with the RuleApp deployed

In this section, you execute the web application.

- 1. Open a web browser at the following web address:

`http://localhost:9080/LoanValidation`

- 2. Look at three different scenarios on the first page:

- The first scenario uses a loan amount of 100,000. The loan is approved.
- The second scenario uses an amount of 200,000. The loan is rejected because the debt-to-income ratio is too high.
- The third scenario has an SSN with letters instead of digits, and a birth year of 1768. The loan is rejected because of the invalid input data.

- 3. Click **Start loan validation** at the top of the page.

The form to enter the borrower information opens.

- 4. Enter the borrower information in the form, according to the scenario of your choice, or leave the default values; and click **Next Step**.

The form to enter the loan information opens.

- 5. Enter the loan information in the form, according to the scenario of your choice, or leave the default values; and click **Calculate loan**.

The Loan Report page opens with the result of the execution of your ruleset.

### End of exercise

## Exercise review and wrap-up

This exercise looked at how to create and deploy a web client application to execute business rules with Rule Execution Server in a Java EE environment. You also explored the required Rule Execution Server API.

To see the solution to this exercise, switch to a new workspace and import the project **Ex 17: Java EE > 02-answer**.



### Troubleshooting

You might see errors on the `loan-rules` project after the workspace finishes building. You can ignore the error.

### For IBM ODM on Cloud users

Use the name and URL of the Rule Execution Server for your IBM ODM on Cloud instance in the **URL** field instead of `Sample` and `localhost`.

# Exercise 18. Executing rules as a hosted transparent decision service (HTDS)

## What this exercise is about

This exercise teaches you how to execute rules as a hosted transparent decision service (HTDS).

## What you should be able to do

After completing this exercise, you should be able to:

- Deploy an HTDS that can be used in a service-oriented architecture (SOA)
- Create a client application that uses an HTDS
- Monitor HTDS execution statistics in the Rule Execution Server console

## Introduction

In this exercise, you review all the steps that are required to use hosted transparent decision services in an SOA environment.

After you deploy the ruleset as a RuleApp to Rule Execution Server, you create an HTDS from this ruleset. You then create a web service client application that uses the HTDS. After you run the application, you monitor the execution results in Rule Execution Server console by using the statistics that are associated with the HTDS.

This exercise is divided into these sections:

- Section 1, "Deploying the decision service to Rule Execution Server"
- Section 2, "Getting the HTDS WSDL file from the deployed ruleset"

## Requirements

This exercise uses the files stored in the following directory:

`<InstallDir>\studio\training\Dev 18 - HTDS\01-start`

## Section 1. Deploying the decision service to Rule Execution Server

In this section, you deploy the RuleApp to Rule Execution Server.

### 1.1. Setting up your environment for this exercise

- \_\_ 1. If the sample server is not running, start it now by clicking **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Start server**.
- \_\_ 2. In Rule Designer, switch to a new workspace:
  - \_\_ a. From the **File** menu, click **Switch Workspace > Other**.
  - \_\_ b. In the Workspace Launcher window, enter the path:  
*<LabfilesDir>\workspaces\htds*
- \_\_ 3. Close the Welcome view.
- \_\_ 4. Import the lab projects.
  - \_\_ a. From the **File** menu, click **Import**.
  - \_\_ b. On the Select page, select **Operational Decision Manager > Samples and Tutorials**, and click **Next**.
  - \_\_ c. In the **Rule Designer > Training** node, select **Ex 18: Executing rules as a hosted transparent decision service > 01-start** and click **Finish**.
- \_\_ 5. After the workspace builds, close the Help view.

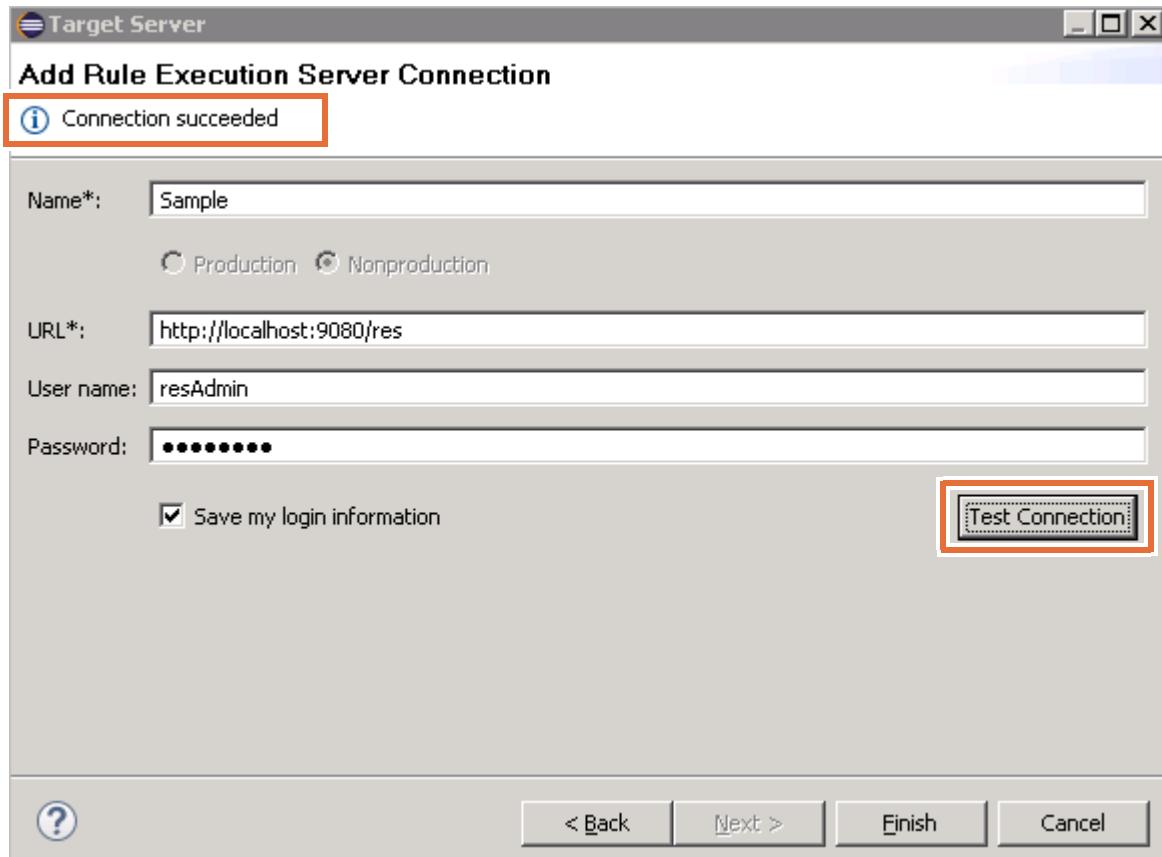
### 1.2. Deploying the RuleApp

- \_\_ 1. In Rule Designer, expand **Miniloan Rules > deployment** and double-click **Miniloan Deployment**.
- \_\_ 2. Define the sample server as the target server.
  - \_\_ a. Click the **Target Servers** tab.
  - \_\_ b. Click the plus sign (+) to add a server.
  - \_\_ c. Leave **Create a Rule Execution Server connection** selected and click **Next**.
  - \_\_ d. Define these fields:
    - **Name:** Sample
    - **URL:** `http://localhost:9080/res`
    - **User name:** resAdmin
    - **Password:** resAdmin

### For IBM ODM on Cloud users

Use the name and URL of the Rule Execution Server for your IBM ODM on Cloud instance in the **URL** field instead of Sample and localhost.

- \_\_ e. Select **Save my login information**.
- \_\_ f. Click **Test Connection**.



You should see a Connection succeeded message.

- \_\_ g. Click **Finish**.
- \_\_ 3. Save your work by pressing Ctrl+S.
- \_\_ 4. Go to the **Overview** tab and deploy the RuleApp.
  - \_\_ a. In the **Deployment** section of the **Overview** tab, click **Proceed to RuleApp deployment**.
  - \_\_ b. On the Deployment Summary page, click **Next**.
  - \_\_ c. On the Credentials page, click **Next**.
  - \_\_ d. On the Version Summary page, click **Finish**.

After the deployment finishes, the Deployment Report page opens.

- \_\_\_ e. Take note of the names of the deployed RuleApp and ruleset:
- RuleApp: my\_deployment
  - Ruleset: my\_operation

## Section 2. Getting the HTDS WSDL file from the deployed ruleset

In this section, you retrieve the HTDS WSDL file, which is the definition of the web service that is associated with the deployed ruleset.

Now that the `htdsRuleApp` is deployed with its associated managed XOM in Rule Execution Server, open the Rule Execution Server console to retrieve the corresponding decision service, and its WSDL interface.

- 1. Open the Rule Execution Server console.
  - a. Click **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Rule Execution Server console**.
  - b. In both the **User Name** field and the **Password** field, enter: `resAdmin`
- 2. Click the **Explorer** tab and in the list of RuleApps, click the `my_deployment` RuleApp that you deployed.
- 3. Click the latest version of the `my_operation` ruleset to open it in the Ruleset View.  
The Ruleset View page opens with the details of this ruleset.
- 4. On the Ruleset View page, click **Show HTDS Options**.



You see a series of options. If you have specific requirements in your enterprise, you can edit these options. For this exercise, you use the default settings.

 A screenshot of a configuration dialog titled "Hosted Transparent Decision Services (HTDS) Options". It contains four entries with URLs:
 

- Location:** <http://localhost:9080/DecisionService>
- Web service endpoint:** <http://localhost:9080/DecisionService>
- Target namespace (SOAP only):** [http://www.ibm.com/rules/decisionservice/My\\_deployment/My\\_operation](http://www.ibm.com/rules/decisionservice/My_deployment/My_operation)
- Parameter target namespace:** [http://www.ibm.com/rules/decisionservice/My\\_deployment/My\\_operation/param](http://www.ibm.com/rules/decisionservice/My_deployment/My_operation/param)

- 5. On the toolbar of the Ruleset View page, click **Retrieve HTDS Description File**.

A screenshot of a toolbar with several buttons. The button labeled "Retrieve HTDS Description File" is highlighted with a red rectangle. Other visible buttons include "Upload Ruleset Archive", "Add Managed URI", "Add Property", and "Edit".

**Information**

A transparent decision service is technically either a strongly typed web service (WSDL) or a REST API that provides an interface to access a deployed rule set. The Decision Service component can pass one or more input parameters to the rule engine and access the return values. The transparent decision service support includes traceability from decision services to rules, runtime monitoring, and version management.

The HTDS is available in two forms: SOAP and REST. You can retrieve the file to view or to download.

**Retrieve HTDS Description File**

**/my\_deployment/1.0/my\_operation/1.1**

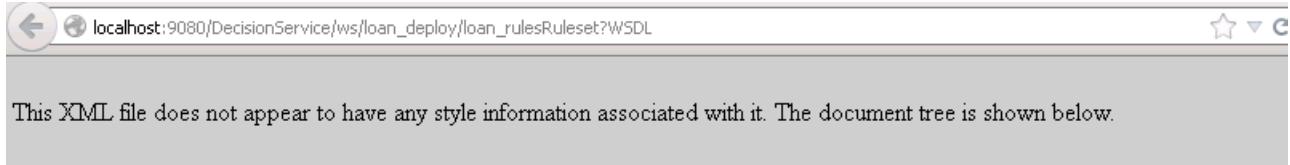
Service protocol type  SOAP  REST

Latest ruleset version  
 Latest RuleApp version  
 Decision trace information  
 Inline types in separate XSD files  
 Compatibility mode

You can retrieve the HTDS with the following options:

<b>Latest ruleset version</b>	You select <b>Latest ruleset version</b> , <b>Latest RuleApp version</b> , or both, to create a WSDL file that either uses a canonical ruleset path or not. In the present case, you select both check boxes to create a WSDL file that is based on a non-canonical ruleset path (where both versions are ignored) to execute the most recent RuleApp and ruleset versions.
<b>Latest RuleApp version</b>	
<b>Decision trace information</b>	Select <b>Decision trace information</b> to set filters in the request, and later retrieve decision traces in the response.
<b>Inline types in separate XSD files</b>	You select or clear this check box, depending on whether you have models with the same name and the same namespace, and on how you want to manage their potential conflict. For more information, see the product documentation. For this exercise, you can clear this check box.
<b>Compatibility mode</b>	You select or clear this check box, depending on whether you want to create a WSDL file that is compatible with a previous version of Operational Decision Manager. If you select it, you must also select the appropriate version. In this course, you do not try to be compatible with any previous version of the product, so clear this check box.

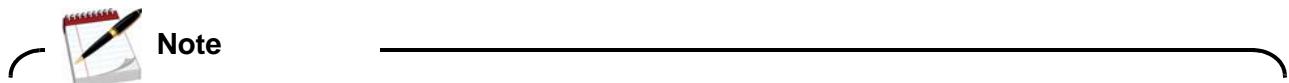
- \_\_\_ 6. Retrieve the SOAP HTDS by selecting these options.
  - \_\_\_ a. For **Service protocol type**, choose **SOAP**.
  - \_\_\_ b. Select the **Latest ruleset version** and **Latest RuleApp version** options.
  - \_\_\_ c. Click **View** to see the contents of the WSDL file.
- \_\_\_ 7. The WSDL file opens in a new browser tab.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <wsdl:definitions name="Loan_rulesRulesetDecisionService" targetNamespace="http://www.ibm.com/rules/decisionservice/Loan_rulesRuleset">
 - <wsdl:types>
 - <xsd:schema targetNamespace="http://www.ibm.com/rules/decisionservice/Loan_deploy/Loan_rulesRuleset/param">
 <xsd:import namespace="http://www.ibm.com/rules/decisionservice/Loan_deploy/Loan_rulesRuleset"/>
 - <xsd:element name="borrower">
 - <xsd:complexType>
 - <xsd:sequence>
 <xsd:element name="borrower" type="xsd:string"/>
```

- \_\_\_ 8. Save the WSDL file to your workspace.
  - \_\_\_ a. Return to the browser window that is running the Rule Execution Server console, ensure that you are still on the Ruleset View page, and click **Retrieve HTDS Description File**.
  - \_\_\_ b. Keep the **SOAP** option selected, and then select **Latest ruleset version** and **Latest RuleApp version**, and click **Download**.
  - \_\_\_ c. Save the WSDL file to the decision service project in your workspace and rename it to MyDecisionService.wsdl. By default, the workspace path is:  
`<LabfilesDir>workspaces\htds\Miniloan Rules`



If you use Firefox to run the Rule Execution Server console, it saves the WSDL file to the **Downloads** folder by default.

After clicking **Download**, select **Save File** and click **OK**. To get to the **Downloads** folder, open Windows Explorer and in the left pane, click **Favorites > Downloads**.

You can then rename the WSDL file and copy it to the **Miniloan Rules** folder in your workspace.

## 2.1. Opening the WSDL in Rule Designer

- \_\_\_ 1. Go back to Rule Designer.
- \_\_\_ 2. In the Rule Explorer, right-click **Miniloan Rules** and click **Refresh**.

You see the `MyDecisionService.wsdl` file in the rule project.

- \_\_\_ 3. Double-click **MyDecisionService.wsdl** to open it in the WSDL editor.  
The WSDL editor shows **My\_operationDecisionService** with an operation of request-response type that has an input message, an output message, and a SOAP fault.
- \_\_\_ 4. Close the WSDL editor.
- \_\_\_ 5. Switch to the Java perspective:
  - \_\_\_ a. From the **Window** menu, click **Open Perspective > Other > Java (default)**.
  - \_\_\_ b. Click **OK**.
- \_\_\_ 6. In the Package Explorer, expand **Miniloan Rules**.
- \_\_\_ 7. Right-click **MyDecisionService.wsdl** and click **Web Services > Test with Web Services Explorer**.
- \_\_\_ 8. In the Web Services Explorer, under Operations, click the **My\_operation** link.  
You see the WSDL binding details.
- \_\_\_ 9. Enter the following values in the empty fields for the borrower:
  - **creditScore**: 100
  - **yearlyIncome**: 70000
- \_\_\_ 10. Enter the following values in the empty fields for loan:
  - **amount**: 8000
  - **duration**: 12
  - **yearlyInterestRate**: 2.5
- \_\_\_ 11. Click **Go** to call the web service on Rule Execution Server.
- \_\_\_ 12. In the **Status** section, review the response and note the **messages** field.  
The loan is rejected because the credit score is below the minimum limit and the debt-to-income ratio is too high.
- \_\_\_ 13. Close the Web Services Explorer, and switch back to the Rule perspective.

## **End of exercise**

## Exercise review and wrap-up

This exercise looked at how you can use hosted transparent decision services by retrieving and examining the HTDS WSDL file.



# Exercise 19. Auditing ruleset execution through Decision Warehouse

## What this exercise is about

This exercise describes how to enable monitoring of ruleset execution and how to audit execution traces in Decision Warehouse.

## What you should be able to do

After completing this exercise, you should be able to:

- Enable monitoring for ruleset execution
- Retrieve decision traces through Decision Warehouse
- Optimize Decision Warehouse
- Delete trace data from Decision Warehouse

## Introduction

After the rules are deployed in the execution environment, you must be able to audit which rules were executed to determine the decision. In this exercise, you learn how to enable monitoring for a ruleset and deploy it from Rule Designer. Next, you test rule execution to generate rule execution traces that you can review in Decision Warehouse.

You also configure monitoring properties and deploy from Decision Center to see how execution traces in Decision Warehouse can link you directly to the corresponding rule artifacts in Decision Center.

The exercise includes these sections:

- Section 1, "Enabling ruleset monitoring"
- Section 2, "Retrieving decision traces in the Rule Execution Server console"
- Section 3, "Optimizing Decision Warehouse"
- Section 4, "Deleting trace information from the database"

## Requirements

This exercise uses the web application that you deployed during Exercise 15, "Managing deployment". You should complete these exercises before proceeding:

- Exercise 16, "Exploring the Rule Execution Server console"
- Exercise 17, "Executing rules with Rule Execution Server in Java EE"

This exercise uses the files that are installed in the  
*<InstallDir>\studio\training\Dev 19 - Decision  
Warehouse\01-start* directory.



**Note**

**For IBM ODM on Cloud users**

This exercise requires that you use the on-premises version of ODM. ODM on Cloud does not support Decision Warehouse.

## Section 1. Enabling ruleset monitoring

You can set properties on your ruleset to capture execution traces and determine the behavior of the ruleset during execution.

When you enable monitoring, you can retrieve information such as:

- How many tasks were executed
- How many rules were executed
- What events took place in the working memory
- Which version of the ruleset was executed

You set monitoring properties at ruleset level. You can set this ruleset property in the Rule Execution Server console by selecting the ruleset and clicking **Add Property**. By default, all the information is retrieved, except working memory events.

In this section, you enable monitoring properties on your ruleset, redeploy it, and then execute it to generate traces. After execution, you review the traces in Decision Warehouse.

### 1.1. Setting up your environment for this exercise

This exercise uses the same project that you used during Exercise 16, "Exploring the Rule Execution Server console".

- \_\_\_ 1. If the sample server is not started, start it now by clicking **Start > All Programs > IBM > Operational Decision Manager V8.8 > Sample server > Start server**.
- \_\_\_ 2. In Rule Designer, switch to a new workspace:
  - \_\_\_ a. From the **File** menu, click **Switch Workspace > Other**.
  - \_\_\_ b. In the Workspace Launcher window, enter the path:  
`<LabfilesDir>\workspaces\decision_warehouse`
- \_\_\_ 3. Close the Welcome view.
- \_\_\_ 4. Import the lab projects.
  - \_\_\_ a. From the **File** menu, click **Import**.
  - \_\_\_ b. On the Select page, select **Operational Decision Manager > Samples and Tutorials**, and click **Next**.
  - \_\_\_ c. In the **Rule Designer > Training** node, select **Ex 19: Auditing ruleset execution > 01-start** and click **Finish**.
- \_\_\_ 5. Close the Help view.



## Troubleshooting

You might see errors on the `loan-rules` project after the workspace finishes building. The error is on the deployment configuration. You can ignore the error for now because you fix it during the exercise.

## 1.2. Define the target server

In this section, you define the target server in the deployment configuration.

- \_\_\_ 1. In Rule Explorer, expand the **loan-rules > deployment** folder and double-click the deployment configuration to open the configuration editor.
- \_\_\_ 2. Click the **Target Servers** tab to define the target server.
  - \_\_\_ a. If the Sample server is listed (without a URL), you can select it and delete it by clicking the X.
  - \_\_\_ b. Click the plus sign (+) to add a server.
  - \_\_\_ c. Leave **Create a Rule Execution Server connection** selected and click **Next**.
  - \_\_\_ d. Define these fields:
    - **Name:** Sample
    - **URL:** `http://localhost:9080/res`
    - **User name:** resAdmin
    - **Password:** resAdmin
  - \_\_\_ e. Select **Save my login information**.
  - \_\_\_ f. Click **Test Connection**.  
You should see the `Connection succeeded` message.
  - \_\_\_ g. Click **Finish**.
- \_\_\_ 3. Save your work.  
The errors on the rule project should now be gone.

## 1.3. Enabling monitoring

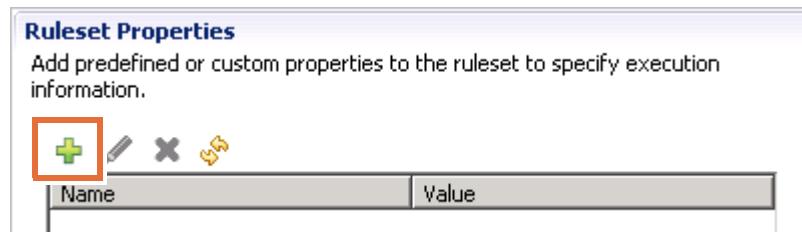
In this section, you use Rule Designer to set these properties on the ruleset:

- `monitoring.enabled` with its value set to `true`
- `rulesetSEQUENTIAL.trace.enabled` with its value set to `true`

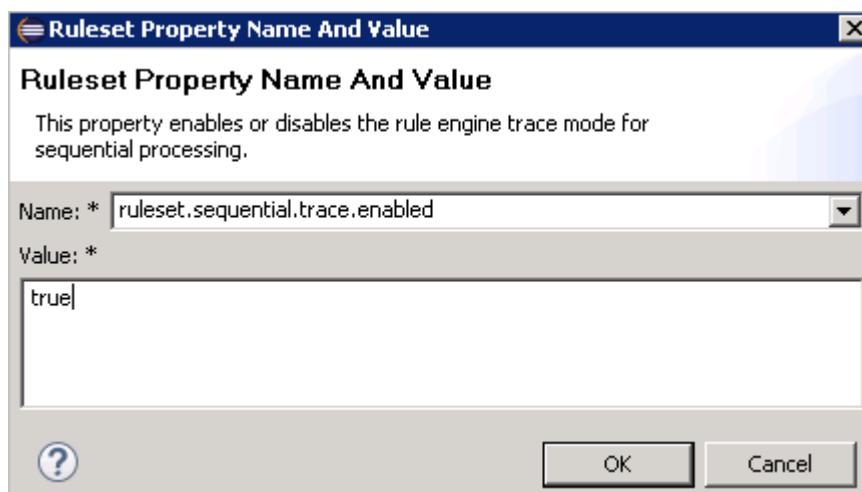
You use the `rulesetSEQUENTIAL.trace.enabled` property when the ruleset contains tasks that use the sequential or the Fastpath execution modes.

## Enabling monitoring of your ruleset

- \_\_\_ 1. Open the **Decision Operations** tab of the deployment configuration.
- \_\_\_ 2. In the **Configured Decision Operations** section, click **loan-rulesOperation**.  
The **Ruleset Properties** section becomes enabled.
- \_\_\_ 3. Define the ruleset properties.
- \_\_\_ a. Click the plus sign (+) to add a ruleset property.



- \_\_\_ b. In the Ruleset Property Name And Value window, in the **Name** field list, select the `rulesetSEQUENTIALTRACEenabled` property.
- \_\_\_ c. In the **Value** field, type `true` and click **OK**.



- \_\_\_ d. Click the plus sign (+) again, select the `monitoring.enabled` property, and set the **Value** field to: `true`
- \_\_\_ 4. Save your work (Ctrl+S).



### Information

Later, you see how to set or modify the ruleset properties through the Rule Execution Server console after they are deployed.

## 1.4. Deploying the updated RuleApp from Rule Designer

In this section, you redeploy the RuleApp from Rule Designer to Rule Execution Server.

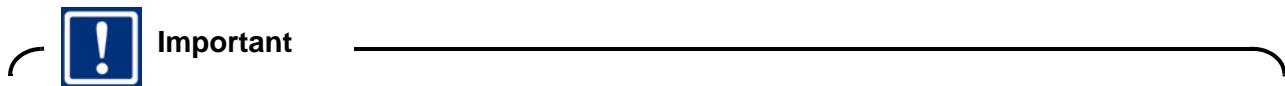
- \_\_\_ 1. In the deployment configuration editor, go to the **Overview** tab.
- \_\_\_ 2. In the **Deployment** section, click **Proceed to RuleApp deployment**.
- \_\_\_ 3. Click **Next** on both the Deployment Summary page and on the Credentials page, and then click **Finish**.
- \_\_\_ 4. In the Deployment Report, make a note of the RuleApp and ruleset versions. You use these version numbers later.

## 1.5. Running the client application

In this section, you run the `Loan Validation` application that you created in Exercise 17, "Executing rules with Rule Execution Server in Java EE" to generate some traces. Now that you defined monitoring properties on your ruleset, the monitoring properties capture decision trace data when the client application calls that ruleset for execution. The data is stored in Decision Warehouse.

- \_\_\_ 1. Open the client application in a web browser at the following URL:

`http://localhost:9080/LoanValidation`



Make sure that you use the correct URL and port for your environment.

- \_\_\_ 2. Click **Start Loan Validation** and run the application with the default values.

Next, you work in Rule Execution Server console to view the execution traces that you generated.

## Section 2. Retrieving decision traces in the Rule Execution Server console

In this section, you work in the Rule Execution Server console. You examine the RuleApp that you deployed from Rule Designer. You then explore the decision traces that are associated with the execution of the ruleset that is packaged in this RuleApp.

- 1. If the Rule Execution Server console is not already open, sign in as an administrator.
  - a. Open a web browser at the following web address:  
`http://localhost:9080/res`
  - b. In the **Username** and **Password** fields, enter `resAdmin` and click **Sign In**.



### Troubleshooting

In some cases, Rule Execution Server or the Loan Validation application might fail to load when you type the URL in a browser.

If you experience this problem, closing and reopening the browser can resolve the issue. Otherwise, close your browsers again and try starting Rule Execution Server console from the **Start** menu.

- 2. In the Rule Execution Server console, examine the RuleApp that you deployed from Rule Designer.
  - a. Click the **Explorer** tab.
  - b. On the RuleApps View page, click the **loan\_deploy** RuleApp version that you deployed in "Deploying the updated RuleApp from Rule Designer" on page 19-6.
  - c. In the list of rulesets, click the most recent version of the `loan_rulesRuleset` ruleset for this RuleApp.

The Ruleset View page opens for the `loan_rulesRuleset` ruleset. You can see in the details of this ruleset that the value of its **Display Name** field is:

`loan-rulesOperation`

- d. Click **Show Properties** for this ruleset.

You can see the ruleset properties that you explicitly defined in Rule Designer for this ruleset:

- `monitoring.enabled`
- `ruleset.sequential.trace.enabled`

- 3. In the Rule Execution Server console, click the **Decision Warehouse** tab.

The **Decision Warehouse** tab includes these tasks:

- **Search Decisions:** To define filters on the data source to find only the events or decisions of interest.

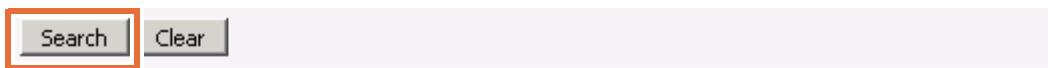
- **Persistence Properties:** To select the data source for Decision Warehouse to query during your web session. Decision Warehouse is installed with a default data source. You can add more sources, such as a historical database.  
When your web session expires, the active data source reverts to the default data source.
- **Clear Decisions:** To delete existing decisions from the Decision Warehouse.

By default, the Search Decisions page opens. You can switch between these pages with the “Select a task” left pane of the **Decision Warehouse** tab.



- 4. On the Search Decisions page, leave all fields empty and click **Search**.
- 5. Scroll down the page to see the results.

All decisions that are stored in Decision Warehouse are listed because you did not specify any search filters.



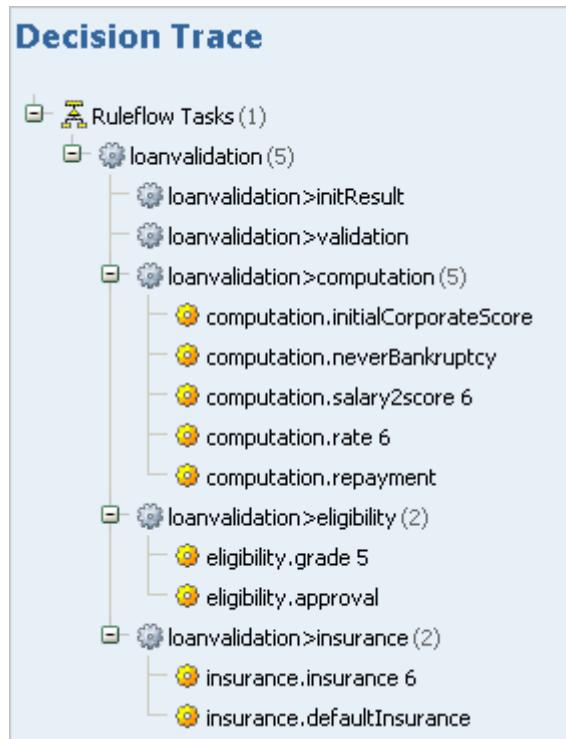
Decision ID	Date	Ruleset Version	Number of rules fired	Decision Trace
d7b06892-dfba-4857-9d2c-15ae69d841280	2016-02-05 12:50:09	/loan_deploy /1.0/loan_rulesRuleset/1.2	9	<a href="#">View Decision details</a>

- 6. In the **Decision Trace** column for the most recent decision, click the **View Decision details** link.  
The Decision Trace page opens, and shows the details of the execution.
- 7. Look at the fields in the **Execution Details** section to relate them to the ruleset execution.

<b>Decision ID:</b>	d7b06892-dfba-4857-9d2c-15ae69d841280
<b>Date:</b>	2016-02-05 12:50:09
<b>Executed ruleset path:</b>	/loan_deploy /1.0/loan_rulesRuleset/1.2
<b>Engine type:</b>	cre
<b>Processing Time (ms)</b>	156
<b>Number of rules fired</b>	9
<b>Number of tasks executed</b>	6

Your RuleApp version and statistics might differ from this screen capture.

8. Expand the Ruleflow Tasks tree in the **Decision Trace** section.



### Questions

Do you understand the trace?

### Answer

Each node corresponds to a rule task or an executed rule artifact, such as a fired action rule, a row in a decision table, or a function. For a row in a decision table, the rank of the row is given. For example, `computation.rate 6` is the sixth row in the `computation.grade` decision table.

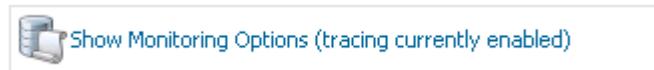
9. Close the Decision Trace window.

## Section 3. Optimizing Decision Warehouse

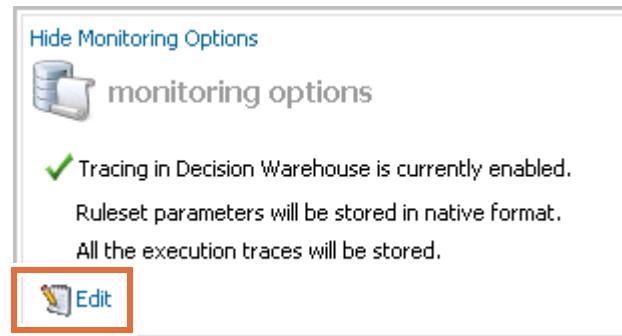
When the trace is enabled, Decision Warehouse captures all execution trace data. You can optimize Decision Warehouse by filtering which parts of the ruleset execution trace you monitor and store, and by turning off BOM serialization.

### 3.1. Working with monitoring options

- \_\_\_ 1. In Rule Execution Server console, click the **Explorer** tab and expand RuleApps in the Navigator pane to reopen the `loan_deploy` RuleApp that you deployed.
- \_\_\_ 2. Open the Ruleset View page for the latest `loan_rulesRuleset` ruleset of your RuleApp. This ruleset is the same one that you worked with during Section 2, "Retrieving decision traces in the Rule Execution Server console," on page 19-7 to set the ruleset properties.
- \_\_\_ 3. Click **Show Properties** to reopen the list of ruleset properties, and note the `monitoring.enabled` ruleset property, which should be set to `true`.
- \_\_\_ 4. Click **Show Monitoring Options**.



- \_\_\_ 5. Identify the connection between the **Enable tracing in Decision Warehouse** option and the `monitoring.enabled` ruleset property.
  - \_\_\_ a. Click **Edit** to open the monitoring options.



- \_\_\_ b. Notice that **Enable tracing in Decision Warehouse** is selected.
- \_\_\_ c. Clear **Enable tracing in Decision Warehouse** and click **Save**.
- \_\_\_ d. If necessary, refresh Rule Execution Server console by pressing F5.



#### Troubleshooting

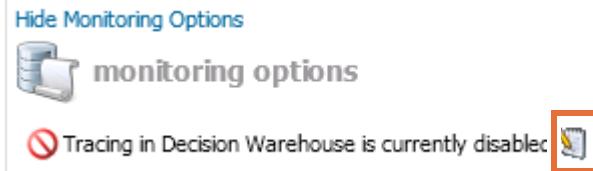
If you see a prompt from the web browser to confirm whether you want to resend information, click **Resend or Retry**.

- \_\_\_ e. Look in the **properties** section to verify that the `monitoring.enabled` ruleset property is now set to: `false`

4 properties		
Select All	Name	Value
<input type="checkbox"/>	monitoring.enabled	false
<input type="checkbox"/>	monitoring.filters	

- \_\_\_ 6. Enable decision tracing again.

- \_\_\_ a. Reopen the **monitoring options** section and click the **Edit** icon.



- \_\_\_ b. Select **Enable tracing in Decision Warehouse**.

- \_\_\_ c. Click **Save**.

Notice that the monitoring options section now lists specific execution traces.

### 3.2. Specifying filters on the trace data

To reduce the information that is stored in an execution trace in Decision Warehouse, you:

- Apply the `monitoring.filters` property to the ruleset
- Select which filters to set so that you can refine the data that is stored

You can set the `monitoring.filters` property filter values in Rule Designer or Decision Center before you deploy a ruleset. You can also set this property on the deployed ruleset in Rule Execution Server console, which you do next.

- \_\_\_ 1. In the **monitoring options** section, click **Edit** to reopen the monitoring properties.  
 \_\_\_ 2. In the **Select the execution traces to store in the Decision Warehouse** list, notice the complete list of execution traces that are available and which traces are selected.

The selected traces are listed as values of the `monitoring.filters` ruleset property.

monitoring.enabled	true
monitoring.filters	INFO_EXECUTION_DATE=true,INFO_EXECUTION_DURATION=true,INFO_TOTAL_T

- \_\_\_ 3. Select *all* the remaining execution traces in the list and click **Save**.  
 \_\_\_ 4. Notice that the `monitoring.filters` property disappears from the list of properties.  
 Selecting all the optional filters is equivalent to turning off the filter property.  
 \_\_\_ 5. Click **Edit** to reopen the monitoring options, clear all the selected traces, and then click **Save**.

- \_\_\_ 6. Notice that the `monitoring.filters` property is in the list of properties again, but its value is empty.

<input type="checkbox"/>	 monitoring.enabled	 true
<input type="checkbox"/>	 monitoring.filters	
<input type="checkbox"/>	 ruleset.bom.enabled	 true

- \_\_\_ 7. Turn on the default monitoring filters again.

- \_\_\_ a. Click **Edit** to reopen the monitoring options, clear **Enable tracing in Decision Warehouse** and click **Save**.
- \_\_\_ b. Again, click **Edit** to reopen the monitoring options, select **Enable tracing in Decision Warehouse** and click **Save**.

### 3.3. Removing BOM serialization

When the `ruleset.bom.enabled` property is set to `true`, it converts the list of objects in the ruleset parameters into a memory buffer by using BOM serialization.



#### Important

For large amounts of input and output data, BOM serialization can result in poor performance.

To optimize performance, you can turn off BOM serialization.

- \_\_\_ 1. If the **properties** section for this ruleset is closed, click **Show Properties**, and click the **Edit** icon for the `ruleset.bom.enabled` ruleset property.
- \_\_\_ 2. Set `ruleset.bom.enabled` to `false` and click the **Save** icon.



ruleset.bom.enabled



false



#### Note

You can edit only one property at a time. If you try to edit the values of more than one property, only one of the values is saved.

- \_\_\_ 3. In the **monitoring options** section, click **Edit** to reopen the monitoring options.

- \_\_\_ 4. Notice that **Native format** is the option for storing the values of ruleset parameters.

When BOM serialization is turned off, the BOM objects that are passed as parameters are stored either in XML for dynamic XOMs or in a string representation for Java XOMs.

- \_\_\_ 5. Rerun the Loan Validation application with the default values to generate a new decision trace in Decision Warehouse.

If the browser is still open, click **Try again**. Otherwise, open a browser at this URL:

<http://localhost:9080/LoanValidation>

- 6. In Rule Execution Server console, open the **Decision Warehouse** tab, and click **Search** to get the latest traces.
- 7. Click **View Decision details** for your latest trace and notice the format for the input and output parameters.

#### Input Parameters

```
borrower = Borrower(firstName: John lastName: Doe born: May 12, 1968 SSN: 123-45-6789 zip code: 06560 income: 100000 credit score: 600)
loan = Loan(amount: 100000 starting: Feb 5, 2016 duration: 72 month LTV: 70%)
```

#### Output Parameters

```
report = Report(validData: true approved: true score: 820 grade: B insuranceRequired: true insuranceRate: 2% message: Low risk loan
Congratulations! Your loan has been approved)
loan = Loan(amount: 100000 starting: Feb 5, 2016 duration: 72 month LTV: 70% rate: 5.7% monthly repayment: 1,643.16)
```

#### Execution output

```
Borrower(firstName: John lastName: Doe born: May 12, 1968 SSN: 123-45-6789 zip code: 06560 income: 100000 credit score: 600
Loan(amount: 100000 starting: Feb 5, 2016 duration: 72 month LTV: 70% rate: 5.7% monthly repayment: 1,643.16)
Report(validData: true approved: true score: 820 grade: B insuranceRequired: true insuranceRate: 2% message: Low risk loan
Congratulations! Your loan has been approved
)
```

- 8. Click **View Decision details** for the previous trace and take a moment to compare the formats of the input and output parameters between the two traces.
- 9. Close the Decision Trace windows.

## Section 4. Deleting trace information from the database

You can delete trace information from the Decision Warehouse database by specifying the ruleset paths or execution dates.

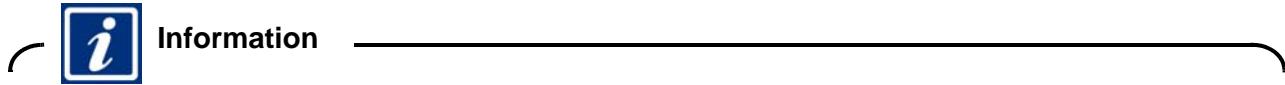
- \_\_\_ 1. In the Rule Execution Server console, click the **Decision Warehouse** tab.
- \_\_\_ 2. Click **Search Decisions > Search** and look at the list of decisions to see RuleApp version numbers and dates.
- \_\_\_ 3. In the “Select a task” pane, click **Clear Decisions**.



- \_\_\_ 4. On the Clear Decisions page, you can either leave all fields empty to delete all traces, or specify which traces to delete, according to the RuleApp versions and dates that you saw listed.

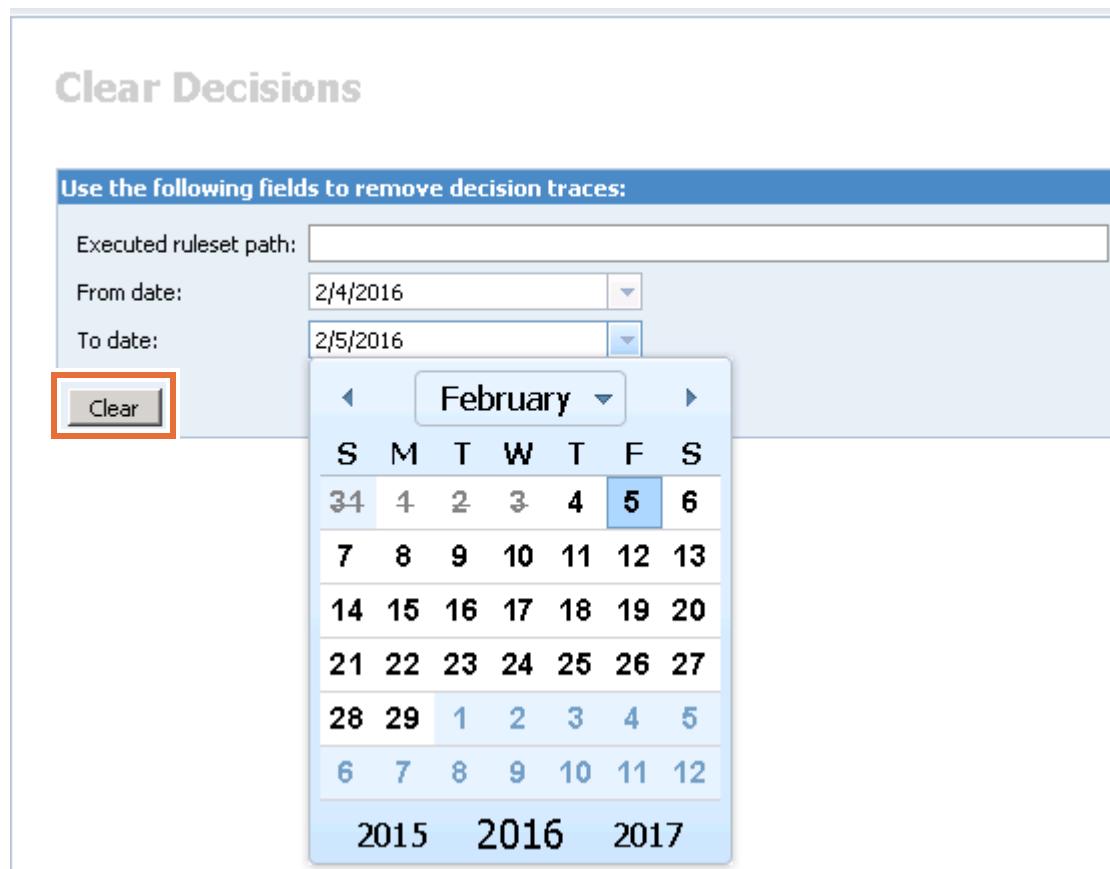
For example, if you want to delete traces for a specific RuleApp version, you can specify which version to delete.

- \_\_\_ a. Leave the **Executed ruleset path** field blank.

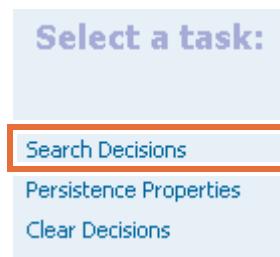


- \_\_\_ b. In the **From Date** field, click the calendar icon to specify a date, such as yesterday's date, or leave the field blank.

- \_\_\_ c. In the **To Date** field, click the calendar to specify a date, such as today's date, or leave the field blank.



- \_\_\_ 5. Click **Clear** to delete the specified traces.  
A warning message prompts you to confirm the deletion.
- \_\_\_ 6. Click **Confirm** to delete the traces.
- \_\_\_ 7. To view the remaining traces, you can click **Search Decisions** and click **Search** to verify the deletion.



**End of exercise**

## Exercise review and wrap-up

This exercise looked at how you can use Decision Warehouse to audit ruleset executions.

# Exercise 20. Working with the REST API

## What this exercise is about

This exercise describes how to use the REST API for ruleset execution and resource management.

## What you should be able to do

After completing this exercise, you should be able to:

- Use the REST API to test ruleset execution and manage RuleApp resources

## Introduction

In this exercise, you learn how to work with the REST API.

The exercise includes these sections:

- Section 1, "Using REST services to test ruleset execution"
- Section 2, "Using REST services to deploy and execute rules"



### Note

#### For IBM ODM on Cloud users

Section 2, "Using REST services to deploy and execute rules" in this exercise is not supported in IBM ODM on Cloud.

## Requirements

You should complete this exercise before proceeding:

- Exercise 16, "Exploring the Rule Execution Server console"

This exercise uses some steps from the REST sample that is provided with the product, along with other steps to test ruleset execution in the RES console.

## Section 1. Using REST services to test ruleset execution

To access REST API test tool for ruleset execution, you first open the Ruleset View page of Rule Execution Server console.

### Testing the ruleset execution for a ruleset

- 1. If Rule Execution Server console is closed, open it and sign in with `resAdmin` for the user name and password.
- 2. Click the **Explorer** tab.
- 3. In the list of RuleApps, click **my\_deployment**.
- 4. Click the most recent version of the `my_operation` ruleset to open it in the Ruleset View page.
- 5. On the toolbar in the Ruleset View, click **Retrieve HTDS Description File**.



- 6. Select the **REST** option and click **Test**.

**Retrieve HTDS Description File**

/my\_deployment/1.0/my\_operation/1.3

Service protocol type  SOAP  REST

Latest ruleset version  
 Latest RuleApp version  
 Decision trace information  
 Inline types in separate XSD files

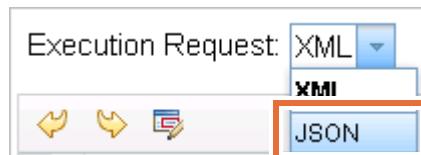
Cancel View Download Test



If the **Latest ruleset version** and other options on this page are selected, you can ignore them as they do not affect the REST test tool.

The test tool opens in a new browser tab. You can modify the execution request to set the ruleset parameter values with test data.

- \_\_\_ 7. Change the Execution Request format to **JSON**.



- \_\_\_ 8. Use the following values for the borrower and loan attributes in the JSON request:

```
{
 "borrower": {
 "name": "Joe",
 "creditScore": 600,
 "yearlyIncome": 80000
 },
 "loan": {
 "amount": 500000,
 "duration": 240,
 "yearlyInterestRate": 0.05
 },
 "__DecisionID__": "12345"
}
```



**Hint**

You can find this code snippet in the `<LabfilesDir>\code\rest.txt` file.

- \_\_\_ 9. Click **Execute Request**.

- \_\_\_ 10. Scroll down to see the server response.

The result includes the `status` attribute, which returns `Rejected` for this test.

- \_\_\_ 11. Close the REST Service browser tab.

## Section 2. Using REST services to deploy and execute rules



### Information

This part of the exercise is based on the REST API sample that is provided with the product.



### Note

#### For IBM ODM on Cloud users

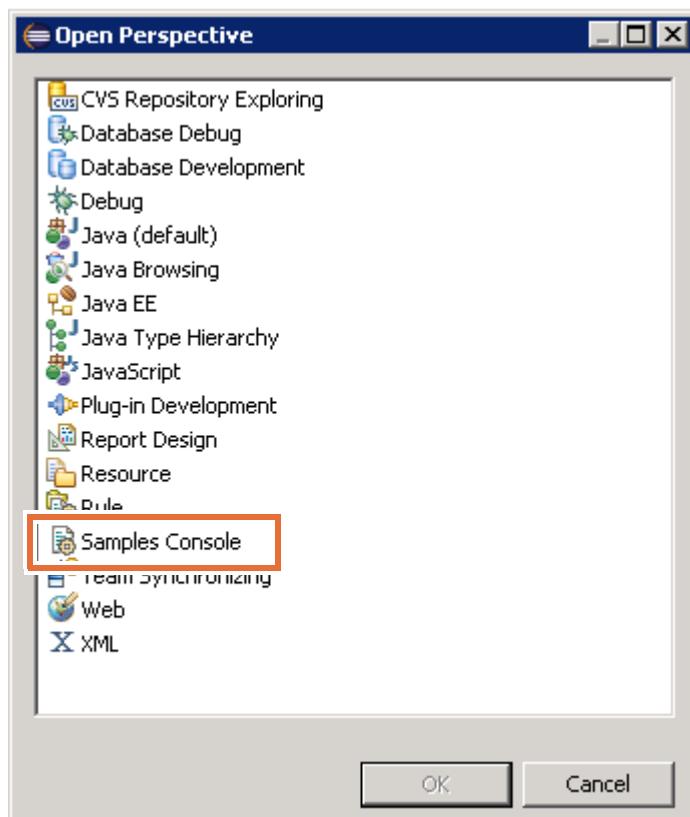
This part of the exercise is not supported in IBM ODM on Cloud.

### 2.1. Setting up your environment for this exercise

- 1. In Rule Designer, switch to a new workspace:
  - a. From the **File** menu, click **Switch Workspace > Other**.
  - b. In the Workspace Launcher window, enter the path:  
`<LabfilesDir>\workspaces\rest`
- 2. Close the Welcome view.
- 3. Switch to the Samples Console.
  - a. In the upper-right corner, click the **Open Perspective** icon to switch from the Rule perspective.



- \_\_\_ b. In the Open Perspective list, select **Samples Console**, and click **OK**.



The “Samples and Tutorials” page opens.

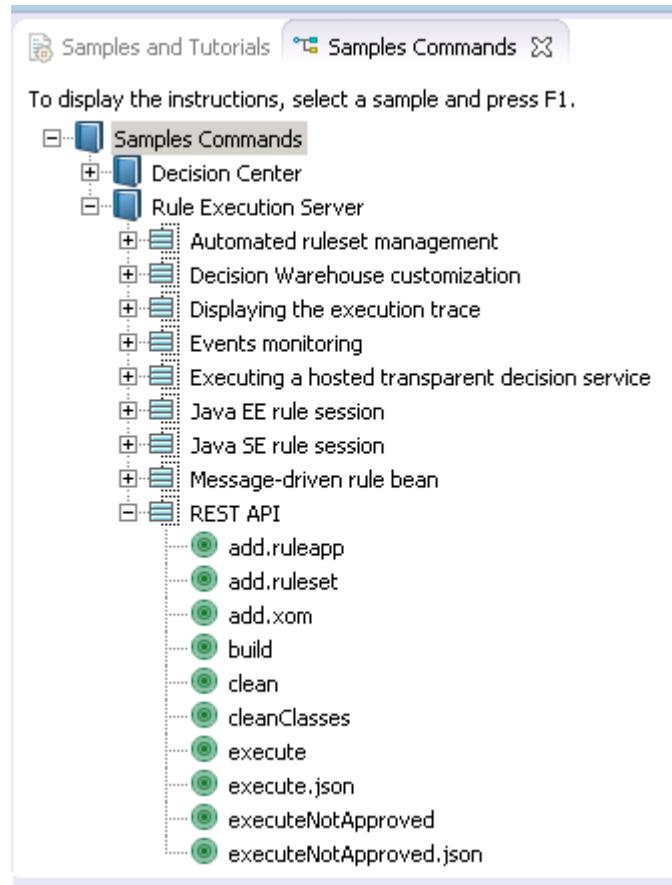
- \_\_\_ 4. In the **Rule Execution Server** section, expand **Samples**, and scroll down in the Samples Console window to find the **REST API in Java** sample.



- \_\_\_ 5. In the **REST API in Java** section, click **View sample commands** to open the Samples Commands page.



The **Samples Commands** tab opens to the REST API sample commands.



6. Go back to the Samples and Tutorials page and click **Import projects**.



The workspace opens in the Java perspective with the projects that are used in this sample. You can expand the projects to view the code that was used and to better understand this sample.

## 2.2. Deploying the RuleApp resources

1. Switch back to the Samples Console perspective.
2. On the **Samples Commands** tab, double-click the `add.ruleapp` command.

The Console view opens so you can see the results of this command, which compiles the code and completes these tasks:

- Checks whether the scenario can run
- Lists all the RuleApps
- Adds a RuleApp
- Lists all the RuleApps again to show the new one

- Modifies the RuleApp
  - Shows the modified RuleApp
- \_\_\_ 3. Verify the results of adding a RuleApp in the Rule Execution Server console.
- \_\_\_ a. If Rule Execution Server console is closed, open it and sign in with `resAdmin` for the user name and password.  
If your session timed out, sign in again.
- \_\_\_ b. Click the **Explorer** tab.  
You might need to refresh the view (F5). If your session timed out, sign in again with `resAdmin` as the user name and password.
- \_\_\_ c. On the **Explorer** tab, notice that the RuleApps View now lists the newly added `myRuleApp` RuleApp.
- \_\_\_ d. Click the new RuleApp to see its description:  
`My REST modified RuleApp`  
The RuleApp does not contain a ruleset.
- \_\_\_ 4. Go back to the **Samples Commands** tab in Rule Designer, and double-click the `add.xom` command.  
The command deploys the XOM.
- \_\_\_ 5. Return to the **Explorer** tab in Rule Execution Server console, and expand **Resources** to see the newly added resource: `myxom.zip/1.0`  
This XOM is not referenced by any rulesets.
- \_\_\_ 6. Back in the **Samples Commands** tab in Rule Designer, double-click the `add.ruleset` command.
- \_\_\_ 7. Return to the **Explorer** tab in Rule Execution Server console, and expand **RuleApps > myRuleApp/1.0** in the Navigator pane to see that `myRuleApp/1.0` now contains a ruleset: `myRuleset`.



If you do not see `myRuleApp/1.0 > myRuleset`, refresh the page (F5).

If you see a message that the browser must resend information, click **Resend** or **Retry**.

- \_\_\_ 8. Click `myRuleset` to open it in the Ruleset View and see the value of the **Description** field:  
`My REST ruleset`

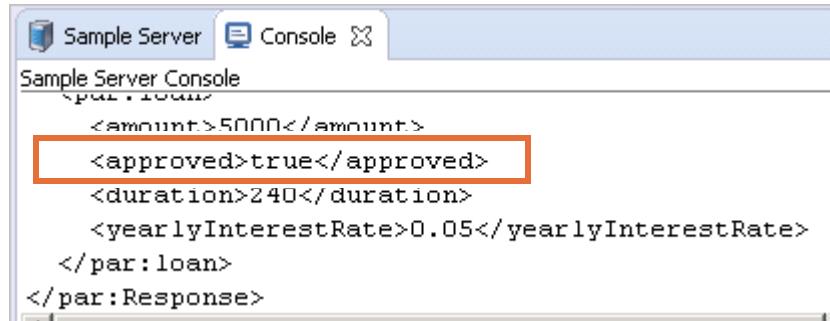
## 2.3. Testing ruleset execution

- \_\_\_ 1. Go back to the **Samples Commands** tab in Rule Designer, and double-click the **execute** command.

The command compiles the code and processes as follows:

- Checks whether the scenario can run.
- Passes the ruleset parameters in XML format and executes the ruleset to apply for a loan for an amount of 5000.

In the Console view, when you see the `BUILD SUCCESSFUL` message, scroll up to see the XML response with the **approved** field of the loan output parameter set to `true`.



```

Sample Server Console
<par:loan>
 <amount>5000</amount>
 <approved>true</approved>
 <duration>240</duration>
 <yearlyInterestRate>0.05</yearlyInterestRate>
</par:loan>
</par:Response>

```

- 2. On the **Samples Commands** tab, double-click the `executeNotApproved` command.

This time, the amount of the requested loan is 500000 and the loan is not approved. The Console view shows the **approved** field set to `false`.

Next, because the execution trace is enabled on the ruleset, you can view the execution trace in Decision Warehouse.

## 2.4. Viewing execution traces in Decision Warehouse

1. Go back to the Rule Execution Server console and click the **Decision Warehouse** tab.
2. On the Search Decisions page, click **Search** to search for the latest decisions.

You should see two decisions that are identified as **idApproved** and **idNotApproved**. Decision **idNotApproved** rejects the loan as the result of the execution of one rule.

## 2.5. Cleaning the scenario

1. Go back to the **Samples Commands** tab in Rule Designer and double-click the `clean` command.

After you see the `BUILD SUCCESSFUL` message in the Console view of Rule Designer, scroll up to see the list of all the RuleApps, which no longer includes `myRuleApp`. You also see the list of all the resources, which no longer includes `myxom.zip`.

2. Go back to the Rule Execution Server console and click the **Explorer** tab.

The RuleApps View does not list `myRuleApp` and the Resources View does not list `myxom.zip`.

## End of exercise

## Exercise review and wrap-up

The first part of the exercise looked at how you can access the REST API and generate a WADL representation of the REST API and its documentation. You also learned how to use REST services to view and manage deployed resources, and test ruleset execution.





**IBM**  
®