

Exercise Guide

Developing Rule Solutions in IBM Operational Decision Manager V8.10.5

Course code WB404 / ZB404 ERC 2.0



January 2021 edition

Notices

This information was developed for products and services offered in the US.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

© Copyright International Business Machines Corporation 2021.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Trademarks	ix
Exercises description	x
General exercise information	xiii
How to follow the exercise instructions	xiii
File references for exercises	xiv
Projects for exercises	xv
Sample server port	xvii
Using the product documentation	xvii
Checking for course corrections	xviii
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	1-7
2.1. Editing rules in Decision Center Business console	1-7
2.2. Reviewing the rule history	1-14
2.3. Changing the credit score requirements	1-16
Section 3. Validating rule changes	1-20
Section 4. Deploying updated rules from Decision Center to Rule Execution Server	1-23
4.1. Preparing the decision operation for deployment	1-23
4.2. Deploying the decision service	1-25
4.3. Customizing Decision Center settings	1-27
Section 5. Monitoring the deployed rules in Rule Execution Server	1-29
Section 6. Viewing the effects in the Miniloan application	1-32
Section 7. Exploring the development environments in Rule Designer	1-33
7.1. Opening Rule Designer	1-33
7.2. Taking a quick tour of Rule Designer	1-35
7.3. Synchronizing Rule Designer and Decision Center environments	1-38
Section 8. Shutting down the modules	1-45
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	2-8
2.3. Creating the BOM	2-10
2.4. Exploring the BOM	2-14
Section 3. Creating the main rule project for the decision service	2-16
3.1. Create the main rule project	2-16
3.2. Identify 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 variables for parameters	2-22
4.3. Binding the variables to parameters	2-24
4.4. Creating a ruleset variable	2-26

Section 5. Creating rule packages	2-28
Section 6. Publishing from Rule Designer to Decision Center	2-30
6.1. Publishing the rule project to Decision Center	2-30
6.2. Opening the project in the Decision Center Business console	2-34
6.3. Synchronizing Rule Designer with Decision Center	2-36
Section 7. Deleting a decision service from Decision Center	2-38
Section 8. Importing a decision service from Decision Center	2-39
8.1. Creating a project in Rule Designer from Decision Center	2-39
8.2. Finalizing the rule project in Rule Designer	2-41
Exercise 3. Modeling decisions	3-1
Section 1. Creating a diagram	3-2
1.1. Open Decision Center Business console	3-2
1.2. Create a model	3-3
1.3. Design the Match Race to Runner decision model	3-4
1.4. Model the node structure	3-4
1.5. Create custom data types	3-6
1.6. Set the output types for the data nodes	3-8
Section 2. Author and test the decision logic	3-9
2.1. Define the Runner category decision node and logic	3-9
2.2. Define the Matching race logic	3-10
2.3. Test the model	3-12
2.4. Add a rule to handle cases when no race matches the runner	3-14
2.5. Validate the model	3-15
2.6. Editing rule sequence	3-15
Exercise 4. Working with the BOM	4-1
Section 1. Finalizing the XOM	4-2
1.1. Setting up your environment for this exercise	4-2
1.2. Understanding the problem	4-3
1.3. Finish implementing the Borrower class	4-4
1.4. Finish implementing the Test class	4-11
Section 2. Creating a rule project with a BOM	4-12
2.1. Create a standard rule project with a BOM	4-12
2.2. Create a BOM entry in the loan-rules-bom project	4-12
Section 3. Working with the vocabulary that is required to author rules	4-15
3.1. Creating the default vocabulary for the Report class	4-15
3.2. Creating the default vocabulary for the Loan class	4-17
3.3. Examining the verbalization for members of the Borrower class	4-18
3.4. Editing the default verbalization	4-20
Exercise 5. Refactoring	5-1
Section 1. Refactoring rule artifacts after vocabulary changes	5-2
1.1. Setting up your environment for this exercise	5-2
1.2. Exploring the default verbalization	5-2
1.3. Changing the default verbalization of the getBankruptcyAge method	5-3
1.4. Changing the default verbalization of the duration member	5-5
Section 2. Maintaining consistency between the BOM and the XOM	5-6
2.1. Adding a XOM class that is missing in the BOM	5-6
2.2. Adding a XOM method that is missing in the BOM	5-8
2.3. Creating a rule to use this new attribute	5-9
2.4. Deprecating BOM members	5-10
Exercise 6. Working with ruleflows	6-1
Section 1. Exploring a ruleflow diagram	6-2
1.1. Setting up your environment for this exercise	6-2

1.2. Exploring the ruleflow	6-3
1.3. Exploring action tasks and transitions	6-7
1.4. Using parameters and variables in rules	6-10
Section 2. Creating a ruleflow	6-11
2.1. Create the ruleflow	6-11
2.2. Define the rule tasks	6-12
Section 3. Defining a main ruleflow	6-15
Exercise 7. Exploring action rules	7-1
Section 1. Exploring rule structure	7-2
1.1. Setting up your environment for this exercise	7-2
1.2. Exploring the rules	7-3
Section 2. Exploring rule behavior	7-5
2.1. Choose a rule to test	7-5
2.2. Test the rule	7-6
Section 3. Working with automatic variables	7-8
3.1. Create an automatic variable	7-8
3.2. Test the automatic variable in a rule	7-10
Exercise 8. Authoring action rules	8-1
Section 1. Setting up the rule project for authoring	8-2
1.1. Setting up your environment for this exercise	8-2
1.2. Create the rule project	8-3
1.3. Create a decision operation	8-3
1.4. Add a rule package	8-3
Section 2. Authoring rules with the Intellirule editor	8-4
2.1. Add a rule that is called: <code>grade</code>	8-4
2.2. Author the <code>grade</code> rule	8-5
2.3. Create more rules	8-6
Section 3. Authoring actions rules in the Guided editor	8-9
Section 4. Authoring rules with parameters	8-12
Exercise 9. Authoring decision tables	9-1
Section 1. Authoring a decision table	9-2
1.1. Setting up your environment for this exercise	9-2
1.2. Creating the table	9-3
1.3. Completing the table cells with values	9-9
Exercise 10. Working with static domains	10-1
Section 1. Exploring a collection domain	10-2
1.1. Setting up your environment for this exercise	10-2
1.2. Exploring the domain	10-3
1.3. Testing the rule	10-4
Section 2. Working with an enumeration of literals	10-8
2.1. Creating the domain	10-8
2.2. Testing the domain in a rule	10-11
Section 3. Defining an enumeration of static references	10-13
3.1. Creating the static references Java class	10-13
3.2. Authoring an action rule that uses a domain of static references	10-20
Exercise 11. Working with dynamic domains	11-1
Section 1. Creating a dynamic domain in Rule Designer	11-3
1.1. Setting up your environment for this exercise	11-3
1.2. Import the domain file	11-4
1.3. Create the domain	11-7
1.4. Associate a dynamic domain with the class	11-7

1.5. Correct the B2X error	11-12
1.6. Explore the dynamic domain	11-13
Section 2. Using the dynamic domain in a rule	11-15
2.1. Add a BOM attribute based on the dynamic domain	11-15
2.2. Write a rule that uses the new <code>type</code> BOM attribute	11-16
Section 3. Updating the dynamic domain	11-18
3.1. Modify the source Excel spreadsheet and update the dynamic domain	11-18
3.2. Synchronize the domain	11-18
Section 4. Updating the XOM	11-23
4.1. Identifying the inconsistency problem	11-23
4.2. Correcting the inconsistency problem	11-23
Section 5. Publishing the BOM and rule projects to Decision Center	11-24
5.1. Publishing the decision service	11-24
Section 6. Examining rules in Decision Center	11-25
6.1. Opening the published projects in Decision Center	11-25
Section 7. Modifying the dynamic domain in Decision Center	11-28
7.1. Viewing and modifying the values of the dynamic domain	11-28
7.2. Modifying the domain value in the rule	11-31
Section 8. Updating Rule Designer from Decision Center	11-33
8.1. Synchronizing the rule project	11-33
Exercise 12. Working with searches and queries	12-1
Section 1. Searching for rule artifacts	12-2
1.1. Setting up your environment for this exercise	12-2
1.2. Searching for specific criteria across the rules	12-2
1.3. Searching for dependencies between rules	12-4
Section 2. Querying rule projects	12-7
2.1. Searching for rules that may become applicable	12-7
2.2. Searching for rules that may lead to a state	12-9
2.3. Searching for rules that use a BOM member	12-10
2.4. Applying actions with queries	12-11
Section 3. Working with queries in Decision Center	12-13
Exercise 13. Debugging a ruleset	13-1
Section 1. Running the debug launch configuration	13-3
1.1. Setting up your environment for this exercise	13-3
1.2. Running the launch configuration	13-3
1.3. Applying automatic exception handling	13-5
1.4. Adding exception handling logging	13-6
Section 2. Setting breakpoints	13-8
2.1. Searching for the source of the problem	13-8
2.2. Preparing the debug session	13-9
2.3. Running the debugger on a ruleflow	13-10
Section 3. Debugging a rule	13-12
3.1. Setting a breakpoint in a rule	13-12
3.2. Debugging the action rule	13-14
3.3. Fixing the error in the rule	13-15
Section 4. Debugging a decision table	13-17
4.1. Finding the source of the rejection message	13-17
4.2. Setting breakpoints in a decision table	13-18
4.3. Debugging the decision table	13-19
Section 5. Debugging the ruleflow	13-20
5.1. Debugging the ruleflow	13-20
5.2. Modifying the ruleflow	13-21
Exercise 14. Enabling rule validation	14-1

Section 1. Validating the rule project	14-2
1.1. Import your workspace in Rule Designer	14-2
1.2. Validating the rule project	14-2
1.3. Choosing a constructor	14-4
1.4. Editing column headings for the scenario file template	14-5
Section 2. Removing columns from the scenario file template	14-7
Section 3. Testing the rules in Rule Designer	14-8
3.1. Import the scenario file	14-8
3.2. Run the test	14-8
Section 4. Enabling testing from Business console	14-10
Section 5. Testing in Business console	14-11
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
1.3. Define the target server for the configuration	15-4
Section 2. Deploying a RuleApp from Rule Designer	15-7
2.1. Deploying the RuleApp	15-7
Section 3. Adding a ruleset property to a deployment configuration	15-11
Section 4. Managing XOM deployment	15-15
4.1. Include the XOM in the deployment configuration	15-15
4.2. Deploy the managed XOM	15-15
4.3. Modify and redeploy the XOM	15-16
4.4. Deploying a RuleApp associated with a managed XOM	15-18
Section 5. Exporting deployment server definitions	15-20
5.1. Exporting deployment configurations	15-20
5.2. Importing deployment configurations	15-20
Section 6. Managing deployment with the Decision Center API tool	15-23
6.1. Explore deployed decision services	15-23
6.2. Build a decision service	15-26
6.3. Verify your snapshot in the Business console	15-27
6.4. Deploy the decision service	15-28
6.5. Confirm that the updated Miniloan Service is deployed and accessible to client applications	15-29
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	16-2
Section 2. Exploring the deployed RuleApps	16-4
Section 3. Working with deployed resources	16-6
Section 4. Exploring the Decision Warehouse tab	16-7
Section 5. Exploring the Diagnostics and Server Info tabs	16-9
Section 6. Managing RuleApps in the console	16-10
6.1. Creating a RuleApp	16-10
6.2. Adding a ruleset to your RuleApp	16-10
Section 7. Managing RuleApps with REST API tools	16-12
7.1. Exploring REST API commands	16-12
7.2. Deploying with REST API	16-13
7.3. Adding references to the XOM	16-15
7.4. Deleting the RuleApp	16-16
Exercise 17. Executing rules using the REST service	17-1
Section 1. Retrieving the HTDS description and viewing the WSDL	17-2
1.1. View the deployed ruleset	17-2
1.2. View the WSDL file	17-3

Section 2. Retrieving and testing the WADL file	17-4
2.1. Retrieve the WADL file	17-4
2.2. Test the WADL file	17-4
Section 3. Retrieving and testing the OpenAPI description	17-7
3.1. Retrieve retrieve the OpenAPI description of your HTDS	17-7
3.2. Test execution	17-8
Exercise 18. Auditing ruleset execution through Decision Warehouse	18-1
Section 1. Enabling ruleset monitoring	18-3
1.1. Enabling ruleset execution monitoring	18-3
1.2. Generate execution traces	18-6
Section 2. Retrieving decision traces in the Rule Execution Server console	18-8
2.1. View decision traces	18-8
2.2. Adjust monitoring options	18-9
Section 3. Optimizing Decision Warehouse	18-11
3.1. Filtering the monitoring options	18-11
3.2. Using filters on trace data	18-12
3.3. Removing BOM serialization	18-12
Section 4. Deleting trace information from the database	18-14
4.1. Clearing traces	18-14

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®
IBM Cloud™
SPSS®

Express®
ILOG®
Tivoli®

IBM API Connect®
Redbooks®
WebSphere®

Intel, Intel Xeon and Xeon are trademarks or registered trademarks 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 is a registered trademark or trademark of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

Social® is a trademark or registered trademark of TWC Product and Technology, LLC, an IBM Company.

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 Business console
- 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
- 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, "Modeling decisions"](#)
- [Exercise 4, "Working with the BOM"](#)
- [Exercise 5, "Refactoring"](#)
- [Exercise 6, "Working with ruleflows"](#)
- [Exercise 7, "Exploring action rules"](#)
- [Exercise 8, "Authoring action rules"](#)
- [Exercise 9, "Authoring decision tables"](#)
- [Exercise 10, "Working with static domains"](#)
- [Exercise 11, "Working with dynamic domains"](#)
- [Exercise 12, "Working with searches and queries"](#)
- [Exercise 13, "Executing rules locally"](#)

- [Exercise 13, "Debugging a ruleset"](#)
- [Exercise 14, "Enabling rule validation"](#)
- [Exercise 15, "Managing deployment"](#)
- [Exercise 16, "Exploring the Rule Execution Server console"](#)
- [Exercise 18, "Auditing ruleset execution through Decision Warehouse"](#)
- [Exercise 17, "Executing rules using the REST service"](#)

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 Decision Center tools.

Enabling rule authoring and customizing vocabulary

- [Exercise 2, "Setting up decision services"](#)
- [Exercise 3, "Modeling decisions"](#)
- [Exercise 4, "Working with the BOM"](#)
- [Exercise 5, "Refactoring"](#)
- [Exercise 10, "Working with static domains"](#)
- [Exercise 11, "Working with dynamic domains"](#)

Rule authoring

- [Exercise 6, "Working with ruleflows"](#)
- [Exercise 7, "Exploring action rules"](#)
- [Exercise 8, "Authoring action rules"](#)
- [Exercise 9, "Authoring decision tables"](#)

Reviewing and testing rulesets

- [Exercise 12, "Working with searches and queries"](#)
- [Exercise 13, "Debugging a ruleset"](#)
- [Exercise 14, "Enabling rule validation"](#)

Deployment and ruleset execution

- [Exercise 15, "Managing deployment"](#)
- [Exercise 16, "Exploring the Rule Execution Server console"](#)
- [Exercise 17, "Executing rules using the REST service"](#)
- [Exercise 18, "Auditing ruleset execution through Decision Warehouse"](#)

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. If necessary, you can skip exercises or do

them out of sequence. However, you are encouraged to complete them in the order in which they are presented.

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	passw0rd
Windows 2008 R2	administrator	passw0rd
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) describe what actions to do.
- The lettered substeps (a, b, c) provide detailed guidance on how to complete the action.

As shown in this example, the numbered step ("3") tells you to change the value in a rule. Substeps "a" and "b" provide details on how to edit.

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

a. Click **Edit this rule** (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**

If you already know how to edit rules and change values, you can skip the details in substeps 3a and 3b.

Text highlighting in exercises

Different text styles indicate various elements in the exercises.

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

- Menu items
- Field names
- Icons

Words that are highlighted with a `code 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:\IBM\ODM8105

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.

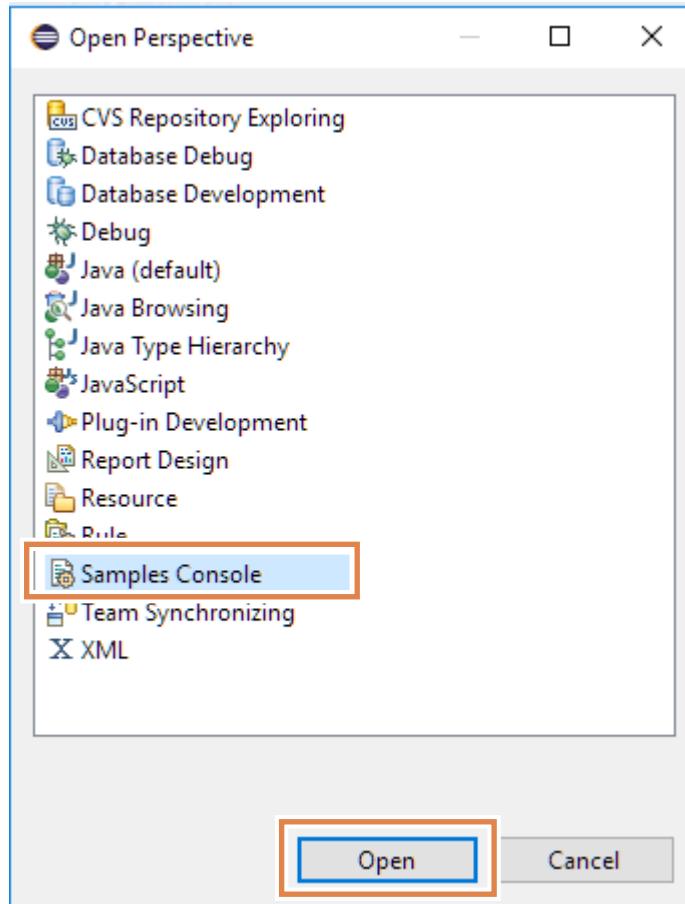
To open Rule Designer, you use the **Rule Designer 8.10.5** shortcut on the Taskbar or on the Windows **Start** menu.

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 Console perspective.



You expand the **Rule Designer > Training** node to see a list of all exercises for this course, and click **Import projects** to import the projects for the exercise.

The screenshot shows the 'Samples and Tutorials' view for Operational Decision Manager. The title bar has tabs for 'Samples and Tutorials' (selected) and 'Samples Commands'. Below the title bar, there's a breadcrumb trail: 'Samples and Tutorials > Rule Designer > Training'. The 'Rule Designer' section is expanded, showing the 'Samples' and 'Training' nodes. The 'Training' node has two children: 'Ex 01: Operational Decision Manager in action' and 'Ex 02: Setting up decision services'. Under 'Ex 01: Operational Decision Manager in action', there is a folder named '01-start' containing a link labeled 'Import projects', which is highlighted with a blue rectangular border. There is another 'Import projects' link under '02-answer'.

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 in the upper right.



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: 9090

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:9090/decisioncenter`: to access the Decision Center Business console
- `http://localhost:9090/teamserver`: to access the Decision Center Enterprise console
- `http://localhost:9090/res`: to access the Rule Execution Server console



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. From the Windows **Start** menu, click the down arrow to get to the **Apps** page, and in the **IBM Operational Decision Manager V8.10.5** section, click **Start Knowledge Center (local)**.

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

www.ibm.com/support/knowledgecenter/en/SSQP76_8.10.x/welcome/kc_welcome_odmV.html



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.

Checking for course corrections



Important

To check whether updates exist for the course material, see <http://ibm.biz/CourseInfo>.



Exercise 1. Operational Decision Manager in action

Estimated time

02:00

Overview

In this exercise, you see how the Operational Decision Manager modules work together to provide comprehensive decision management across the business and development environments.

Objectives

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 tasks that apply to your role in your organization

Introduction

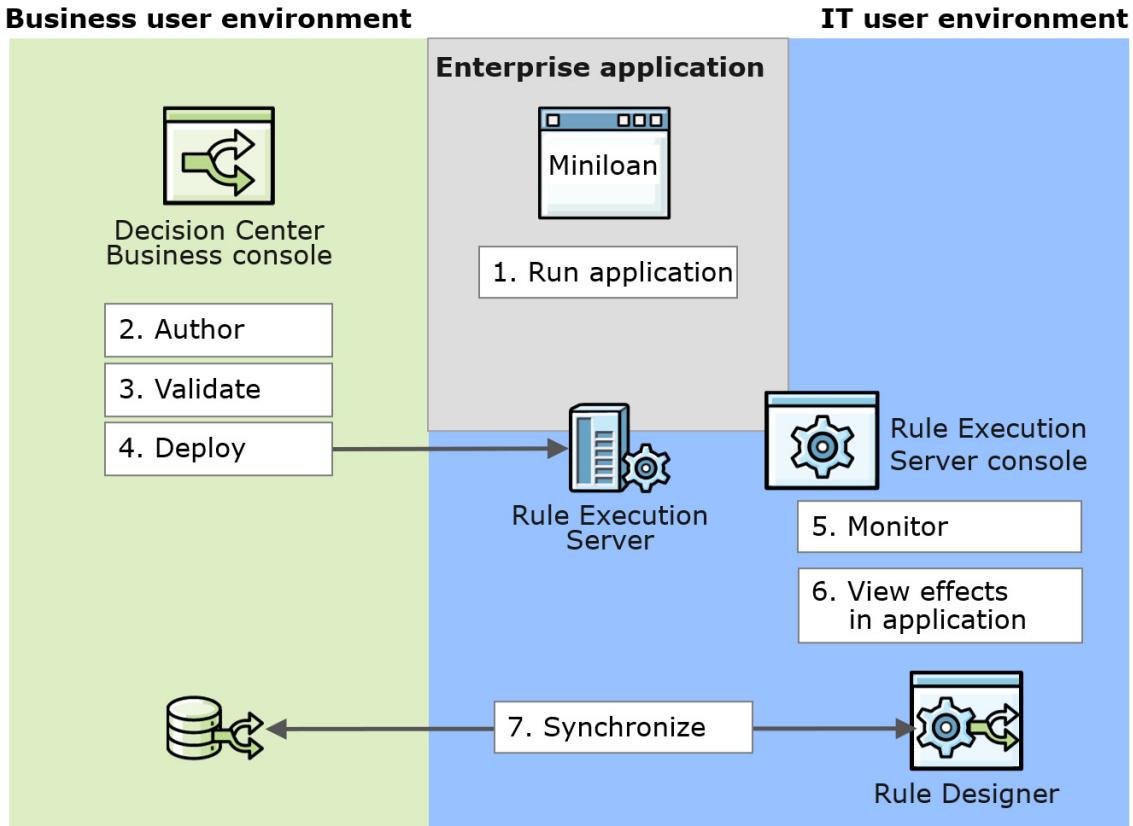
The exercise is based on a fictional financial institution, Miniloan, which provides online quotations for loan requests.

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

- [Section 1, "Running the Miniloan web application"](#)
- [Section 2, "Managing business rules in Decision Center"](#)
- [Section 3, "Validating rule changes"](#)
- [Section 4, "Deploying updated rules from Decision Center to Rule Execution Server"](#)
- [Section 5, "Monitoring the deployed rules in Rule Execution Server"](#)
- [Section 6, "Viewing the effects in the Miniloan application"](#)
- [Section 7, "Exploring the development environments in Rule Designer"](#)
- [Section 8, "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. 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 requirements.

User accounts

Type	User ID	Password
Access to VMware image	administrator	passw0rd
Decision Center administrator	rtsAdmin	rtsAdmin
Business user	rtsUser1	rtsUser1
Rule Execution Server administrator	resAdmin	resAdmin



Important

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

C:\IBM\ODM8105\studio\training

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 uses some features that are not supported in IBM ODM on Cloud. For ODM on Cloud, follow the instructions that are provided in the IBM ODM on Cloud documentation:

www.ibm.com/support/knowledgecenter/SS7J8H/welcome/kc_welcome_cloud.html

The documentation points you to software downloads and projects that are available on GitHub:

<https://github.com/ODMDev/odm-on-cloud-getting-started>

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.

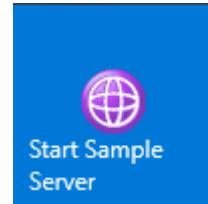
For IBM ODM on Cloud users

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 `Miniloan Service` decision service and install the `miniloan-webapp` web application.

— 1. Start the sample server.

— a. Click the **Start** menu, and click the **Start Sample Server** shortcut.



— 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 Sample Server
ath resource [ilog/rules/teamserver/client/client-applicationContext.xml]
[set-samples-settings] Oct 27, 2020 6:35:03 PM com.ibm.rules.decisioncenter.remoting.internal.RemoteSessionHttpExecutorBuilder log
[set-samples-settings] INFO: {"url":"http://localhost:9090/decisioncenter", "datasource":"jdbc/ilogDataSource", "username":"rtsAdmin", "message": "Sign out successfully", "date": "October 27, 2020 6:35:03 PM"}

delete.new.installation.files:
[samples.echo] Oct 27, 2020 6:35:05 PM 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: 4 minutes 3 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:9090/miniloan-server>

The Miniloan application opens.

The screenshot shows the 'Sample' MiniLoan application. It has three main sections: 'Borrower' (left), 'Loan' (right), and 'Execution' (bottom). The 'Borrower' section contains fields for Name (Joe), Yearly Income (80000), and Credit Score (600). The 'Loan' section contains fields for Amount (500000), Duration (240), and Yearly Interest ... (0.05). Below these sections is an 'Execution' section. At the bottom center is a blue button labeled 'Validate Loan' with a checkmark icon.

Borrower		Loan	
Name:	Joe	Amount:	500000
Yearly Income:	80000	Duration:	240
Credit Score:	600	Yearly Interest ...	0.05

Execution

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** and scroll down to see view the response.

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

Validate Loan

Rejected

```
{  
    "decision": "Your 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 use this application again 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.

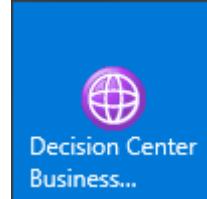
Miniloan decided to review its eligibility policies about debt-to-income ratios. Miniloan uses Decision Center to store and manage business rules, so next, you open Decision Center to see which rule to change to solve the loan rejection issue.

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.
 - a. From the **Start** menu, click the **Decision Center Business console** shortcut.



Information

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

<http://localhost:9090/decisioncenter>

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

- ___ b. If 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**.

- ___ c. 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.

IBM
Decision Center | Business Console

Username:

Password:

Keep me logged in

Log in

Licensed Materials - Property of IBM. © Copyright IBM Corporation 2000, 2017

- ___ d. Click **Log in**.

For IBM ODM on Cloud users

You start the Business console by logging in to the ODM on Cloud User Portal, and 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.



Note

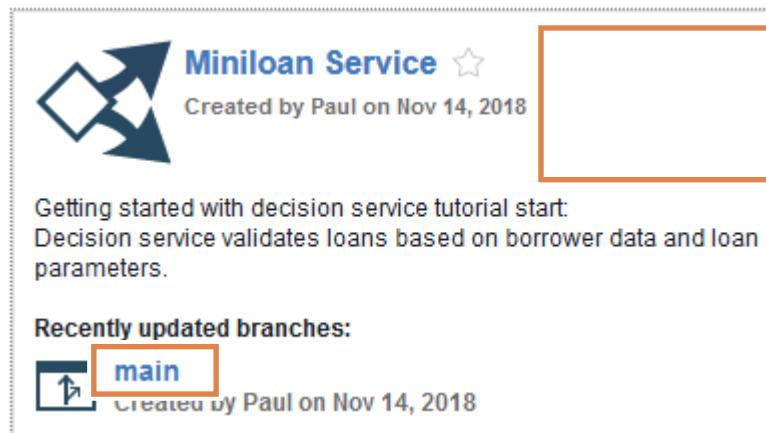
If you see a message to store your password, you can click **Not for this site**.

- ___ 2. On the Decision Center menu, click **Library**.



On the Decision Services page, you see the list of decision services that you can work on in Decision Center.

- ___ 3. Click the **Miniloan Service** box (in the empty white space) to expand the box, and then click **main**.



The **Decision Artifacts** tab opens, and in the navigator pane, you see two rule folders: **eligibility** and **validation**. These folders contain the rule artifacts for Miniloan Service.



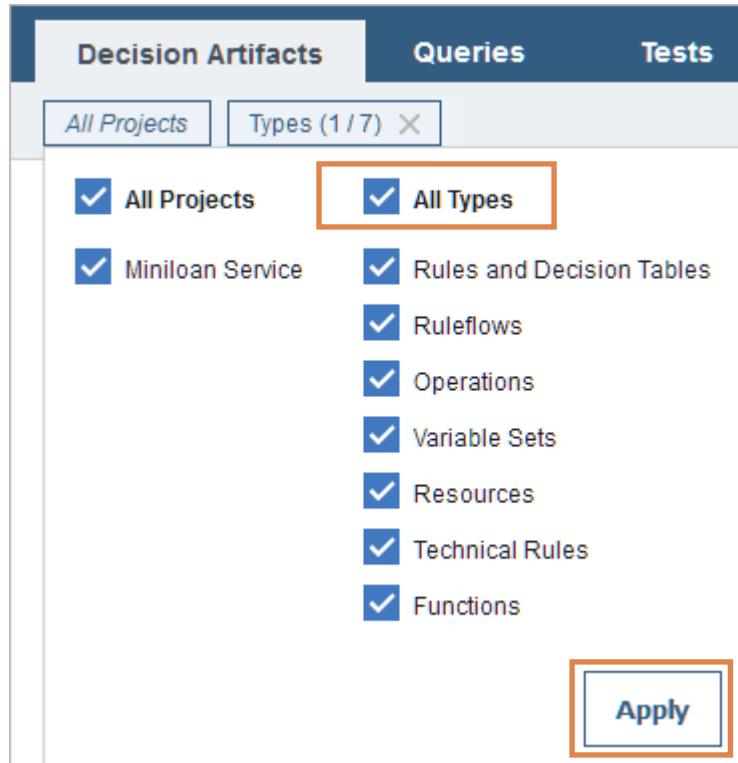
Information

By default, the Business console shows only rule artifacts, such as rules and decision tables. However, you can select various decision artifact types for display in the navigator pane.

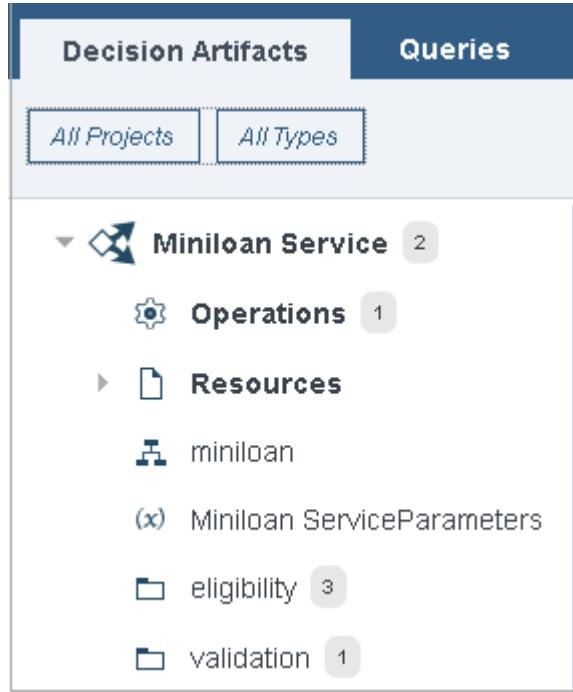
- 4. View all of the decision artifacts that are associated with Miniloan Service.
— a. On the **Decision Artifacts** tab, click **Types**.



- b. In the Types menu, select **All Types**, and then click **Apply**.



- ___ c. The navigator pane now shows all decision artifact types of Miniloan Service.



The decision artifacts in Miniloan Service include:

- **Operations** (artifact that defines input and output parameters for decision services)
 - **Resources** (artifact that stores decision service resources, such as the execution object model, or XOM)
 - **miniloan** (ruleflow)
 - **Miniloan ServiceParameters** (artifact that defines the input and output parameters for a decision service)
 - **eligibility** (folder that contains rules)
 - **validation** (folder that contains rules)
- ___ 5. View the **minimum income** rule in the **eligibility** folder.
- ___ a. Click the **eligibility** folder to see the rules that it contains.
 - ___ b. Click **minimum income** to review the minimum income policy.

The screenshot shows the 'minimum income' rule editor. At the top, there are navigation icons and a toolbar with icons for save, edit, and search. The main area displays the rule logic:

```

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.

- ___ 6. Edit the rule and change the debt-to-income ratio from 0.3 to 0.5.

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



The rule opens in the Rule View.

- ___ 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**

- ___ 7. 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

- 8. When you are finished with your changes, save your work and add a comment to provide version information for your changes.

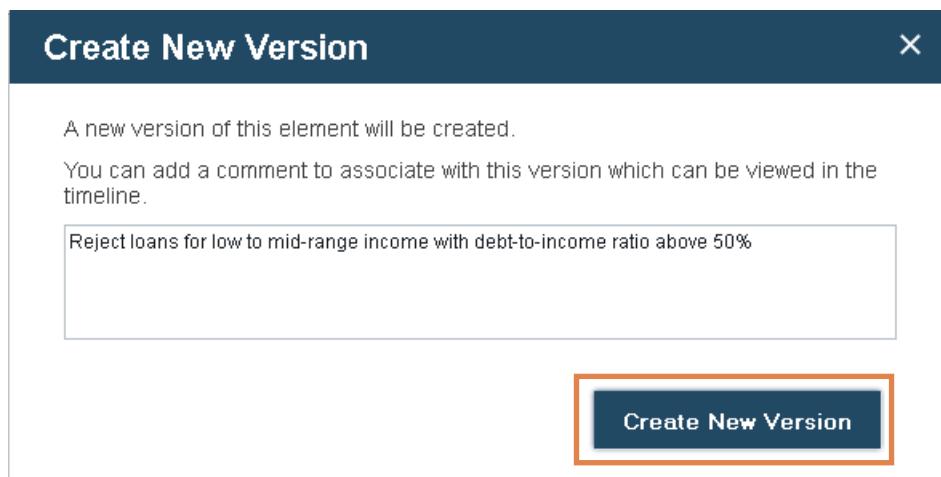
- a. Click **Save**.



- 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**.



You can review the updated rule.

```

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. Click the **Open Rule View** icon.

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**' ;

- b. On the toolbar in the Rule View, 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**.

Compare minimum income Versions

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

Current Version (1.1) Created by rtsUser1 on Jun 18, 2018	<input checked="" type="checkbox"/>
Reject loans for low to mid-range income with debt-to-income ratio above 50%	
Version 1.0 Created by Paul on Mar 1, 2018	<input checked="" type="checkbox"/>

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 comparison between Version 1.0 and Current Version (1.1) of the 'minimum income' rule. The summary indicates that '0.3 was changed to 0.5' and 'and the yearly income of 'the borrower' is less than 500000' was added. The additions are highlighted in purple.

You might need to maximize your browser to see the red triangle that indicates modified values. Additions to the rule are shown in purple, underlined text.

version 1.1 (current)
Created by rtsUser1 on Jun 18, 2018

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.

Decision Center [HOME](#) [LIBRARY](#) WORK

Miniloan Service : **main**

minimum income ⓘ

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

- 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.

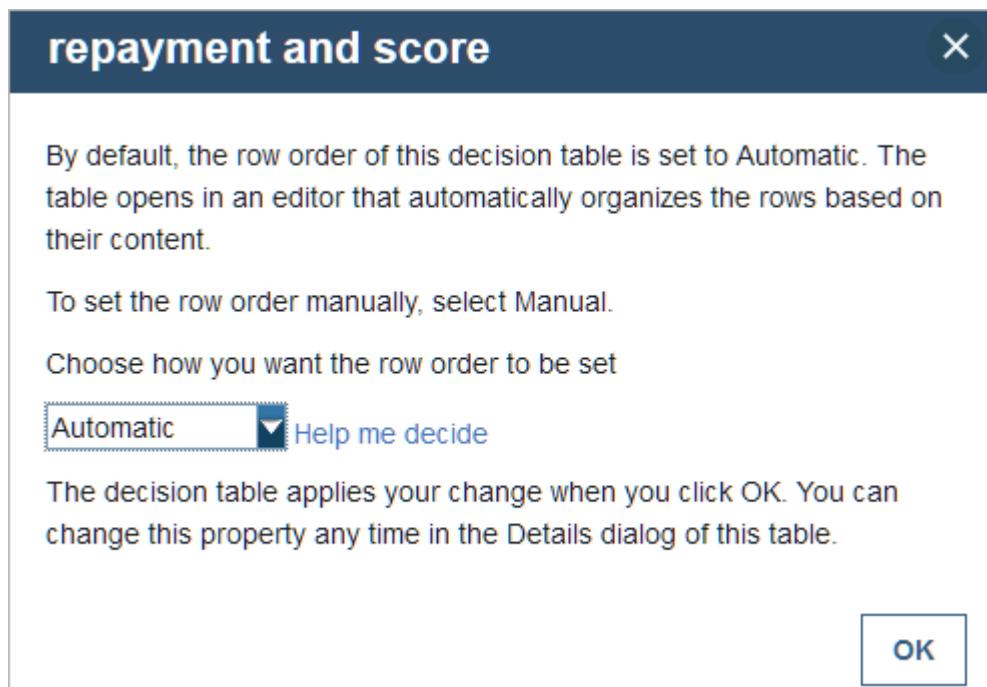
- 1. Edit the repayment and score decision table to modify the credit score values in row 5 to become: [0; 500[
- a. Click the **eligibility** folder, hover your mouse on **repayment and score** and click the **Edit this decision table** icon to open the repayment and score decision table.

The screenshot shows the IBM Operational Decision Manager (ODM) interface. On the left, there is a navigation tree for the 'Miniloan Service' project, which includes 'Operations' (1 item), 'Resources' (with 'xom-libraries' (1 item)), 'miniloan' (1 item), 'Miniloan ServiceParameters' (with 'eligibility' (3 items) highlighted by an orange box), and 'validation' (1 item). On the right, the 'eligibility' decision table is displayed. The table has columns for 'Name', 'minimum credit score', 'minimum income', and 'Last Ch'. It contains three rows:

Name	minimum credit score	Last Ch
Paul	500	2021-09-01
rtsUser1	500	2021-09-01
Paul	500	2021-09-01

A blue vertical bar highlights the 'repayment and score' row. A red box surrounds the 'Edit' icon (pencil) in the row's toolbar.

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



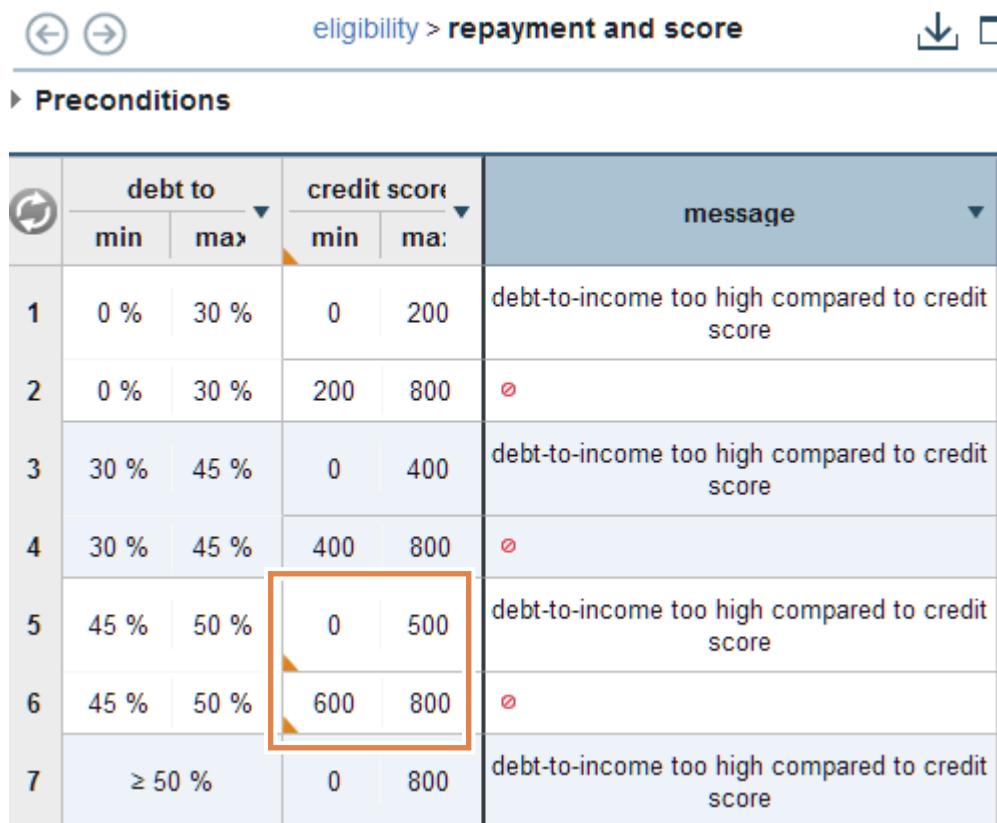
- ___ c. In row 5, double-click the cell in the credit score column with the value: 600
 ___ d. Change 600 to 500 and press Enter.

	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	≥ 50 %		0	800

- ___ 2. Save your edits.
- ___ a. Click **Save**.
- ___ b. 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
- ___ c. Click **Create New Version**.

- ___ d. If the rule closed, click it to reopen and view it.

After gap error is detected in the table. The problem cells are highlighted with a gold triangle in the lower-left corner.



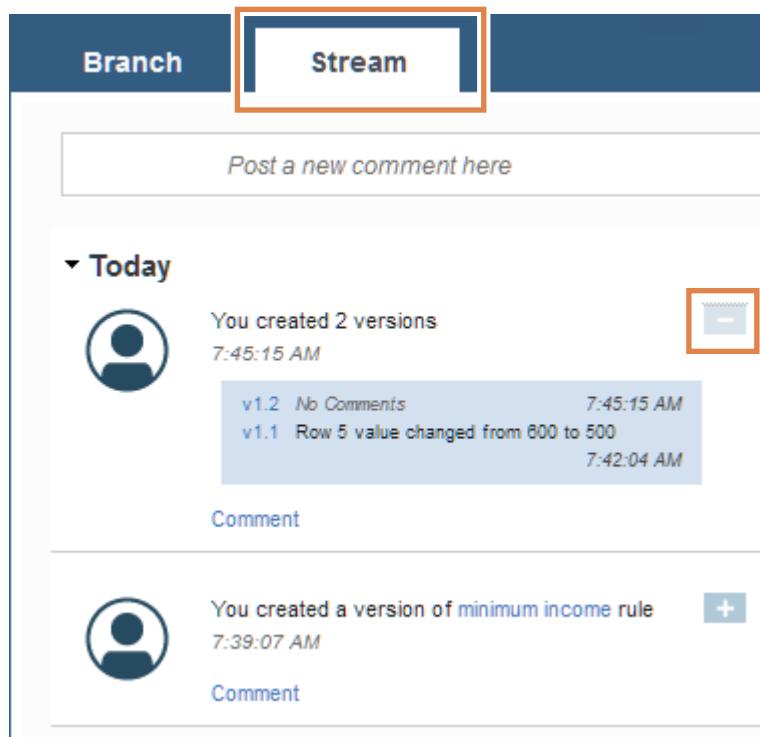
The screenshot shows a table titled "eligibility > repayment and score". The table has columns for "debt to" and "credit score", each with "min" and "max" headers. A "message" column contains descriptive text. Row 6 is highlighted with a red border and a yellow triangle in the "credit score min" cell, indicating a gap error. The table rows are numbered 1 through 7.

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

- ___ 3. Edit the table to resolve the gap error.
- ___ a. Click the **Edit** icon to reopen the editor.
 - ___ b. In row 6, select the cell in the credit score column with the value: 600
 - ___ c. Change the value from 600 to 500, and press Enter.
 - ___ d. Click **Save**.
 - ___ e. In the Create New Version window, click **Create New Version**.

The table no longer shows any errors.

- 4. Click the **Stream** tab on the right pane and click the **plus** icon  to see the change details. You can use Stream to notify others or be notified of an updates.



Post a new comment here

▼ Today

You created 2 versions
7:45:15 AM

v1.2 No Comments 7:45:15 AM
v1.1 Row 5 value changed from 600 to 500 7:42:04 AM

Comment

You created a version of minimum income rule +
7:39:07 AM

Comment

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.

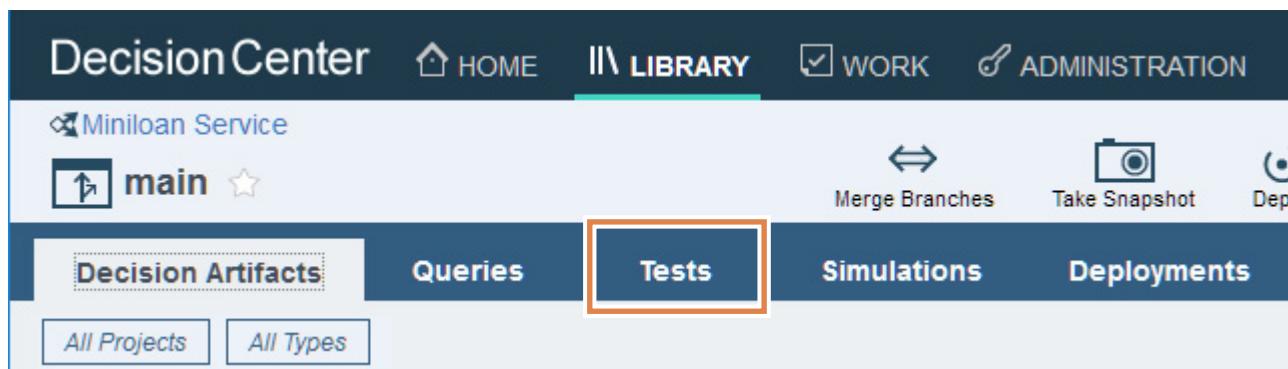
To run a scenario in Decision Center, you must create a test suite. For this exercise, the scenario file for the test suite is already created for you. The `miniloan.xlsx` scenario file is in this directory:

```
<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 **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.

- __ 2. Hover your mouse over **Miniloan test suite**, and click the **Run this test suite** icon.

The screenshot shows the 'Tests' tab selected in the top navigation bar. Below it, the 'Test Suites' section displays a list of suites. The first suite in the list, 'Miniloan t...', has its run button (a blue circle with a white play icon) highlighted with an orange border, indicating it is the target for the next step.

- __ 3. In the Run Test Suite window, click **OK** to close the window.
__ 4. Wait for the test to complete and then click the report link to open the report.

Name	Operation	Status
Report - 2018-06-18_07-29-...	Miniloan ServiceOperation	!

Because you changed the rules, you should not expect a 100% success rate. The results show that 1 test failed.



- __ 5. In the upper right, click **Close** to close the report.

Information

Later in this course, you learn how to define test suites and interpret the results.

- __ 6. 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**.



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.

For IBM ODM on Cloud users

For this section, make sure that you run the steps on a Development Rule Execution Server. For information about the ODM on Cloud roles and environment, see:

www.ibm.com/support/knowledgecenter/SS7J8H/com.ibm.odm.cloud.admin/topics/con_work_env.html



Information

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

4.1. Preparing the decision operation for deployment

- 1. Sign in to the Business console as a user with administrative privileges:
 - **User name:** rtsAdmin
 - **Password:** rtsAdmin
- 2. Click **Library** and reopen the **main** branch of the Miniloan Service decision service.
- 3. On the **Decision Artifacts** tab, enable all types of decision artifacts for display in the navigation pane by clicking the **Types** tab, selecting **All Types**, and clicking **Apply**.

4. In the navigation pane, click **Operations**, and click **Miniloan ServiceOperation**.

The screenshot shows the 'Decision Artifacts' tab selected in the top navigation bar. The left sidebar displays a tree structure under the 'Miniloan Service' project: 'Operations' (1 item), 'Resources', 'miniloan', '(x) Miniloan ServiceParameters', and 'eligibility' (3 items). The 'Operations' node is highlighted with a grey background. On the right, the 'Operations' panel is open, showing a list of operations. At the top of this panel are navigation icons: back, forward, add, delete, and copy. Below these are buttons for 'Total: 1 Selected: 0' and page navigation (1). A search bar labeled 'Name' is present. The list contains one item: 'Miniloan ServiceOperation', which is highlighted with an orange border.

5. In the upper right, click **Edit** (the pencil icon).



- ___ 6. Change the value in the **Ruleset name** field to: my_operation

The screenshot shows the 'Miniloan Service > main' interface. At the top, there's a gear icon and the title 'Miniloan ServiceOperation'. Below this, there are two input fields: 'Name' (containing 'Miniloan ServiceOperation') and 'Description'. Under the 'Overview' section, there are several dropdowns and input fields: 'Source Project' (set to 'Miniloan Service'), 'Extracted Rules' (empty), 'Query' (set to 'No query'), 'Validator' (set to 'Default Validator'), 'Ruleset Name' (highlighted with an orange border and containing 'my_operation'), and 'Main Ruleflow' (set to 'miniloan').

- ___ 7. Save your changes.
- ___ a. Click **Save** to exit the editor.
 - ___ b. In the Create New Version window, click **Create New Version**.

4.2. Deploying the decision service

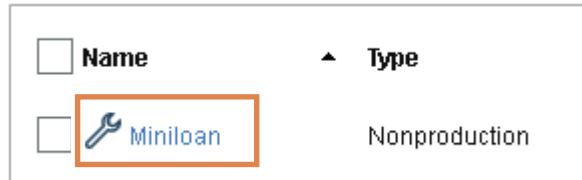
- ___ 1. Click the **Deployments** tab and make sure that you are on the Configurations page.



For IBM ODM on Cloud users

For this section, you use the Development configuration with the Development Environment server as the target.

- ___ 2. Click the **Miniloan** deployment configuration.



- ___ 3. In the upper-right area of the page, click the **Edit** icon.



- ___ 4. On the **General** tab, change the **RuleApp name** to: my_deployment

General	Operations	Targets	Ruleset Properties	Groups	Deployment
* Configuration name: Miniloan			* RuleApp name: my_deployment		
Configuration type: <input checked="" type="radio"/> Nonproduction <input type="radio"/> Production			* RuleApp base version number: 1.0		

- ___ 5. Click the **Targets** tab and make sure that **Decision Service Execution** is selected.

General	Operations	Targets	Ruleset Properties
Select one or more servers for the deployment. If you do not select a server, you can select it later.			
Target servers: <input checked="" type="checkbox"/> Servers <input checked="" type="checkbox"/> Decision Service Execution			

- ___ 6. Click **Save** and click **Create New Version**.

- ___ 7. Deploy the updated decision service.
 - ___ a. In the toolbar, click the **Deploy** icon.
- 
- ___ b. In the “Deploy: main” window, click **Deploy**.
 - ___ c. In the “Deployment status” window, click **OK**.
- ___ 8. After deployment is finished, click the **Report** link.
 - ___ 9. View the report details, and when you are finished, click **Close**.



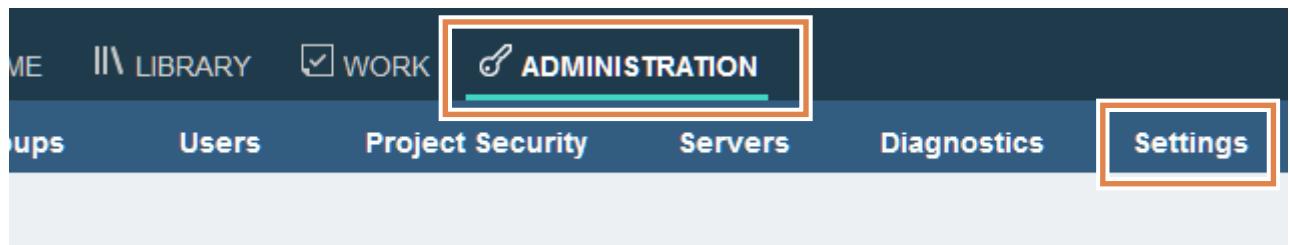
Information

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.

4.3. Customizing Decision Center settings

Before leaving the Business console, note that as an administrator (rtsAdmin) you can customize some environment settings.

- ___ 1. Click the **Administration** menu and click the **Settings** tab.



- ___ 2. In the navigation pane, click **Business Console**.
- ___ 3. In the **Customer Survey** section, enable **Do not show customer survey**.
This setting disables the survey popup.

- 4. In the **Session Timeout** section, enable **Save edited artifact before timeout** to avoid losing work.

The screenshot shows the 'Settings' tab selected in the top navigation bar. Under the 'Business Console' section, there is a toggle switch labeled 'Show number of artifacts in Decision Artifacts tree (may impact performances)' which is turned on. In the 'Session Timeout' section, there are three settings: 'Maximum time of session inactivity (minutes)' set to 30, 'Delay for timeout warning (minutes)' set to 10, and a toggle switch for 'Save open rule artifact before timeout' which is turned on. The 'Session Timeout' section and the 'Save open rule artifact before timeout' toggle switch are both highlighted with orange boxes.

Decision Center ▾ Decision Artifacts Tab

Business Console

Show number of artifacts in Decision Artifacts tree (may impact performances)

Custom Settings ▾ Decision Tables

Default row ordering for decision tables

▀ Customer Survey

Do not show customer survey

▀ Session Timeout

Maximum time of session inactivity (minutes) ⓘ

Delay for timeout warning (minutes) ⓘ

Save open rule artifact before timeout ⓘ

- 5. Sign out of the 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 clicking the Rule Execution Server shortcut on the Windows Start menu or by entering the following URL in a browser:

`http://localhost:9090/res`



Information

You can also open the Rule Execution Server by using the **Start** menu shortcut.

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 in the Development Environment, 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.

4. In the **Navigator** pane, expand **RuleApps** > click **my_deployment** to see the details in the RuleApp view

RuleApps View

Add RuleApp Deploy RuleApp Archive Update RuleApps

RuleApps

Total Number of RuleApps 1

Select All	Name	Version	Creation Date
<input type="checkbox"/>	my_deployment	1.0	Dec 14, 2020, 3:34:56 PM GMT

RuleApp 1 - 1 of 1

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.

Ruleset View

Test Ruleset View Statistics View Execution Units Upload Ruleset Archive

/my_deployment/1.0/my_operation/1.0

Name	my_operation
Version	1.0
Creation Date	Nov 9, 2020, 6:02:13 PM GMT-5
Display Name	Miniloan ServiceOperation
Description	
Rule engine	Decision Engine - 1.40.10
Status	✓ enabled
Debug	disabled

Permanent link

- 6. Scroll down and click the various options, such as Show Managed URIs and **Show Properties**.

The screenshot shows the 'Ruleset Parameters' interface. At the top, there's a table with columns 'Direction' and 'Name'. Two rows are listed: 'borrower' and 'loan'. Below this table, it says 'Ruleset Parameters 1 - 2 of 2'. There are three buttons at the bottom:

- Show Managed URIs (1)
- Show Properties (18)
- Show Monitoring Options (tracing currently enabled)

The first two buttons are highlighted with orange boxes.

These properties are just some of the details that are available in Rule Execution Server.



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:9090/miniloan-server>

- 2. Click **Execution** to expand it and select **Use rules**.

- 3. Click **Validate Loan**.

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

```
{
  "__DecisionID__": "810211d2-61fe-47c3-96bf-40f672a062980",
  "loan": {
    "amount": 500000,
    "duration": 240,
    "yearlyInterestRate": 0.05,
    "yearlyRenavment": 39597,
    "approved": true,
    "messages": []
  }
}
```

- 4. Close the browser.

Section 7. Exploring the development environments in Rule Designer

For IBM ODM on Cloud users

The steps in this section do not apply to ODM on Cloud.

For more information about installing and using Rule Designer with ODM on Cloud, see the documentation.

www.ibm.com/support/knowledgecenter/SS7J8H/welcome/kc_welcome_cloud.html

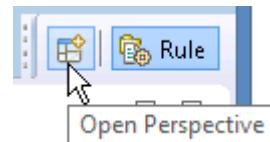
7.1. Opening Rule Designer

- ___ 1. On the taskbar, click the **Rule Designer** shortcut.

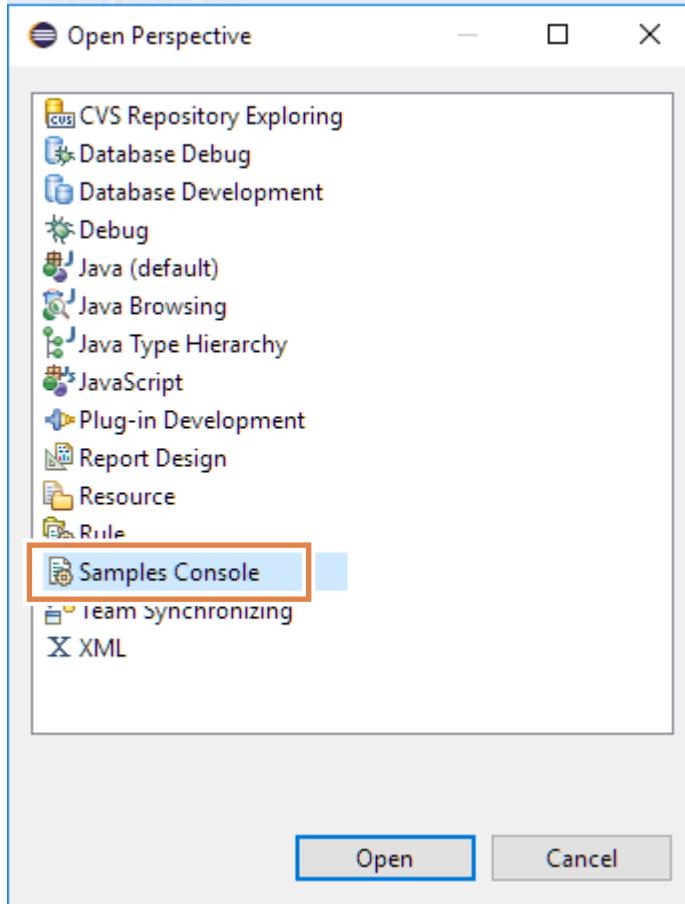


You can also click the **Rule Designer 8.10.5** shortcut on the **Start** menu.

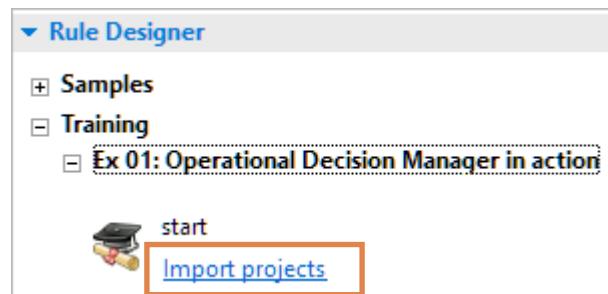
- ___ 2. When prompted for a workspace, use the following path and click **Launch**:
C:\labfiles\workspaces\intro
- ___ 3. Close the Welcome view.
By default, the Rule Designer perspective opens in Eclipse.
- ___ 4. Switch to the Samples Console and import the project for this exercise.
 - ___ a. In the upper-right area of the window, click the **Open Perspective** icon to switch from the Rule perspective.



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



- ___ c. In the Rule Designer section, expand **Training > Ex 01: Operational Decision Manager in action > start**, and click **Import projects**.



The Eclipse perspective automatically switches back to the Rule perspective.

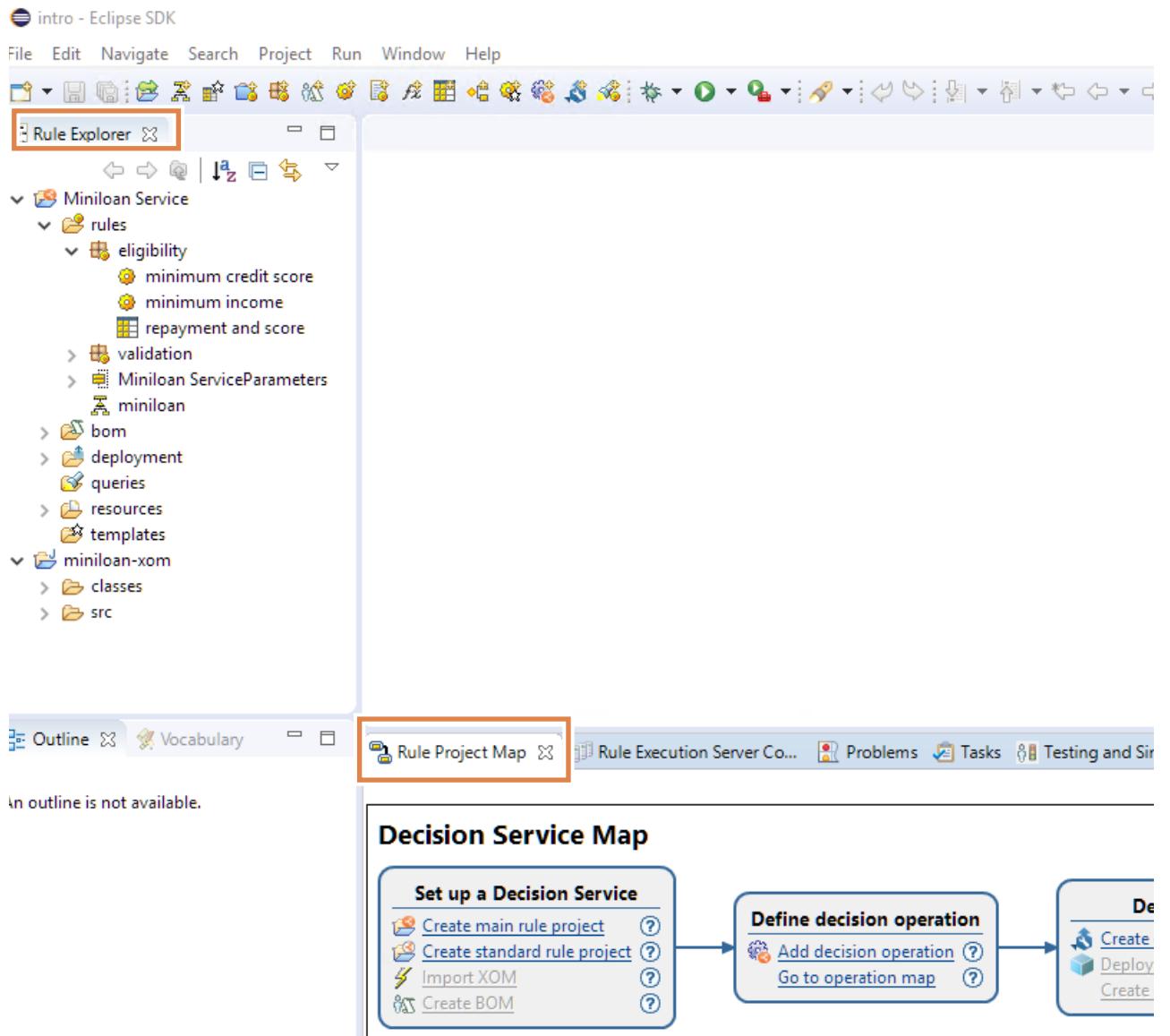
- ___ d. Close the Help view by clicking the X to give yourself more working area.



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 these views:
 - **Rule Explorer:** Expand and view the contents of the projects
 - **Rule Project Map:** Notice the various links that 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.
- 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.
 - In the **eligibility** folder, double-click **repayment and score** to open the decision table.
 - 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.
 - Close the decision table.

debt to income		
	min	max
1	0 %	30 %
2		

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.

- Expand **Borrower** and **Loan** to see their members in Rule Explorer.
- Double-click **Borrower** to open it in the BOM editor and view its properties and members. For example, here you see that the Borrower class has a name, a credit score and a yearly income.

Class Borrower (package: miniloan)

General Information

Name:	Borrower	
Namespace:	miniloan	Change...
Superclasses:	java.lang.Object	Change...
Interfaces:		Change...
<input type="checkbox"/> Deprecated		

Members

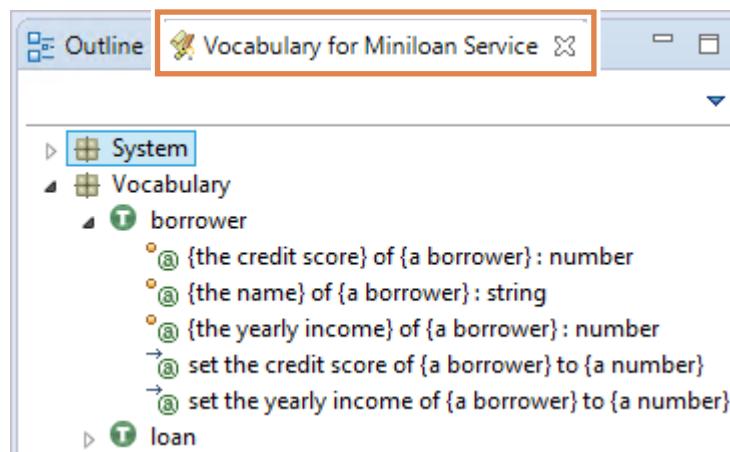
Specify the members of this class.

creditScore	New...
name	Delete
yearlyIncome	Edit
Borrower()	
Borrower(String,int,int)	

- ___ 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?

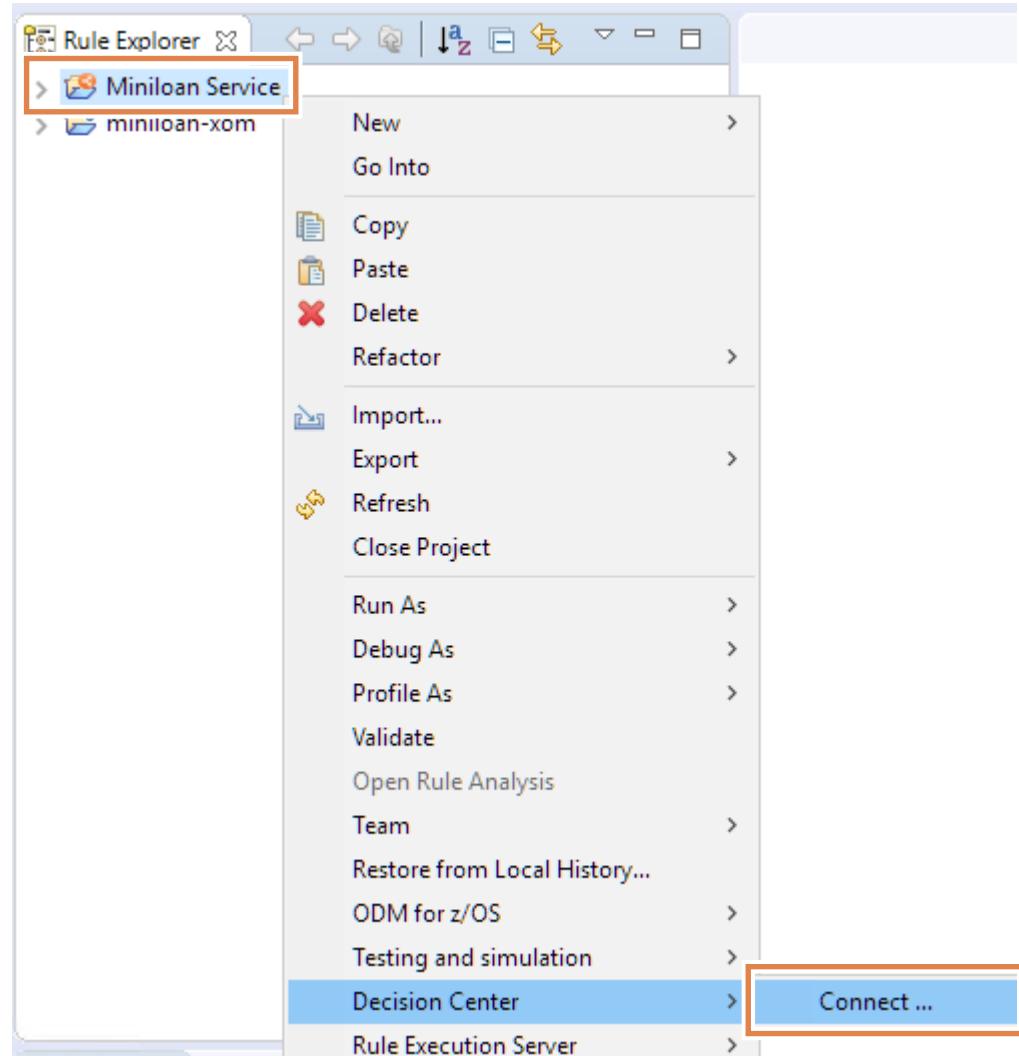
Remember this comparison as a first look at the connection between rules and vocabulary.

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

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

7.3. Synchronizing Rule Designer and Decision Center environments

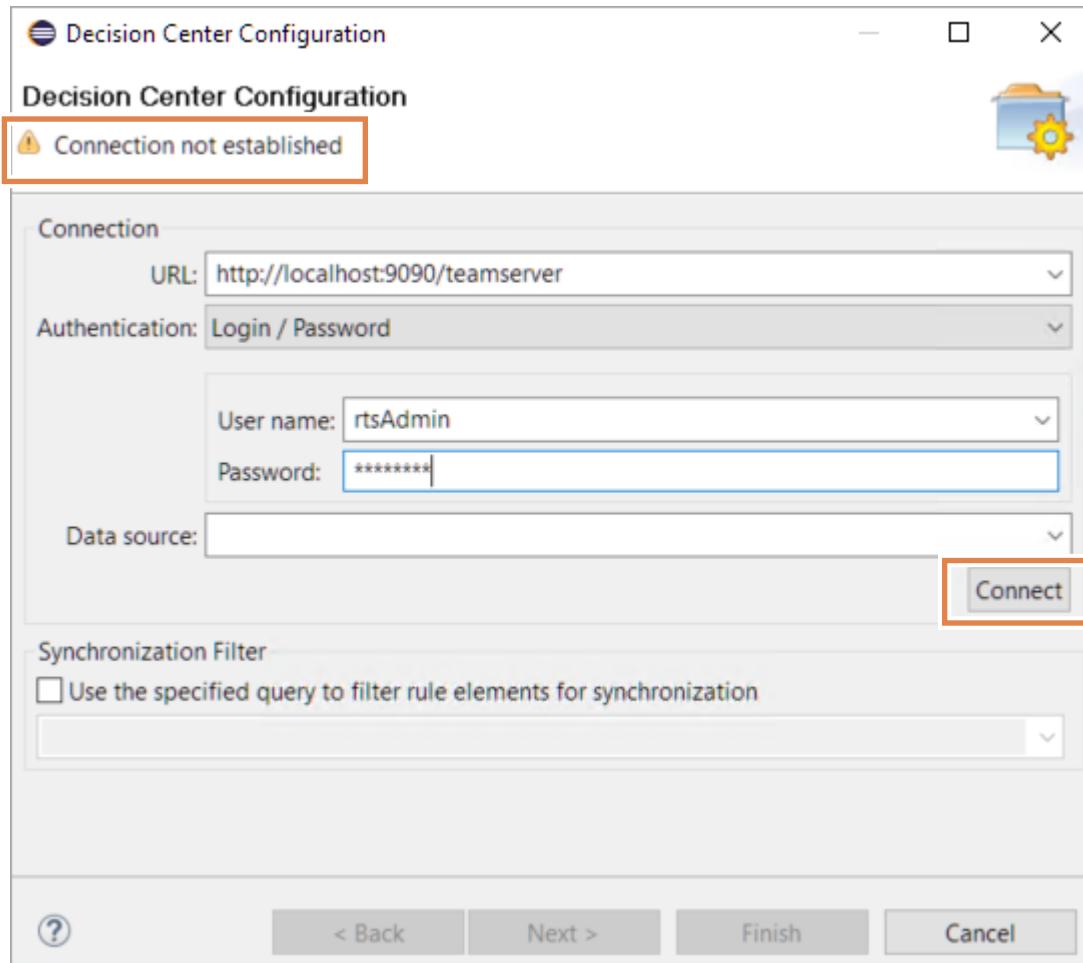
- 1. In Rule Explorer, right-click Miniloan Service, and click **Decision Center > Connect**.



- 2. Enter the following values in the Decision Center Configuration window fields.

- **URL:** `http://localhost:9090/teamserver`
- **User name:** `rtsAdmin`
- **Password:** `rtsAdmin`
- **Data source:** (Leave the field empty)

3. Click **Connect**.



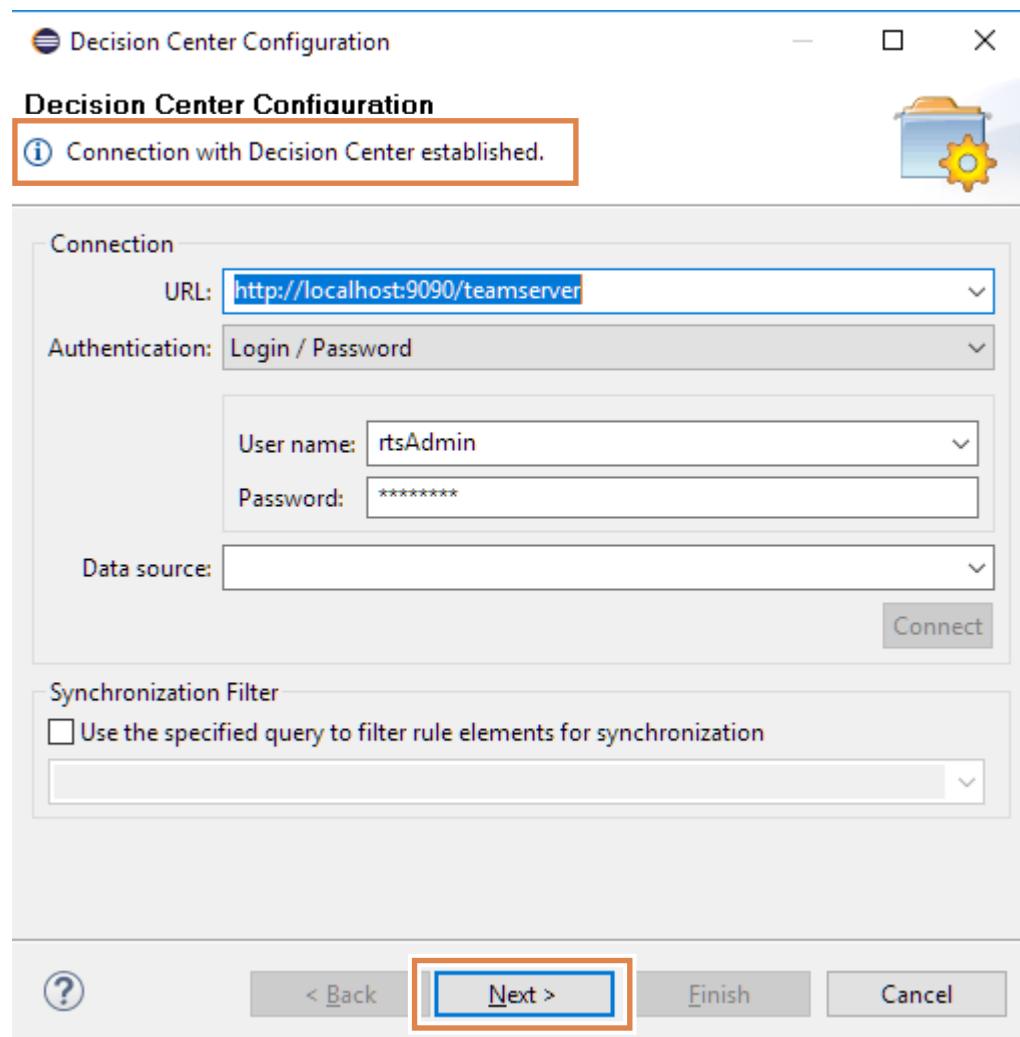
When the connection is established, the warning message disappears and the Project configuration area becomes active.



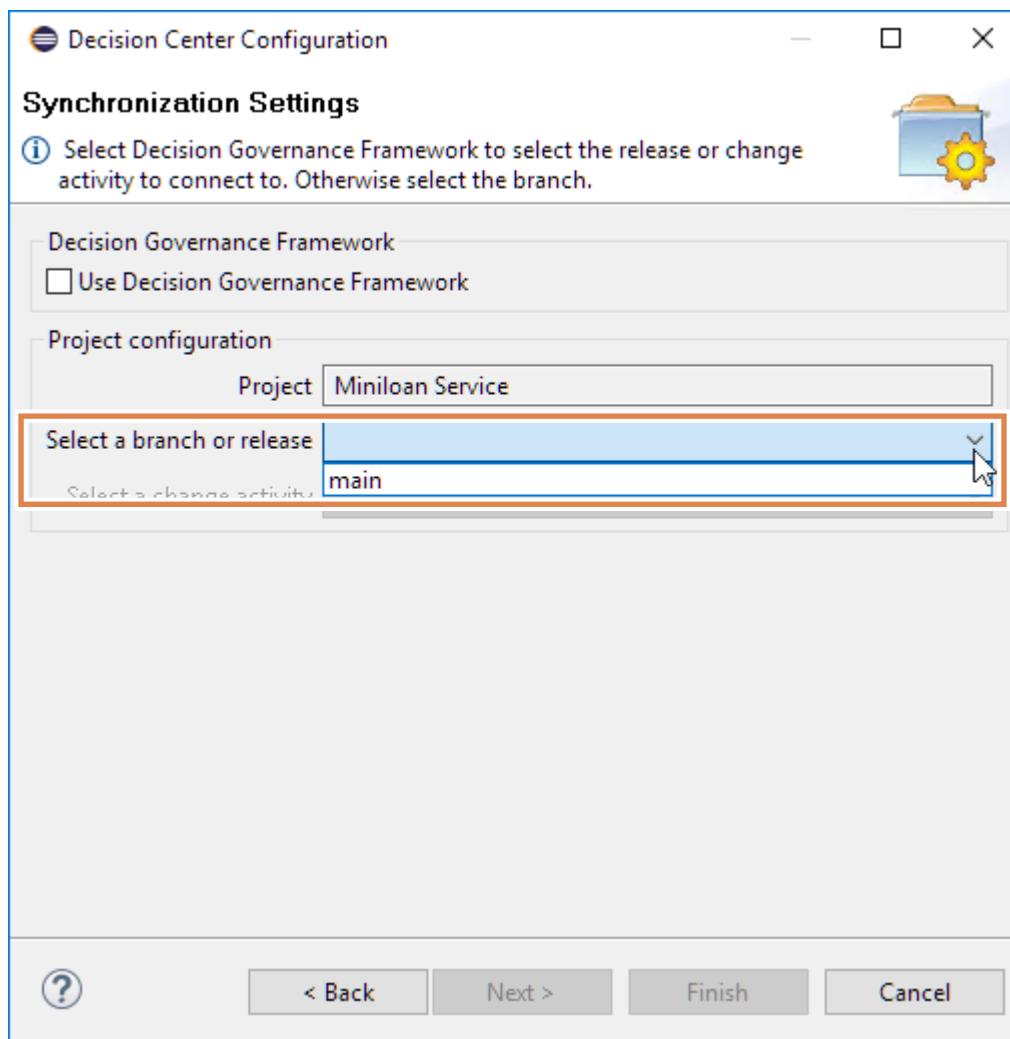
Troubleshooting

Sometimes the connection window disappears to the background of your desktop. Click the Rule Designer window to bring the connection wizard back into focus.

4. After the connection is established, click **Next**.



- 5. In the **Select a branch or release** field, choose **main**.

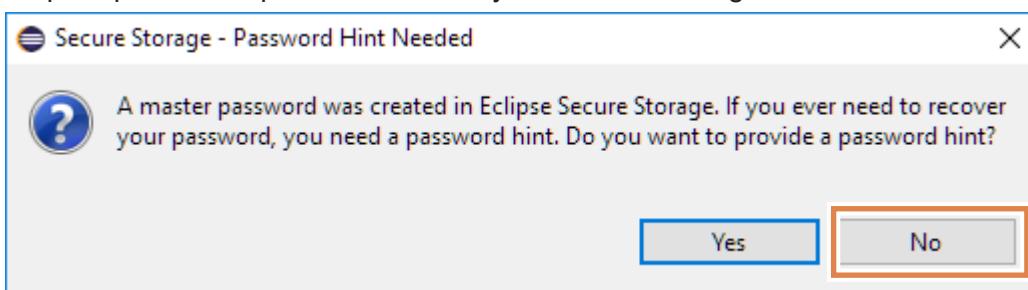


- 6. Click **Finish**.

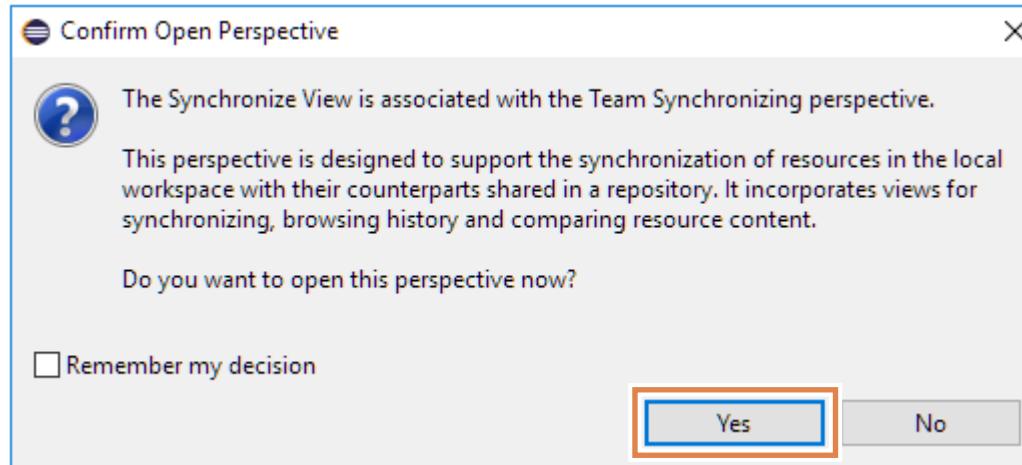


Troubleshooting

If you are prompted about password recovery for Secure Storage, click **No**.



- ___ 7. When the window prompts you to change to Team Synchronizing perspective, click **Yes**.

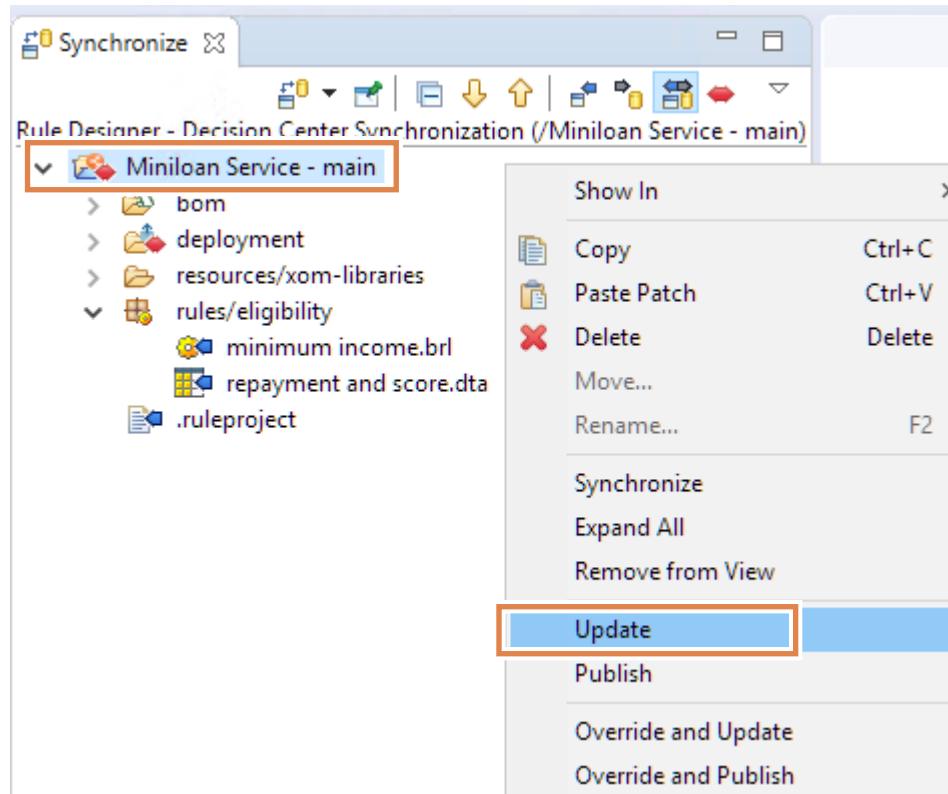


After synchronization is complete, the Synchronize view shows the `miniloan-rules` rule project, and indicates that changes were made to rules in that rule project.

- ___ 8. In the Synchronize view, expand `Miniloan Service - main` and the `rules/eligibility` folder to see which rules need to be updated.
- ___ a. Expand the `rules/eligibility` folder to see the rules that you updated in Decision Center.
- `minimum income.brl`
 - `repayment and score.dta`
- ___ b. Double-click `minimum income.brl` to open it in the **Text Compare** view and see the differences that now require synchronization.

Text Compare	
Local File	Remote File
<pre> 1 <?xml version="1.0" encoding="UTF-8"?> 2 <iolog.rules.studio.model.brl:ActionRule xm... 3 <name>minimum income</name> 4 <uuid>f5ad9c90-3892-433d-88ad-12de47e9d6... 5 <locale>en_US</locale> 6 <definition> 7 the yearly repayment of 'the loan' is mo... 8 then 9 10 reject 'the loan' ;]]></definition> 11 </iolog.rules.studio.model.brl:ActionRule> 12 </pre>	<pre> 1 <?xml version="1.0" encoding="UTF-8"?> 2 <iolog.rules.studio.model.brl:ActionRule xm... 3 <name>minimum income</name> 4 <uuid>f5ad9c90-3892-433d-88ad-12de47e9d6... 5 <locale>en_US</locale> 6 <definition> 7 the yearly repayment of 'the loan' i... 8 and the yearly income of 'the borrower' 9 10 add "Too big Debt-To-Income ratio" to... 11 reject 'the loan' ;]]></definition> 12 </iolog.rules.studio.model.brl:ActionRule... 13 </pre>

- ___ c. Right-click **Miniloan Service - main** and click **Update**.



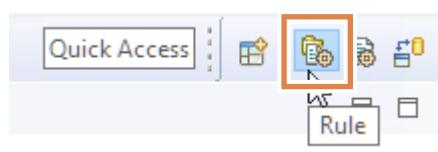
After the update is complete, the out-of-sync rules in the **rules/eligibility** folder are removed from the Synchronize view, which means that your local Rule Designer workspace is correctly updated with the edited rules from Decision Center.

-
- ___ 9. Return to the Rule perspective.
-



Hint

To return to the Rule perspective, click **Rule** at the upper-right corner of the perspective.



-
- ___ 10. If the minimum income rule is closed, reopen it by expanding the **Miniloan Service > rules > eligibility** folder in Rule Explorer, and double-clicking **minimum income**.

The condition statement should now match your edits in Decision Center and state:

```
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
```

- ___ 11. Close Rule Designer, and confirm closing when prompted to confirm.
-



Information

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

Section 8. Shutting down the modules

For IBM ODM on Cloud users

The steps in this section do not apply to ODM on Cloud.

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

- __ 1. Close any open browsers.
- __ 2. Close Rule Designer and click **Exit** when prompted to confirm your exit from Eclipse.
- __ 3. Stop the sample server.
 - __ a. From the Windows **Start** menu, click the **Stop Sample Server** shortcut.
 - __ 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

Estimated time

01:45

Overview

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

Objectives

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 variables and parameters. 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 support files, which are installed in the `<InstallDir>\studio\training` directory:

- Start project: Dev 02 – Decision services\start

Section 1. Creating a standard rule project

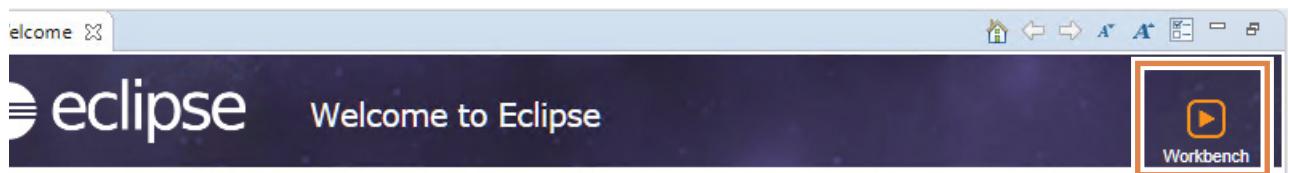
In this section of the exercise, you set up the environment for the exercise and create a standard rule project for the decision service.

1.1. Setting up your environment for this exercise

- 1. Open Rule Designer by clicking the **Rule Designer** shortcut on the taskbar.

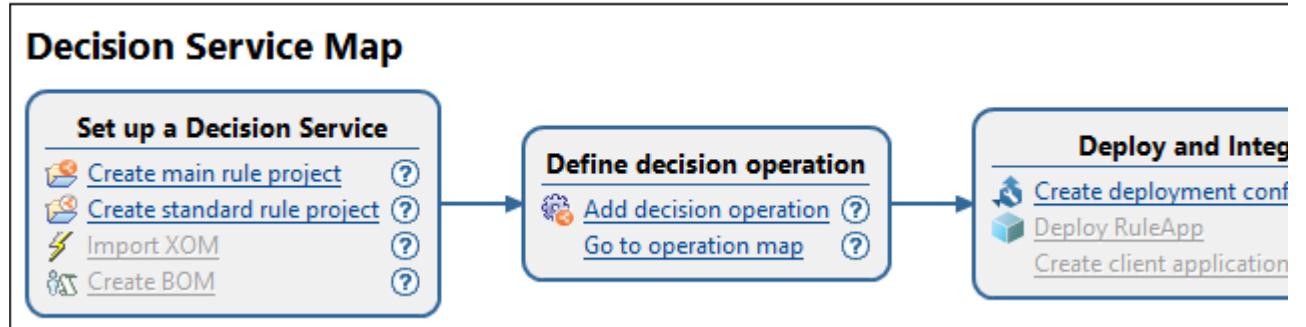


- 2. In the Workspace Launcher window, create a workspace.
 - a. In the **Workspace** field, type a workspace path and name:
C:\labfiles\workspaces\decision_service
 - b. Make sure that **Use this as the default and do not ask again** is *not* selected.
 - c. Click **Launch**.
- 3. Click **Workbench**.



The Rule Designer workspace opens and you see the Decision Service Map in the lower part of the workspace, in the Rule Project Map view. The steps for setting up a decision service are outlined in the Decision Service 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.



Hint

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

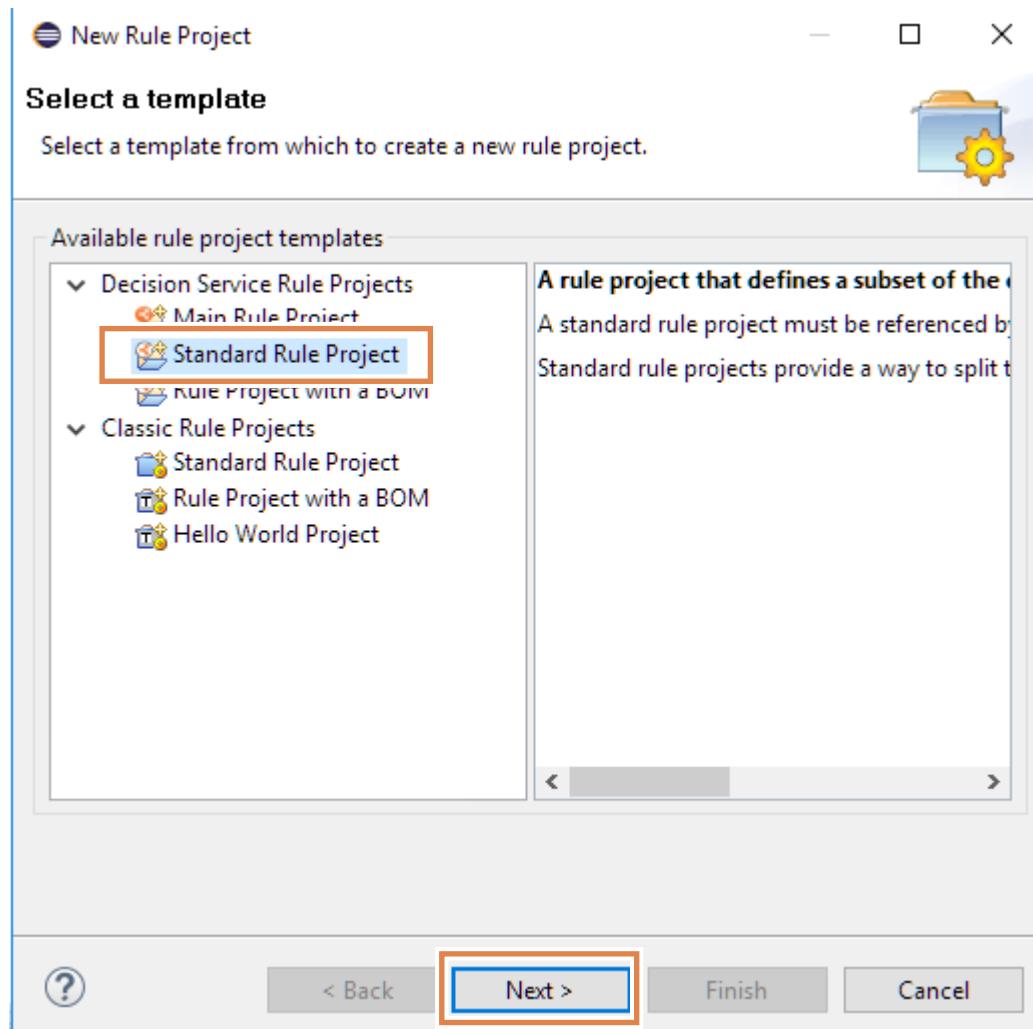
In this section of the exercise, you create an empty standard rule project with a BOM in preparation for the next section of the exercise, where you import the XOM and create the BOM.

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

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

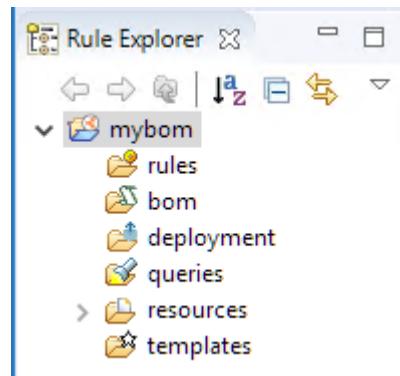


- 2. In the **Decision Service Rule Projects** section of the New Rule Project window, click **Standard Rule Project** and click **Next**.



- 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.

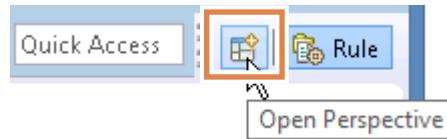


Information

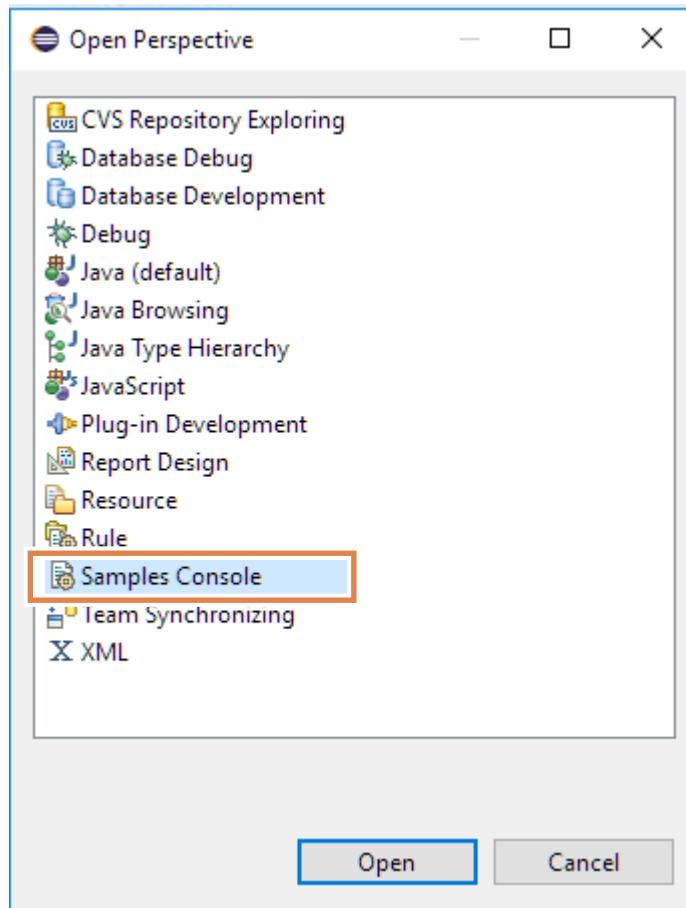
The XOM is the code implementation of the model against which your business rules are executed. For this exercise, the XOM is provided for you. You learn more about XOMs later in this course.

For this exercise, the required XOM is the `loan-xom` Java project and is provided for you.

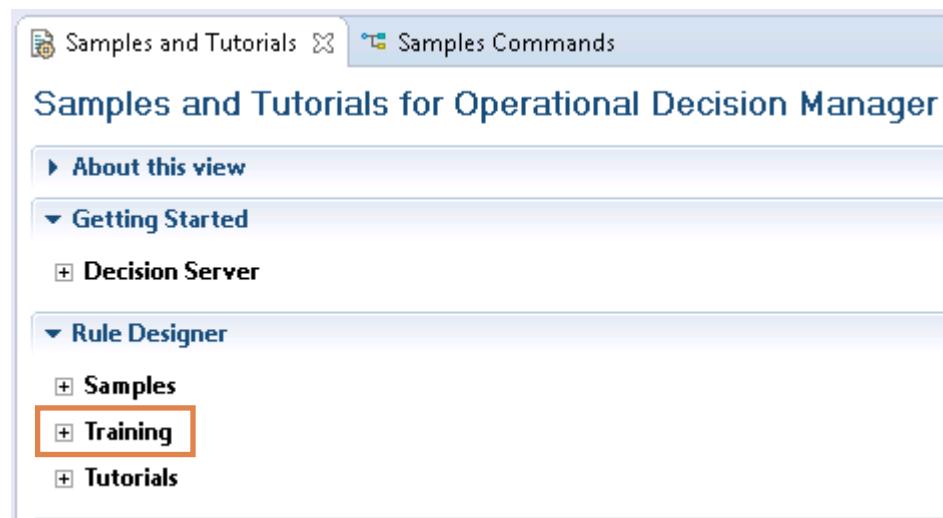
- 1. In the upper-right area of the Rule perspective, click the **Open Perspective** icon.



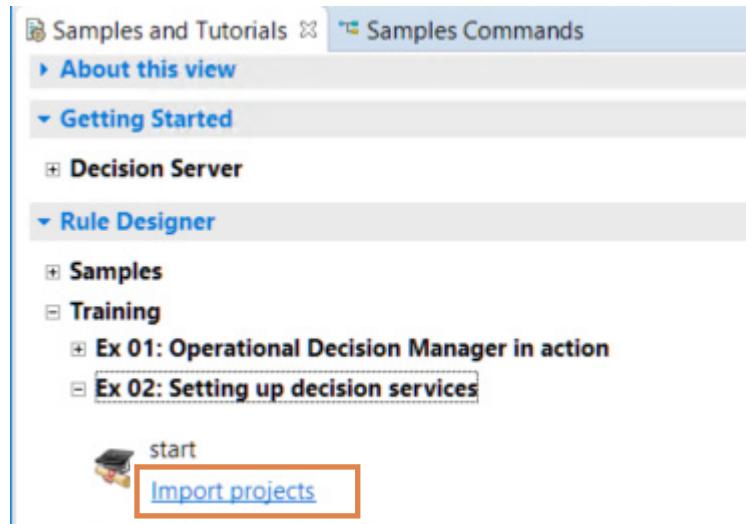
- 2. Click **Samples Console** and then click **Open**.



- 3. In the “Samples and Tutorials” page, go to the Rule Designer section and expand **Training**.

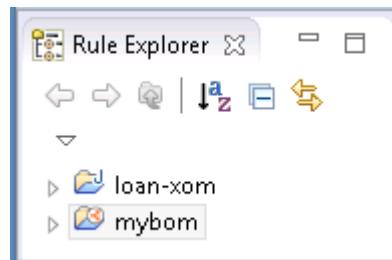


- 4. Expand **Ex 02: Setting up decision services**, and under **start**, click **Import projects**.



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

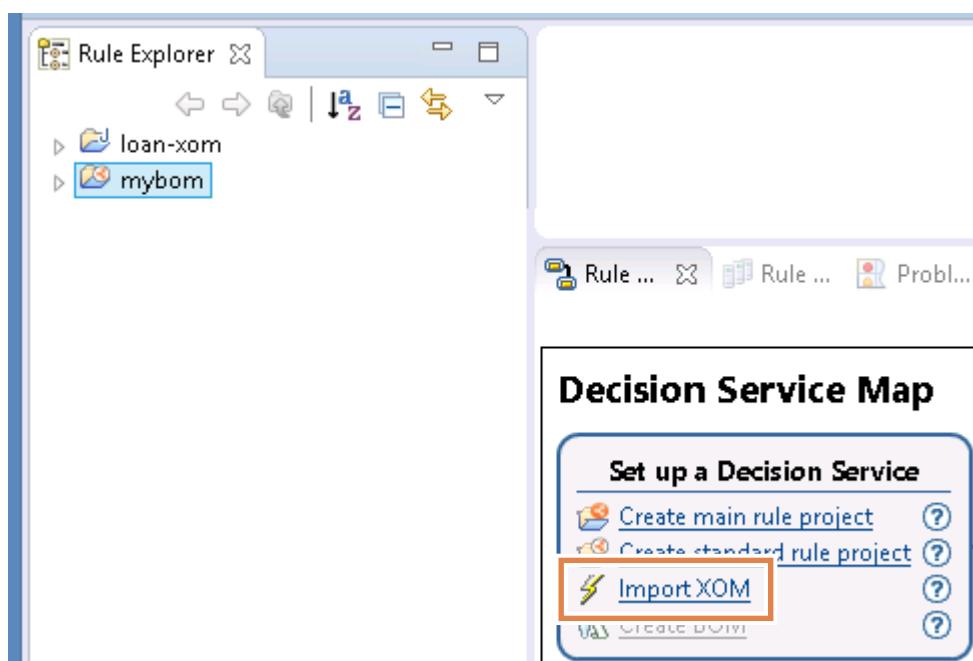
For each exercise, you import projects into your workspace. The procedure for importing is also available at the beginning of this document, in "[Projects for exercises](#)" on page xv.

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

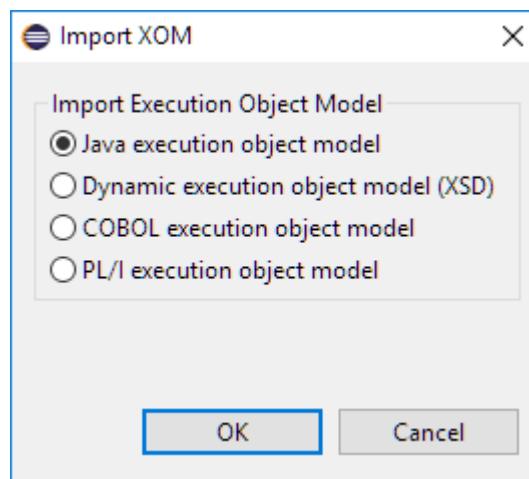
Now that you have a XOM in your workspace, use the Decision Service Map to import it into your decision service.

- 1. Click the `mybom` project in the Rule Explorer.

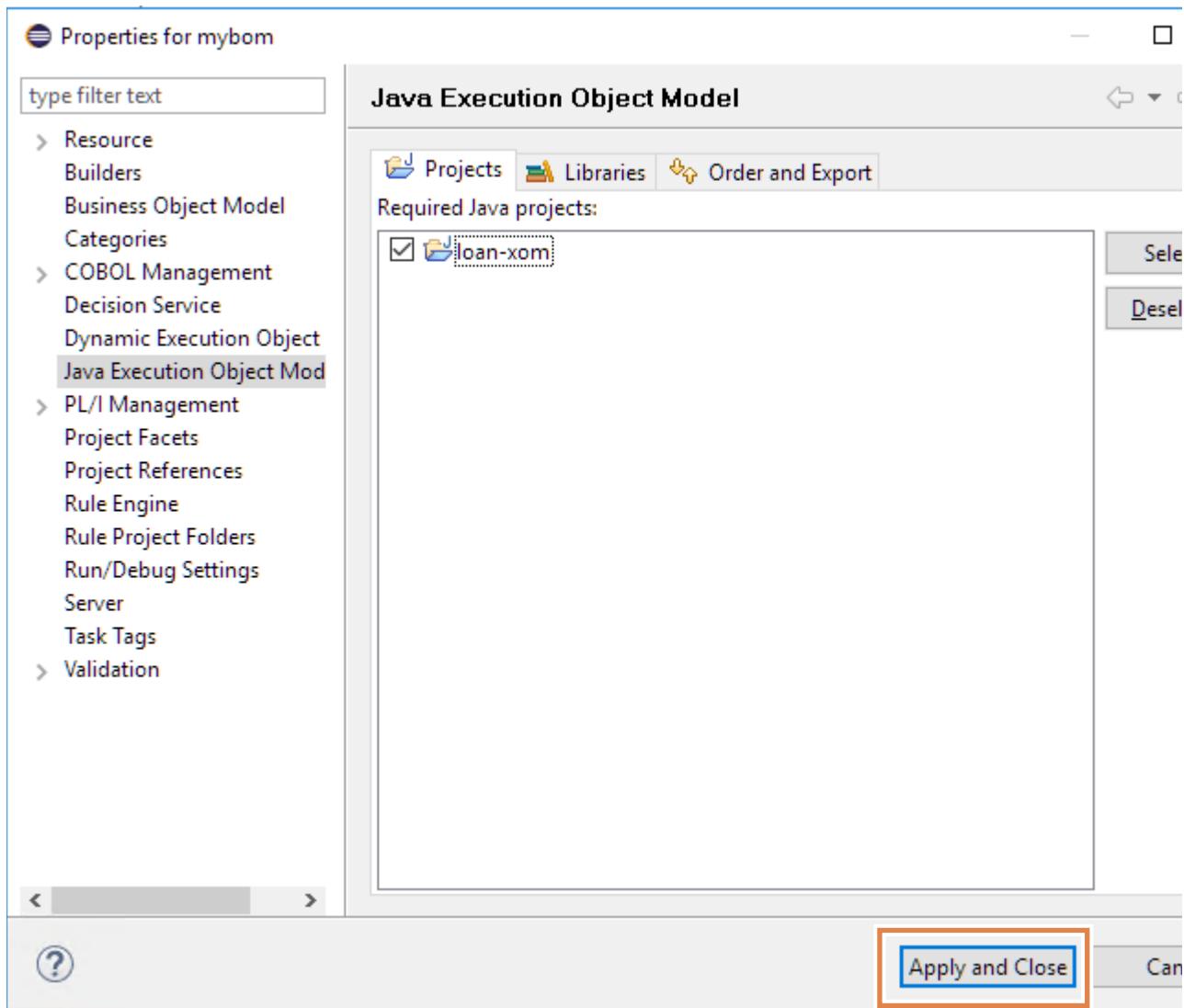
- 2. In the **Set up a Decision Service** part of the Decision Service Map, click the **Import XOM** link to import the XOM into your decision service.



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



4. In the “Properties for mybom” window, under **Java Execution Object Model**, select the **loan-xom** check box, and click **Apply and Close**.

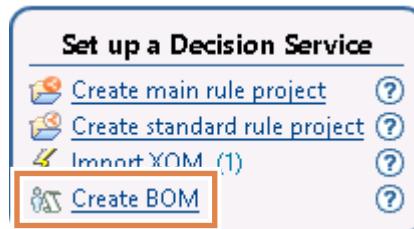


By selecting **loan-xom**, the `mybom` project now references the `loan-xom` Java project.

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**.

BOM Entry

Create a new BOM entry.

BOM folder: /mybom/bom

Folder:

Name: model

Create a BOM entry from a XOM
 Create an empty BOM entry

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

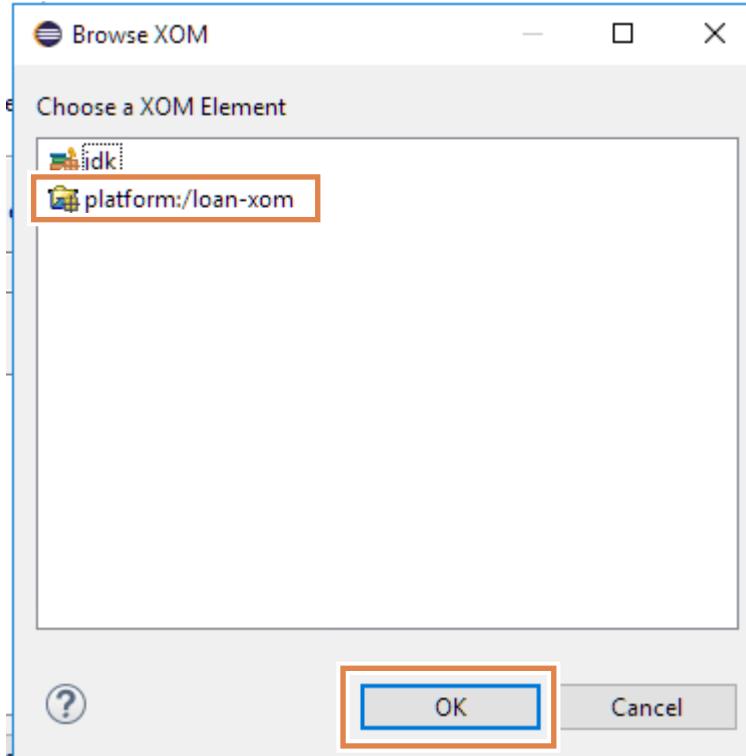
BOM Entry

Create a BOM entry from a XOM.

Choose a XOM entry:

Select classes:

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



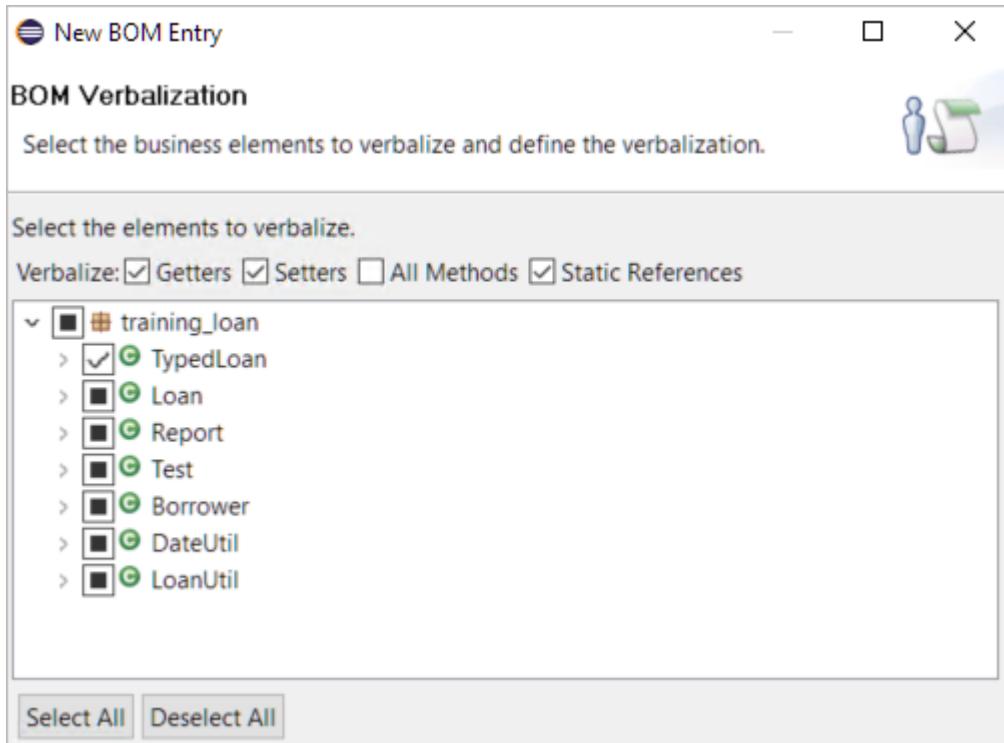
- ___ 6. Make sure that your BOM entry fully reflects the classes and methods in the XOM as follows:
- Expand **training_loan** and select all classes by clicking **Select All**.
 - Make sure that the **Load getters and setters as attributes** check box is selected.



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

- ___ c. Click **Next**.

The BOM Verbalization page opens.



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



Information

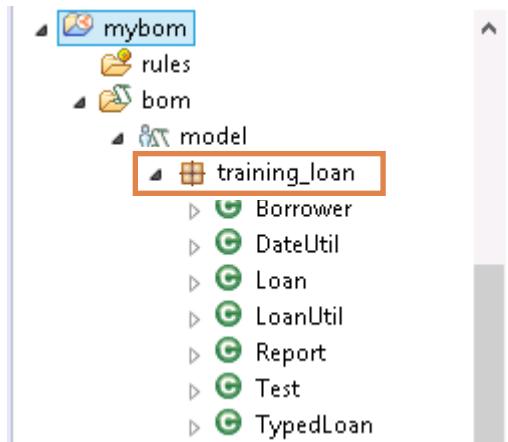
Verbalization attaches natural-language terms to the BOM elements to create the vocabulary for your business rules. You learn more about verbalization later in this course.

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.



Hint

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

✖ [Remove](#) the verbalization. 📖 [Edit](#) the documentation.

Generate automatic variable

Term: ✏️ [Edit](#) term.

ℹ️ 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
- zipCode

New... Delete Edit

- 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

Navigation : "the first name of a borrower"

Template: {first name} of {this}

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

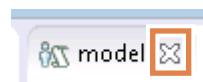
{first name} of {this}



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.



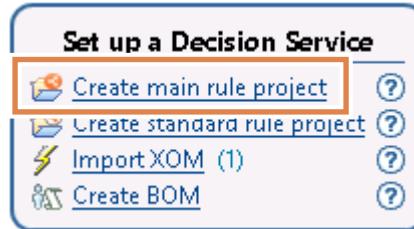
Section 3. Creating the main rule project for the decision service

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.

3.1. Create the main rule project

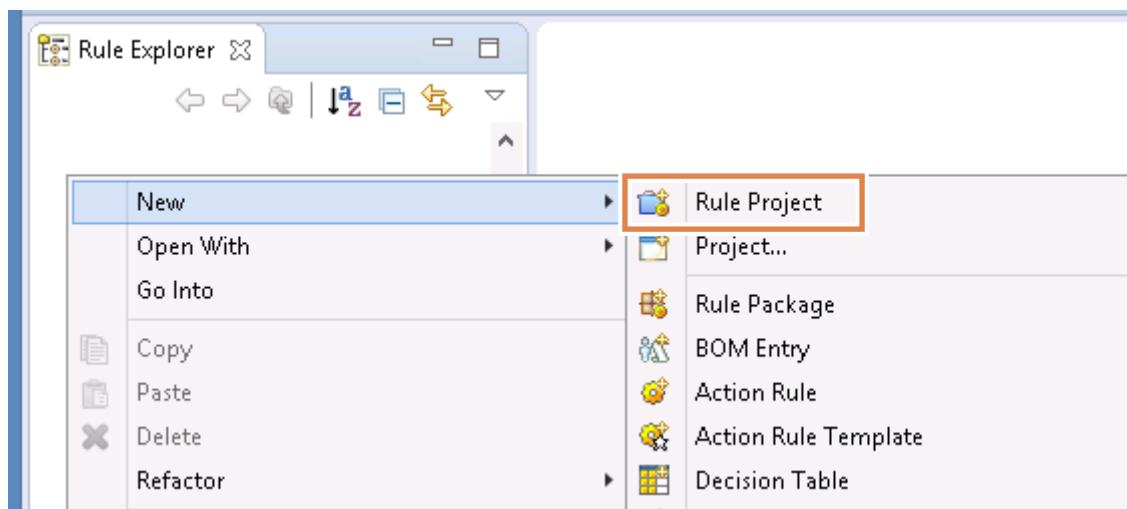
In this section of the exercise, you create a main rule project for the decision service.

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

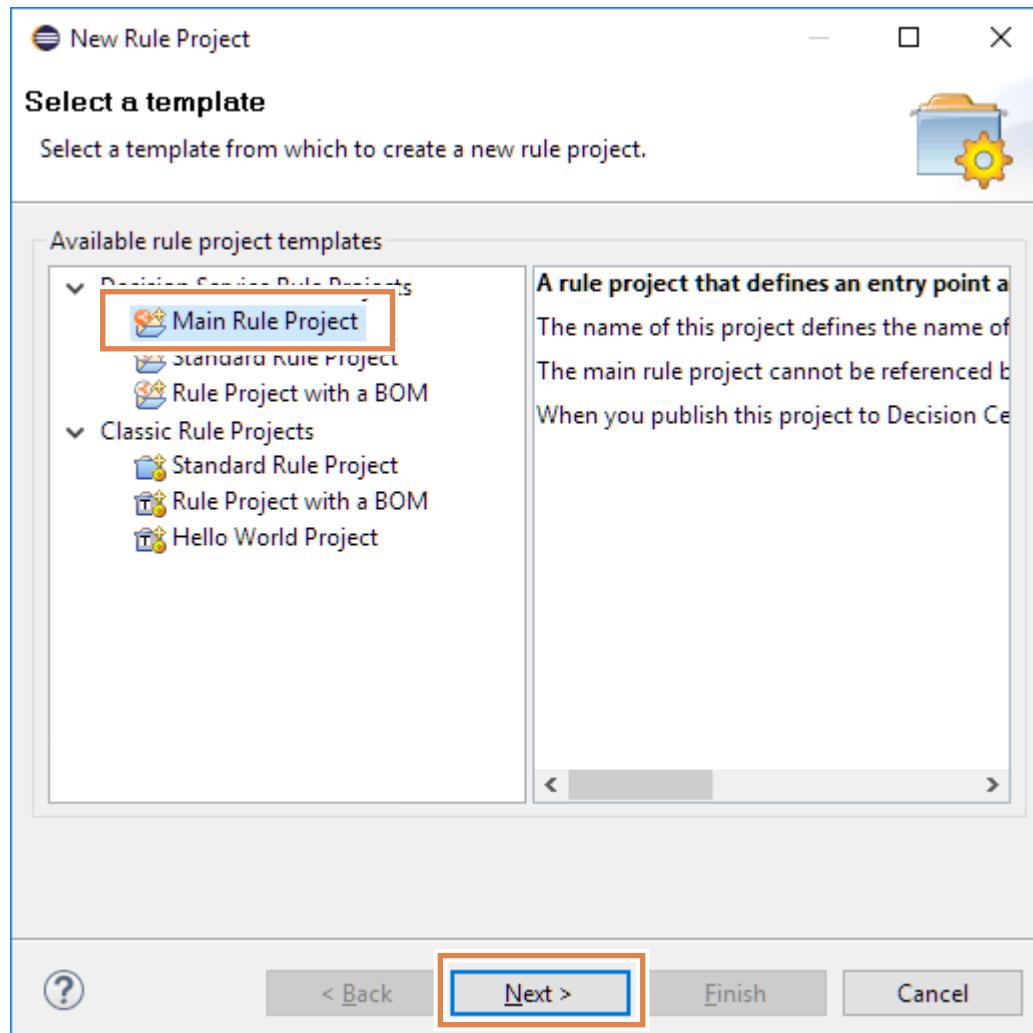


Note

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**.

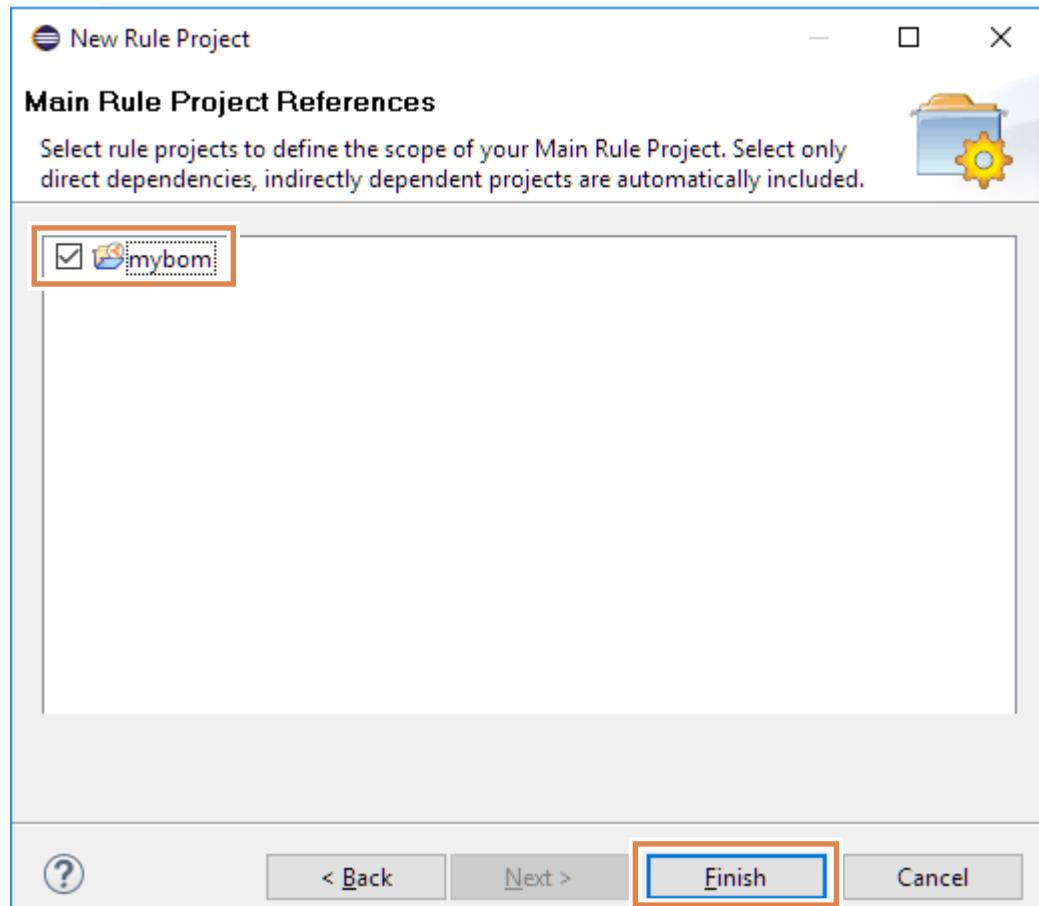


- 2. In the New Rule Project window, create a main rule project for a decision service.
— a. Select **Decision Service Rule Projects > Main Rule Project**, and click **Next**.

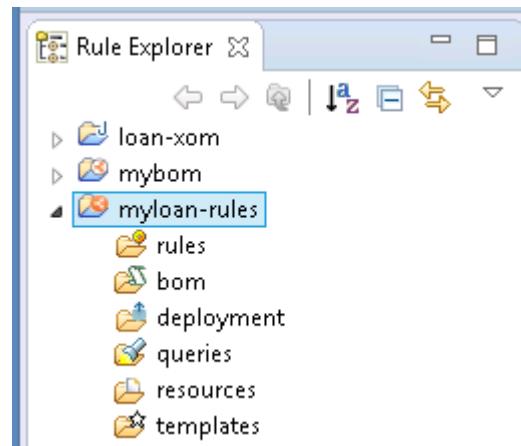


- b. In the **Project name** field, type `myloan-rules` and click **Next**.

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



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

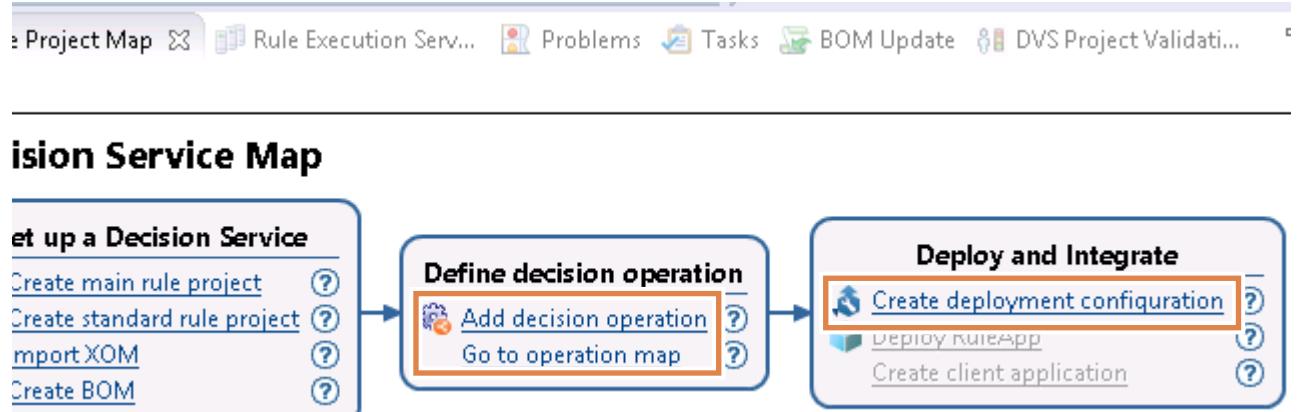


3.2. Identify the next tasks in decision service development

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

- ___ 1. In Rule Explorer, select the `myloan-rules` decision service, and notice 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 parameters) for a decision. The 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 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

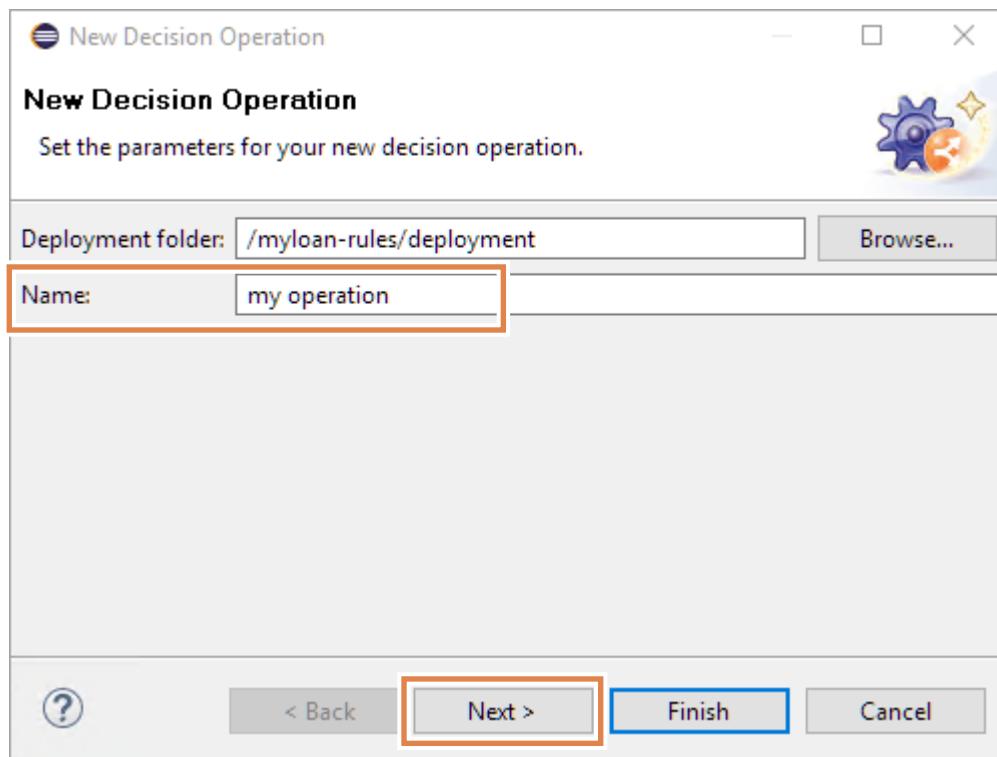
These parameters do not have default values.

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**.



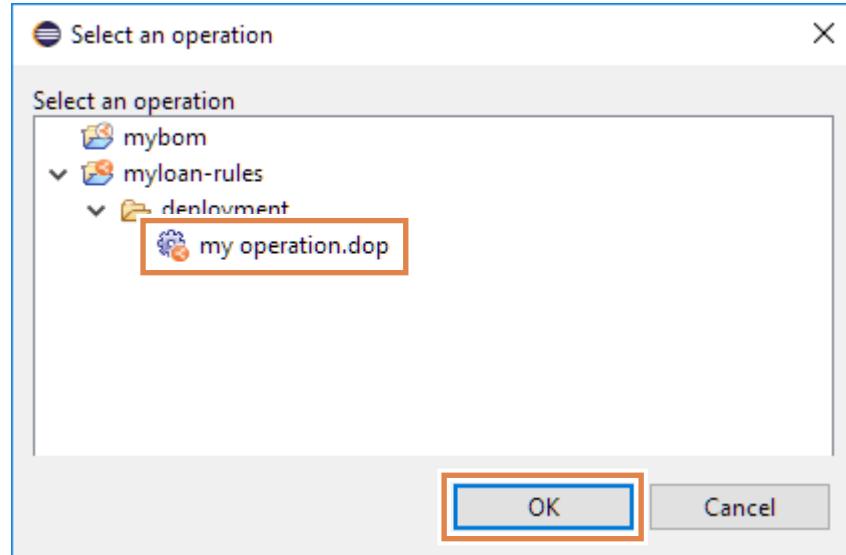
- 3. In the New Decision Operation wizard, in the **Name** field, type **my operation** and click **Next**.



- 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**.

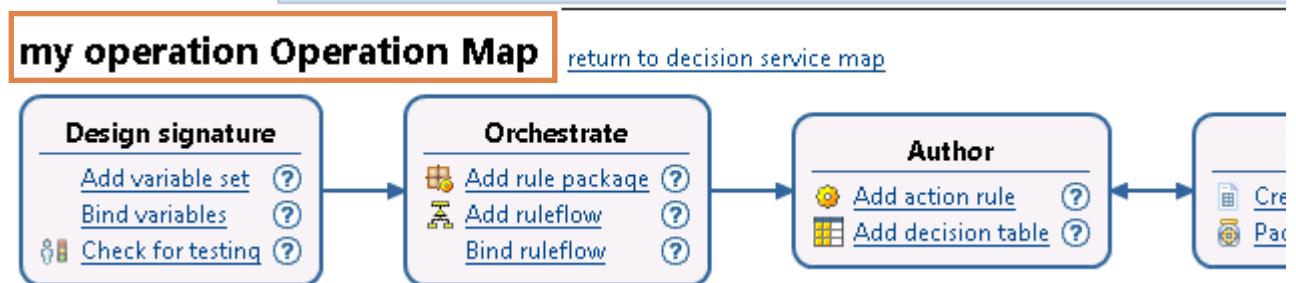


- ___ 6. Select **myloan-rules > deployment > my operation.dop** and click **OK**.



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

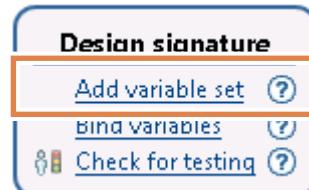
- Design signature
- Orchestrate
- Author
- Test



4.2. Creating variables for parameters

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

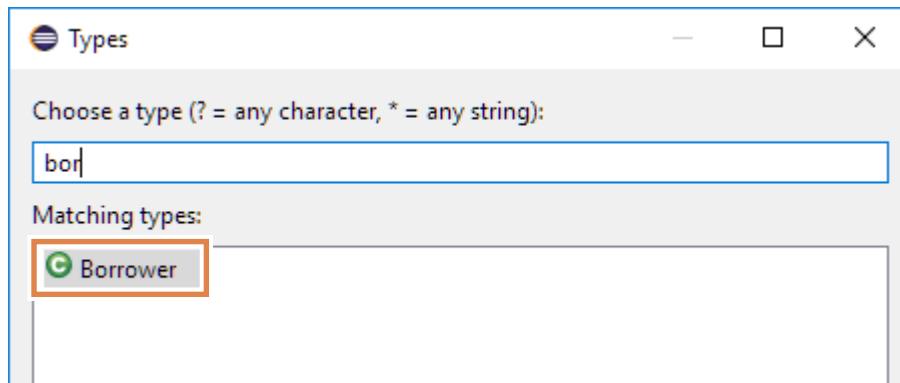
4. In the Variable Set editor, define the `borrower` parameter.

- a. Click **Add**.

Name	Type	Verbalization	Initial Value	
				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 and click **OK**.



- e. In the **Verbalization** field, overwrite the value by typing: `the borrower`

Variable Set: my parameters

Name	Type	Verbalization	Initial Value
<code>borrower</code>	<code>training_loan.Borrower</code>	<code>the borrower</code>	

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 start typing `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**

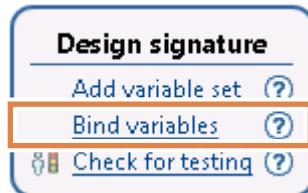
When you set the **Type** value, you can start typing `Report` and Rule Designer finds the `training_loan`.`Report` type for you.

-
- ___ 7. Save your work by pressing **Ctrl+S**.

4.3. Binding the variables to parameters

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

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



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

2. Expand **my parameters** in the **Eligible variables** section of the Decision Operation Signature editor to see the borrower, loan, and report variables.

The screenshot shows the 'Decision Operation Signature - my operation' interface. The 'my parameters' tab is selected. In the 'Eligible variables' section, there is a tree view under 'myloan-rules' with nodes: 'my parameters', 'borrower', 'loan', and 'report'. The 'my parameters' node is highlighted with a red box. In the 'Input Parameters' section, there is a table with one row for 'Parameter name' and 'Verbalization'. In the 'Input - Output Parameters' section, there is a table with one row for 'Parameter name' and 'Verbalization'. In the 'Output Parameters' section, there is a table with one row for 'Parameter name' and 'Verbalization'. At the bottom, there are tabs for 'Overview', 'Signature' (which is selected and highlighted with a red box), and 'Source'.

3. Map the variables to the input and output parameters.
- Drag `borrower` to the **Input Parameters** table.
 - Drag `loan` to the **Input-Output Parameters** table.
 - 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>

- 4. Save your work (Ctrl+S).

4.4. Creating a ruleset variable

Parameters carry the objects that are passed between the calling application and the rule engine. The rules evaluate these objects during rule execution.

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

Rather than manipulating this parameter directly during rule execution, you implement the business request by creating a 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 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 add the variable to the `my parameters` variable set. However, you do not bind `loanApproved` to a parameter.

- 1. In Rule Explorer, in **myloan-rules > rules** folder, double-click **my parameters** to open the variable set.
- 2. Add the `loanApproved` variable.
 - a. In the Variable Set editor, click **Add**.
 - b. In the **Name** column, type over `myVar` to enter: `loanApproved`
 - c. Set the **Type** to boolean (lowercase) by clicking the ellipsis (...) and in the Types window, typing: boolean.
 - d. In the **Verbalization** column, type over `myVar` to enter: the loan is approved
 - e. In the **Initial Value** column, type: true
- 3. Save your work (Ctrl+S).
- 4. Close the variable set editor by clicking the X for that window.

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

Section 5. Creating rule packages

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



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



Requirements

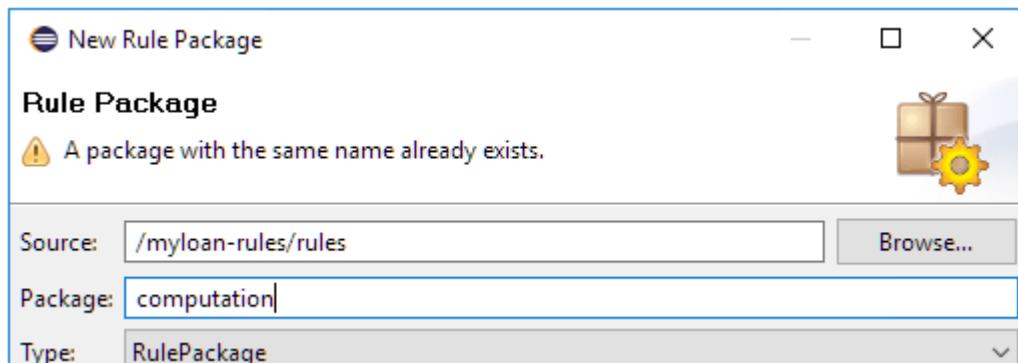
The business analysts provide you with an initial outline of how the rules can be organized and the naming conventions to use. Based on these requirements, the rule artifacts should be logically managed within the following packages and subpackages:

- 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.

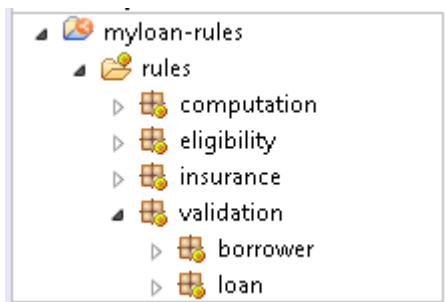
- ___ 3. Add the remaining packages by using the instructions in [Step 2](#):
 - eligibility
 - insurance
 - validation
- ___ 4. Create the borrower and loan subpackages of the validation package.
 - ___ a. In the Orchestrate part of the Operation Map, click the **Add rule package** link.
 - ___ b. In the **Package** field, type validation.borrower and click **Finish**.
 - ___ c. Add another package, type validation.loan in the **Package** field, and click **Finish**.



Note

Adding a **period (.)** to the package name creates 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 6, "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. Start the sample server by clicking the **Start Sample Server** shortcut in the Windows **Start** menu.



Reminder

Decision Center Business console runs 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 trying to open the Business console in a browser, for example: `http://localhost:9090/teamserver`

If the console page opens correctly, the server is running and you can close the browser.

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:9090/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

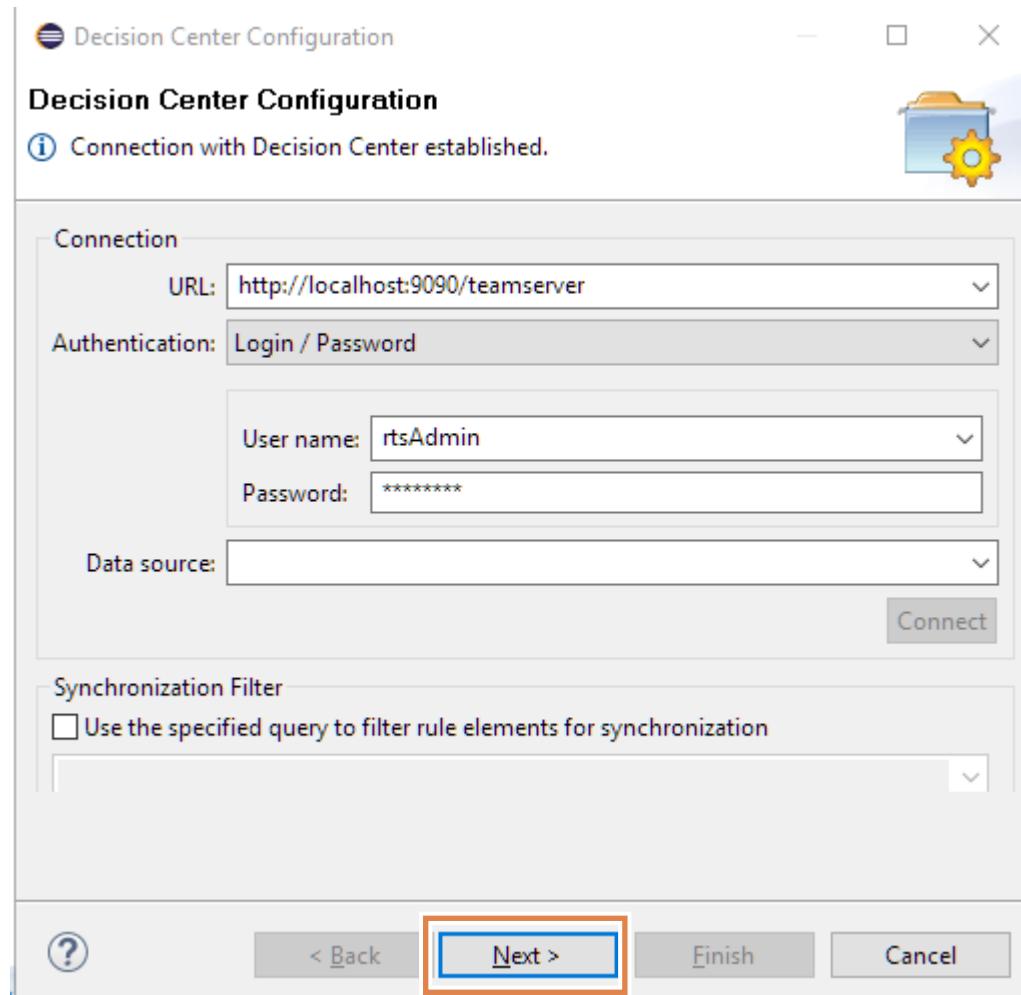
When you synchronize between Rule Designer and the ODM on Cloud environment, enter the URL of your IBM ODM on Cloud instance in the **URL** field instead of `localhost` and make sure to use “`https`”.

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**.



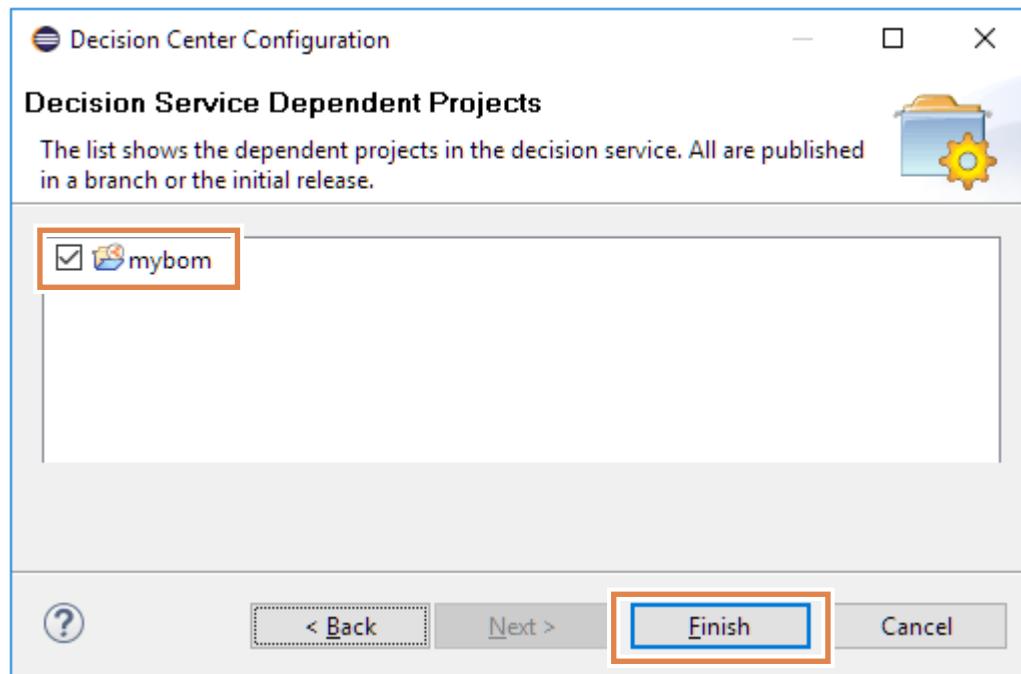
Troubleshooting

Sometimes, when you connect to Decision Center, the Decision Center Configuration window disappears behind other windows. You can click your Rule Designer instance or minimize other windows to see the Decision Center Configuration window.

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

Note: 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**.



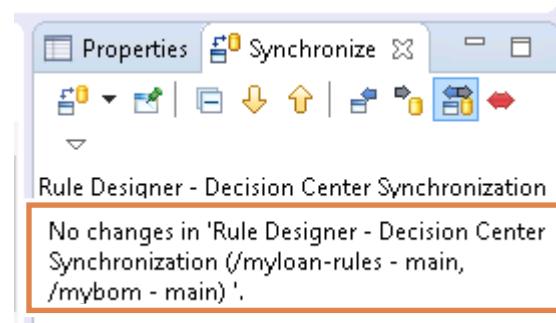
Note

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

8. When prompted to switch to the Team Synchronizing perspective, click **No**.

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

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. From the Windows **Start** menu, click the **Decision Center Business console** shortcut.

**Information**

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

`http://localhost:9090/decisioncenter`

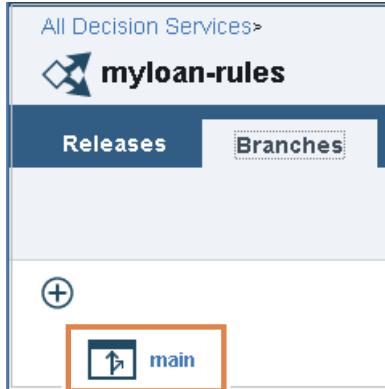
Make sure that you use the correct URL and port for your environment.

- ___ 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 your `myloan-rules` decision service is now listed.

The screenshot shows the Decision Center Business console interface. At the top, there is a navigation bar with tabs: HOME, LIBRARY (which is selected), WORK, and ADMINISTRATION. Below the navigation bar, the title 'Decision Services' is displayed. Under this title, there are two filter options: 'Date' and 'Name'. A large blue arrow icon points upwards. To its right, there is a card for the 'myloan-rules' service. The card contains the service name 'myloan-rules' in blue, a small blue icon, and the text 'Created by rtsAdmin on Jun 19, 2018'.

- ___ 4. Click the **myloan-rules** link.

- ___ 5. On the **Branches** tab, click **main** to open the main branch of myloan-rules.



- ___ 6. If you are not on the **Decision Artifacts** tab, click **Decision Artifacts**.



Troubleshooting

You might need to use the scrolling arrows on the toolbar to access different tabs.



- ___ 7. On the **Decision Artifacts** page, click **Types**, select **All Types**, and click **Apply**.
 ___ 8. Expand **myloan-rules**.
 ___ 9. Click **my parameters** and click the **Edit** (pencil) icon.

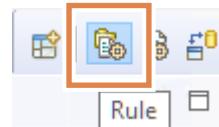
Name	Type	Verbalization	Initial Value
borrower	borrower	the borrower	
loan	loan	the loan	
loanApproved	boolean	the loan is approved	true
report	report	the report	

- ___ 10. Set the value of the **loanApproved** variable to: **false**
 ___ 11. Click **Save** and **Create new version**.
 ___ 12. Leave Business console open.

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. Synchronizing Rule Designer with Decision Center

- ___ 1. Go back to Rule Designer and make sure that you are 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 variable set is changed.
- ___ 4. Double-click `my parameters.var` to open it in the Text Compare view and check your changes.

Local File	Remote File
1	1
2 :rsion="2.0" xmlns:xmi="	2 :rsion="2.0" xmlns:xm
3	3
4 iid>	4 iid>
5 i.Borrower" initialValue	5 i.Borrower" initialValue
6 n" initialValue="" verb	6 n" initialValue="" \
7 leport" initialValue=""	7 leport" initialValue=
8 initialValue="true" ver	8 initialValue="false'



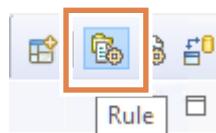
Information

In the Synchronize view, entries show whether 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 that are 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 be `false`.

- ___ 5. Update your local copy of the project in Rule Designer by right-clicking **my parameters.var** and clicking **Update**.
- ___ 6. When 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.
- ___ 7. Close the Text Compare view.
- ___ 8. Return to the Rule perspective by clicking **Rule** on the toolbar.



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. Return to the Business console. If you need to sign in again, use `rtsAdmin` as the user name and password.
- ___ 2. Click the **Library** tab.
- ___ 3. Click the arrow beside **myloan-rules** and click **Delete**.

The screenshot shows the IBM Decision Center interface. At the top, there's a navigation bar with tabs: HOME, LIBRARY (which is currently selected), and WORK. Below the navigation bar, the title 'Decision Services' is displayed. Underneath, there's a table header with columns for Date and Name. A single row is visible, representing a decision service named 'myloan-rules'. To the right of this row, a context menu is open, featuring a downward-pointing arrow and a 'Delete...' option. This 'Delete...' option is specifically highlighted with a red rectangular box.

- ___ 4. When prompted to confirm deletion of the decision service, click **Delete**.
- ___ 5. Sign out and close the Business 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.

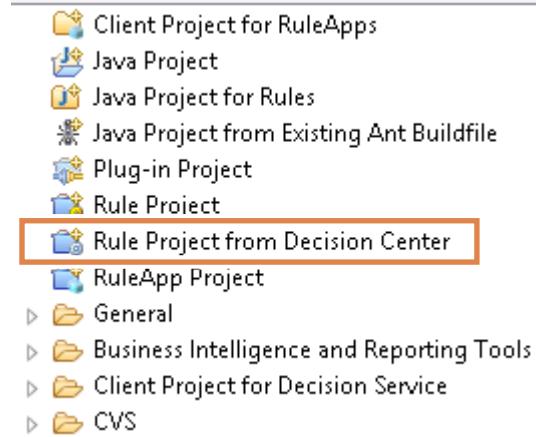


Requirements

Business analysts indicate that they worked with another team and developed a complementary decision service, called Miniloan Service. The most recent 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**.

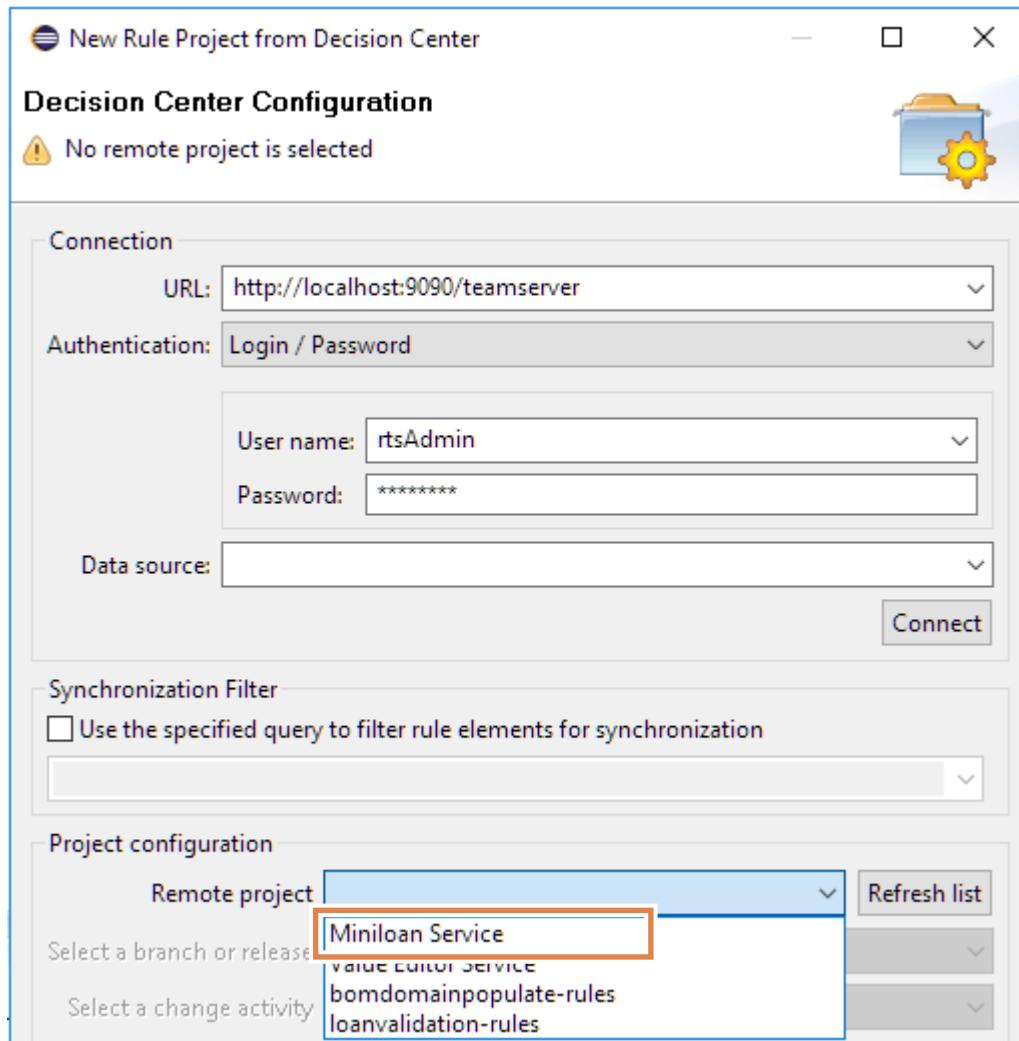


The New Rule Project from Decision Center wizard opens.

- 3. Verify the connection entries, which should be prepopulated from the previous exercise.
 - **URL:** `http://localhost:9090/teamserver`
 - **User name:** `rtsAdmin`
 - **Password:** `rtsAdmin`
- 4. Click **Connect**.

When the connection is established, the list of remote projects becomes available.

- 5. In the **Project configuration** section, select **Miniloan Service** from the **Remote project** list.



- 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** to import Miniloan Service into Rule Designer.

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.
The Miniloan Service project is now listed in Rule Explorer.
- ___ 9. In Miniloan Service, disconnect Rule Designer from Decision Center.
 - ___ a. Right-click **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.

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, click **Problems** to open the Problems view for the Miniloan Service project.

Notice that one of the issues is about the project path for the `miniloan-xom`, which is the XOM project for the Miniloan Service decision service.

4 errors, 0 warnings, 0 others

Description

Errors (4 items)

- An internal error occurred while building this element. See error log for more information.
- The deployment configuration 'Miniloan' only references Java SE Rule Execution Server connectio...
- The deployment configuration 'Miniloan' references a target 'Decision Service Execution' that can...
- The project path platform:/miniloan-xom cannot be resolved.

- ___ 2. In Rule Explorer, right-click **Miniloan Service** and click **Properties** from the menu.

- ___ 3. In the left pane of the Properties window, select **Java Execution Object Model**.
The Java Execution Object Model page opens and you see that `miniloan-xom` is selected.
 - ___ 4. In the left pane of the Properties window, select **Business Object Model**.
 - ___ 5. On 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 XOM 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.

- ___ 6. Click **Cancel** to close the Properties window.
- ___ 7. Close Rule Designer.

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 demonstrated how you start developing a business rule application. You set up a decision service in Rule Designer, and publish it to Decision Center for the business users.

Exercise 3. Modeling decisions

Estimated time

00:30

Overview

This exercise demonstrates how to create and test a decision model service.

Objectives

After completing this exercise, you should be able to:

- Create a model diagram
- Define the decision and data node structure
- Create custom data types
- Author business logic in decision modeling language
- Test the model

Introduction

In this exercise, you learn how to model decisions in Decision Center Business console. You also see how to test your model and deploy it as a decision service to an execution environment.

This exercise includes the following sections:

- [Section 1, "Creating a diagram"](#)
- [Section 2, "Author and test the decision logic"](#)

Requirements

The second part of this exercise uses IBM Decision Composer online, and requires that you have an IBM ID. If you do not have an IBM ID, you can follow the instructions in the exercise to obtain one.

Section 1. Creating a diagram

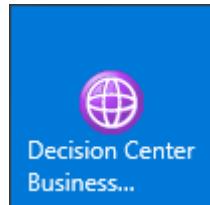
In this section of the exercise, you set up the environment for the exercise and create a standard rule project for the decision service.

1.1. Open Decision Center Business console

- ___ 1. If the sample server is not started, start it now.
 - ___ a. Click the **Start** menu, and click the **Start Sample Server** shortcut.



- ___ b. Wait until the server is started.
- ___ 2. Open the Business console from the **Start** menu, by clicking the **Decision Center Business console** shortcut.



- ___ 3. Sign in as a business user with the following values for the **Username** and **Password** fields.
 - **Username:** rtsAdmin
 - **Password:** rtsAdmin
- ___ 4. Click **Log in**.

- 5. Click the **Home** tab, and notice the **Start modeling decisions** section, which links you to the IBM Knowledge Center for more information about modeling decisions.

1.2. Create a model

- 1. Click the **Library** tab.

The Library includes regular decision services and decision services that are based on models, or *decision model services*. Decision model services are indicated by the model icon.



- 2. Click the **Create new Decision Model Service** (plus sign) icon.

- 3. In the name field, enter my decision model and click **Create**.

- 4. Click **main**.

A new model diagram opens with a decision node and a data node as your starting point.

- 5. Click the **Edit** icon.



Next, you create a model that matches runners to races that suit their abilities.

1.3. Design the Match Race to Runner decision model

To model a decision, first consider these questions:

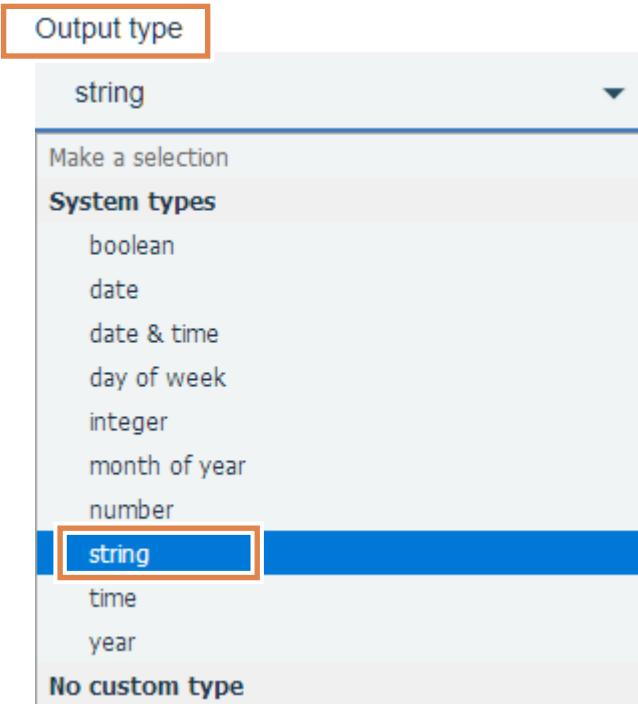
Design questions	Answers
What is the actual decision you want to make?	Match a runner to the appropriate race
What is the business data that is involved in the decision?	The runner The races
What data influences the outcome?	The location and ability of the runner The location and difficulty level of the races
Is the main decision dependent on other decisions?	Yes, determining the runner's level of ability
What data is involved in the subdecision?	How fast and how far this runner runs regularly

Next, you translate this design into a diagram.

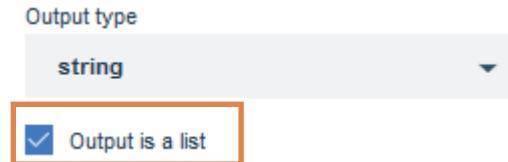
1.4. Model the node structure

- 1. Define the Matching race decision node to output a list of string values.
 - a. On the diagram, click the **Decision 1** node.
 - b. On the **Model** tab, change the name from **Decision 1** to **Matching race** and press Enter.

- ___ c. In the **Output type** list, choose **string**.



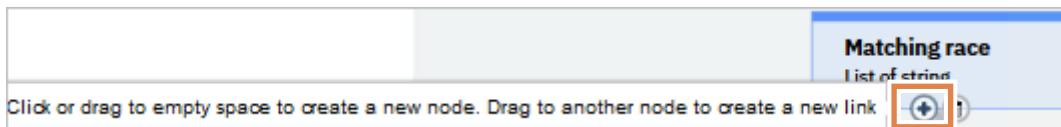
- ___ d. Select the **Output is a list** option.



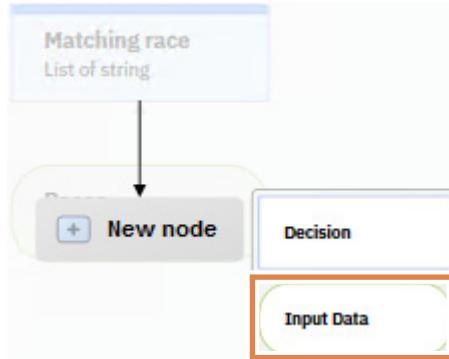
To match a race to a runner, the decision depends on the race and the runner data. Next, you add the dependency to these data nodes.

- ___ 2. On the diagram, click the **Input Data 1** node and set the name to: **Runner**
- ___ 3. Add another data node called **Races**.

- ___ a. Hover your mouse on the **Matching race** node and click the plus sign.

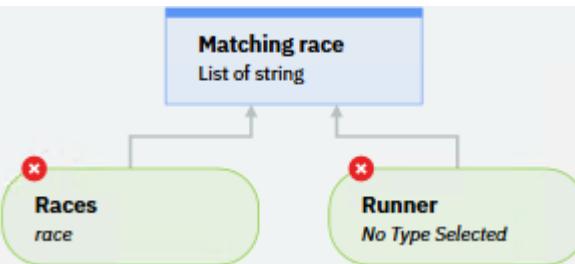


- __ b. Click **Input Data**.



- __ c. Set the name to: Races

You see errors on your data nodes because they need to be set to a type. For this model, you create custom types.



1.5. Create custom data types

- __ 1. Define the **category** type with the values: advanced, intermediate, and beginner.

- __ a. In the left pane, click the **Types** tab.



- __ b. In the **Data Types** section, click the plus sign and click **Add enumeration type**.

- __ c. In the **Name** field, type **category** and press Enter.

- __ d. In the **List of values** section, click the plus sign to add these values.

advanced
intermediate
beginner

- __ 2. Define the **runner** type with these attributes:

name	string
area	string
average speed	number
weekly mileage	number
category	category

- __ a. In the **Data Types** section on the **Types** tab, click the plus sign and click **Add type**.
- __ b. In the **Name** field, type `runner` and press Enter.
- __ c. In the **Attributes** section, click the plus sign, and in the **Name** field, type `name` and leave the **Type** field set to `string`.
- __ d. Repeat [Step c](#) to define the remaining attributes listed in the table above.

Note that the **category** attribute is based on the custom `category` type that you created.

- __ 3. Define the custom **race** type.

- __ a. In the **Data Types** section on the **Types** tab, click the plus sign and click **Add type**.
- __ b. In the **Name** field, type `race` and press Enter.
- __ c. Add the following attributes by clicking the plus sign.

name	string
area	string
category	category



Information

Notice that both the runner and the race have a `category` attribute. Next, you build the decision logic to match the runner to appropriate races by using this attribute.

- __ 4. Save your work.
 - __ a. On the toolbar, click **Save and Continue**.



- __ b. Click **Create new version**.



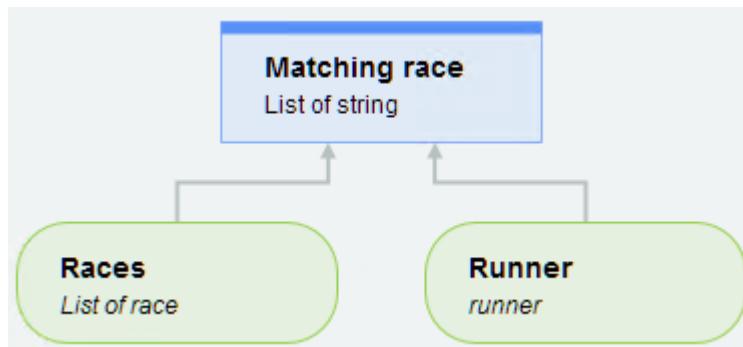
Note

Save often to avoid losing your work in case of session timeouts. Each time that you save, you create new version of the model.

1.6. Set the output types for the data nodes

- ___ 1. Make sure that you are on the **Model** tab to edit the diagram.
- ___ 2. On the diagram, click the **Runner** data node and set the **Output type** to **runner**.
- ___ 3. Click the **Races** data node and set the **Output type** to **race**.
- ___ 4. Select **Output is a list** option.

The data nodes do not show errors anymore.



- ___ 5. Save your work.
- ___ a. On the toolbar, click **Save and Continue**.



- ___ b. Click **Create new version**.

Section 2. Author and test the decision logic

The **Matching race** decision needs as input the location and ability of the runner, plus the location and difficulty level of the races. The runner's ability is defined by the **category** attribute.

Before a match can be made, a subdecision must determine the runner's category based on how fast and how far the runner runs each week. For this subdecision, you add another decision node called **Runner category** as input to the **Matching race** decision node

2.1. Define the Runner category decision node and logic

- ___ 1. Create the **Runner category** decision node.
 - ___ a. On the toolbar, click the **Decision** plus sign.



- ___ b. Hover your mouse over the new node, click the plus sign, and use your mouse to drag it to the **Matching race** node.



Hint

By dragging the plus sign on one node to another node, you create a dependency between those nodes.

- ___ c. Set the name of the node to: **Runner category**
- ___ d. Set the **Output type** to **category**.
- ___ e. The **Runner category** node requires data input from the **Runner** data node, so hover your mouse over the **Runner** data node, click the plus sign, and use your mouse to drag it to the **Runner category** decision node.

- ___ 2. Define the **Runner category** logic.

- ___ a. In the **Decision logic** section, click **Add table**.

▼ Decision logic



- ___ b. In the **Select the condition columns for your decision table** section, choose:
 - the weekly mileage of the runner
 - the average speed of the runner

These selections become condition columns in your table.

- ___ c. Click **Create table**.
- ___ d. Keep the default table name.
- ___ e. Set the values in the table to match these values.

weekly mileage		average speed		Runner category
min	max	min	max	
0	10	3	6	beginner
10	20	6	9	intermediate
20	30	9	12	advanced

- ___ 3. Change the operator in the **weekly mileage** columns to **]..]**
 - ___ a. Right-click the cell in the min or max column of row 1
 - ___ b. Click **Change Operator** and select **]..]**
The result is equivalent to: more than 0 and at most 10
 - ___ c. Change the operator for rows 2 and 3.

Your decision table should match this screen capture.

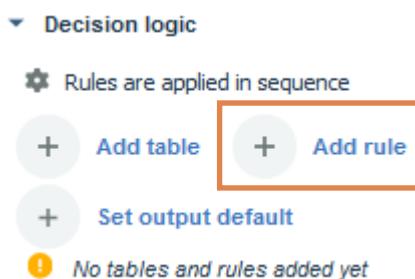
	weekly mileage		average speed		Runner category
	min	max	min	max	
1	[1] 0	10]	3	6	beginner
2	[2] 10	20]	6	9	intermediate
3	[3] 20	30]	9	12	advanced

- ___ 4. Close the table.

- ___ 5. Click **Save and Continue**.

2.2. Define the Matching race logic

- ___ 1. Click the **Matching race** decision node.
- ___ 2. In the **Decision logic** section of the **Model** tab, click **Add rule**.



- ___ 3. Choose the following criteria in the **Select the criteria for your rule** section:
- the races
 - the runner category
 - the area of the runner
- ___ 4. Click **Create rule**.
- ___ 5. Copy and paste the following text for this rule:
- ```
for each race called 'the race' , in 'the races'
if
 the area of 'the runner' is the area of 'the race'
 and the category of 'the race' is 'the runner category'
then
 add "Here is a race that we recommend for you: " + the name of 'the race'
 +" . Are you in?" to decision;
```
- Your rule should match this screen capture.

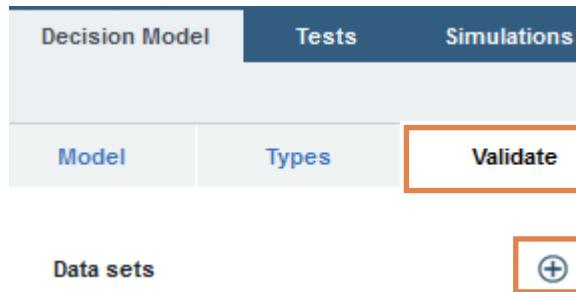
```
for each race called 'the race' , in 'the races'
if
 the area of 'the runner' is the area of 'the race'
 and the category of 'the race' is 'the runner category'
then
 add "Here is a race that we recommend for you: " + the name of 'the race' +" . Are you in?" to decision;
```

| Available inputs | Output          |
|------------------|-----------------|
| Name             | Type            |
| ▶ Races          | List of race    |
| ▶ Runner         | runner          |
| category         | category (enum) |

- \_\_\_ 6. Close the rule.
- \_\_\_ 7. Click **Save and Continue**.

## 2.3. Test the model

- 1. Click the **Validate** tab, and click the plus sign to create a new data set.



- 2. For the **New input** field, click the **Edit** (pencil) icon and set the name to: Mark

A screenshot of a configuration interface. At the top, it says 'Data sets' and has a red-bordered 'New input' button. To the right of 'New input' are a red-bordered edit icon (pencil) and a trash icon. Below this is a 'Run' button. The main area shows a JSON editor titled 'Races' with a single item '1'. The item 'race' has fields 'name' (with placeholder 'Enter a text string, eg. Hello') and 'area' (with placeholder 'Enter a text string, eg. Hello'). The 'category' field has a dropdown menu showing 'advanced'. At the bottom of the JSON editor is a blue 'Add race' button with a red-bordered '+' icon.

\_\_\_ 3. In the **Runner** section, enter the following data:

- **name:** Mark
- **area:** Toronto
- **average speed:** 3
- **weekly mileage:** 7
- **category:** beginner

| Runner         |          |
|----------------|----------|
| name           | Mark     |
| area           | Toronto  |
| average speed  | 3        |
| weekly mileage | 7        |
| category       | beginner |

\_\_\_ 4. Define input data for the **Races** section

\_\_\_ a. In the **race** fields, enter the following data:

- **name:** Paris Marathon
- **area:** Paris
- **category:** advanced

\_\_\_ b. Click **Add race**, and enter the following data:

- **name:** Citadel Trail
- **area:** Montsegur
- **category:** intermediate

\_\_\_ c. Click **Add race again**, and enter the following data:

- **name:** Terry Fox Run
- **area:** Toronto
- **category:** beginner

- \_\_\_ 5. Click **Run**.

The results are correct.

**Mark** - Last run on 2/28/2019, 12:09:18 PM

Show JSON output

▼ Decision outputs

| Matching race output                                                              | Matching race type |
|-----------------------------------------------------------------------------------|--------------------|
| [<br>"Here is a race that we recommend for you: Terry Fox Run . Are you in?"<br>] | string             |

- \_\_\_ 6. In the **Runner** section, change **area** to **Montreal** and click **Run**.

The test with **area** set to **Montreal** does not provide any output.



## Questions

Why is there no output when a matching race isn't found?

**Mark** - Last run on 2/28/2019, 12:04:30 PM

Show JSON output

X

▼ Decision outputs

| Matching race output | Matching race type |
|----------------------|--------------------|
| [<br>]               | string             |

▼ Messages

|                                |
|--------------------------------|
| No output messages to display. |
|--------------------------------|

Answer: Because the decision logic doesn't handle cases when no races match the runner. You must author a rule to handle a "no match" situation.

## 2.4. Add a rule to handle cases when no race matches the runner

- \_\_\_ 1. Return to the diagram and click the **Matching race** node.
- \_\_\_ 2. Add a second rule to handle the case when no races match the runner.
  - \_\_\_ a. In the **Decision logic** section, click **Add rule**.
  - \_\_\_ b. In the **Select the criteria for your rule** section, choose:
    - **the races**
    - **the name of the runner**
  - \_\_\_ c. Click **Create rule**.

- \_\_\_ 3. Overwrite the rule contents with the following text:

```
if the number of elements in 'the matching races' equals 0
then
 print "Hi, " + the name of 'the runner' + "! We have no recommendation for
your category and area , please come back later" ;
```

## 2.5. Validate the model

- \_\_\_ 1. Click the **Validate** tab.  
 \_\_\_ 2. Make sure that `area` is set to `Montreal`, and click **Run**.

The Decision outputs section correctly shows that no race matched.

**Mark** - Last run on 2/28/2019, 12:59:03 PM [Show JSON output](#)

▼ **Decision outputs**

| Matching race output | Matching race type |
|----------------------|--------------------|
| [<br>]<br>]          | string             |

▼ **Messages**

- Hi, Mark! We have no recommendation for your category and area , please come back later

- \_\_\_ 3. In the **Runner** section on the Validate tab, change `area` to `Toronto`, and click **Run**.

The Decision outputs section correctly shows a match. However, the Messages section also includes a message that no match was found.



## Questions

Why does the output show conflicting results?

**Mark** - Last run on 2/28/2019, 1:05:02 PM [Show JSON output](#)

▼ **Decision outputs**

| Matching race output                                                                        | Matching race type |
|---------------------------------------------------------------------------------------------|--------------------|
| [<br>"Here is a race that we recommend for you: Terry Fox Run . Are you in?"<br>]<br>string |                    |

▼ **Messages**

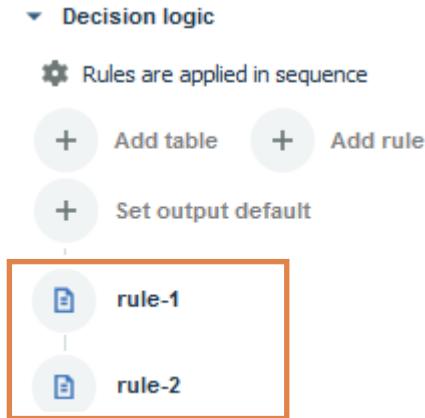
- Hi, Mark! We have no recommendation for your category and area , please come back later

## 2.6. Editing rule sequence

- \_\_\_ 1. Return to the diagram and click the **Matching race** decision node.

Notice the setting: Rules are applied in sequence. That means rule-2 executes before rule-1 matches a race to the runner.

- 2. Select **rule-1** and drag it above **rule-2**.



- 3. Return to the **Validate** tab and click **Run**.

This time, only the correct result for Toronto is returned.

- 4. Set area to Montreal and click **Run**.

The correct result of no recommendation is returned for Montreal.

- 5. Click **Save and Close** on the toolbar and click **Create new version**.

- 6. Return to the **Library** tab.

- 7. Close Business console.

## End of exercise

## Exercise review and wrap-up

This exercise demonstrated how to create, test, and deploy a decision model service.

---

# Exercise 4. Working with the BOM

## Estimated time

00:45

## Overview

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

## Objectives

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 support files, which are installed in the `<InstallDir>\studio\training` directory:

- Start project: Dev 04 – BOM\start

## 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.

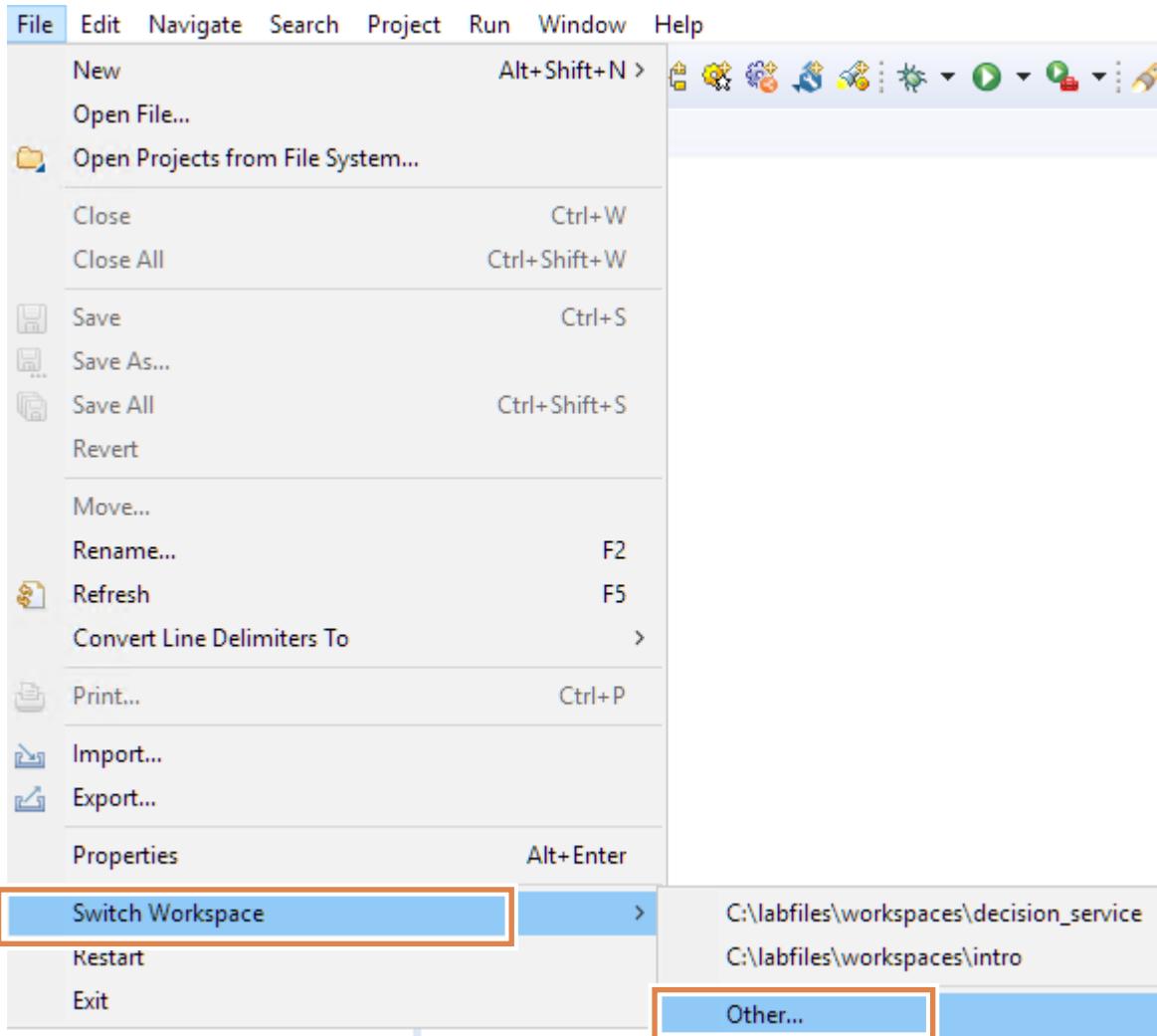


### Note

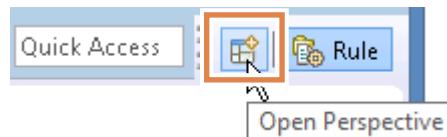
The purpose of this exercise is not to teach you how to create Java classes, but to illustrate the types of tasks that developers are required to do when implementing the XOM.

### 1.1. Setting up your environment for this exercise

- 1. In Rule Designer, switch to a new workspace and name it: `bom`
  - a. From the **File** menu, click **Switch Workspace > Other**.

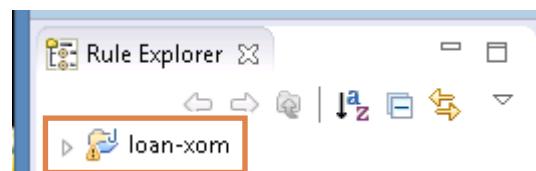


- \_\_\_ b. In the Workspace Launcher window, enter the path:  
C:\labfiles\workspaces\bom
- \_\_\_ c. Click **Launch**.
- \_\_\_ 2. Close the Welcome view.
- \_\_\_ 3. Use the Samples Console to import **Training > Ex 04: Working with BOMs > start**.
  - \_\_\_ a. In the upper-right area of the Rule perspective, click the **Open Perspective** icon.



- \_\_\_ b. In the Open Perspective window, click **Samples Console** and click **Open**.
- \_\_\_ c. In the **Rule Designer** section, expand **Training > Ex 04: Working with BOMs > start**, and click **Import projects**.
- \_\_\_ 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 |
|---------------------------------------------------------------------------------------|---------------|-----------------------|----------|
| <b>Warnings (1 item)</b><br><b>The value of the field Borrower.spouse is not used</b> | Borrower.java | /loan-xom/src/trai... | line 27  |

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.

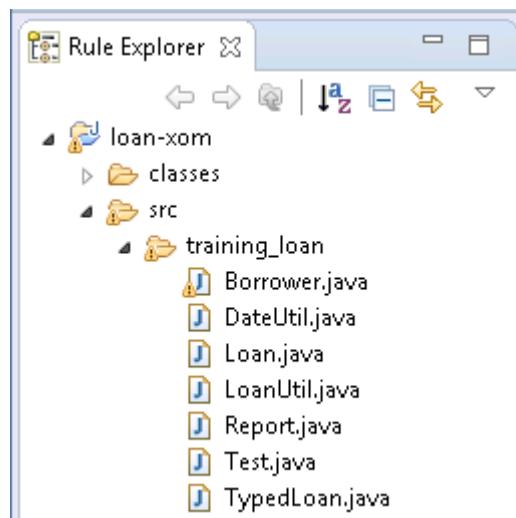
|   | Description                                                 | Resource      | Path                  | Location |
|---|-------------------------------------------------------------|---------------|-----------------------|----------|
| ! | TODO : Uncomment when the getters and setters of t...       | Test.java     | /loan-xom/src/trai... | line 19  |
| ! | TODO : Uncomment when the getters and setters of t...       | Test.java     | /loan-xom/src/trai... | line 35  |
| ! | TODO: Create a getter called getBirthDate for the birt...   | Borrower.java | /loan-xom/src/trai... | line 18  |
| ! | TODO: Create the getters (not the setters) for the follo... | Borrower.java | /loan-xom/src/trai... | line 16  |
| ! | TODO: Create the getters and the setters for the follo...   | Borrower.java | /loan-xom/src/trai... | line 17  |

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**.

The screenshot shows a Java code editor with the file 'Borrower.java' open. The 'Borrower' class name is selected. A context menu is displayed, with the 'Source' submenu expanded. The 'Generate Getters and Setters...' option is highlighted with a blue selection bar.

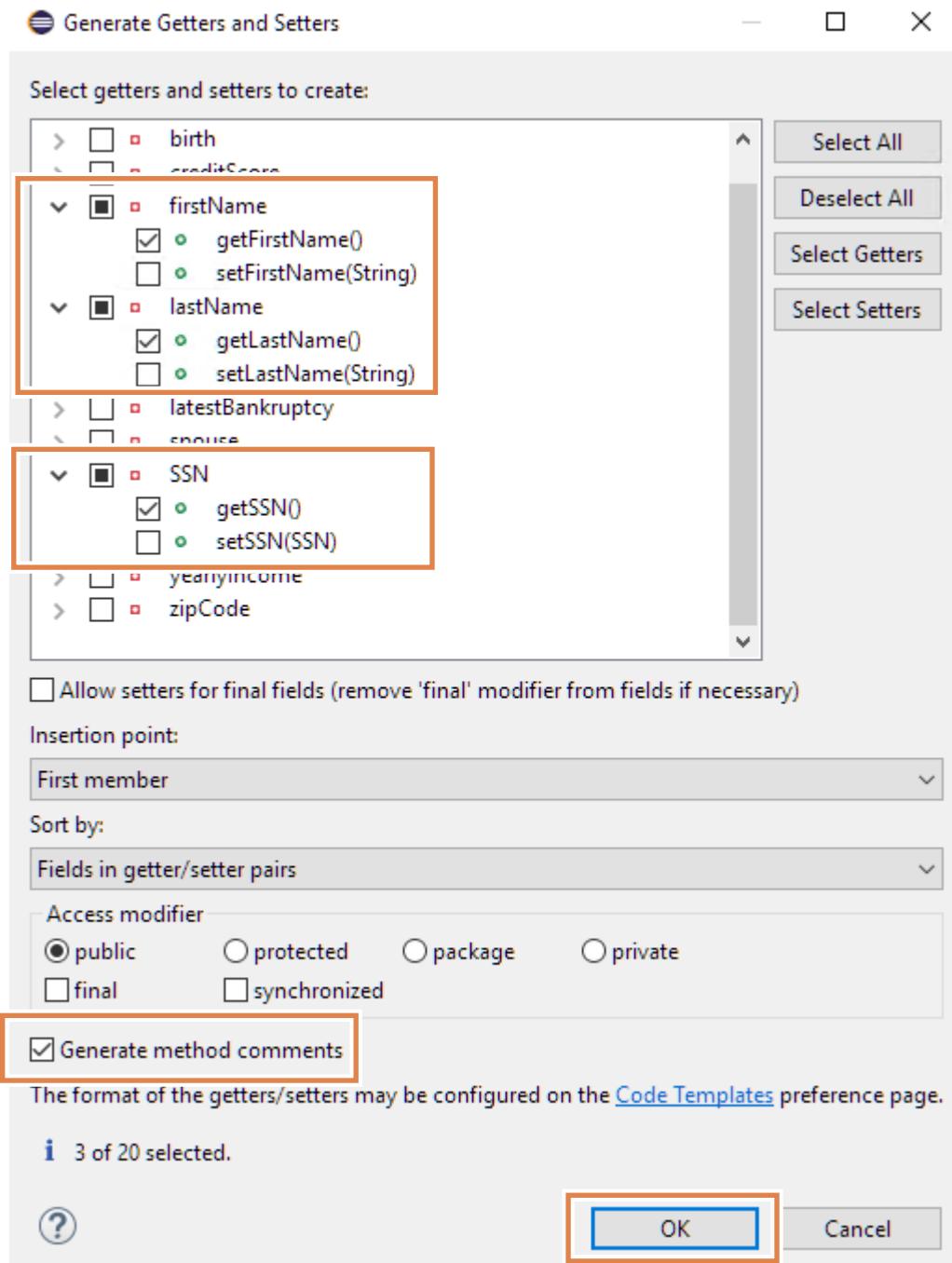
```

2+ * (c) Copyright IBM Corp. 1987, 2017
7
8 package training_loan;
9
10+ import java.util.Calendar
11
12
13 public class Borrower {
14
15+ /**
16 * TODO: Create t
17 * TODO: Create t
18 * TODO: Create a
19 */
20 private static final String
21 private String
22 private String
23 private SSN
24 private int
25 private String
26 private int
27 private Borrower
28 private Calendar
29
30 private Bankruptc
31
32+ @SuppressWarnings("unchecked")
33 private Borrower()

```

|             |        |                                          |
|-------------|--------|------------------------------------------|
| Undo        | Ctrl+Z | Toggle Comment                           |
| Revert File |        | Add Block Comment                        |
| Save        | Ctrl+S | Remove Block Comment                     |
|             |        | Generate Element Comment                 |
|             |        | Correct Indentation                      |
|             |        | Format                                   |
|             |        | Format Element                           |
|             |        | Add Import                               |
|             |        | Organize Imports                         |
|             |        | Sort Members...                          |
|             |        | Clean Up...                              |
|             |        | Override/Implement Methods...            |
|             |        | Generate Getters and Setters... <b>▼</b> |
|             |        | Generate Delegate Methods...             |
|             |        | Generate hashCode() and equals()         |
|             |        | Generate toString()...                   |
|             |        | Generate Constructor using Field...      |
|             |        | Generate Constructors from Super...      |
|             |        | 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.

```
15 /**
16 * @return the firstName
17 */
18 public String getFirstName() {
19 return firstName;
20 }
21
22 /**
23 * @return the lastName
24 */
25 public String getLastName() {
26 return lastName;
27 }
28
29 /**
30 * @return the sSN
31 */
32 public SSN getSSN() {
33 return SSN;
34 }
```



### Note

The new methods were appended where your mouse was positioned in the file.

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

- creditScore
- spouse
- yearlyIncome
- zipCode

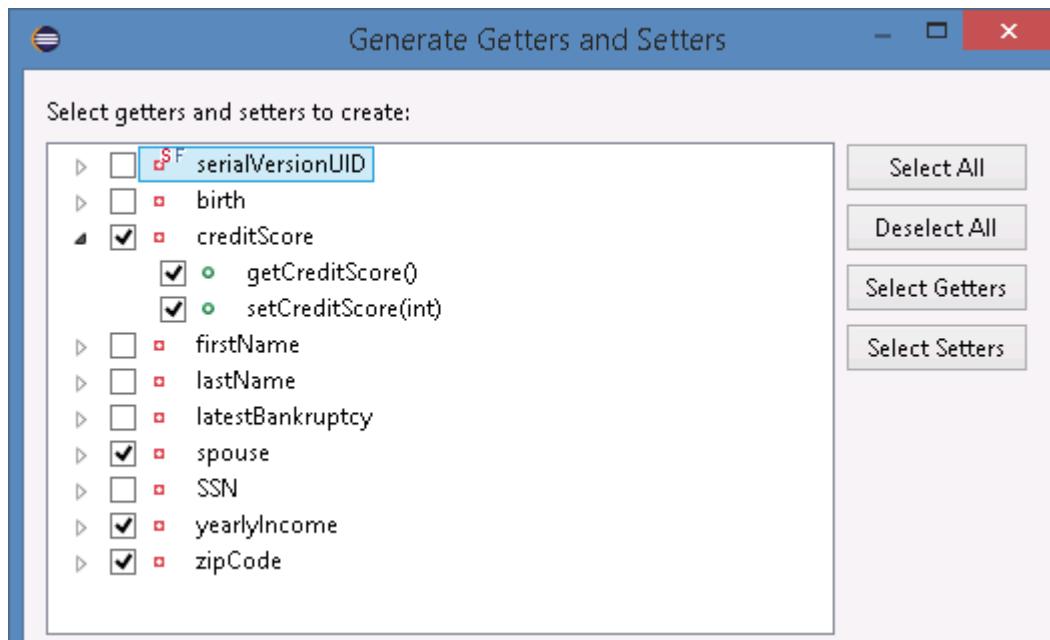
## 1+1=2 Example

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` and close the editing window.

## 1.4. Finish implementing the Test class

- \_\_\_ 1. Edit the **Test.java** file according to the instructions in the `TODO` comments.
  - \_\_\_ a. In Rule Explorer, double-click **Test.java** to open it in the code editor.
  - \_\_\_ a. Find the `TODO` comments and 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());
```

---

```
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(12);
```

---

- \_\_\_ b. Delete the `TODO` comments in the **Test** class.
- \_\_\_ 2. Press **Ctrl+S** to save the **Test** class.
- \_\_\_ 3. Verify that the Problems view and the Tasks view are empty, which indicates that the **loan-xom** Java project compiled successfully.
- \_\_\_ 4. Close the **Test.java** editing window.

## 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.

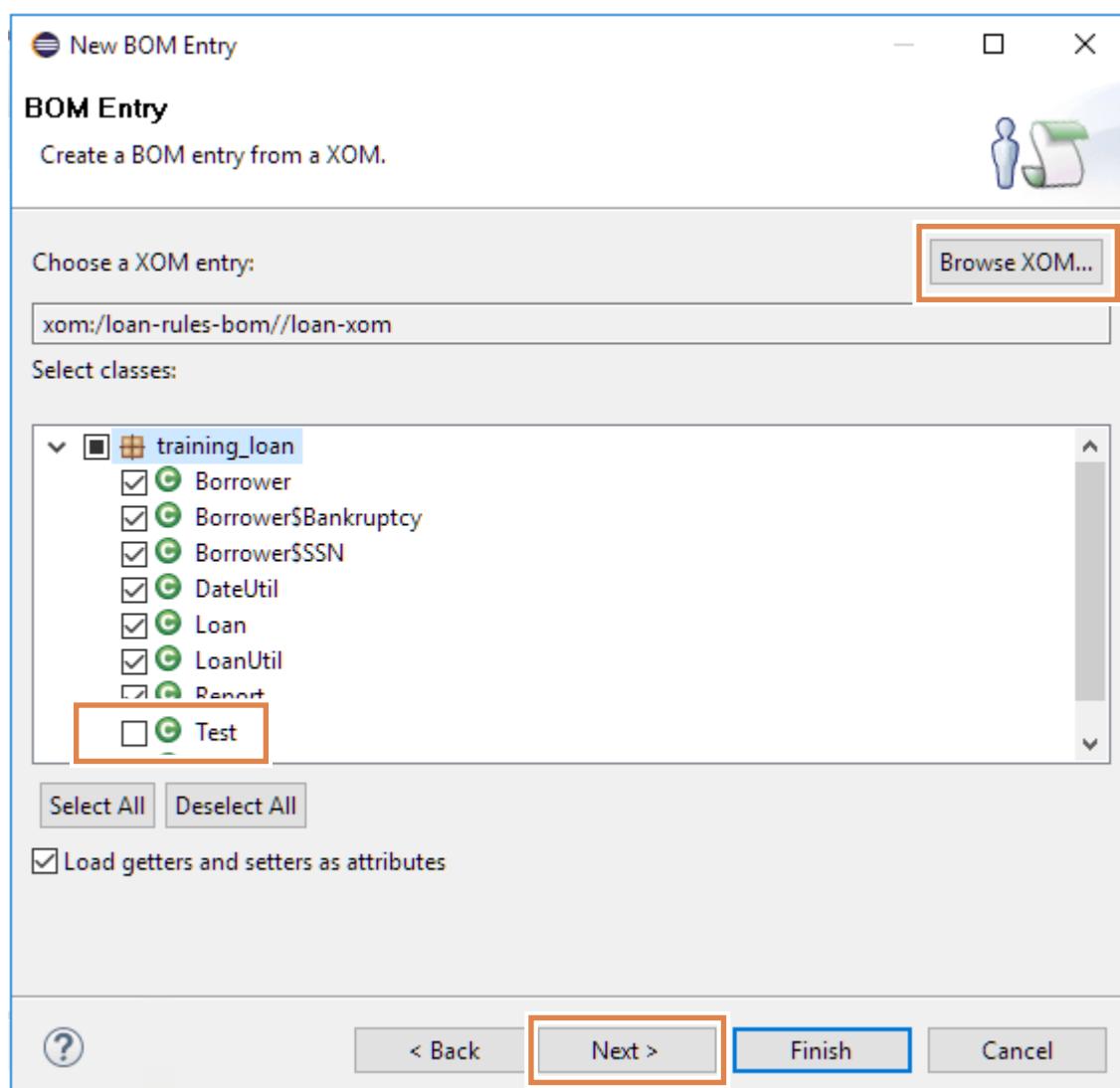
### 2.1. Create a standard rule project with a BOM

- \_\_\_ 1. Right-click anywhere in the Rule Explorer and click **New > Rule Project**.
- \_\_\_ 2. On the “Select a template” page, in the **Decision Service Rule Projects** list, select **Standard Rule Project** and click **Next**.
- \_\_\_ 3. In the **Project name** field, type `loan-rules-bom` and click **Finish**.
- \_\_\_ 4. Create a reference to the `loan-xom` project.
  - \_\_\_ a. Right-click **loan-rules-bom** and click **Properties**.
  - \_\_\_ b. In the Properties pane, click **Java Execution Object Model**, select `loan-xom`, and click **Apply and Close**.

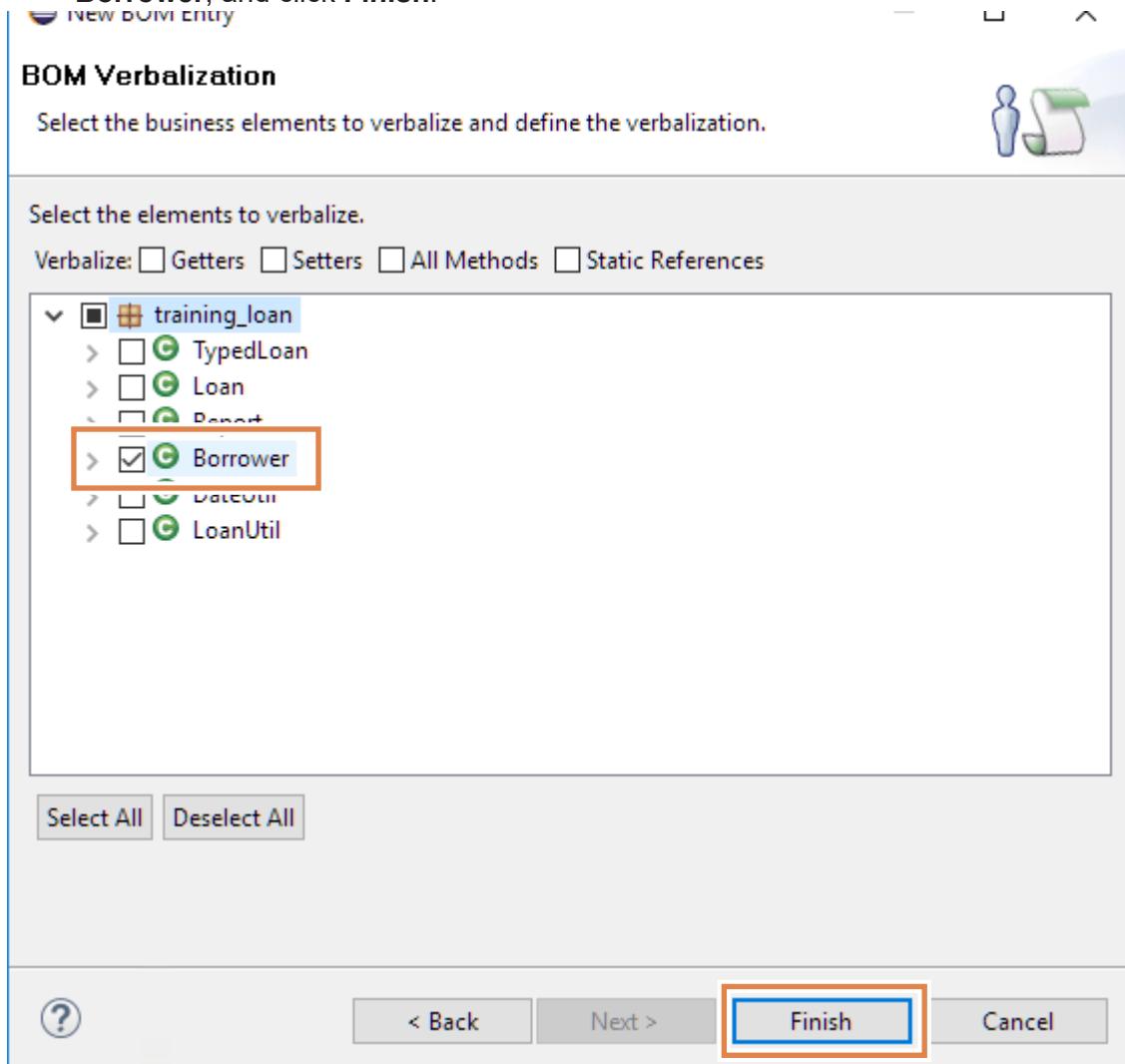
### 2.2. Create a BOM entry in the `loan-rules-bom` project

- \_\_\_ 1. In Rule Explorer, right-click **loan-rules-bom > bom** and click **New > BOM Entry**.
- \_\_\_ 2. 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**.
- \_\_\_ 3. For the **Choose a XOM entry** field, click **Browse XOM**.
- \_\_\_ 4. In the Browse XOM window, select `platform:/loan-xom`, and click **OK**.

5. In the **Select classes** section, expand **training\_loan**, select the check boxes for all classes except **Test**, and click **Next**.



- \_\_\_ c. 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.

- \_\_\_ 6. Save your work (Ctrl+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.



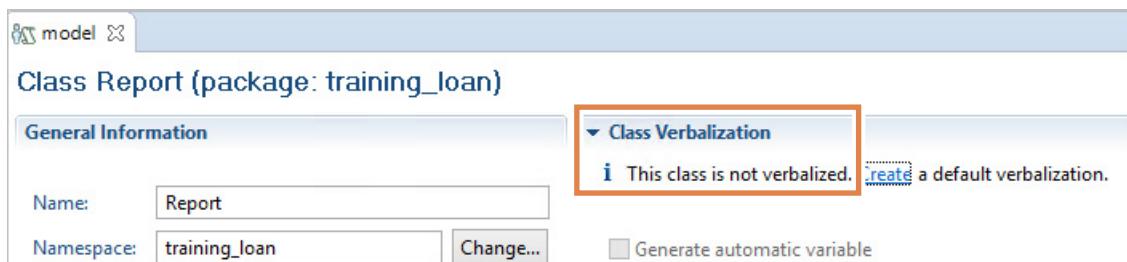
### Requirements

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 that you revise the vocabulary that is associated with the `Borrower` class.

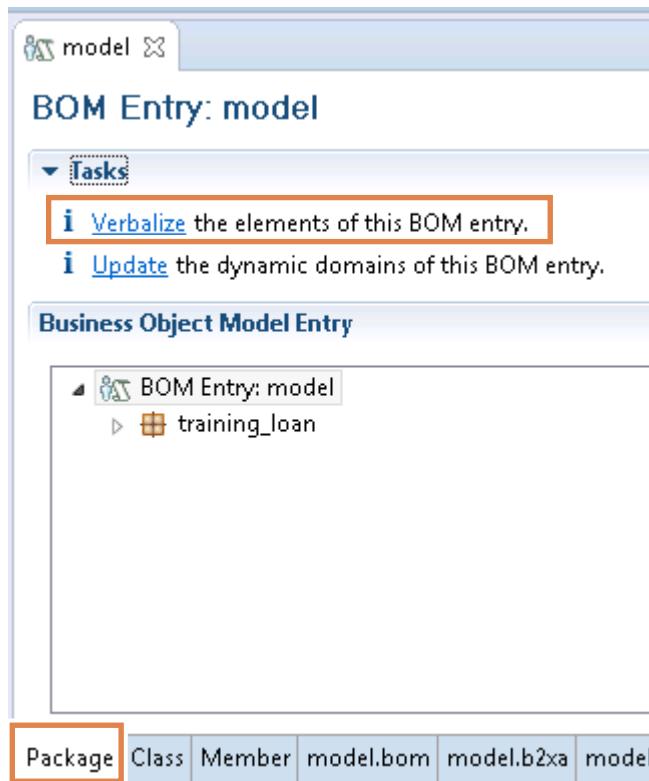
In this section, you learn now how to create or modify the vocabulary in the BOM editor.

### 3.1. Creating the default vocabulary for the Report class

- 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).
- \_\_\_ 4. In the **Business Object Model Entry** section of the **Package** page of the BOM editor, expand **training\_loan** and double-click **Report** to open it in the **Class** page of the BOM editor.

- 5. On the **Class** page, in the **Members** section, double-click **approved** to open it in the Member page and view the default verbalization.

The screenshot shows the 'Members' section of the BOM editor. A list of members is displayed, with 'approved' highlighted by a red box. To the right of the list are buttons for 'New...', 'Delete', and 'Edit'. Below the list is a vertical scrollbar. At the bottom of the window, there is a navigation bar with tabs: 'Package', 'Class' (which is highlighted with a red box), 'Member', 'model.bom', 'model.b2xa', and 'model\_e'.

The default verbalization for the setter of the Boolean `approved` member is:  
`make it {approved} that {this} is approved`

### 3.2. Creating the default vocabulary for the Loan class

- 1. Return to the **Package** page and double-click **Loan** to open it in the **Class** page.  
 — 2. In the **Class Verbalization** section, click **Create a default verbalization**.

The screenshot shows the 'Class Verbalization' section. It contains a note: 'This class is not verbalized. [Create a default verbalization.](#)' This note is highlighted with a red box. Below the note is a checkbox labeled 'Generate automatic variable'.



#### Note

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.

- 3. Create the default verbalization for the `duration` member of the `Loan` class.  
 — a. On the Class page for the `Loan` class, double-click `duration`.

- \_\_\_ b. In the **Member Verbalization** section of the Member page, click **Create a default verbalization**.

The screenshot shows a dialog box titled "Member Verbalization". Inside, there is a warning icon followed by the text "⚠ This member is not verbalized." and a blue link "Create a default verbalization." A red box highlights the "Create a default verbalization" link.

- \_\_\_ c. Notice the default verbalization:

Navigation: "the duration of a loan"  
Template: {duration} of {this}

- \_\_\_ 4. Save your work (Ctrl+S).

### 3.3. Examining the verbalization for members of the Borrower class

- \_\_\_ 1. On the **Package** page of the BOM editor, double-click the `Borrower` class to open it in the **Class** page.  
 \_\_\_ 2. Click **Edit term** in the **Class Verbalization** section.

The screenshot shows a dialog box titled "Class Verbalization". It contains several buttons and links: "Remove the verbalization.", "Edit the documentation.", and "Generate automatic variable". Below these is a "Term:" field containing "borrower" and an "Edit term." button with a pencil icon. A red box highlights the "Edit term." button. At the bottom, there is a tooltip "the borrower, a borrower, the borrowers....".

The Edit Term window opens.

The screenshot shows the "Edit Term" window for the term "borrower". The window has a title bar "Edit Term" and a toolbar with a help icon and a save icon. The main area is titled "Term: borrower" and contains a tooltip "the borrower, a borrower, the borrowers...". Below this are sections for "Singular" (set to "borrower") and "Plural" (set to "borrowers"). There are also sections for "Definite article" (checkboxes for "Singular" and "Plural" both checked, with "the" in the respective fields) and "Indefinite article" (checkboxes for "Singular" and "Plural" both checked, with "a" in the "Singular" field). At the bottom are "OK" and "Cancel" buttons.

Notice that you can edit the `borrower` term to set the vocabulary for singular and plural forms, and for the definite and indefinite articles.

- \_\_\_ 3. Click **Cancel** to close the Edit Term window.
- \_\_\_ 4. On the **Class** page, 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 `hasLatestBankruptcy` method name. As a result, the name and the default verbalization of the `hasLatestBankruptcy` method in the BOM also have the same incorrect spelling.

- \_\_\_ 5. 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: `{this}.setLatestBankruptcy({0},{1},{2})`





## 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.

| Name | Type             | D         |
|------|------------------|-----------|
| arg1 | java.util.Date   | Add...    |
| arg2 | int              | Remove... |
| arg3 | java.lang.String | Up        |
|      |                  | Down      |
|      |                  | Edit...   |

## 3.4. 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. From the Class page of the BOM editor, double-click the `setLatestBankruptcy` method.
  - \_\_\_ 3. Change the verbalization in the **Template** field to match the following text:  
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.
  - \_\_\_ 4. Save your work.
  - \_\_\_ 5. Close the BOM editor.

**End of exercise**

## Exercise review and wrap-up

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 5. Refactoring

## Estimated time

00:45

## Overview

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

## Objectives

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 support files, which are installed in the `<InstallDir>\studio\training` directory:

- Start project: Dev 05 – Refactoring\start

## 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 and name it: `refactor`
  - \_\_\_ a. From the **File** menu, click **Switch Workspace > Other**.
  - \_\_\_ b. In the Workspace Launcher window, enter the path:  
`C:\labfiles\workspaces\refactor`
  - \_\_\_ c. Click **Launch**.
- \_\_\_ 2. Close the Welcome view.
- \_\_\_ 3. Use the Samples Console to import the start projects for the exercise.
  - \_\_\_ a. Click the **Open Perspective** icon.
  - \_\_\_ b. In the Open Perspective window, click **Samples Console** and click **Open**.
  - \_\_\_ c. In the **Rule Designer** section, expand **Training > Ex 05: Refactoring > start**, and click **Import projects**.
- \_\_\_ 4. 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 **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 **loan-rules > rules** folder.
  - \_\_\_ a. In Rule Explorer, 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` condition 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 “bankruptcy age” term to indicate the *latest* bankruptcy.

### 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.

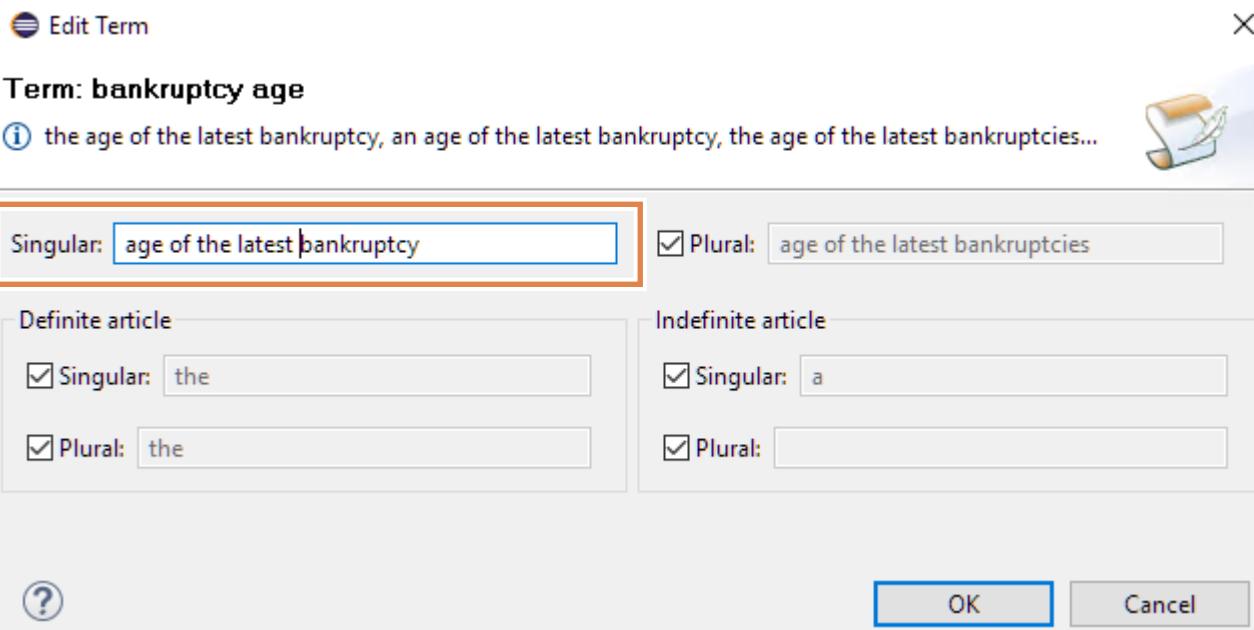
- 1. Edit the `getBankruptcyAge` BOM member verbalization to become: age of the latest bankruptcy
  - a. Switch back to the **model** tab in the editing window.
  - b. Click the **Edit the subject used in phrases** link in the **Member Verbalization** section of the `getBankruptcyAge` method.

**Member getBankruptcyAge (class: training\_loan.Borrower)**

|                                     |                                              |                                                        |                            |
|-------------------------------------|----------------------------------------------|--------------------------------------------------------|----------------------------|
| <b>General Information</b>          |                                              | <b>Member Verbalization</b>                            |                            |
| Name:                               | <code>getBankruptcyAge</code>                | <a href="#">Remove</a> the verbalization.              |                            |
| Type:                               | <code>int</code>                             | <a href="#">Create a navigation phrase.</a>            |                            |
| Class:                              | <code>training_loan.Borrower</code>          | <a href="#">Edit</a> the subject used in phrases.      |                            |
| <input type="checkbox"/> Static     | <input type="checkbox"/> Final               | <b>Navigation : "the bankruptcy age of a borrower"</b> |                            |
| <input type="checkbox"/> Deprecated | <input type="checkbox"/> Update object state | Template:                                              | {bankruptcy age} of {this} |

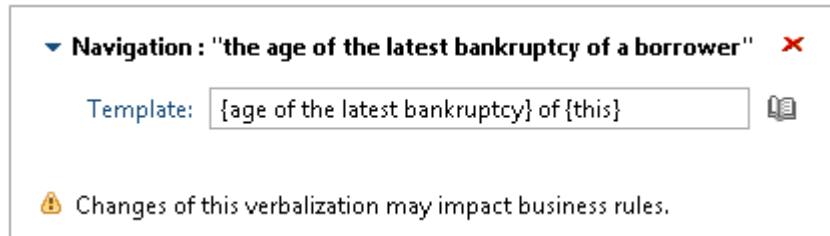
- \_\_\_ c. In the **Singular** field of the Edit Term window, enter:

age of the latest bankruptcy



- \_\_\_ d. Click **OK**.

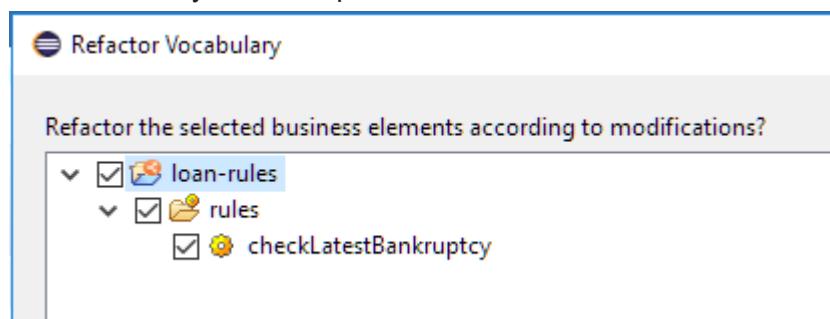
Note the changes to the navigation phrase and the template of Borrower.getBankruptcyAge.



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, click **Select All**, 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, in the **rules** folder, double-click the **checkDuration** rule.

The **if** statement checks whether the duration is between 1 and 30.



### Requirements

The business analysts request that you change the navigation template for the `duration` member of the `Loan` class should indicate that the duration is in years.

- \_\_\_ 2. In Rule Explorer, expand **bom > model > training\_Loan > Loan**, double-click the `duration` BOM member, and look at the current verbalization.
- \_\_\_ 3. Edit the **Template** field in the **Navigation** section by inserting `(in years)` after `{duration}`  
`{duration} (in years) of {this}`

The Navigation and Template phrases are updated.

▼ Navigation : "the duration (in years) of a loan"
✖

Template: `{duration} (in years) of {this}`
✖

- \_\_\_ 4. Save the BOM (Ctrl+S).

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

- \_\_\_ 5. In the Refactor Vocabulary window, make sure that **Select All** is selected and click **Yes**.
- \_\_\_ 6. Open the `checkDuration` rule to see how it was modified.
- \_\_\_ 7. Close the editors for the `checkLatestBankruptcy` and `checkDuration` rules.

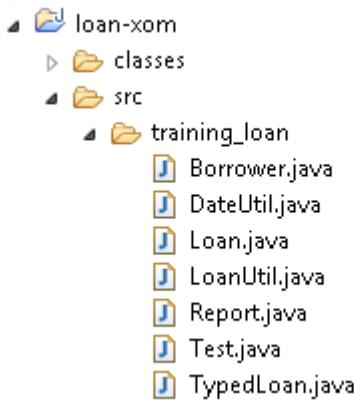
## Section 2. Maintaining consistency between the BOM and the XOM

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

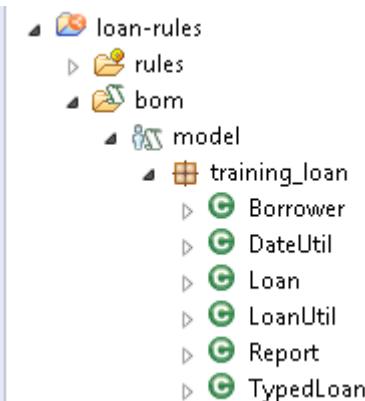
### 2.1. Adding a XOM class that is missing in the BOM

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` to see 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**.

The screenshot shows the BOM Update view in a software interface. At the top, there are tabs for Rule Project M..., Rule Execution..., Problems, Tasks, Testing and Si.., and BOM Update. The BOM Update tab is highlighted with a red box. Below the tabs, a message says: "⚠ Differences have been detected between XOM and BOM classes. Resolve using the actions below." Two tree views are displayed side-by-side. The left tree, labeled "XOM", contains nodes for XOM Root, training\_loan (with Borrower, Bankruptcy, SSN, DateUtil, Loan, LoanUtil, Report, Test), and a few others. The right tree, labeled "BOM", contains nodes for BOM Entry: model, training\_loan (with Borrower, Bankruptcy, SSN, DateUtil, Loan, LoanUtil, Report, TypedLoan), and a few others. A vertical scrollbar is positioned between the two trees. Below the trees, a message says "No action performed." At the bottom, a button labeled "Differences and Actions" is highlighted with a blue box.

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 XOM class `training_loan.Test`.

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

The screenshot shows the "Differences and Actions" dialog. At the top, there is a dropdown menu labeled "Actions:" with the option "Import the XOM class "training\_loan.Test"" selected. To the right of the dropdown is a "Perform and save" button, which is highlighted with a red box. Below the dropdown, a message says "Perform action for your selection in the trees, or select a difference in the table below to perform an action to solve this." A table is shown with the following data:

| Origin | Class Name                      | Type             | Description                                      |
|--------|---------------------------------|------------------|--------------------------------------------------|
| XOM    | <code>training_loan.Test</code> | Missing from BOM | XOM class "training_loan.Test" not found in BOM. |

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



### Requirements

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. You must also 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. Click the **Open Perspective** icon.



— b. Select **Java (default)** from the perspectives list, and click **Open**.

— 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** ( ) in the upper-right area of the pane.



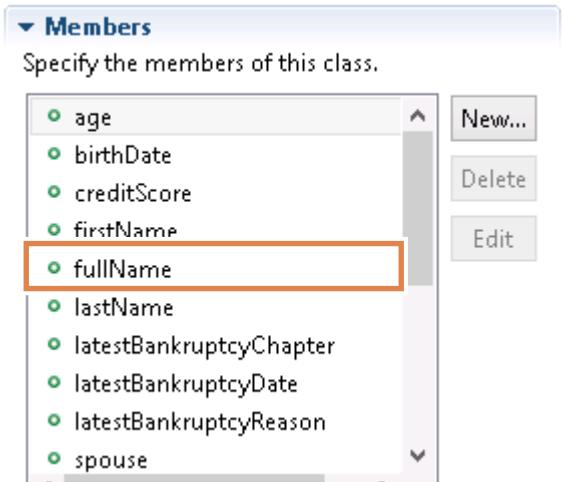
— b. Expand **Differences and Actions** and notice the new line that states that the XOM attribute `training_loan.Borrower.fullName` was not found in the `training_loan.Borrower` BOM class.

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

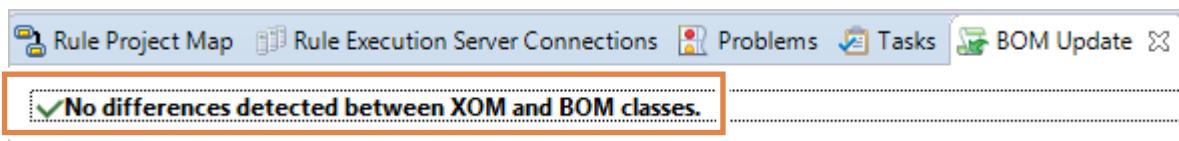
Update the BOM class "training\_loan.Borrower"

- 7. 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`.



- 8. 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

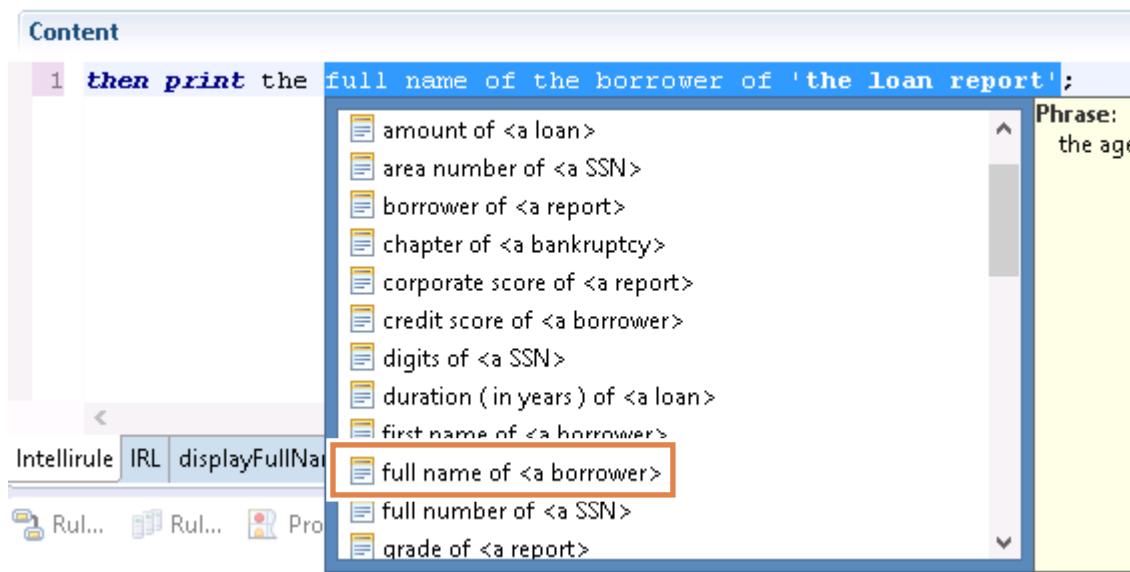
- 1. Add an action rule to the `rules` folder of the `loan-rules` project:
- Right-click the `rules` folder, and click **New > Action Rule**.
  - In the **Name** field, type: `displayFullName`
  - Click **Finish**.

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 5-8.
  - \_\_\_ a. In Rule Explorer, expand `loan-xom > src > training_loan`, and double-click the `Borrower.java` file.
  - \_\_\_ b. 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"`

The screenshot shows the 'Rule Project Map' interface with the 'Problems' tab selected. A message indicates '1 error, 0 warnings, 0 others'. The details pane shows an error entry: '[B2X] GBREX0021E: Cannot find attribute 'fullName' in execution class 'training\_loan.Borrower'' associated with '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.

The screenshot shows the 'Differences and Actions' view. It lists three actions for the 'Deprecate the BOM attribute "training\_loan.Borrower.fullName"' difference:

- Actions: Deprecate the BOM attribute "training\_loan.Borrower.fullName"
- Perform action: Delete the BOM attribute "training\_loan.Borrower.fullName"
- Deprecate the BOM attribute "training\_loan.Borrower.fullName"

The third option is currently selected. The 'Origin' column shows 'XOM' and 'Modified' for the 'BOM attribute "training\_loan.Borrower.fullName"'.

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**.



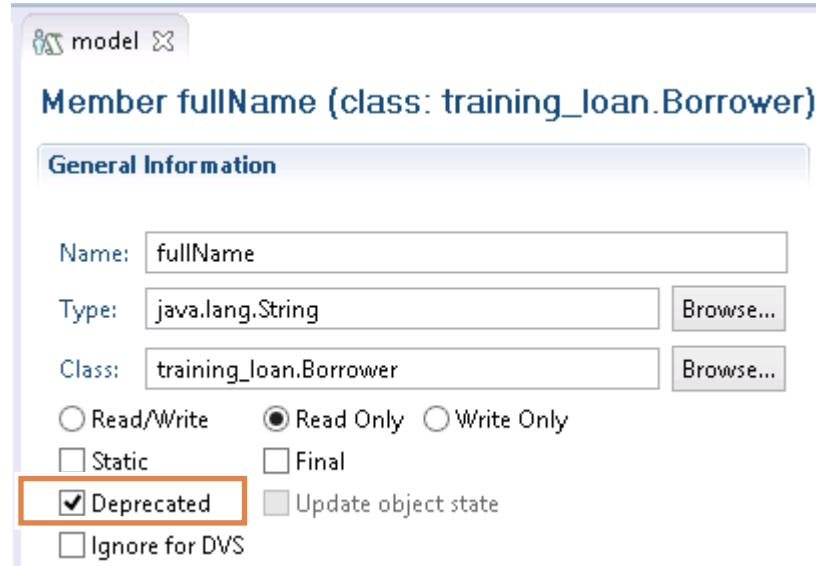
### Information

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.



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

- \_\_\_ 7. View the deprecation result in the `displayFullName` rule.
- Expand **loan-rules > rules**.
  - Reopen the `displayFullName` action rule.
  - Hover your mouse over the **Warning** icon.



- \_\_\_ 8. View the result in the Problems view.
- Notice the error:  
Cannot find attribute "fullName" in execution class  
`"training_loan.Borrower"`
  - Notice the warning messages and the resources to which they apply:
    - The deprecated message 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.

---

# Exercise 6. Working with ruleflows

## Estimated time

00:45

## Overview

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

## Objectives

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 06 - Ruleflows\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 and name it: `ruleflow`
  - \_\_\_ a. From the **File** menu, click **Switch Workspace > Other**.
  - \_\_\_ b. In the Workspace Launcher window, enter the path:  
`C:\labfiles\workspaces\ruleflow`
  - \_\_\_ c. Click **Launch**.
- \_\_\_ 2. Close the Welcome view.
- \_\_\_ 3. Use the Samples Console to import **Training > Ex 06: Ruleflows > start**.
  - \_\_\_ a. Click the **Open Perspective** icon.
  - \_\_\_ b. In the Open Perspective window, click **Samples Console** and click **Open**.
  - \_\_\_ c. In the Rule Designer section of the “Samples and Tutorials” page, expand **Training > Ex 06: Ruleflows > start**, and click **Import projects**.
- \_\_\_ 4. When the workspace finishes building, close the Help view.

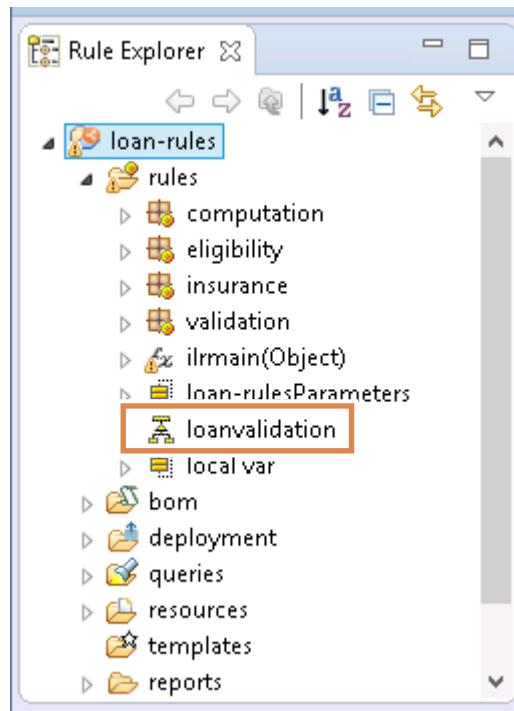


#### Note

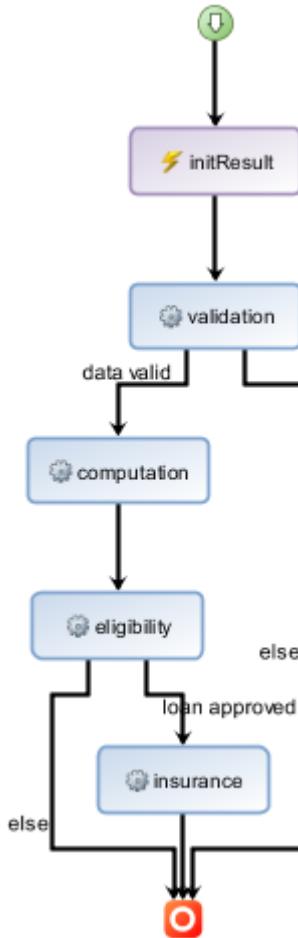
Ignore any warnings that appear on the project.

## 1.2. Exploring the ruleflow

- 1. In Rule Explorer, expand **loan-rules > rules** 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

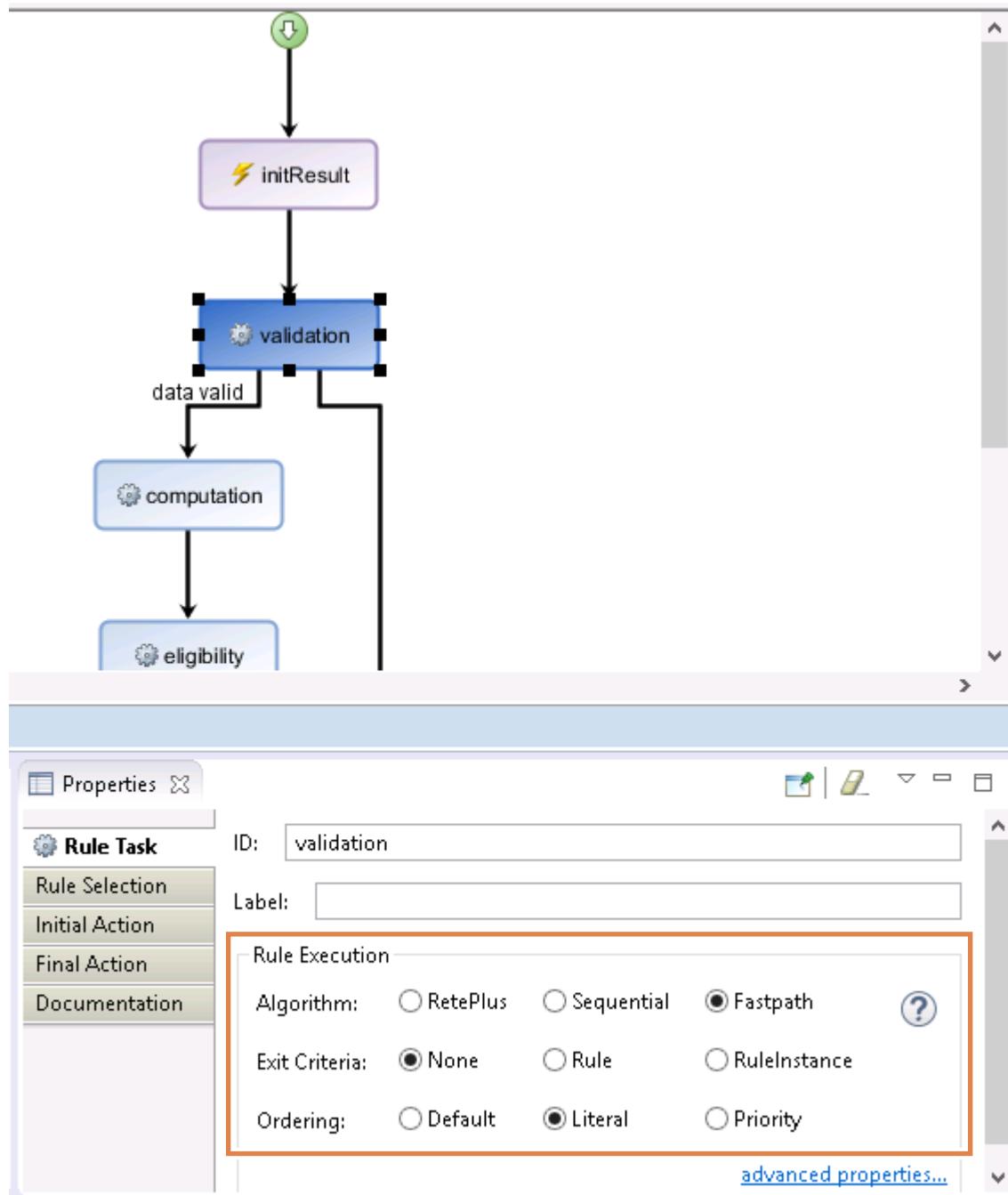
The ruleflow organizes the flow of rule execution in a ruleset to produce a decision.

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.

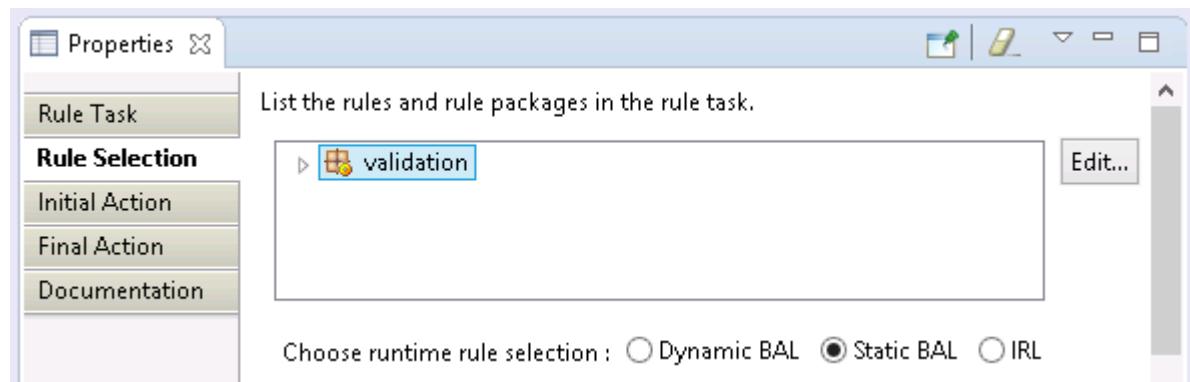
Transitions between tasks define the path through the ruleflow. You set conditions on transitions to define which path to take according to the results of evaluating a rule task.

3. Double-click the **validation** rule task in the diagram to open 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.

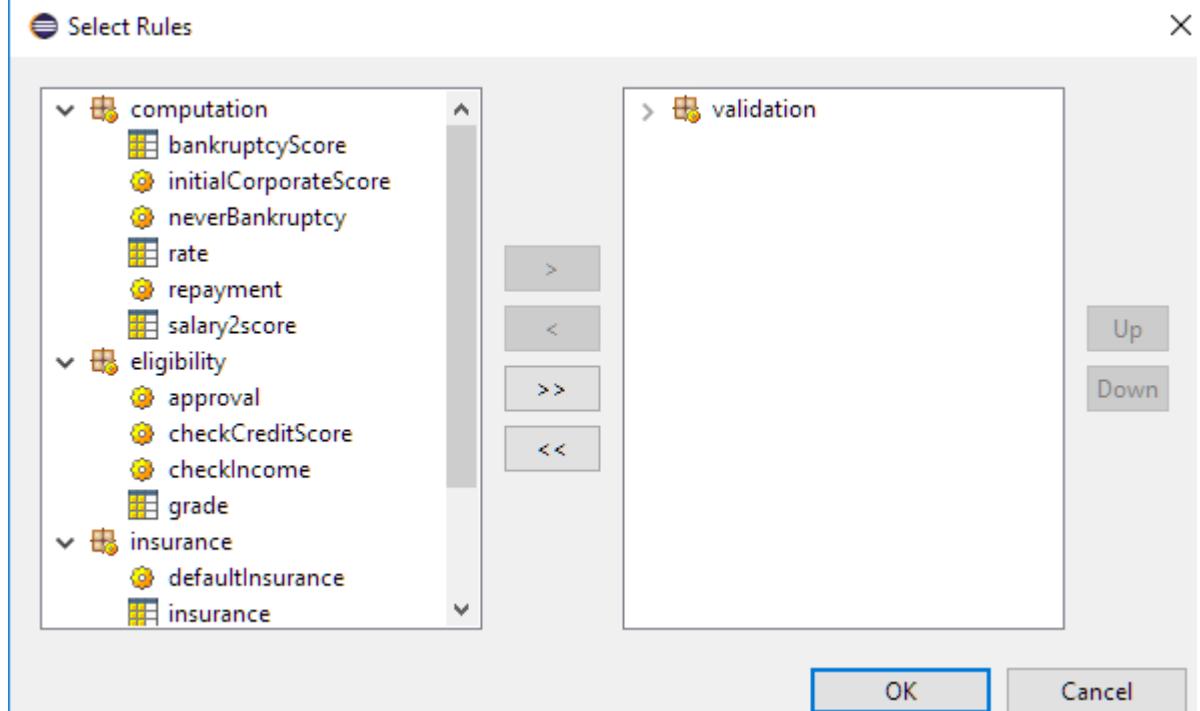


These execution properties dictate how instances of the rules in this rule task are executed.

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



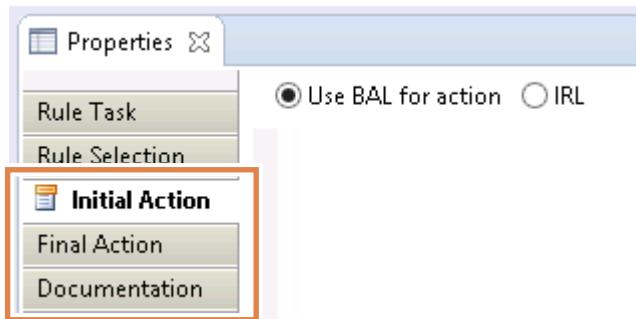
- \_\_\_ c. On the **Rule Selection** tab of the Properties view, click **Edit** to open the Select Rules window.
- \_\_\_ d. Take some time and experiment with selecting which rules to include in this task.



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.

### 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 parameters and variables.



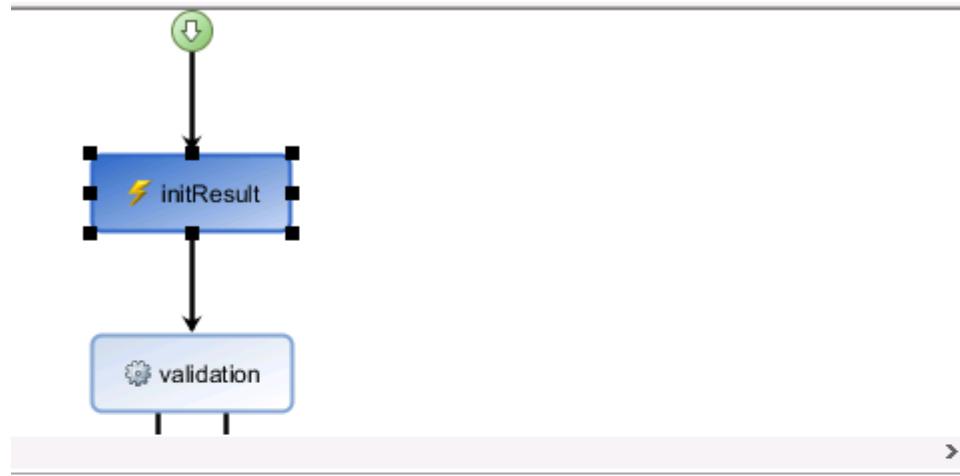
#### Questions

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 parameter.



## Questions

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

## Answer

You do not have to set the initial value of the `loanApproved` variable in the ruleflow. This initial value is already given as part of the 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?

---



---

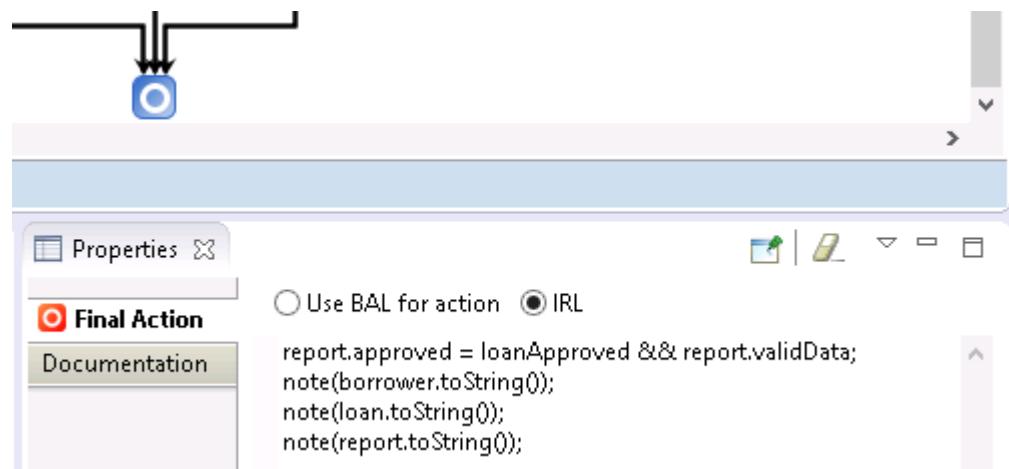
## Answer

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 parameter, a Boolean 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 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. Also, take time to explore 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` parameter is set according to the latest values of the `loanApproved` variable and the `validData` attribute of the `report` 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 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 parameters and the variables 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` variable and the `report` 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` variable, which is verbalized as:  
'the loan is approved'
  - These action rules use the `report` parameter, which is verbalized as:  
'the loan report'
- 
- 3. Close the rule editor windows for these rules.

## Section 2. Creating a ruleflow

In this section, you re-create parts of the `loanvalidation` ruleflow in the Ruleflow editor.

### 2.1. Create the ruleflow

- \_\_\_ 1. Right-click the `rules` folder in the `loan-rules` project and click **New > Ruleflow**.
- \_\_\_ 2. 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.



#### 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.

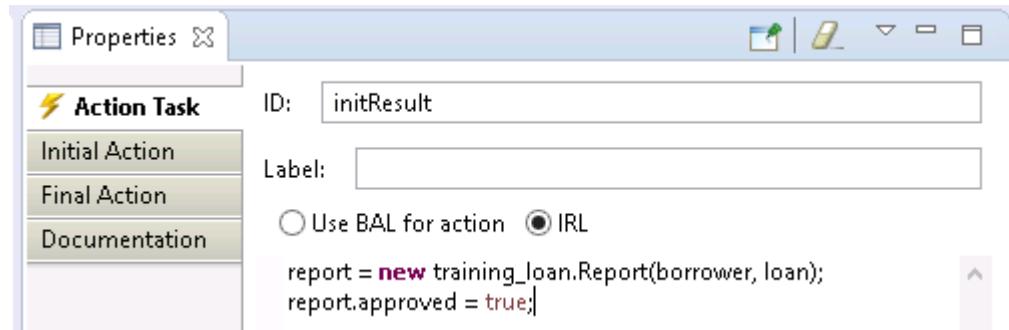
- \_\_\_ 3. Add the start node of the `loanvalidation_2` ruleflow.
  - \_\_\_ a. Click  **Create a start node**.
  - \_\_\_ b. In the Ruleflow editor main area, click anywhere on the canvas to place the node.
- \_\_\_ 4. Add the end node of the `loanvalidation_2` ruleflow.
  - \_\_\_ a. Click  **Create an end node**.
  - \_\_\_ b. In the Ruleflow editor main area, click anywhere on the canvas to place the node.

### Create an action task

- \_\_\_ 1. 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 anywhere on the canvas to place 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`

- \_\_\_ e. Select the **IRL** option and enter the following IRL function code:

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



- \_\_\_ 2. Save your work (Ctrl+S).



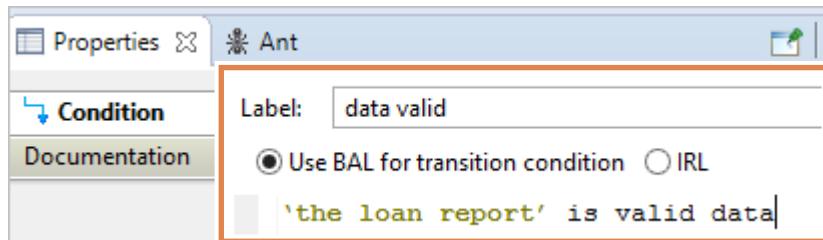
### Troubleshooting

You can ignore the errors that you see.

## 2.2. Define the rule tasks

- \_\_\_ 1. Create the validation, computation, eligibility, and insurance rule tasks by dragging each of those rule packages from Rule Explorer onto the Ruleflow editor.
- \_\_\_ 2. 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**
    - o Algorithm: **RetePlus**
    - o Exit Criteria: **None**
    - o Ordering: **Default**
- 3. 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.
- 4. Disable the transition creation mode in the Ruleflow editor palette, by clicking  **Create a transition** again.
- 5. To adjust the ruleflow diagram layout, go to the Ruleflow editor toolbar and click the  **Layout All Nodes** icon.
- 6. 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, the **Condition** tab should be open, and in the **Label** field, enter: `data valid`
  - c. Select **Use BAL for transition condition**.
  - d. In the condition editor, enter: 'the loan report' is valid data



Notice that the transition from `validation` to the `end node` is automatically labeled `else`.



## Troubleshooting

If you find it difficult to select transitions and other ruleflow elements for editing in the Properties window, try closing the ruleflow and reopening it in the ruleflow editor.

- \_\_\_ 7. Save your work.
  - \_\_\_ 8. 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.

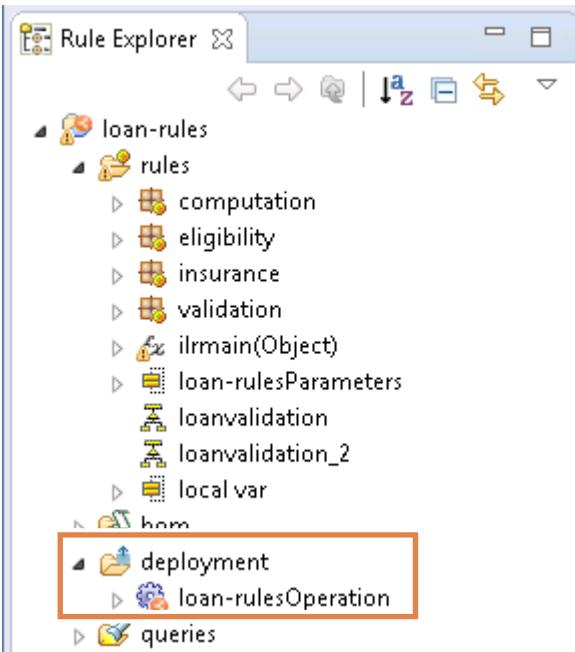
- \_\_\_ 1. Add a transition between the `eligibility` rule task and the end node.
  - \_\_\_ 2. Add a label and a condition to the transition between the `eligibility` and `insurance` tasks.
    - \_\_\_ a. Select the transition between the `eligibility` and `insurance` rule tasks.
    - \_\_\_ b. On the **Condition** tab, in the **Label** field, enter: `loan approved`
    - \_\_\_ c. Select **Use BAL for transition condition**.
    - \_\_\_ d. 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-rulesOperation** 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**
- **Do not use a ruleflow**

In this exercise, the loanvalidation ruleflow is already selected as the main ruleflow.

**Signature**

Define the input and output parameters for the ruleset.

|                 |          |
|-----------------|----------|
| Input:          | borrower |
| Input - output: | loan     |
| Output:         | report   |

**Ruleset Name**

Enter the name of the ruleset part of the ruleset path that is called by Rule Execution Server.

Ruleset name:

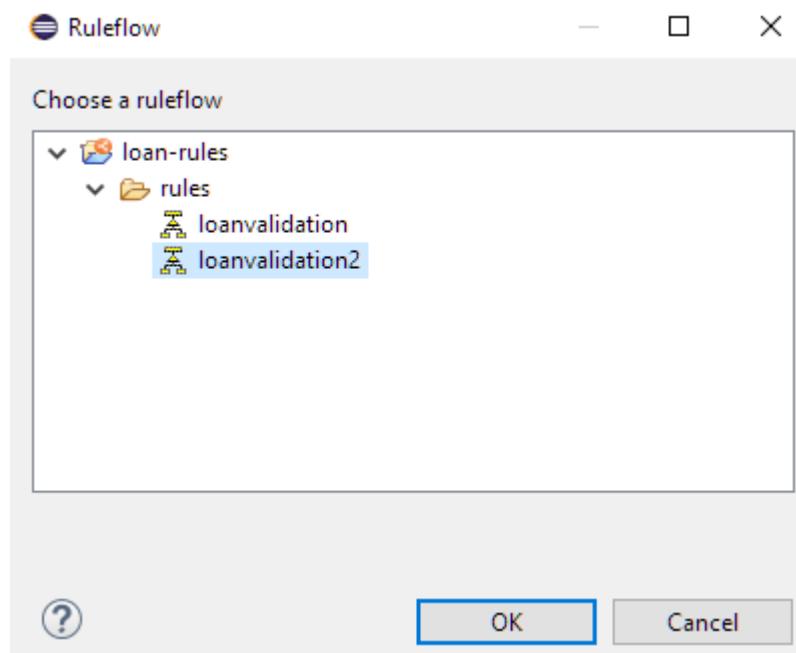
**Ruleflow**

Define the main ruleflow that is used as the entry point for execution.

Use main ruleflow: [loanvalidation](#)

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.

**Ruleflow**

Define the main ruleflow that is used as the entry point for execution.

Use main ruleflow: [loanvalidation\\_2](#)

Do not use a 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 7. Exploring action rules

## Estimated time

00:30

## Overview

In this exercise, you learn how to write action rules.

## Objectives

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 support files, which are installed in the `<InstallDir>\studio\training` directory:

- Start project: Dev 07 – Exploring rules\start

## 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 and name it: `explore`
  - \_\_\_ a. From the **File** menu, click **Switch Workspace > Other**.
  - \_\_\_ b. In the Workspace Launcher window, enter the path:  
`C:\labfiles\workspaces\explore`
  - \_\_\_ c. Click **Launch**.
- \_\_\_ 2. Close the Welcome view.
- \_\_\_ 3. Use the Samples Console to import **Training > Ex 07: Exploring rules > start**.
  - \_\_\_ a. Click the **Open Perspective** icon.
  - \_\_\_ b. In the Open Perspective window, click **Samples Console** and click **Open**.
  - \_\_\_ c. In the **Rule Designer** section, expand **Training > Ex 06: Exploring rules > start**, and click **Import projects**.
- \_\_\_ 4. When the workspace finishes building, close the Help view.



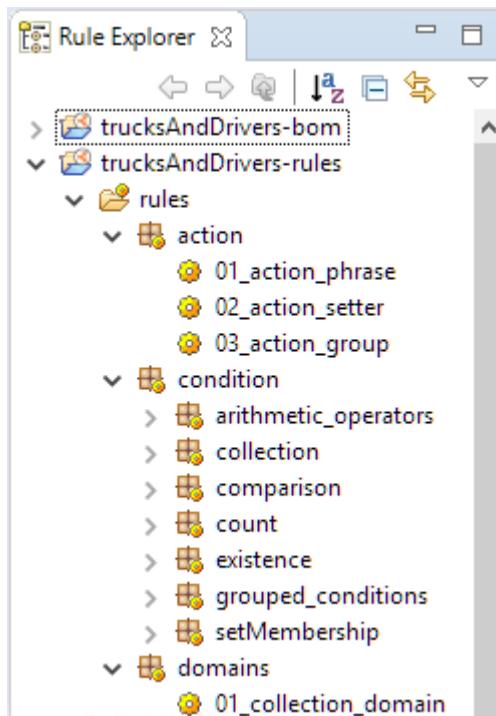
#### Note

Ignore any warnings that appear on the project.

## 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**: Action rules in this rule package show how to use operators that count the number of occurrences of an object.
  - **existence**: 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.
  - **setMembership**: 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:

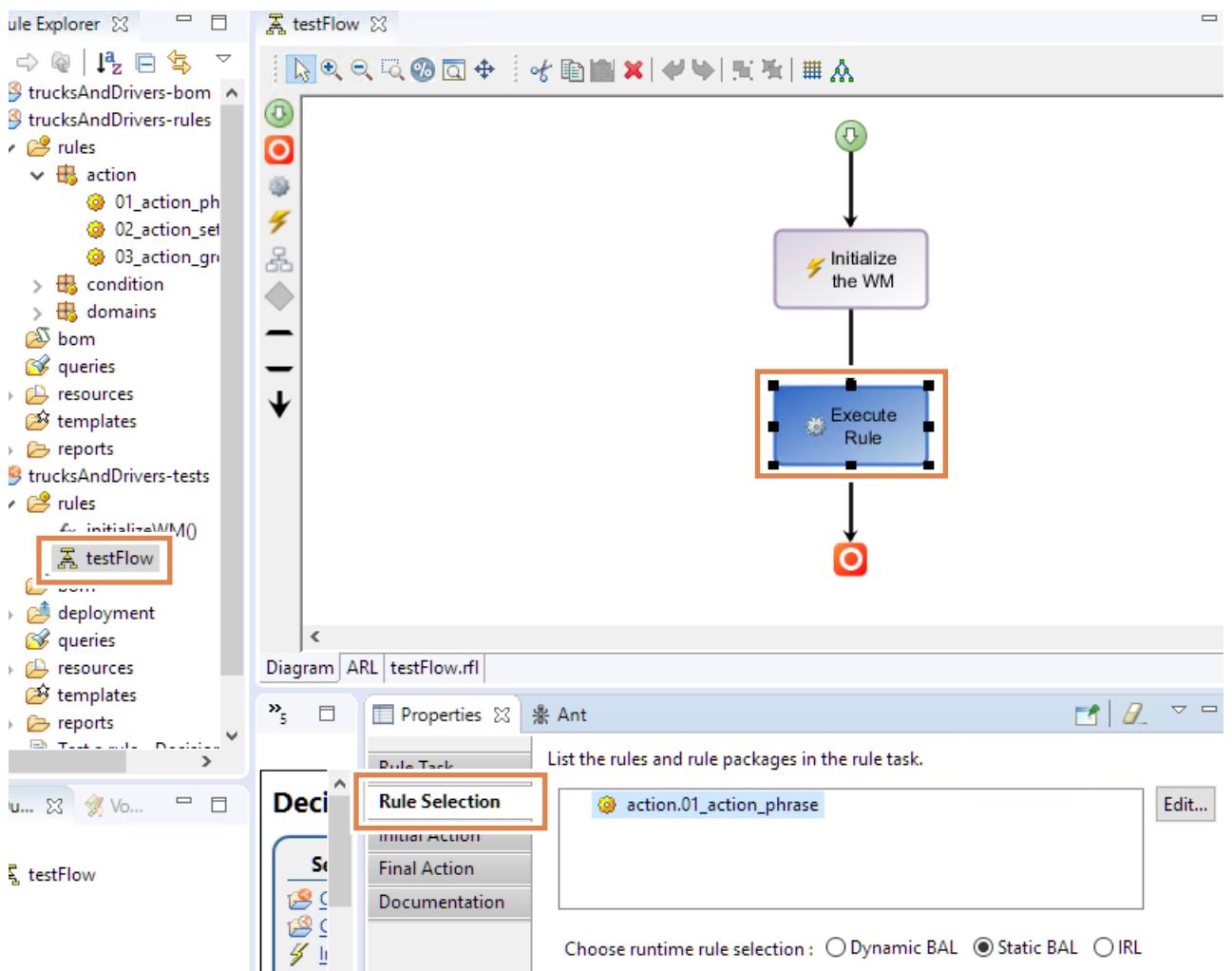
- Can you identify the objects on which the action rules work?
  - Are there any objects for which you cannot identify the origin?
  - Based on your experience with Java programming, can you determine what the terms *definitions*, *conditions*, *actions*, and *variables* mean in relation with the design 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.

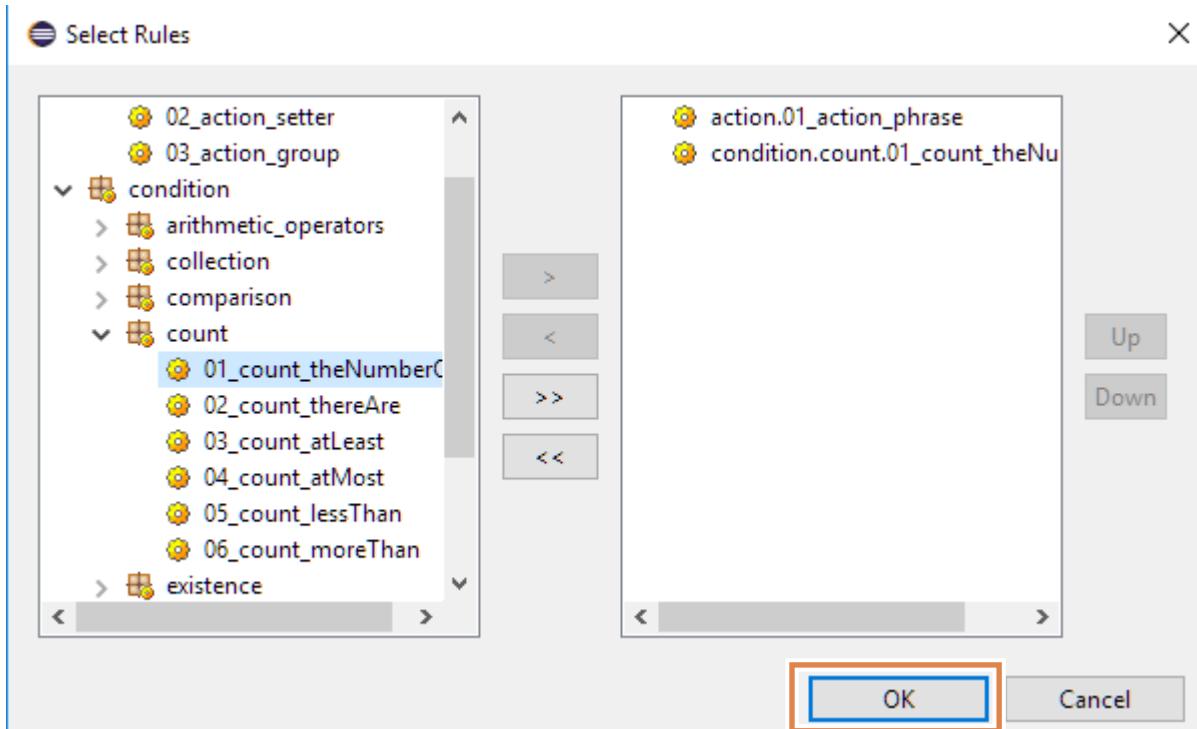
### 2.1. Choose a rule to test

- \_\_\_ 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, double-click the **Execute Rule** rule task to 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 to see the available rules.

- \_\_ b. Expand **condition > count**, and double-click **01\_count\_theNumberOf** to add that rule to the task.
- \_\_ c. Click **OK**.

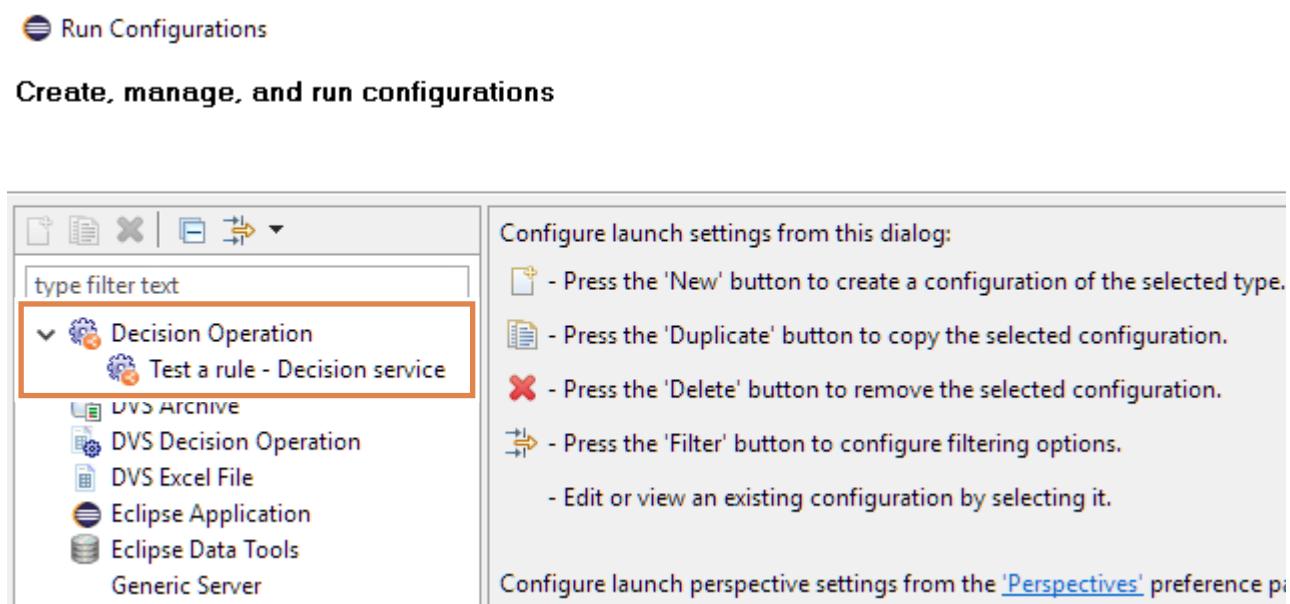


- \_\_ 6. Save your work (Ctrl+S).

## 2.2. Test the rule

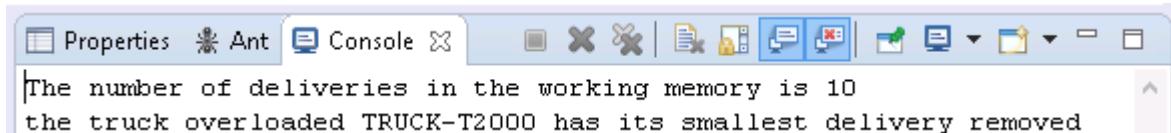
- \_\_ 1. From the **Run** menu, click **Run Configurations** to set up rule execution.

2. In the Run Configurations window, expand **Decision Operation** and click **Test a rule - Decision service**.



3. 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.



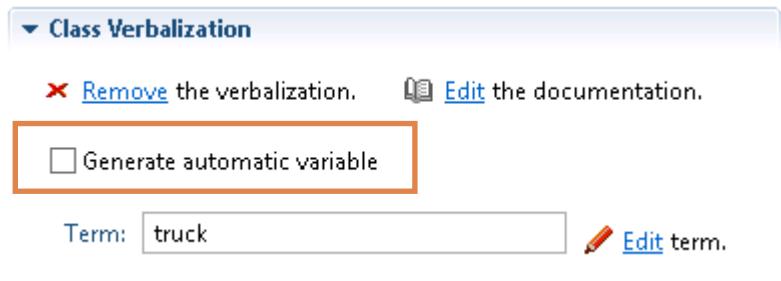
4. Try selecting different rules and running them, one at a time, to view their behavior.  
 5. After you finish testing, remove the rules that you added, so that only the 01\_action\_phrase rule remains.

## 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.

### 3.1. Create an automatic variable

- \_\_\_ 1. In Rule Explorer, expand **trucksAndDrivers-bom > bom > model > drivers** and double-click **Truck** to edit it in the BOM editor.
- \_\_\_ 2. In the **Class Verbalization** section, select **Generate automatic variable**.



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.

The screenshot shows the IBM Rational Rules IDE interface. On the left is the Rule Explorer tree, which includes sections for 'trucksAndDrivers-bom' (containing 'rules', 'bom', and 'model' with sub-nodes like 'drivers', 'feeder', 'queries', 'resources', 'templates', 'reports'), and 'trucksAndDrivers-rules' (containing 'rules' with sub-nodes like '01\_action\_phrase', '02\_action\_setter', etc.). The main workspace shows the 'testFlow' project with the 'model' tab selected. A specific rule, '01\_setMembership\_isOneOf', is highlighted. The 'Content' tab displays the rule's code:

```

1 definitions
2 set 'truck' to a truck ;
3 if
4 the model of truck is one of { the F150 , the
5 then
6 display the message "The truck model of " + th

```

The second line ('set 'truck' to a truck ;') is highlighted with a red box. Below the code, the status bar shows 'Intellirule ARL | 01\_setMembership\_isOneOf.brl'. At the bottom, the 'Problems' view shows '1 error, 6 warnings, 0 others'. A detailed view of the errors table shows one error: 'An automatic variable 'truck' is already declared.' This error is also highlighted with a red box.



## 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.



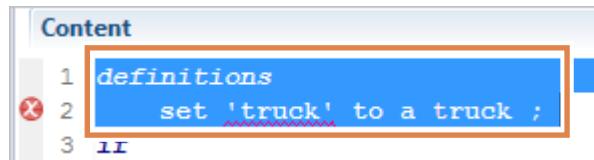
## Questions

How can you rewrite the rule so that it compiles?

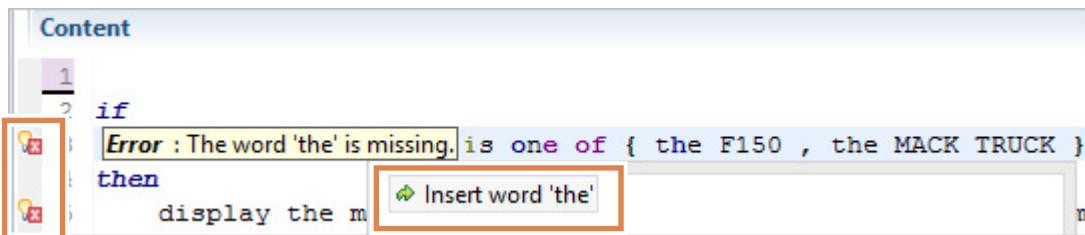
### Answer

- 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. Select the ***definitions*** statement and delete it.



- 6. Change the `01_setMembership_isOneOf` action rule to use the automatic variable by clicking the **Quick Fix** icons, and then double-clicking the proposed fix: **Insert word 'the'**



The resulting rule should match this rule:

```

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 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 and close it in the editing window.

## 3.2. Test the automatic variable in a rule

- 1. In Rule Explorer, expand **trucksAndDrivers-rules > 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

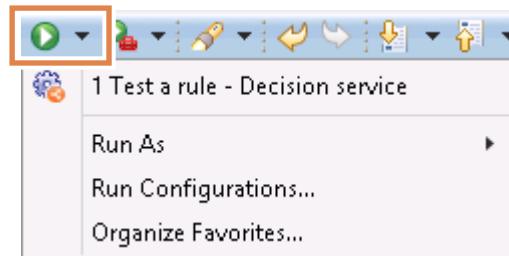
- The definition in the `01_action_phrase` action rule is useless.
- Where the `01_action_phrase` action rule uses the rule variable '`'the truck'`' (with single quotation marks), it can use the automatic variable `the truck` (without quotation marks) instead.

- 2. To use the automatic variable, delete the definition statement and modify the `01_action_phrase` rule in the `action` rule package to use the automatic variable.

Your rule should match the rule statement here:

```
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";
```

- 3. Save the `01_action_phrase` action rule and close it.
- 4. 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 7-5.

**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.

# Exercise 8. Authoring action rules

## Estimated time

00:45

## Overview

In this exercise, you learn how to author action rules.

## Objectives

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 parameters in rule statements

## Introduction

In this exercise, you author the action rules that use rule variables, automatic variables, and parameters.

The exercise includes these tasks:

- [Section 1, "Setting up the rule project for authoring"](#)
- [Section 3, "Authoring actions rules in the Guided editor"](#)
- [Section 4, "Authoring rules with parameters"](#)

## Requirements

You should complete these exercises before proceeding:

- [Exercise 2, "Setting up decision services"](#)
- [Exercise 7, "Exploring action rules"](#)

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

- Start project: Dev 08 – Authoring rules\start

## Section 1. Setting up the rule project for authoring

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.

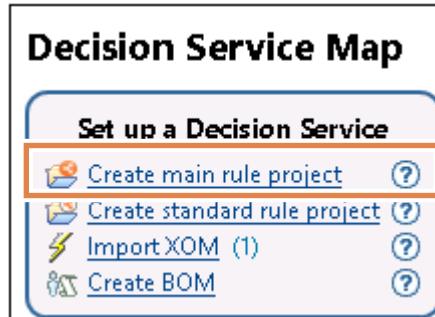
### 1.1. Setting up your environment for this exercise

- 1. In Rule Designer, switch to a new workspace and name it: `author`
  - a. From the **File** menu, click **Switch Workspace > Other**.
  - b. In the Workspace Launcher window, enter the path:  
`C:\labfiles\workspaces\author`
  - c. Click **Launch**.
- 2. Close the Welcome view.
- 3. Use the Samples Console to import **Training > Ex 08: Authoring rules > start**.
  - a. Click the **Open Perspective** icon.
  - b. In the Open Perspective window, click **Samples Console** and click **Open**.
  - c. In the **Rule Designer** section, expand **Training > Ex 08: Authoring rules > start**, and click **Import projects**.
- 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.

## 1.2. Create the rule project

- 1. In Rule Designer, create a main rule project named: loan-rules
- 1. In the “Set up a Decision Service” part of the Decision Service Map in Rule Designer, click **Create main rule project**.



- 2. In the **Decision Service Rule Projects** section of the New Rule Project wizard, make sure that **Main Rule Project** is selected, and click **Next**.
- 3. In the **Project name** field, enter `loan-rules` and click **Next**.
- 4. In the Main Rule Project References page, select **loan-bom**.  
The `loan-rules` project uses the vocabulary from this BOM project.
- 5. Click **Finish**.

## 1.3. Create a decision operation

- 1. In Rule Explorer, select the `loan-rules > deployment` folder.
- 2. In the “Define decision operation” part of the Decision Service Map, click **Add decision operation**.
- 3. In the New Decision Operation window **Name** field, enter: `loan-rules operation`
- 4. Click **Next**.
- 5. Make sure that **loan-rules** is selected and click **Finish**.

## 1.4. Add a rule package

- 1. In the “Define decision operation” part of the Decision Service Map, click **Go to operation map**.
- 2. In the “Select an operation” window, select `loan-rules operation.dop`, and click **OK**.
- 3. In the Orchestrate part of the Operation Map, click **Add rule package**.
- 4. In the **Package** field, enter: `loan`
- 5. Click **Finish**.
- 6. Close the `loan-rules operation` tab.

## Section 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 the 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.

#### 2.1. Add a rule that is called: grade

- 1. In Rule Explorer, right-click the **loan** rule package, and click **New > Action Rule**.
- 2. In the **Name** field, enter: **grade**
- 3. Click **Finish**.

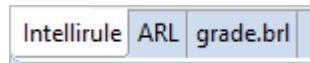
The empty rule opens in the Intellirule editor.



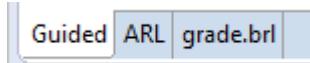
### Note

To verify whether you are using the Intellirule editor or the Guided editor, look at the tab of the editor window:

- **Intellirule** for the Intellirule editor



- **Guided** for the Guided 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**.

---

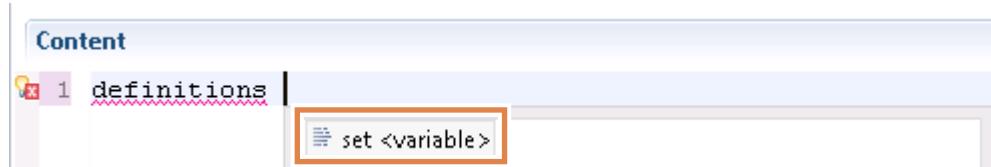
## 2.2. Author the grade rule

The `grade` rule must define a rule variable to manipulate the Report object that is passed to the rule engine by the `report` parameter.

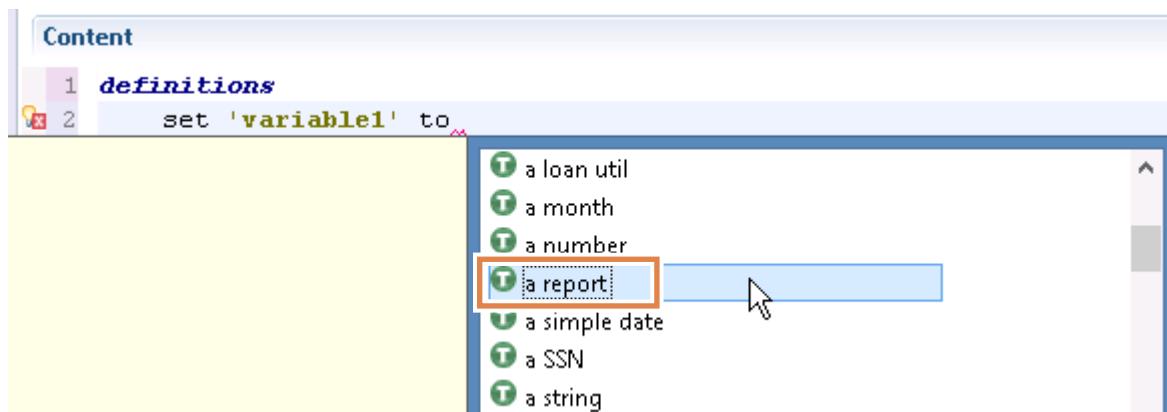
- 1. Press **Ctrl+Space** to open the Content Assist menu and double-click **definitions**.



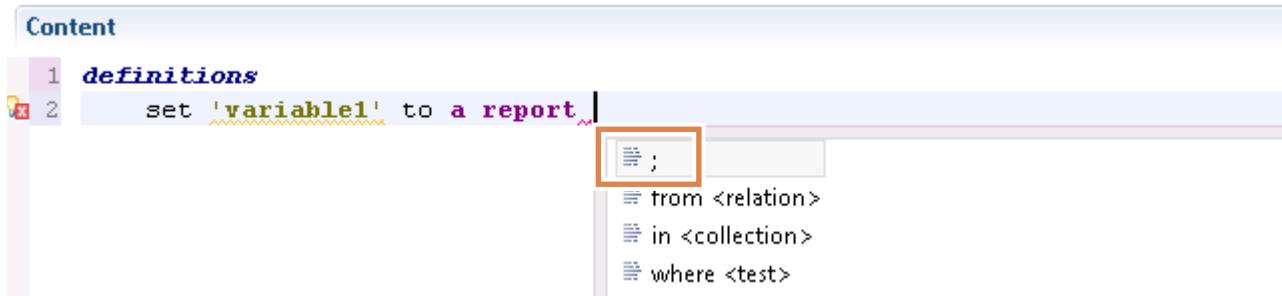
- 2. Content Assist continues to prompt you, so double-click **set <variable>**, and then double-click **variable1**.



- 3. Put the cursor in the space after `set 'variable1'` and press **Ctrl+Space** to open Content Assist.
- 4. Double-click **to <binding type>**.  
The Content Assist menu now prompts you with a vocabulary list.
- 5. You want the variable to be of type Report, so scroll through the vocabulary list to find and double-click **a report**.

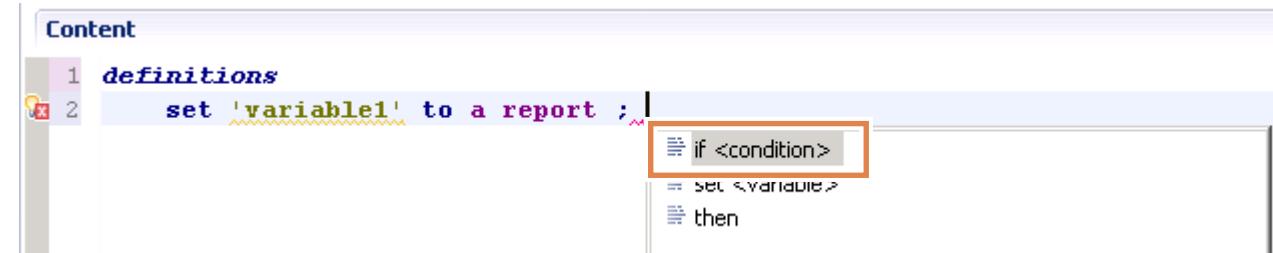


- \_\_\_ 6. In the Content Assist menu, double-click the semicolon (;).



The semicolon is required in the Intellirule editor, or to add another clause to your statement. For this rule, the variable definition is complete.

- \_\_\_ 7. When prompted, select **if <condition>**.



The editor automatically formats the condition statement to start on a new line.

- \_\_\_ 8. 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 ('').

- \_\_\_ 9. 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";

```

- \_\_\_ 10. Save the rule (Ctrl+S) and close the editing window.

## 2.3. Create more rules

This rule must refuse a loan application when the corporate score is less than 1000.

- \_\_\_ 1. In Rule Explorer, add a new rule to the **loan** rule package called: **invalid corporate score**

- 2. 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 `C:\labfiles\code` directory.

Press **Ctrl+Shift+F** to format the rule.

---



### Note

Notice that you must use the correct punctuation to avoid errors in the rule statement.

---

- 3. Save the rule (**Ctrl+S**).
  - 4. 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**

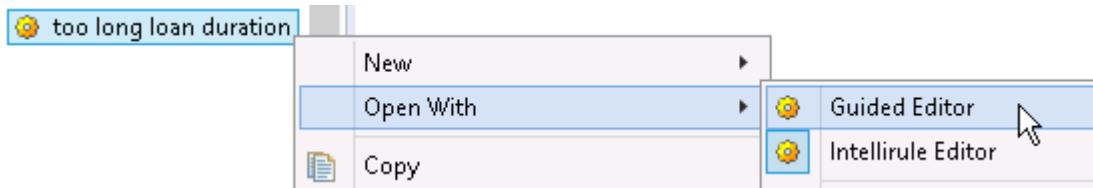
You can find the text of the too big loan rule in the `author.txt` file in the `C:\labfiles\code` directory.

- 
- \_\_\_ 5. Save your work and close the editing windows.

## Section 3. 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.
- \_\_\_ 3. In Rule Explorer, right-click the rule and click **Open With > Guided Editor**.



- \_\_\_ 4. 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 by clicking **definitions**.
- \_\_\_ b. Click **variable1** and set the variable name by typing: **the report**
- \_\_\_ c. Click **<select a choice>** and click **an <object>** from the list.
- \_\_\_ d. Click **<select an object>** and select **a report** from the list.
- \_\_\_ e. Continue editing the condition and action statements to complete the rule.



### Hint

To access the **is more than** BAL construct, click **is**.

The screenshot shows the Intellirule editor interface. A context menu is open over the word "is". The menu items include: does not equal, equals, is, is at least, is at least <min> and less than <max>, is at most, is between <min> and <max>, is less than, **is more than** (which is highlighted with a blue background), is more than <min> and at most <max>, is not, and is not one of.

```

if
 the duration (in years) of the loan of the report [±] is '200 [±]
 ↴
 then
 <select an action>
 ↴
 [else]

```

Guided ARL too long loan duration.brl

For the refusal message, you can type directly in the **enter a string** field.

Your rule should match the following statements:

```

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.

```

The screenshot shows the Guided editor interface with the following content:

```

Content

definitions
 set the report to ▼ a report [from/in]
 [where]
 ↴
 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 [±]
 ↴
 [else]

```



### 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.

---

- 5. Save your work.
- 6. Close the editing window.

## Section 4. Authoring rules with parameters

For this part of the exercise, you use a parameter instead of the local rule variable. Recall that to define a parameter:

- You first create a variable set.
- Then, you bind the variables to parameters.



### Reminder

For more information about defining parameters, see [Section 4.2, "Creating variables for parameters"](#) and [Section 4.3, "Binding the variables to parameters"](#) in [Exercise 2, "Setting up decision services"](#).

- 1. Expand **loan-bom > bom > model > training\_loan > Report**.

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:

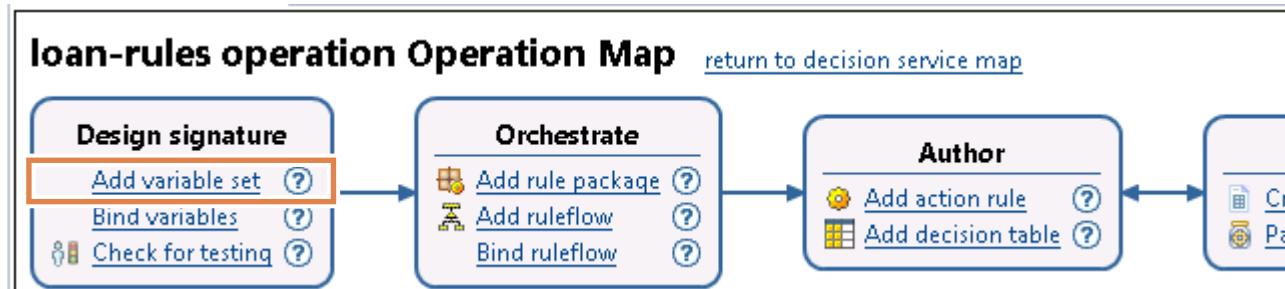
- messages
- insuranceRequired
- insuranceRate
- validData

Because objects of type `Report` are designed to carry output information, you can create an output parameter that is based on this object type.

- 2. Add a variable set called `my parameters` to the `loan-rules` decision service.

- a. In Rule Explorer, select the **loan-rules** decision service.

- \_\_\_ b. Make sure the Rule Project Map view is open, and in the “Design signature” part of the Operation Map, click **Add variable set**.



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

- \_\_\_ d. Click **Finish**.

The Variable Set editor opens.

3. 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`, and click the ellipsis (...) to open the Types window.
- In the **Choose a type** field, type: `report` and in the **Matching types** field, double-click **Report**.
- Click **OK**.

- **Verbalization:** the loan report

- \_\_\_ c. Save your work and close the **my parameters** tab.

4. Bind the report variable to an output parameter.

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

- \_\_\_ b. In the **Eligible variables** section, expand **my parameters**.

**Eligible variables**  
Select the ruleset variables that you want to use as parameters for the decision operation. Ruleset variables are defined in variable sets.

- \_\_\_ c. Select **report** and drag it to the Output Parameters table.

| Parameter name | Verbalization   | Type                 |
|----------------|-----------------|----------------------|
| report         | the loan report | training_loan.Report |

- \_\_\_ d. Save your work and close the **loan-rules** operation tab.

- \_\_\_ 5. Rewrite the `grade` rule to use the `report` parameter.

You can use your existing local rule variable to manipulate the `report` parameter (which shows up in the vocabulary list as “the loan report”).

You can also delete the local rule variable and use the parameter directly.

```

Content
1 definitions
2 set 'the report' to a report;

```

- \_\_\_ 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

```

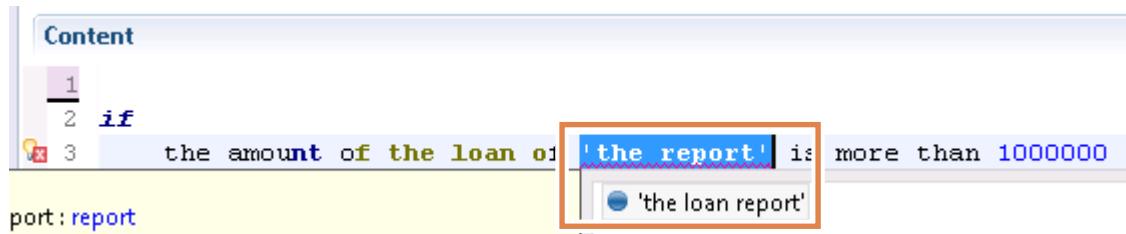
You should now see errors in the rule.

```

Content
1
2 if
3 the amount of the loan of 'the report' is more than 1000000
4 then
5 set the grade of 'the report' to "GOLD";

```

- \_\_\_ c. In the condition and action statements, double-click ‘the report’ and double-click ‘the loan report’ to replace it with the new parameter.



The errors should now be gone.

- \_\_\_ 6. Save your work (Ctrl+Shift+S).  
\_\_\_ 7. When you are finished editing rules, close the rule editors.  
\_\_\_ 8. In the Problems view, verify that no messages are listed.

## 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 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.

# Exercise 9. Authoring decision tables

## Estimated time

00:45

## Overview

In this exercise, you learn how to author decision tables.

## Objectives

After completing this exercise, you should be able to:

- Use the decision table editor to create a decision table

## Introduction

In this exercise, you work with patterns of rules to implement them as decision tables.

This exercise includes these tasks:

- [Section 1, "Authoring a decision table"](#)

## Requirements

You should complete these exercises before proceeding:

- [Exercise 8, "Authoring action rules"](#)

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

- Start project: Dev 09 – Authoring decision tables\start

## 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 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                  |     |                 |

A status bar at the bottom says: "Select a row header to see the text of the rule."

At the bottom of the interface, there are tabs: General, Decision Table, IRL, and rate.dta. The "Decision Table" tab is selected.

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).

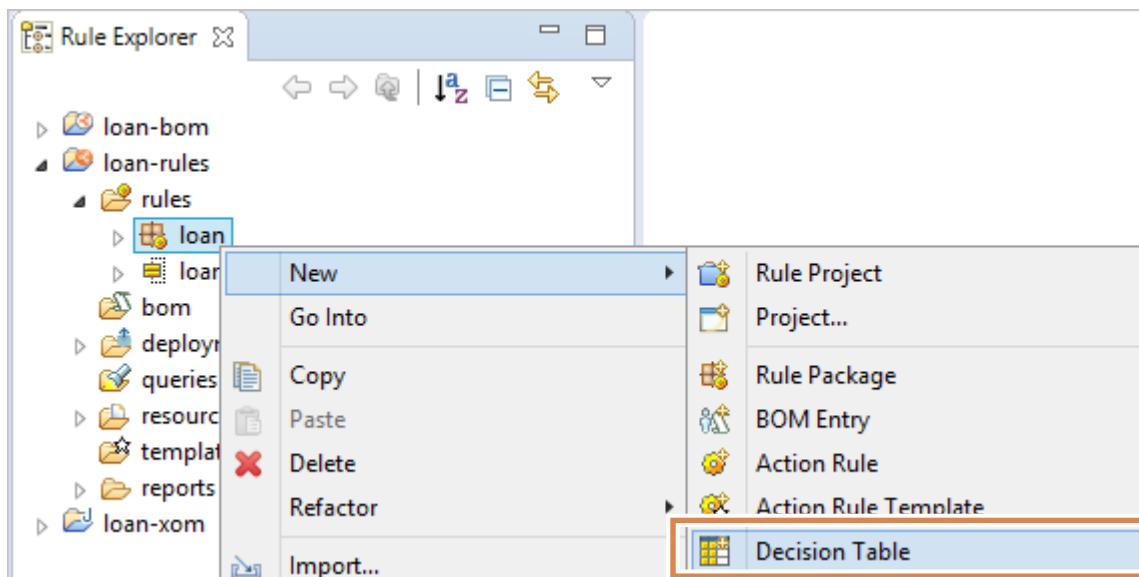
### 1.1. Setting up your environment for this exercise

- 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:  
C:\labfiles\workspaces\dtable
- \_\_\_ c. Click **Launch**.
- \_\_\_ 2. Close the Welcome view.
- \_\_\_ 3. Use the Samples Console to import **Training > Ex 09: Authoring decision tables > start**.
  - \_\_\_ a. Click the **Open Perspective** icon.
  - \_\_\_ b. In the Open Perspective window, click **Samples Console** and click **Open**.
  - \_\_\_ c. In the **Rule Designer** section, expand **Training > Ex 09: Authoring decision tables > start**, and click **Import projects**.
- \_\_\_ 4. When the workspace finishes building, close the Help view.

## 1.2. Creating the table

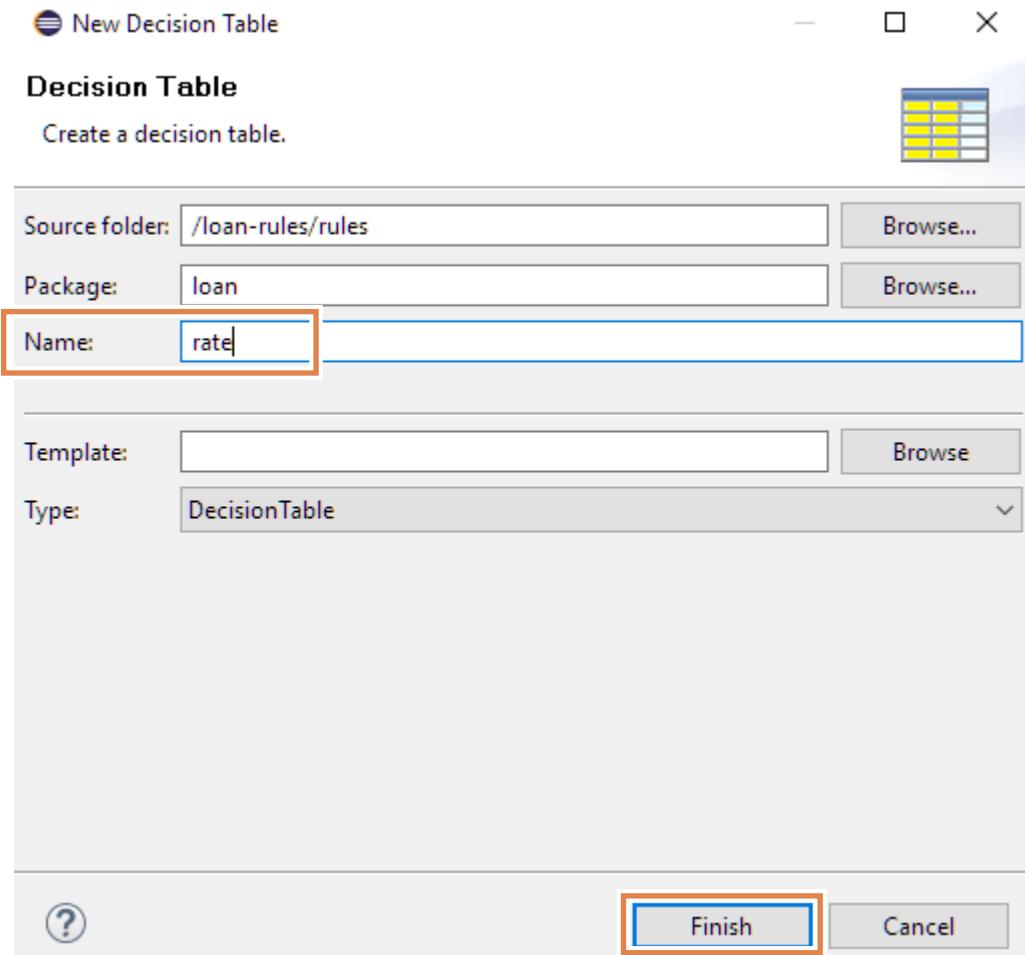
- \_\_\_ 1. In the `loan` rule package, create a decision table named: `rate`
  - \_\_\_ a. In Rule Explorer, expand the `loan-rules > rules` project.
  - \_\_\_ b. Right-click the `loan` rule package, and click **New > Decision Table**.



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.

|    | A | B | C | D |
|----|---|---|---|---|
| 1  |   |   |   |   |
| 2  |   |   |   |   |
| 3  |   |   |   |   |
| 4  |   |   |   |   |
| 5  |   |   |   |   |
| 6  |   |   |   |   |
| 7  |   |   |   |   |
| 8  |   |   |   |   |
| 9  |   |   |   |   |
| 10 |   |   |   |   |
| 11 |   |   |   |   |
| 12 |   |   |   |   |

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';
```

Decision Table: rate

**General Information**

Name : rate

**Documentation**

**Preconditions**

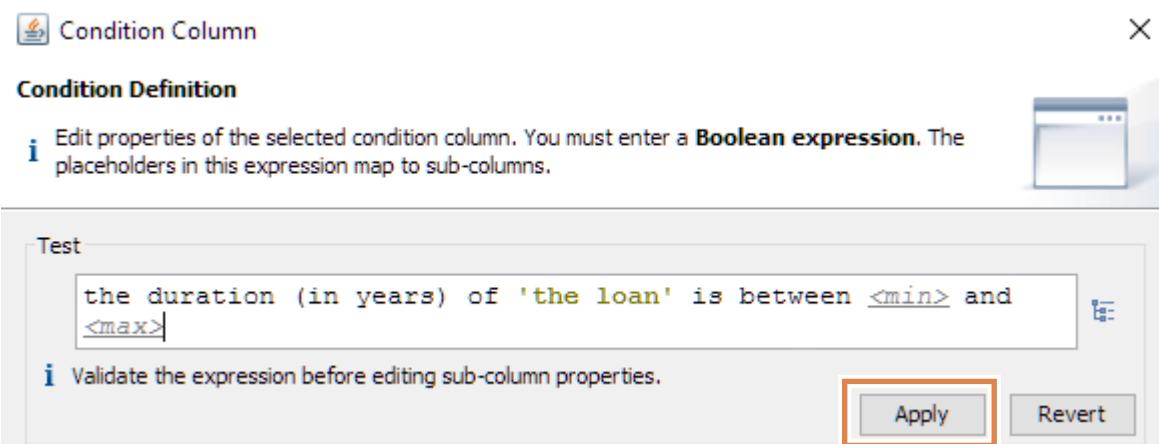
1 |

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**

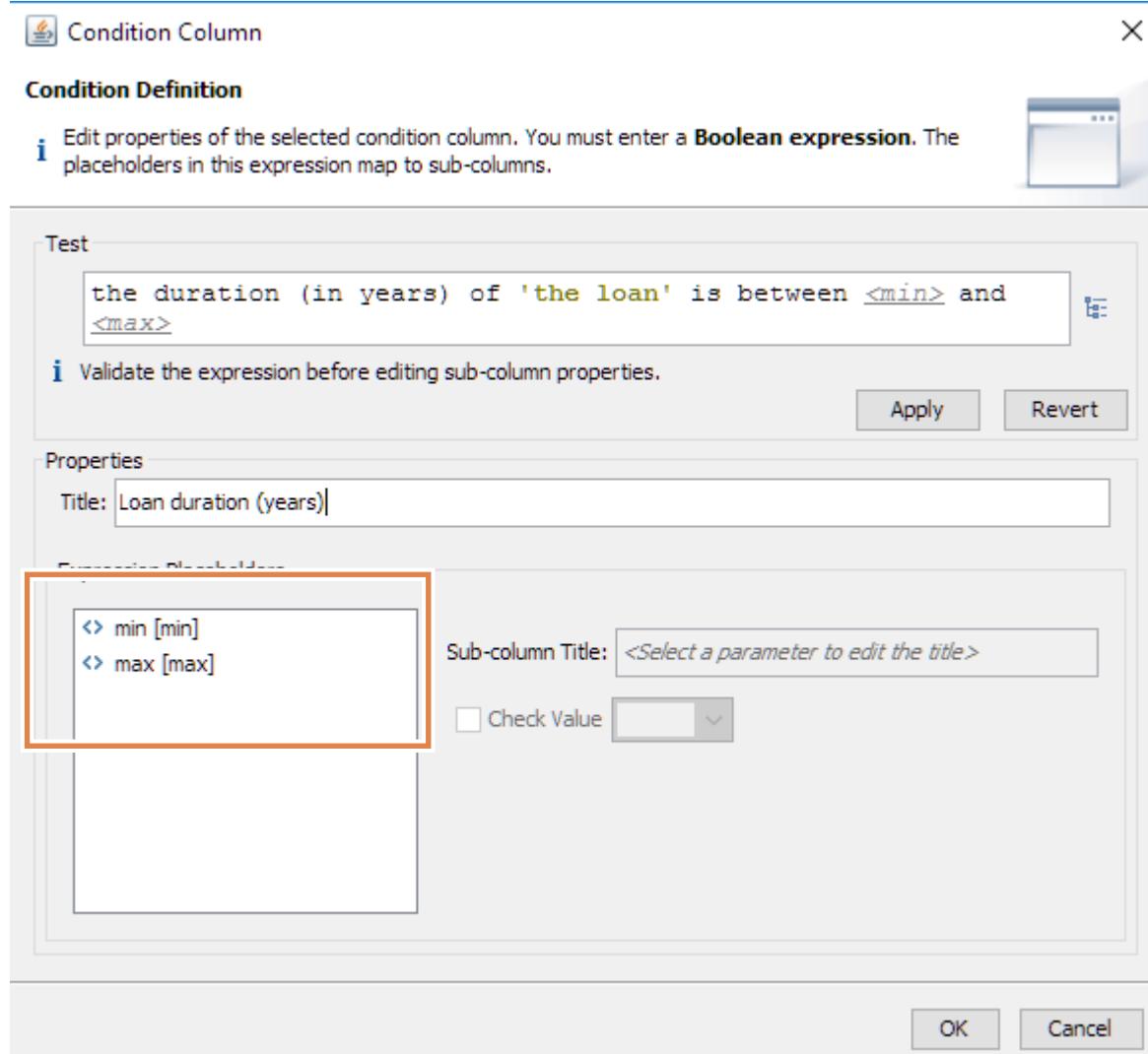
You can press Ctrl+Space to open Content Assist to build your precondition, or you can type directly in the editor. If you type directly, you can use Ctrl+Shift+F to format the content.

- \_\_\_ 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. Click **Apply**.



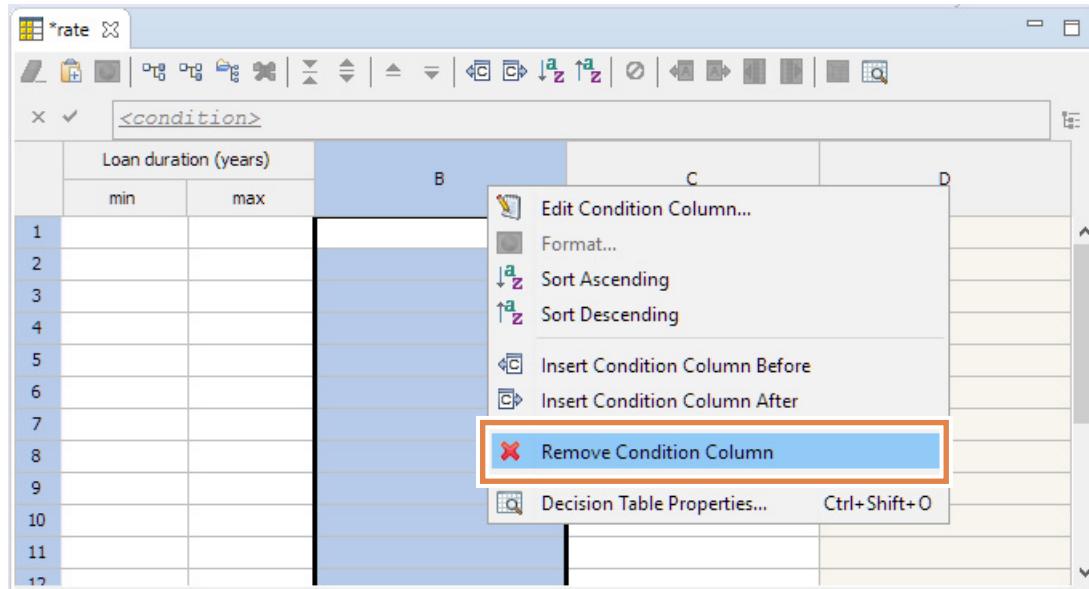
- \_\_\_ d. In the **Title** field, replace **A** with: Loan duration (years)

Notice that the placeholders in the condition test are now listed in the **Expression Placeholders** list.



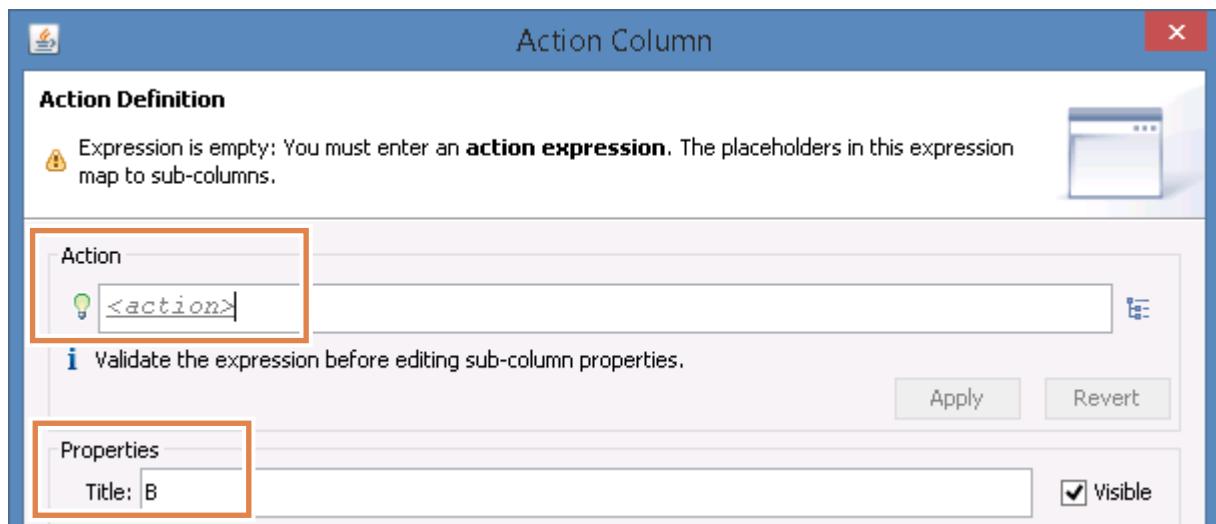
\_\_\_ e. Click **OK**.

- 5. Right-click each of the next two condition columns and remove them.



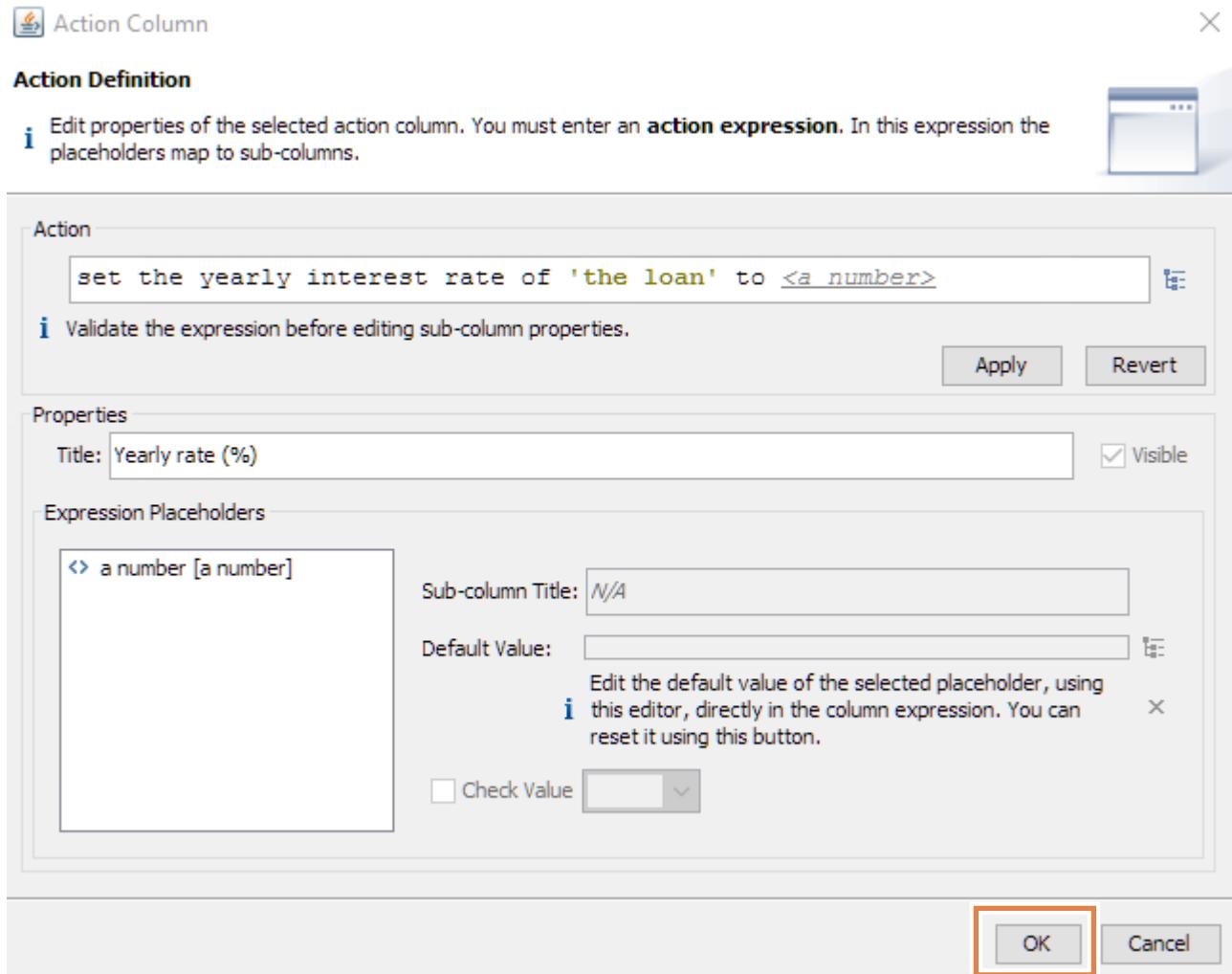
- 6. Define the action column.

- 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. Click **OK** to close the window.



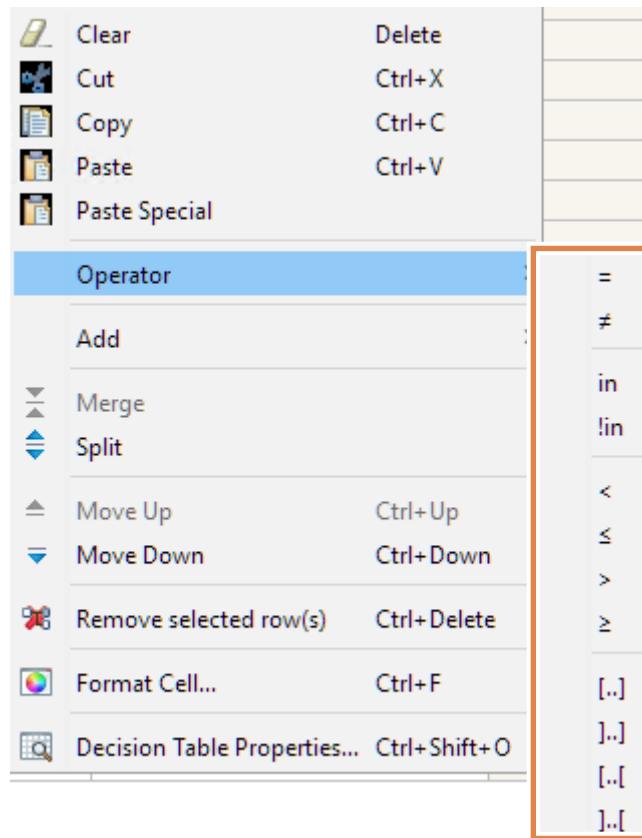
### 1.3. Completing the table cells with values

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.

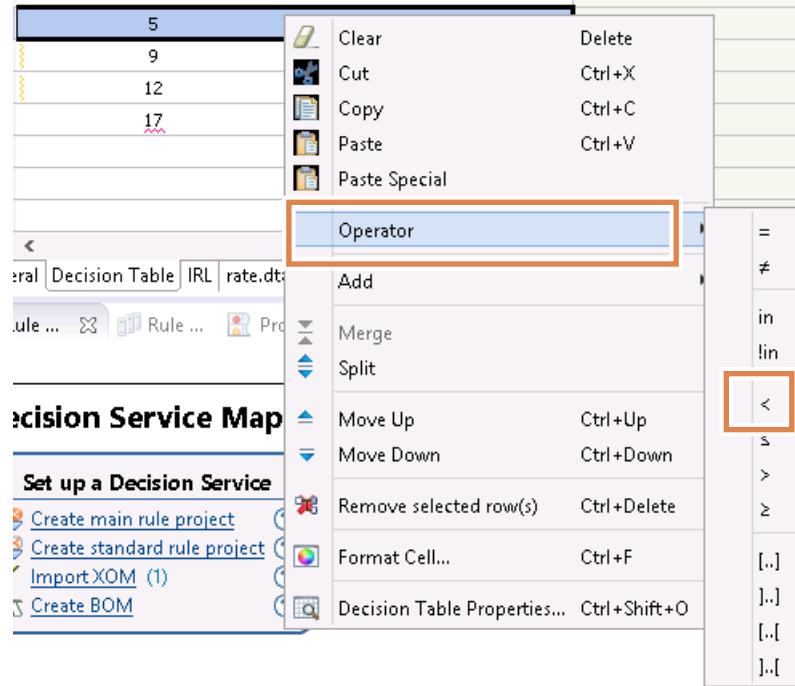
- 1. Review the five rules that you must write and the corresponding lines in the decision table, recalled at the beginning of this exercise. This table lists the values and operators that you use to complete the table.
- 2. In the decision table editor, first enter the values (not the operator) in the cells of the `rate` decision table:

| Loan duration (years)              | Yearly rate (%) |
|------------------------------------|-----------------|
| < 5                                | 5               |
| <code>&gt;= 5 AND &lt;= 8</code>   | 5.8             |
| <code>&gt;= 9 AND &lt; 12</code>   | 6.7             |
| <code>&gt;= 12 AND &lt;= 16</code> | 7.4             |
| <code>&gt;=17</code>               | 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.



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: [...]

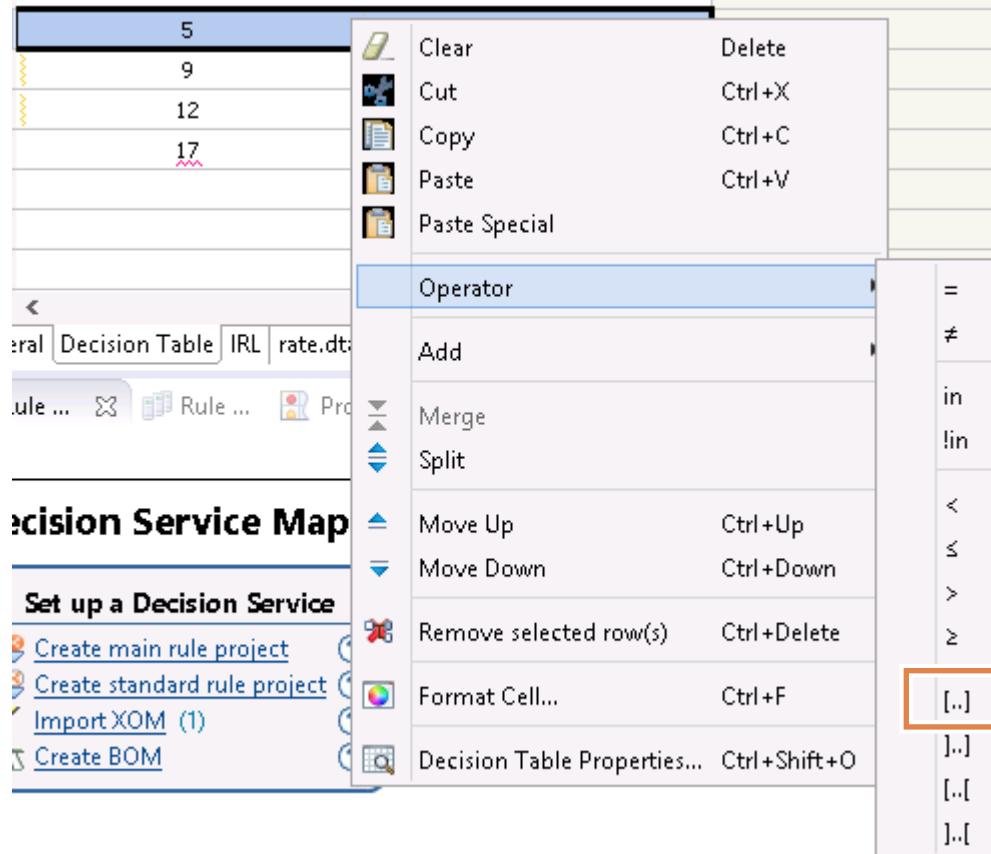
The screenshot shows a decision table with three rows. The first row has a condition 'Loan duration (years) < 5'. The second row has a condition '5 <= [ ] AND [ ] <= 8', where the range '5 <= [ ] AND [ ] <= 8' is highlighted by a red box. The third row has a condition '9 < 12'. The columns are labeled 'min' and 'max'.

|   | Loan duration (years) |     | Yearly rate (%) |
|---|-----------------------|-----|-----------------|
|   | min                   | max |                 |
| 1 | < 5                   |     |                 |
| 2 | 5                     | 8   |                 |
| 3 | 9                     | 12  |                 |

You want the row to state:  $\geq 5 \text{ AND } \leq 8$

In this case, these operators are equivalent to **is between** or [...].

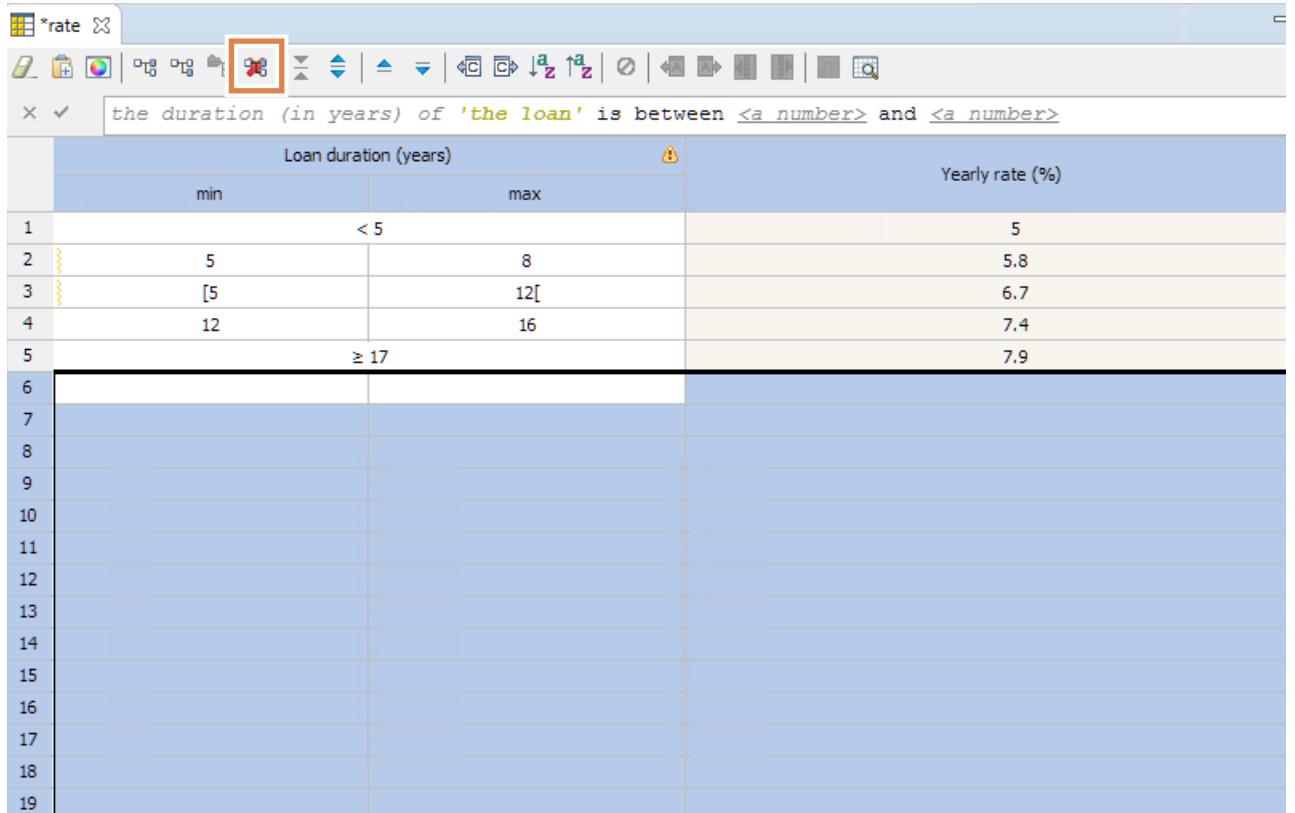
- \_\_\_ d. Right-click the second row and click [ . . ] from the **Operator** menu.



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.



|    | Loan duration (years) |     | Yearly rate (%) |
|----|-----------------------|-----|-----------------|
|    | min                   | max |                 |
| 1  |                       | < 5 | 5               |
| 2  | 5                     | 8   | 5.8             |
| 3  | [5                    | 12[ | 6.7             |
| 4  | 12                    | 16  | 7.4             |
| 5  | ≥ 17                  |     | 7.9             |
| 6  |                       |     |                 |
| 7  |                       |     |                 |
| 8  |                       |     |                 |
| 9  |                       |     |                 |
| 10 |                       |     |                 |
| 11 |                       |     |                 |
| 12 |                       |     |                 |
| 13 |                       |     |                 |
| 14 |                       |     |                 |
| 15 |                       |     |                 |
| 16 |                       |     |                 |
| 17 |                       |     |                 |
| 18 |                       |     |                 |
| 19 |                       |     |                 |

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             |



### 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.

The screenshot shows a decision table with one row selected. The selected row is highlighted with a blue background and has a red box around its first column cell containing the number '1'. A tooltip window is displayed below the table, showing the rule definition:

```

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.

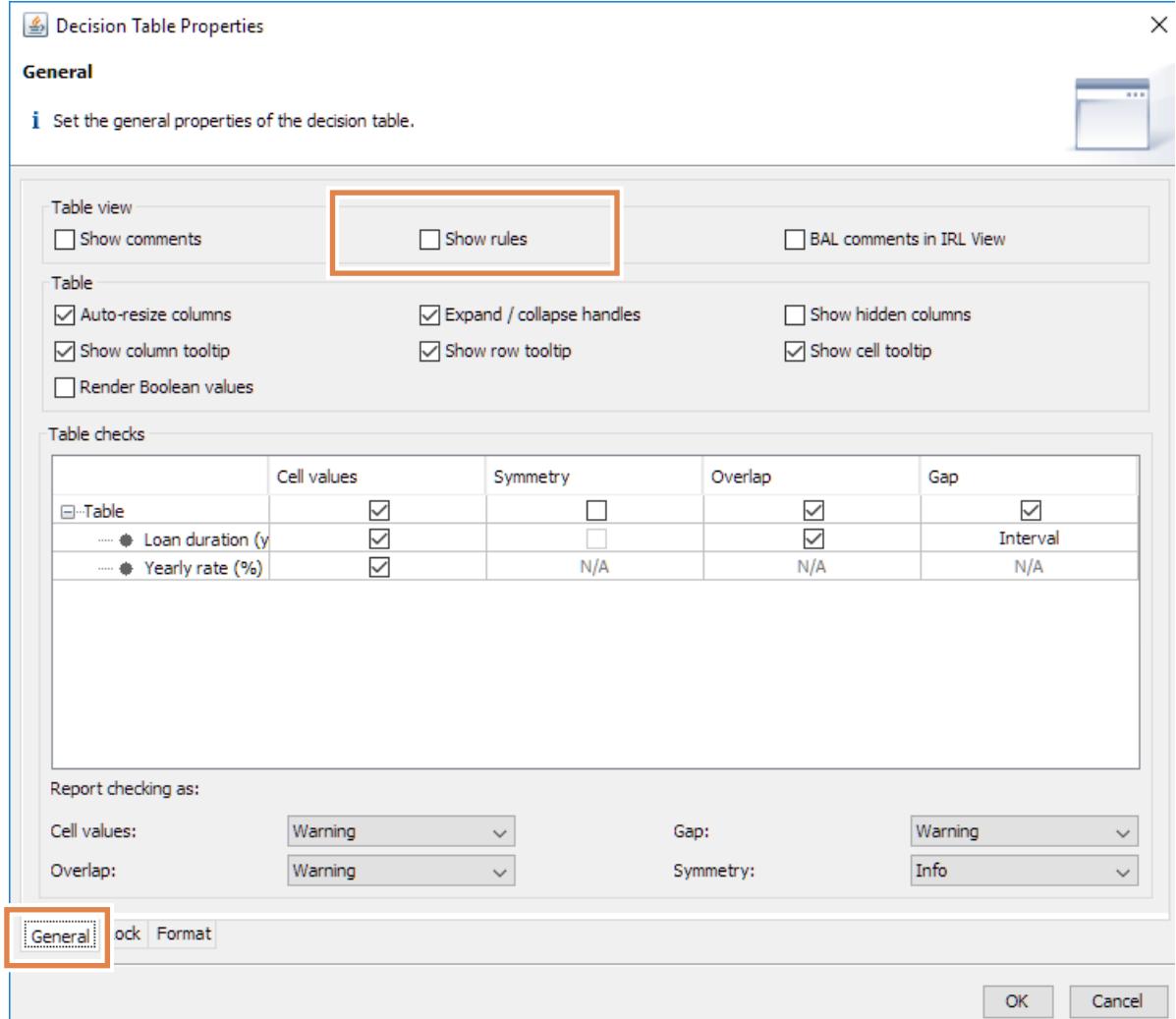


The Decision Table Properties window opens.

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.

|   | Loan duration (years) |     |
|---|-----------------------|-----|
|   | min                   | max |
| 1 | < 5                   |     |
| 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 ;

```



## 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.

## End of exercise

## Exercise review and wrap-up

The exercise looked at how to author a decision table.

# Exercise 10. Working with static domains

## Estimated time

00:45

## Overview

In this exercise, you learn how to simplify rule authoring by defining static domains in the BOM.

## Objectives

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 support files, which are installed in the `<InstallDir>\studio\training` directory:

- Start project: Dev 10 – Static domains\start

## 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 `C:\labfiles\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 and name it: `static_domains`
  - a. From the **File** menu, click **Switch Workspace > Other**.
  - b. In the Workspace Launcher window, enter the path:  
`C:\labfiles\workspaces\static_domains`
  - c. Click **Launch**.
- 2. Close the Welcome view.
- 3. Use the Samples Console to import **Training > Ex 10: Static domains > start**.
  - a. Click the **Open Perspective** icon.
  - b. In the Open Perspective window, click **Samples Console** and click **Open**.
  - c. In the **Rule Designer** section, expand **Training > Ex 10: Static domains > start**, and click **Import projects**.
- 4. When the workspace finishes building, close the Help view.

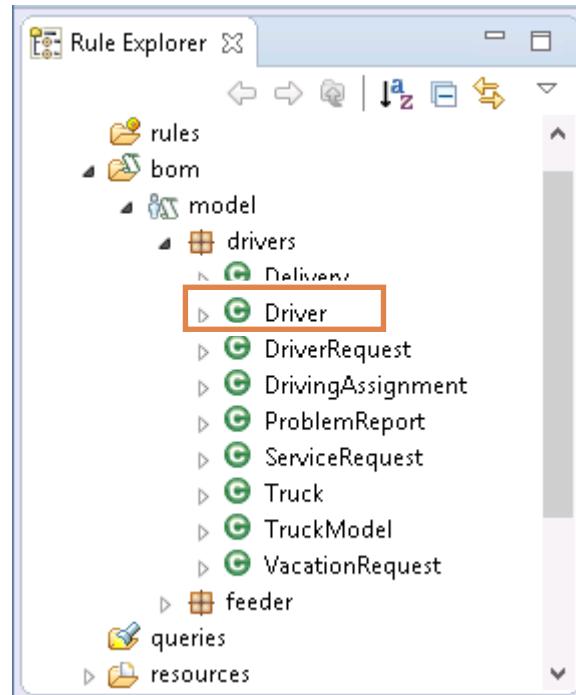
In your workspace, you now have the following projects in the `trucksAndDrivers-tests` decision service:

- `trucksAndDrivers-bom`
- `trucksAndDrivers-rules`
- `trucksAndDrivers-tests` (main rule project)
- `trucksAndDrivers-xom`

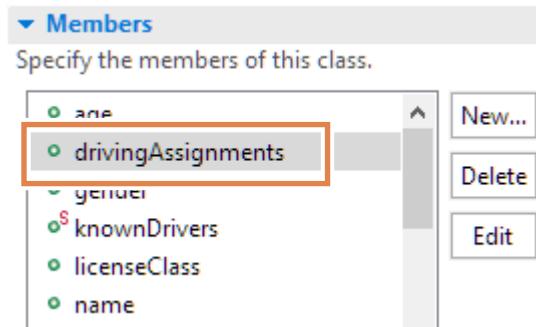
These projects define the business rule solution in which you explore and create domains.

## 1.2. Exploring the domain

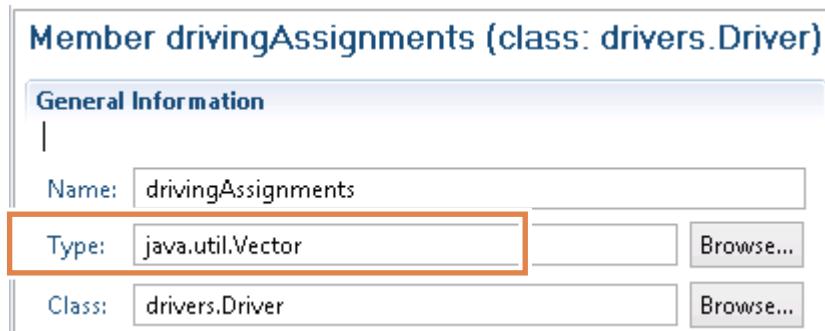
- 1. In Rule Explorer, expand **trucksAndDrivers-bom > bom > model > drivers** and double-click the **Driver** class to open it in the BOM editor.



- 2. In the **Members** section of the Class page, double-click the **drivingAssignments** member.



- 3. On the Member page, notice that the **drivingAssignments** member is defined as a vector.



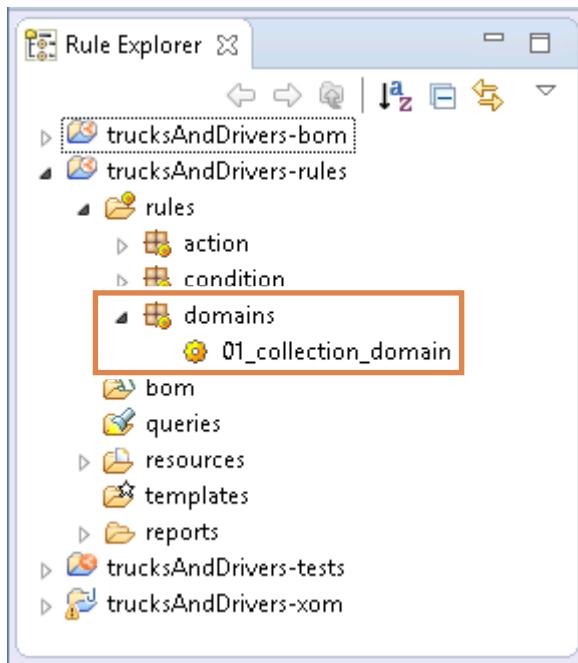
The vector is defined with a collection domain, and each member of the collection is of type drivers.DrivingAssignment.

**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>`

- \_\_\_ 4. In Rule Explorer, expand **trucksAndDrivers-rules > rules > domains**.
- \_\_\_ 5. Double-click the `01_collection_domain` rule to open it and see how this rule uses the collection domain.



- \_\_\_ 6. When you are done, close the rule.

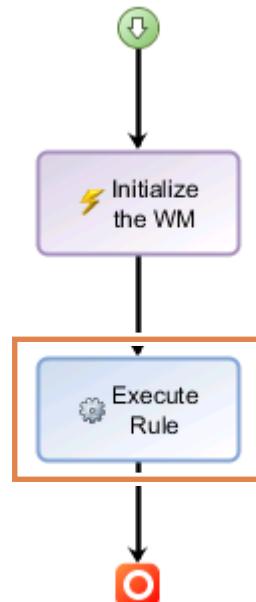
### 1.3. Testing the rule

With the next steps, you test the `domains\01_collection_domain` action rule.

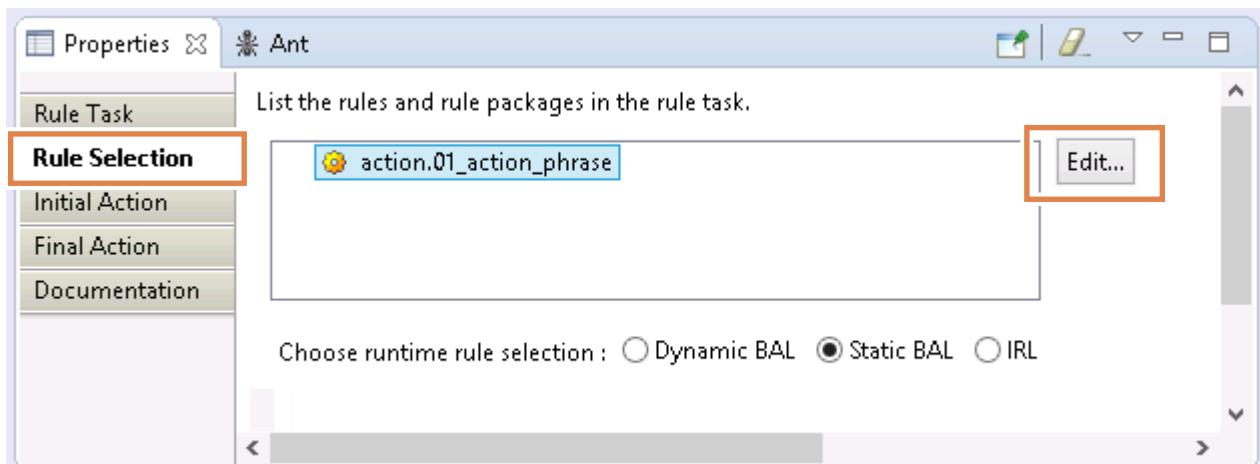
First, you verify that objects in the working memory exist that match the definitions of this rule. Then, you use the `trucksAndDrivers-tests` project to test this rule in the rule project.

- \_\_\_ 1. Expand the **trucksAndDrivers-xom > src > feeder** folder and double-click the `WIFeeder.java` file.

- \_\_ 2. In the `WMFeeder.java` file, scroll towards the end of the file to see the `DrivingAssignment` object for the `Driver` with the name: John Jongle
- \_\_ 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.

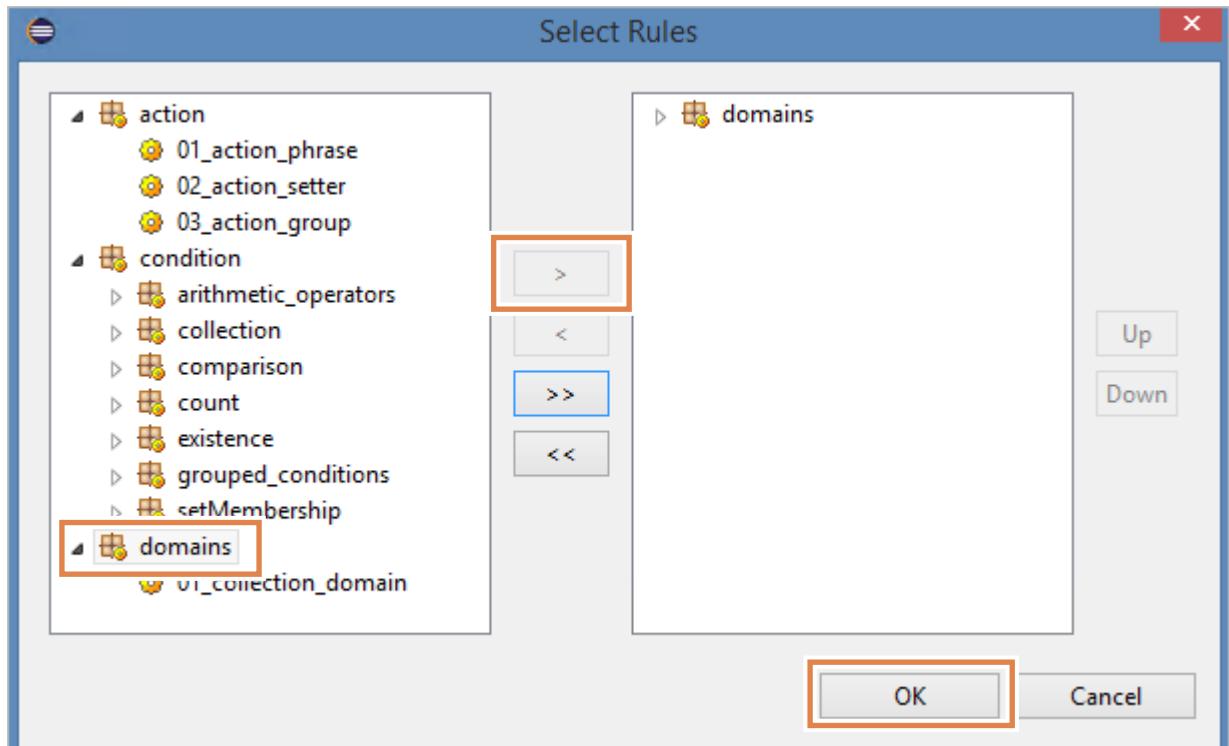


- \_\_ c. In the Rule Task properties, click **Rule Selection** and click **Edit**.



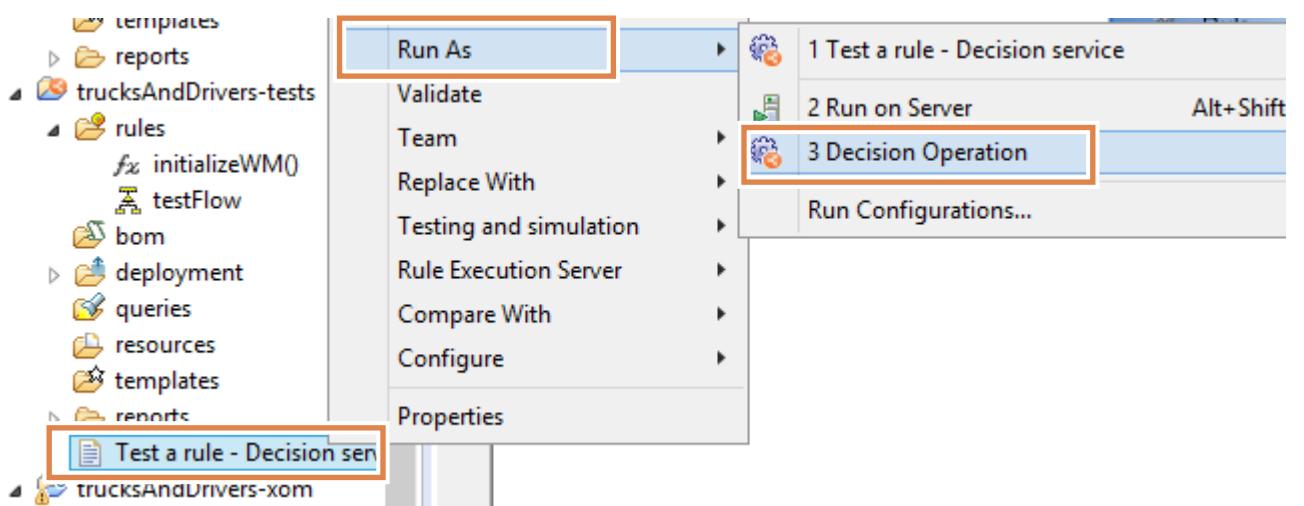
The Select Rules window opens.

- \_\_\_ 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

**Note**

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
 ...

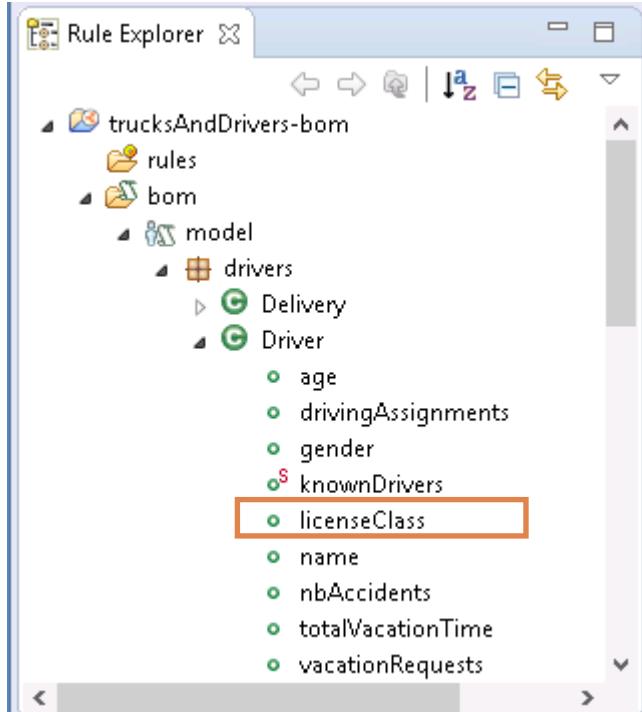
```

## Section 2. Working with an enumeration of literals

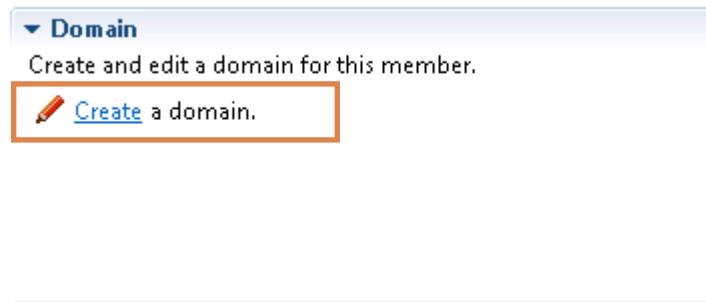
In this section, you define a domain as a list of literal values {A, B, C, D} for the `licenseClass` member of the `Driver` and `TruckModel` classes, and then use it to author an action rule.

### 2.1. Creating the domain

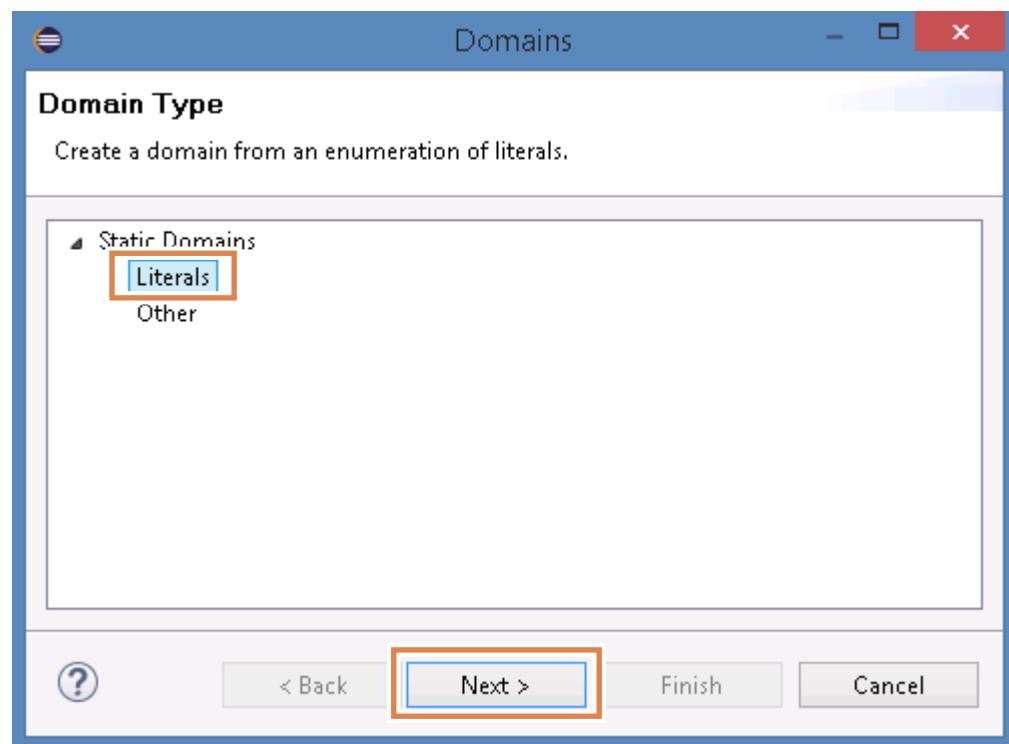
- 1. In the `trucksAndDrivers-bom` project, expand the `drivers.Driver` class and double-click the `licenseClass` BOM member to open it in the BOM editor.



- 2. In the BOM editor, in the **Domain** section, create a domain of literal values:
  - a. Scroll down and 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**.



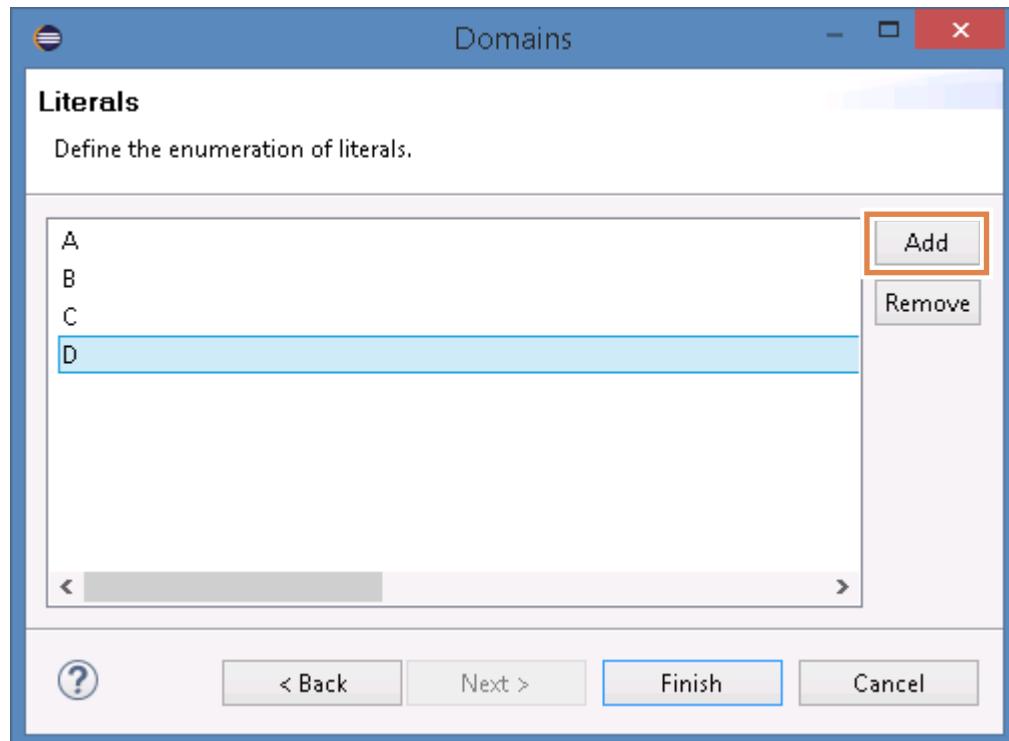
- \_\_ c. On the Literals page of the Domains window, click **Add** and type over the new domain value to rename it to: A



### 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

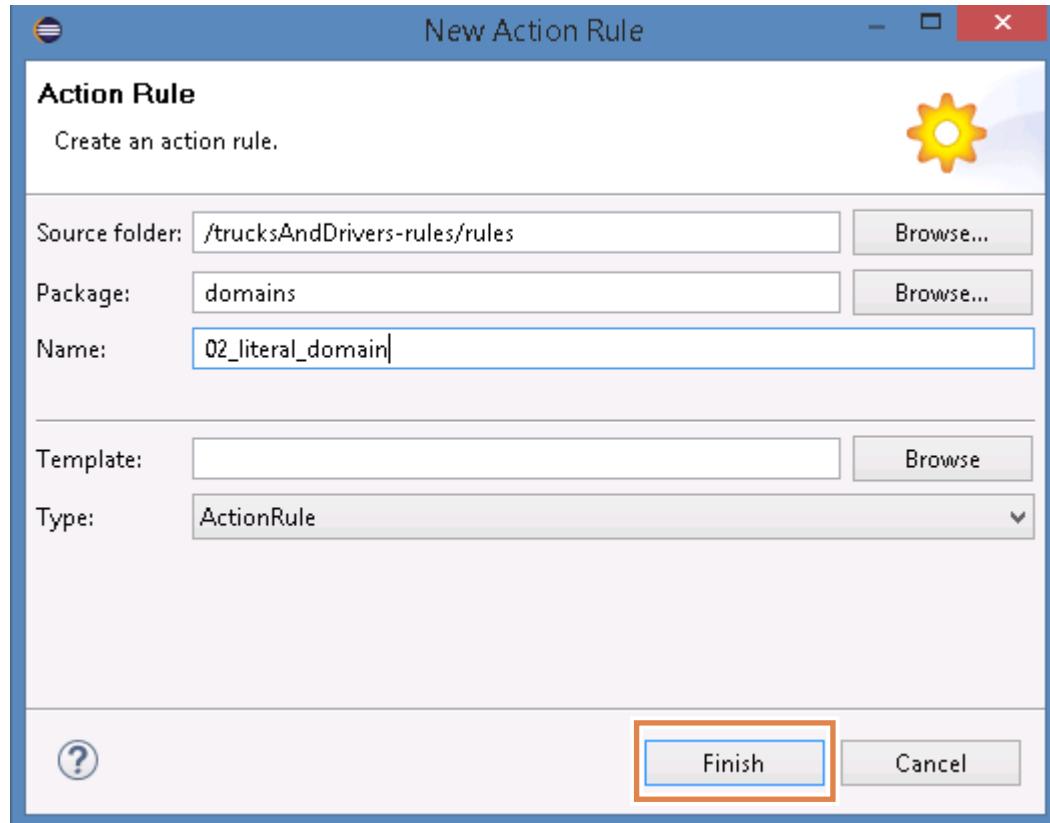
- A
- B
- C
- D

- \_\_\_ 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:

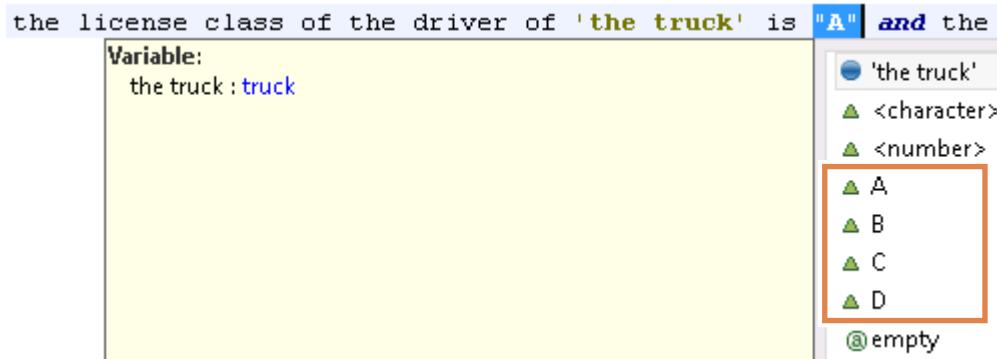
```
definitions
 set 'the truck' to a truck where the driver of this truck is not null;
if
 the license class of the driver of 'the truck' is "A" and the license
 class of the model of 'the truck' is one of {"B", "C", "D"}
then
 display the message "The driver (" + the name of the driver of 'the truck'
 + ") is not eligible to drive the truck " + the serial number of 'the
 truck' ;
```



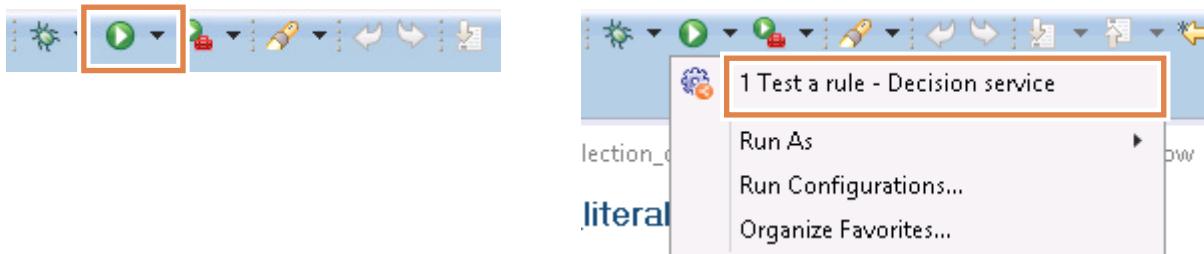
### Hint

Recall that you can type directly in the Intellirule editor or you can paste the code from the code snippet in the `C:\labfiles\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

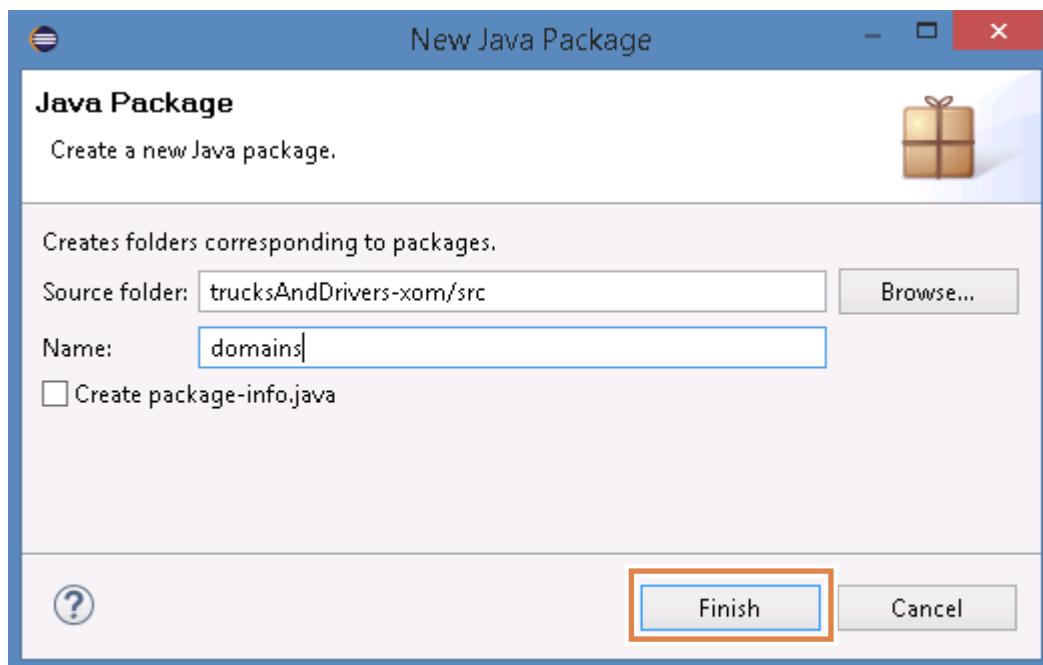
### 3.1. Creating the static references Java class

In this section, you change the attribute `gender` in the `Driver` class to use the static attributes of a new class called `GenderType`.

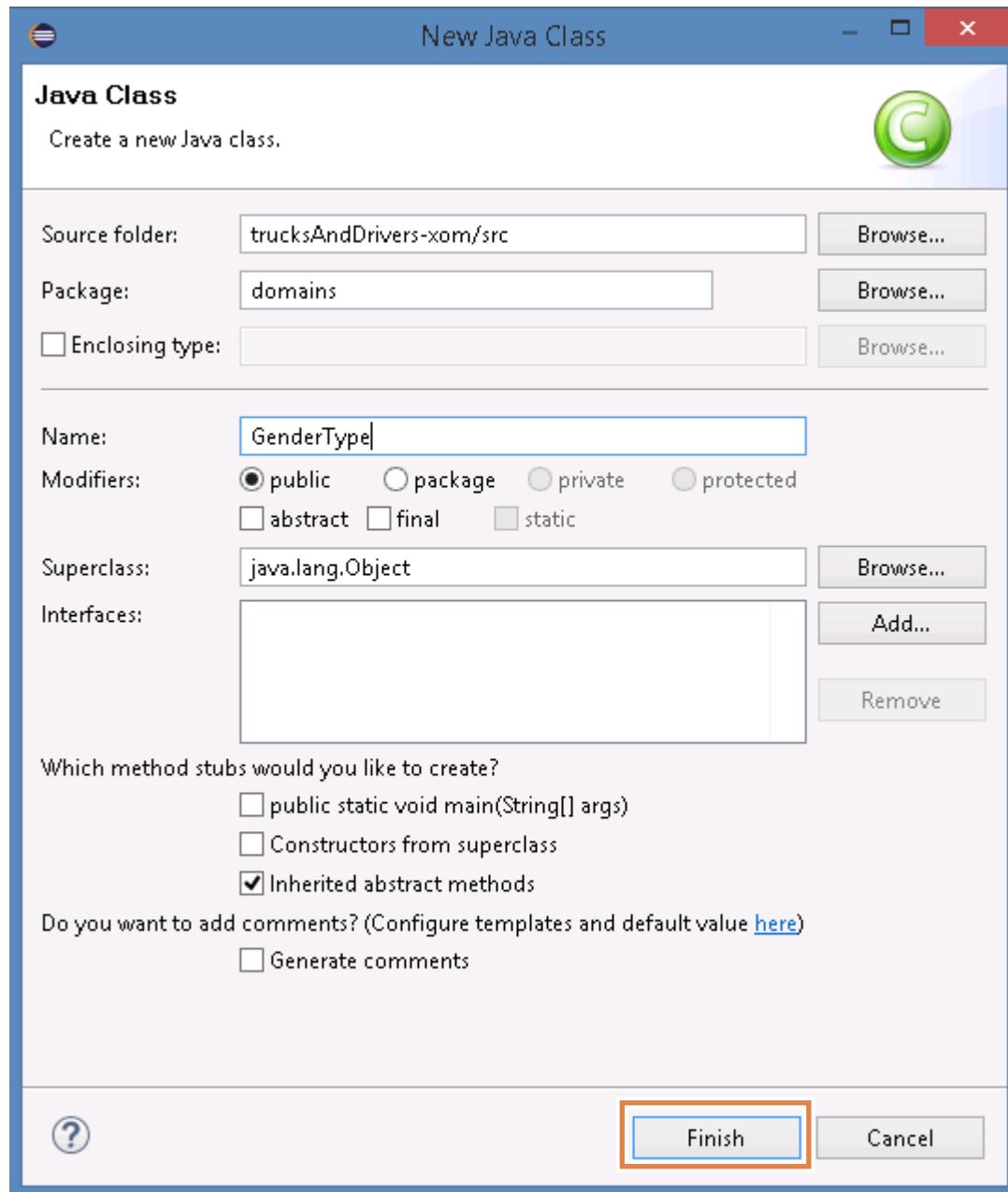
- 1. Switch to the Java perspective.
- a. Click the **Open Perspective** icon in the toolbar, which is in the upper-right corner of the Rule Designer window.



- 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 `C:\labfiles\code\create_domains.txt` file.

- \_\_\_ 4. Save your work (Ctrl+S).
- \_\_\_ 5. Change the `drivers.Driver` class to use the new `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
 private Vector<VacationRequest> vacationRequests; // vacation
 private String gender;
```

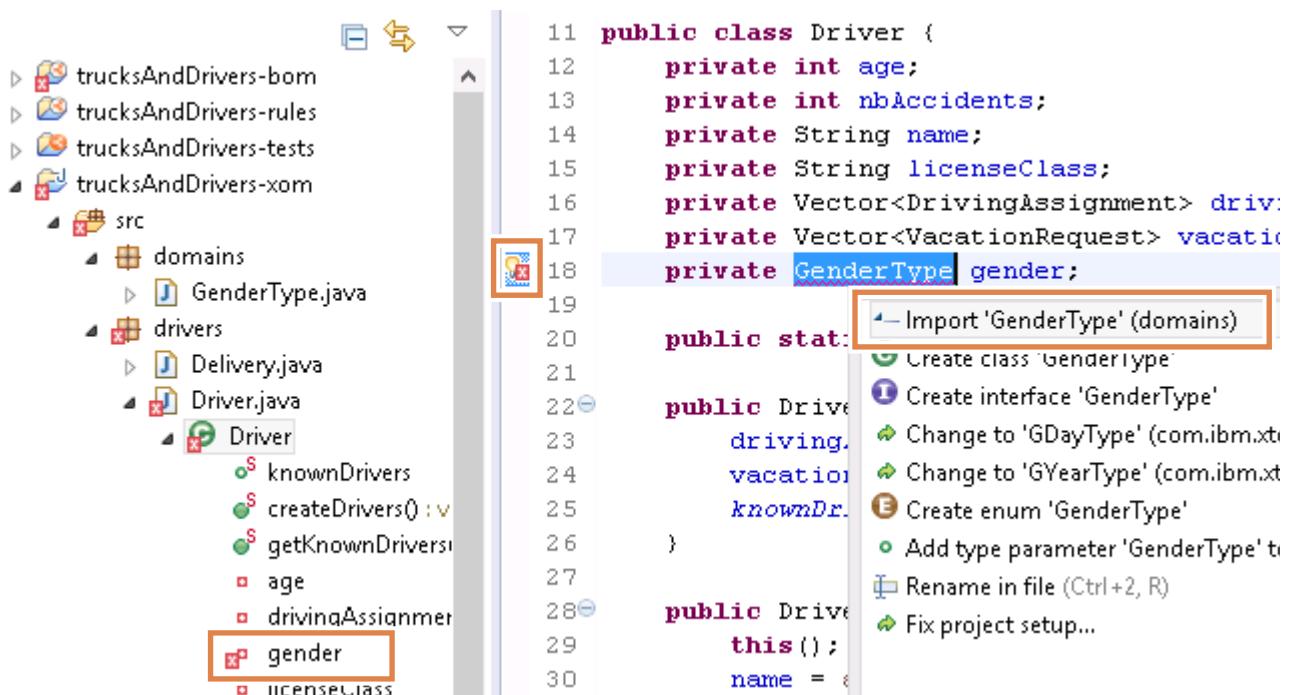
- \_\_\_ d. Edit the line by changing `String` to `GenderType`.

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
 private Vector<VacationRequest> vacationRequests; // vacation
 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. Also in the `src` folder, 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.getLicenseClass());
 dA20.setAge(20);
 dA20.setNbAccidents(3);
 dA20.setGender(GenderType.MALE);

 Driver dA18 = new Driver("John Jongle", TruckModel.MACK_TRUCK.getLicenseClass());
 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);

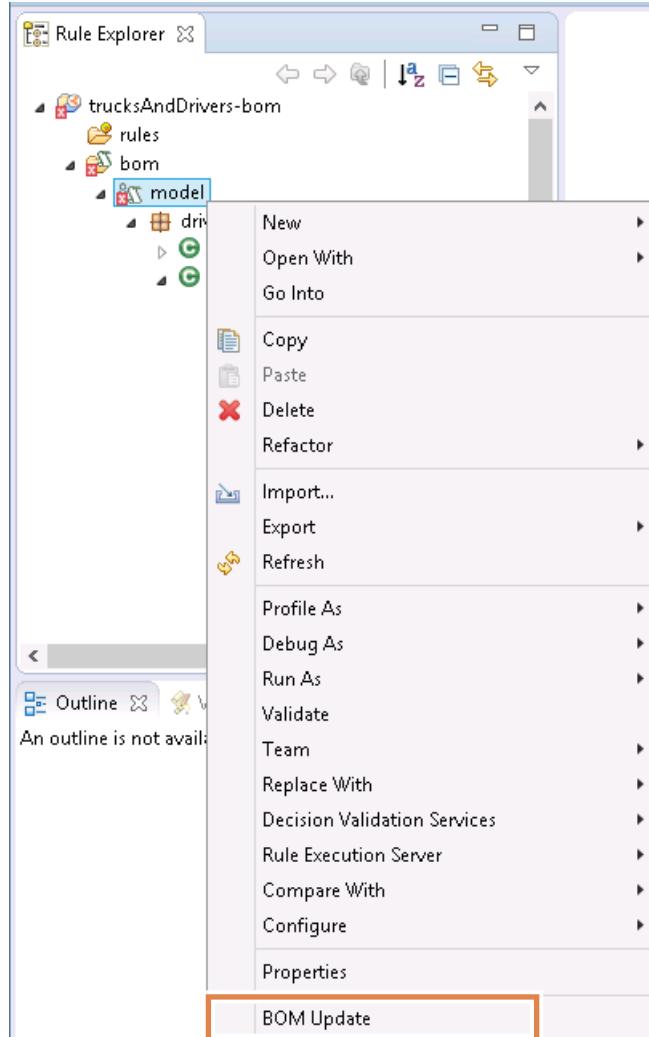
 Driver dB37 = new Driver("Jill Northwood", TruckModel.T2000.getLicenseClass());
 dB37.setAge(37);
 dB37.setNbAccidents(1);
 dB37.setGender(GenderType.UNKNOWN);
```

- \_\_\_ 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 errors on the BOM. You correct these errors 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 > bom** folder, right-click **model** and click **BOM Update**.

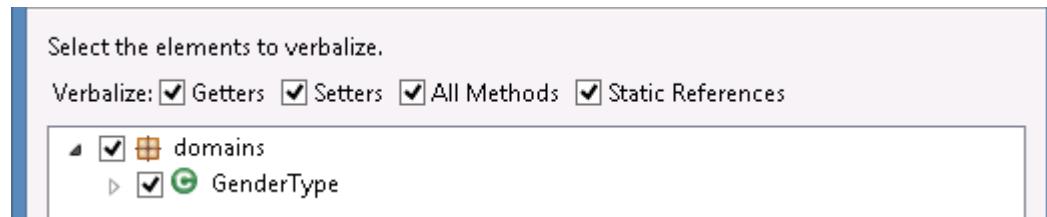


- \_\_\_ a. In the BOM Update tab in the bottom pane, expand the **Differences and Actions** section.
- \_\_\_ b. Notice the differences that were detected between the XOM and BOM classes:
- The XOM class `domains.GenderType` is not found in the BOM.
  - The definition of the attribute `drivers.Driver.gender` differs between the BOM and the XOM.
- \_\_\_ c. In the Actions menu, select **Import the XOM class** and click **Perform and save**.

**Differences and Actions**

|                 |                                                                                 |                                 |       |
|-----------------|---------------------------------------------------------------------------------|---------------------------------|-------|
| Actions:        | Deprecate the BOM method "feeder.WMFeeder.feedWM(ilog.rules.engine.IlrContext)" | Perform and save                | Clear |
| Perform action: | Deprecate the BOM method "feeder.WMFeeder.feedWM(ilog.rules.engine.IlrContext)" | Action to solve this difference |       |
| Origin          | Import the XOM class "domains.GenderType"                                       | not found in BO                 |       |
| XOM             |                                                                                 |                                 |       |

- \_\_\_ d. 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.



- \_\_\_ e. Click **Finish**.

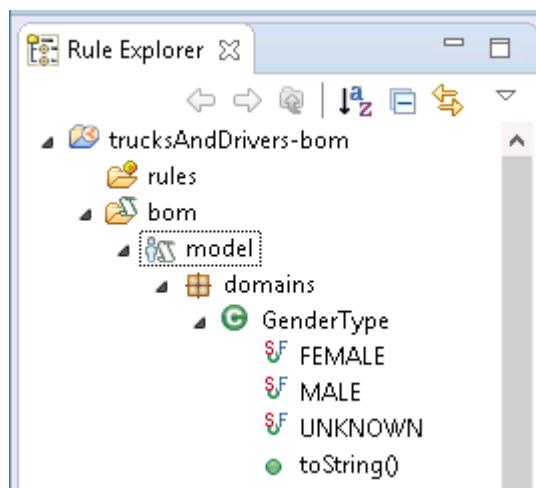
- \_\_\_ 12. Update the BOM class.

- \_\_\_ a. Go back to the BOM Update view.
- \_\_\_ b. Expand **Differences and Actions**.
- \_\_\_ c. Select the **Bulk: Update BOM Class (2)** 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 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`



- The **Type** field of `drivers.Driver.gender` BOM member is changed to: `GenderType`

### Member gender (class: drivers.Driver)

#### General Information

Name: `gender`

Type: `domains.GenderType`

Class: `drivers.Driver`

- \_\_\_ 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 `C:\labfiles\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.

# Exercise 11. Working with dynamic domains

## Estimated time

01:30

## Overview

In this exercise, you learn how to define and use dynamic domains with Microsoft Excel spreadsheets.

## Objectives

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 4, "Working with the BOM"](#)
- [Exercise 7, "Exploring action rules"](#)
- [Exercise 10, "Working with static domains"](#)

This exercise uses a series of Excel files that are stored in the `C:\labfiles\code` directory.

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

- Start project: Dev 11 – Dynamic domains\start



### 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 `C:\labfiles\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 `C:\labfiles\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 and name it: `dynamic_domain`
  - \_\_\_ a. From the **File** menu, click **Switch Workspace > Other**.
  - \_\_\_ b. In the **Workspace Launcher** window, enter the path:  
`C:\labfiles\workspaces\dynamic_domain`
  - \_\_\_ c. Click **Launch**.
- \_\_\_ 2. Close the Welcome view.
- \_\_\_ 3. Use the Samples Console to import the exercise start project.
  - \_\_\_ a. Click the **Open Perspective** icon.
  - \_\_\_ b. In the Open Perspective window, click **Samples Console** and click **Open**.
  - \_\_\_ c. In the **Rule Designer** section, expand **Training > Ex 11: Dynamic domains > start**, and click **Import projects**.
  - \_\_\_ d. 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 when they have changes.

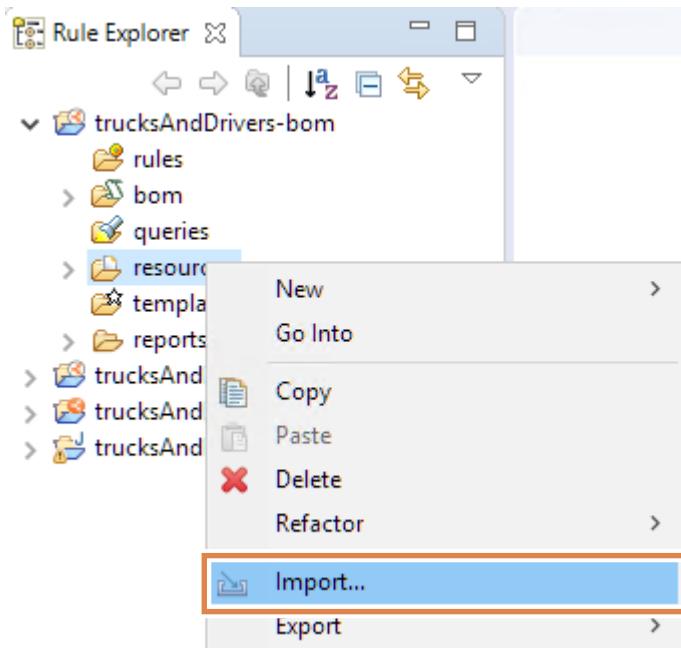
Because of potential change, you cannot use a static domain to implement the requirement. Instead, you use a dynamic domain, which is an enumerated domain (or list of values) that can change dynamically.

---

## 1.2. Import the domain file

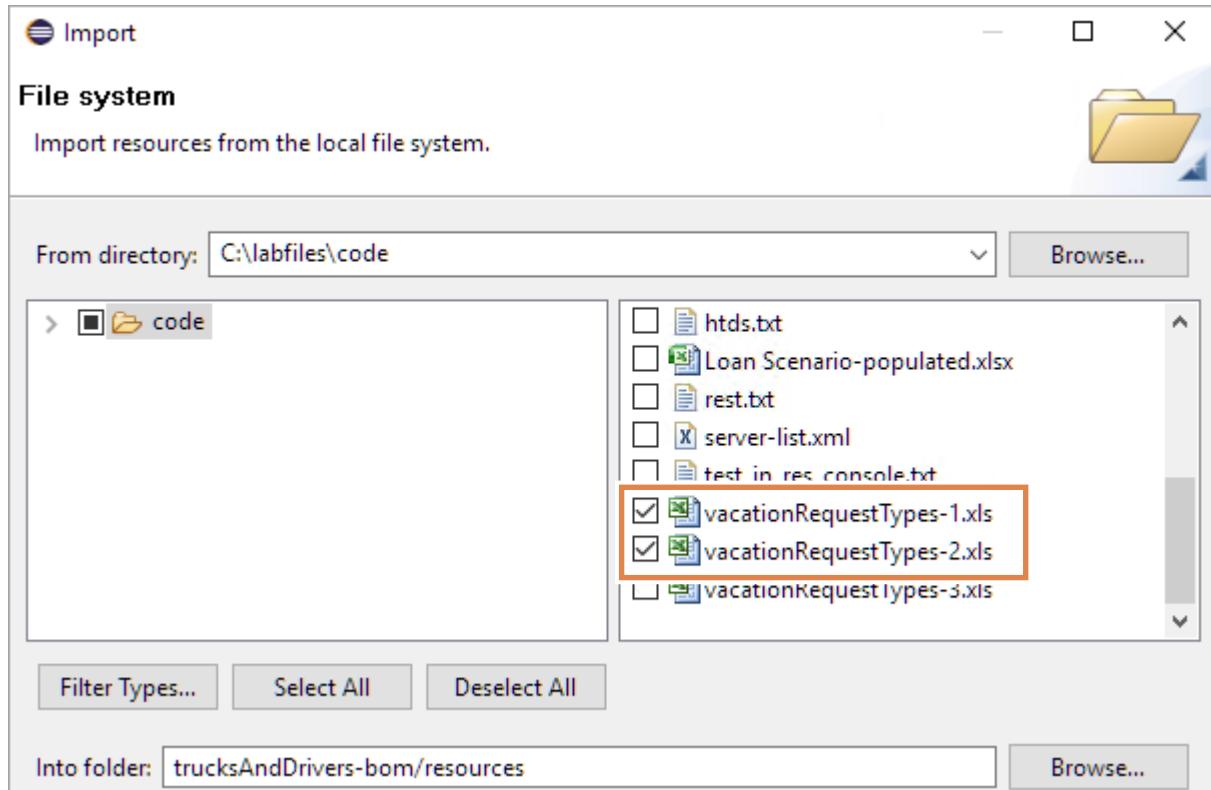
- 1. Go to Windows Explorer and open the `C:\labfiles\code\vacationRequestTypes-1.xls` file and verify that each row of this file contains three columns, where:
  - The first column represents the name to the domain value.  
Example (line 6): `JuryDuty`
  - The second column represents the code in the BOM-to-XOM mapping.  
Example (line 6): `return "V78";`
  - The third column represents the verbalization of the domain value.  
Example (line 6): `Jury Duty`

- \_\_ 2. In Rule Designer, import the Excel domain files into the `resources` folder of your BOM project.
  - \_\_ a. Expand the `trucksAndDrivers-bom` project, right-click the `resources` folder of the `trucksAndDrivers-bom` project, and click **Import**.



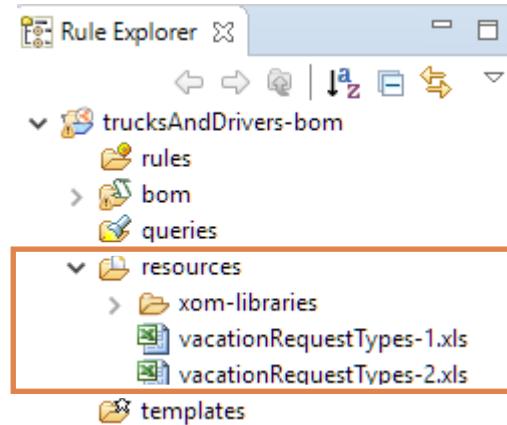
- \_\_ b. In the Select pane of the Import window, select **General > File System**, and click **Next**.
- \_\_ c. In the File System pane, click **Browse**, which is next to the **From directory** field, select the `C:\labfiles\code` folder, and click **OK**.

- \_\_ d. Select the first two “vacationRequestTypes” Excel files:



- \_\_ e. Click **Finish**.

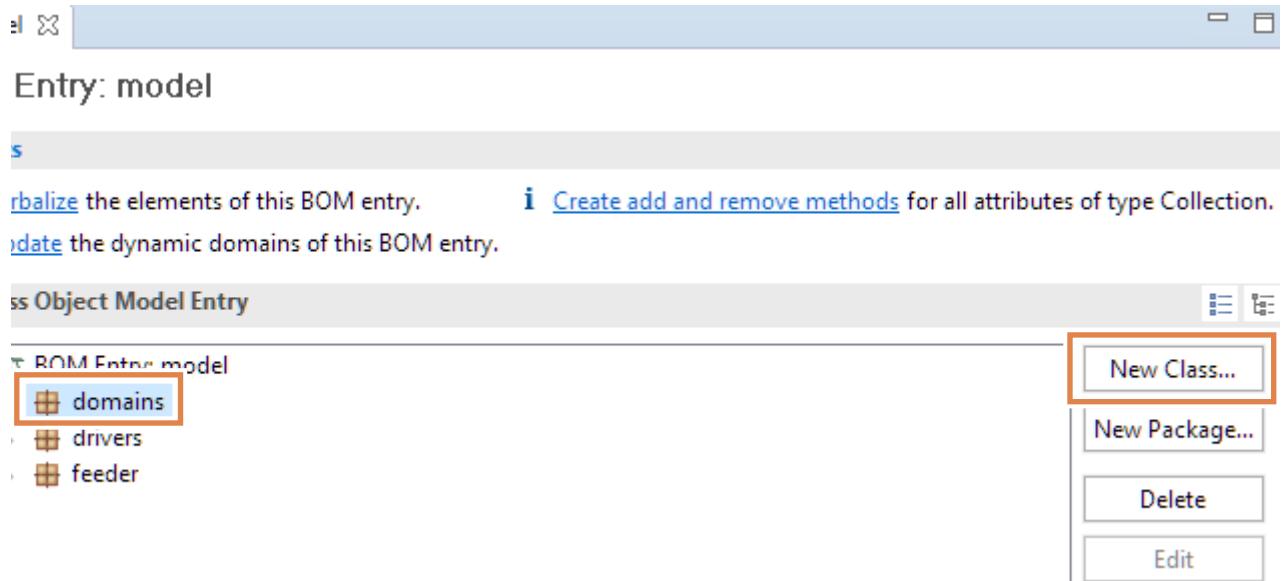
The Excel files are now visible in the resources folder of the trucksAndDrivers-bom project.



- \_\_ 3. In your workspace, rename the `vacationRequestTypes-1.xls` file to `vacationRequestTypes.xls`
  - \_\_ a. Right-click the file and click **Refactor > Rename**.
  - \_\_ b. In the Rename Resource window, in the **New name** field, change the value to: `vacationRequestTypes.xls`
  - \_\_ c. Click **OK** to close the window.

### 1.3. Create the domain

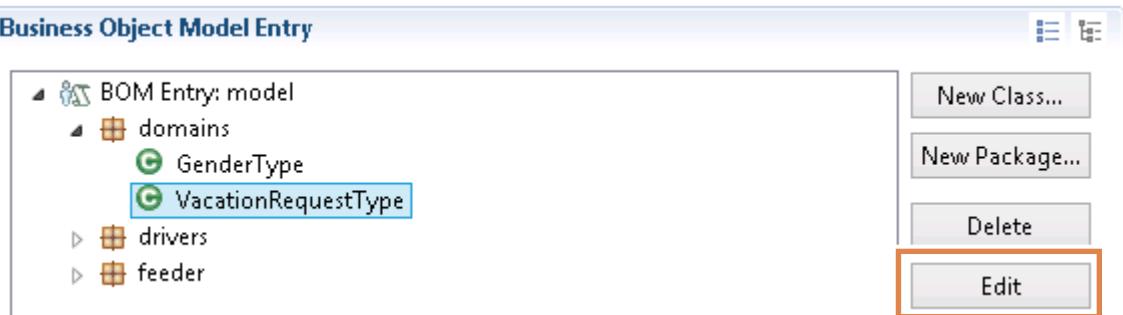
- 1. In Rule Explorer, expand **trucksAndDrivers-bom > bom > model** and double-click **domains** to open the **domains** package in the BOM editor.
- 2. Create a BOM class called **VacationRequestType** in the **domains** package.
  - a. Make sure that **domains** is selected and click **New Class**.



- b. In the New BOM Class window, in the **Name** field, enter: **VacationRequestType**
- c. Click **Finish**.

### 1.4. Associate a dynamic domain with the class

- 1. In the **Business Object Model Entry** section, make sure that **VacationRequestType** is selected, and click **Edit**.



2. In the **Domain** section, click **Create a domain**.

Class VacationRequestType (package: domains)

**General Information**

Name: VacationRequestType  
Namespace: domains  
Superclasses: java.lang.Object  
Interfaces:  
 Deprecated

**Class Verbalization**

This class is not verbalized. [Create](#) a default verbalization.  
 Generate automatic variable

**Members**

Specify the members of this class.  
 [New...](#)

**Domain**

Create and edit a domain for this class.  
[Create a domain.](#)

3. Under Dynamic Domains, select **Excel** and click **Next**.  
4. In the Excel pane, set the **Excel file** field to **vacationRequestTypes.xls**.

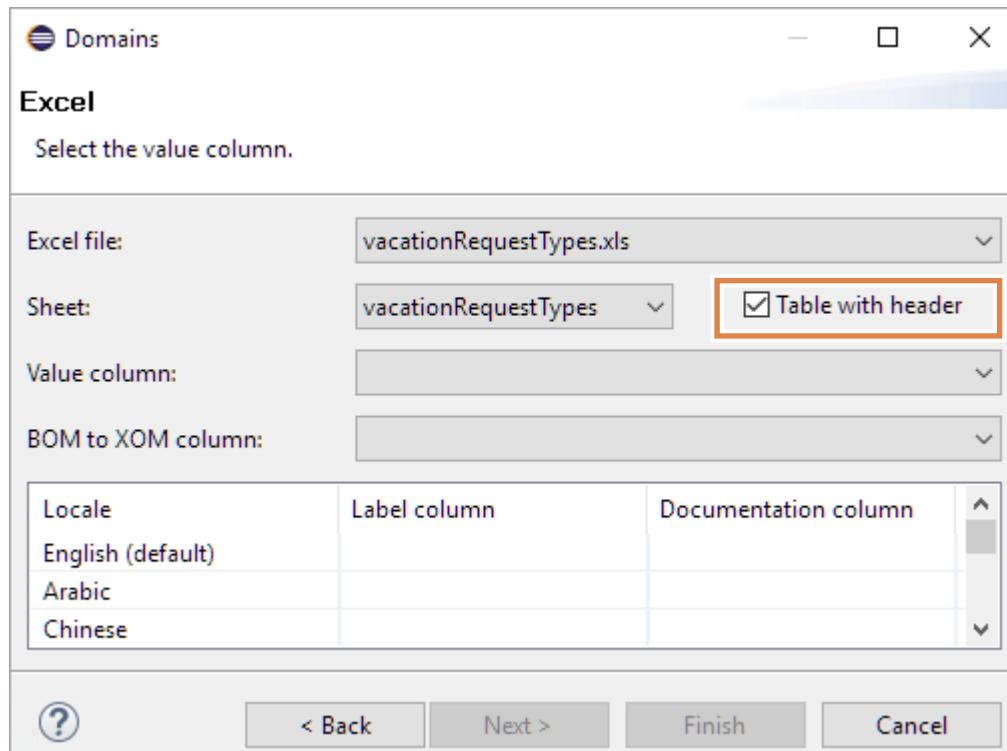


### Note

This choice is available in this window because the 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.

- 5. Select **Table with header** to indicate that the first row in the `vacationRequestTypes` sheet is the header for the columns in that sheet.



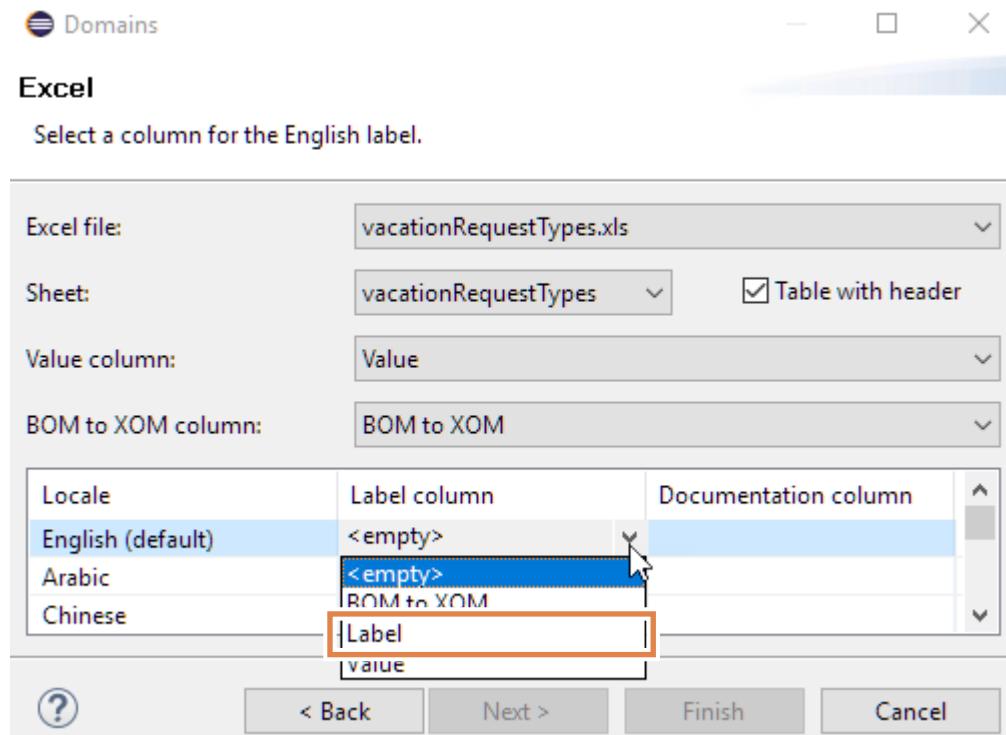
### Information

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.

- 
- 6. 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.
- 7. 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.

- 8. 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.

- 9. Click in another cell to make sure that the **Label** header is set.  
 — 10. Click **Finish** to close the Domains window.  
 — 11. Save the BOM (Ctrl+S).

The dynamic domain is created and its values are available in Rule Designer. The **Synchronize with dynamic values** link is now visible in the **Domain** section of the Class page.

**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

12. Open the **Problems** view and expand **Errors**.

| Description                                                                                                                              | Resource  | Path |
|------------------------------------------------------------------------------------------------------------------------------------------|-----------|------|
| [B2X] GBREX001E: Cannot find execution class 'domains.VacationRequestType' for translating business class 'domains.VacationRequestType'. | model.bom | /tn  |

The **Problems** view indicates an error because the XOM does not include a `domains.VacationRequestType` class to translate your new BOM class.

You can ignore the warnings.



## 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
```

---

## 1.5. Correct the B2X error

To correct the problem in the `VacationRequestType` BOM class, you define the BOM-to-XOM mapping.

- \_\_\_ 1. In the `VacationRequestType` class page, scroll down to the **BOM to XOM Mapping** section and expand it.
- \_\_\_ 2. Next to the **Execution name** field, click **Browse**.
- \_\_\_ 3. In the **Choose a type** field, type `String`
- \_\_\_ 4. Select **String** from the list, and click **OK**. The type field is set to: `java.lang.String`

### BOM to XOM Mapping

Edit the mapping between this BOM class and the XOM.

|                 |                                               |                                          |
|-----------------|-----------------------------------------------|------------------------------------------|
| Execution name: | <input type="text" value="java.lang.String"/> | <input type="button" value="Browse..."/> |
| Extender name:  | <input type="text"/>                          | <input type="button" value="Browse..."/> |

### Imports

- \_\_\_ 5. Save your work.
- \_\_\_ 6. Verify that the **Problems** view no longer indicates any errors on this new class. You can ignore the warnings.

## 1.6. Explore the dynamic domain

- 1. In the **Custom Properties** section of the `VacationRequestType` BOM class, verify that two custom properties were defined.

| Name                    | Value                                    |
|-------------------------|------------------------------------------|
| domainProviderResource  | vacationRequestTypes.xls                 |
| domainValueProviderName | com.ibm.rules.domainProvider.msexcel2003 |

- 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.
- Scroll up to the **Members** section and double-click the **VolunteerFireAndRescue** BOM member to open it.
  - Verify that this member is both static and final, and the **Type** of this BOM member is `domains.VacationRequestType`

|        |                             |
|--------|-----------------------------|
| Name:  | VolunteerFireAndRescue      |
| Type:  | domains.VacationRequestType |
| Class: | domains.VacationRequestType |

Read/Write    Read Only    Write Only  
 Static    Final  
 Deprecated    Update object state  
 Ignore for DVS

- Verify that the verbalization of this BOM member is **Volunteer Fire and Rescue** as shown in the **Label** field of Microsoft Excel file.

|        |                           |
|--------|---------------------------|
| Label: | Volunteer Fire and Rescue |
|--------|---------------------------|

- \_\_\_ d. Scroll down and expand the **BOM to XOM Mapping** section to 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 (in ARL)

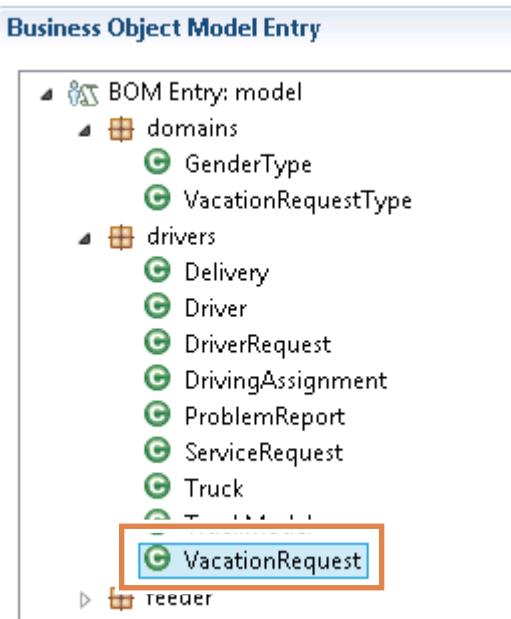
```
1 return "K29";
2
```

## 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.

### 2.1. Add a BOM attribute based on the dynamic domain

- 1. Return to the **Package** page of the BOM editor, expand the **drivers** package, and double-click the `VacationRequest` class to open it.



- 2. In the **Members** section, add a BOM attribute named: `type`
  - a. Click **New**.
  - b. Set **Name** to: `type`
  - c. Set the **Type** field to `domains.VacationRequestType` by clicking **Browse** and typing `VacationRequestType` and clicking **OK**.
  - d. Click **Finish**.
- 3. Set `type` to **Read Only** and create a default verbalization.
  - a. From the Members list, double-click `type` to open the Member page.
  - b. In the **General Information** section, select **Read Only**.

- \_\_\_ c. In the Member Verbalization section, click **Create**.

### Member type (class: drivers.VacationRequest)

|                                                                                                                                                                                                    |                                                                                                                     |                                          |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|------------------------------------------|
| <b>General Information</b>                                                                                                                                                                         |                                                                                                                     | <b>Member Verbalization</b>              |
| <div style="display: flex; align-items: center;"> <span style="color: orange;">⚠ This member is not verbalized</span> <input checked="" type="button" value="Create"/> <span>default</span> </div> |                                                                                                                     |                                          |
| Name:                                                                                                                                                                                              | <input type="text" value="type"/>                                                                                   |                                          |
| Type:                                                                                                                                                                                              | <input type="text" value="domains.VacationRequestType"/>                                                            | <input type="button" value="Browse..."/> |
| Class:                                                                                                                                                                                             | <input type="text" value="drivers.VacationRequest"/>                                                                |                                          |
|                                                                                                                                                                                                    | <input type="radio"/> Read/Write <input checked="" type="radio"/> <b>Read Only</b> <input type="radio"/> Write Only |                                          |

- \_\_\_ 4. Save the BOM (Ctrl+S).



### 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:

Cannot find attribute 'type' in execution class 'drivers.VacationRequest'

You solve this problem later in this exercise, in "[Updating the XOM](#)" on page 11-23. For now, ignore the error and try to use the new dynamic domain in a rule.

## 2.2. Write a rule that uses the new `type` BOM attribute

- \_\_\_ 1. In Rule Explorer, expand **trucksAndDrivers-rules > rules**.
- \_\_\_ 2. In the **domains** package, add an action rule that is named `05_dynamic_domain` 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 `C:\labfiles\code` directory.

Press **Ctrl+Shift+F** to format the rule.

- \_\_\_ d. In the rule editor, double-click one of { } to see list of domain values in.

The screenshot shows a rule editor interface. A dropdown menu is open, displaying a list of domain values: Administrative, Compensatory, Educational, Family/Personal, Jury Duty, and Military. The word "Administrative" is highlighted in the list. To the right of the list, a tooltip-like window titled "Domain value:" shows the selected value: "Administrative: domains.VacationRequestType".

- \_\_\_ 3. Save the rule and close it.

## 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.

---

### 3.1. Modify the source Excel spreadsheet and update the dynamic domain

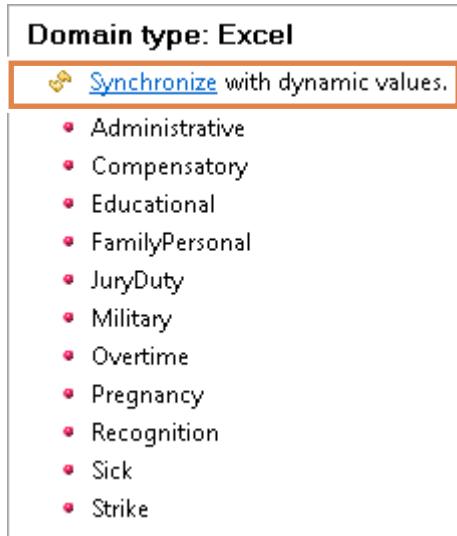
- \_\_\_ 1. In Rule Designer, expand `trucksAndDrivers-bom > resources`.
- \_\_\_ 2. In the `resources` folder, double-click `vacationRequestTypes.xls` to open it in Excel Viewer.
- \_\_\_ 3. In Excel Viewer, notice that line 3 includes these values:
  - **Value:** Compensatory
  - **BOM to XOM:** return "B34";
  - **Label:** Compensatory
- \_\_\_ 4. In Rule Explorer, double-click the `vacationRequestTypes-2.xls` file, and verify that its content is the same as the content of the `vacationRequestTypes.xls` file, except for line 3, which has these values:
  - **Value:** Other
  - **BOM to XOM:** return "O22";
  - **Label:** Other Vacation
- \_\_\_ 5. Close both Excel files.
- \_\_\_ 6. In Rule Designer, rename the `vacationRequestTypes.xls` file back to: `vacationRequestTypes-1.xls`
- \_\_\_ 7. In Rule Designer, rename the `vacationRequestTypes-2.xls` 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.

### 3.2. Synchronize the domain

- \_\_\_ 1. On the **Package** tab of the BOM editor, double-click the `domains.VacationRequestType` BOM class to open it.

- \_\_ 2. In the **Domain** section, click **Synchronize**.



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 new `Other` value is also visible, both under **Synchronize with dynamic values** and in the **Members** section.



### Important

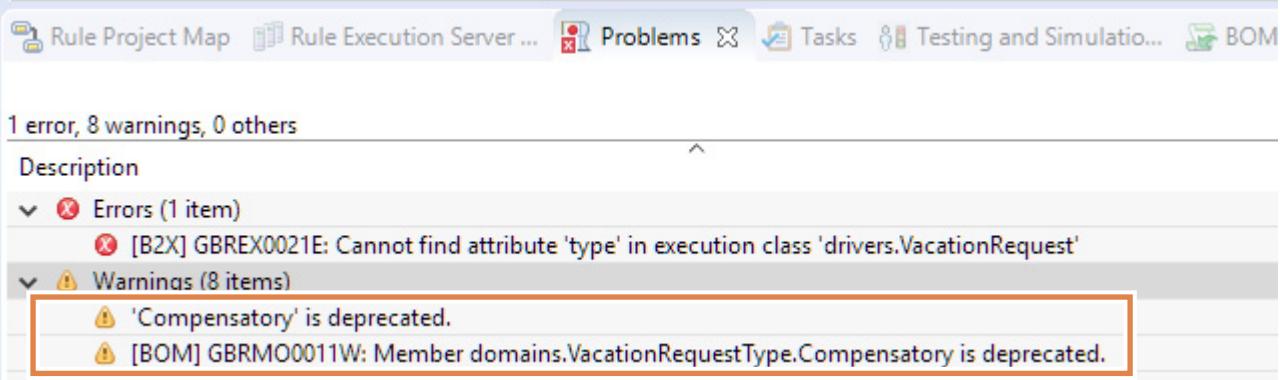
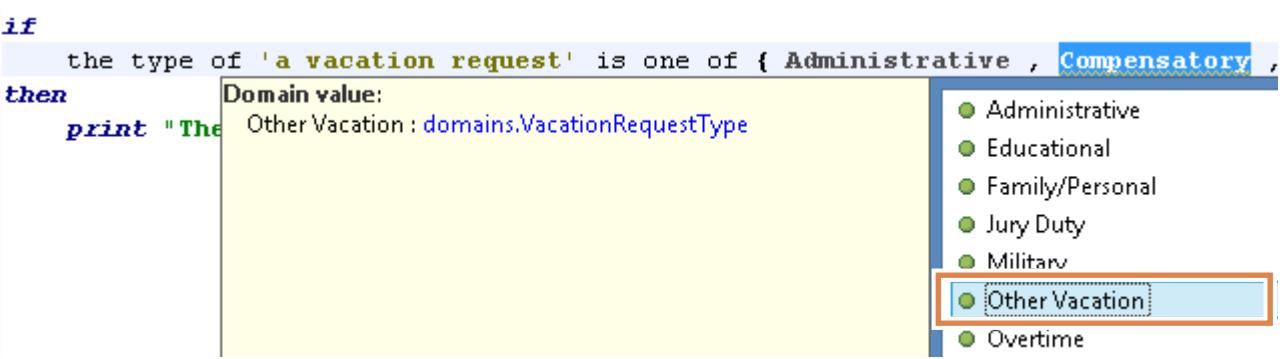
The **Members** section lists all the static references that are, or were, part of this dynamic domain. The references that correspond to values that are no longer present in the dynamic domain are marked as *deprecated*.

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.

- \_\_ 3. Save the BOM (Ctrl+S).

4. Verify that the `Compensatory` value is deprecated.
- In the **Problems** view, verify that the following warning messages are listed:
 

'Compensatory' is deprecated.  
[BOM] GBRMO0011W: Member domains.VacationRequestType.Compensatory is deprecated.


  - In the **Members** section of the `domains.VacationRequestType` BOM class, double-click `Compensatory`.
  - 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.
- Update the `05_dynamic_domain` rule to replace the deprecated value with `Other Vacation`.
    - In the `trucksAndDrivers-rules > domains` rule package, double-click `05_dynamic_domain`.
    - Verify that the term `Compensatory` is underlined, and that the associated warning is the same as in the Problems view:  
'Compensatory' is deprecated
    - In the **if** part of the rule, double-click `Compensatory` and replace it with `Other Vacation`.

```

if
 the type of 'a vacation request' is one of { Administrative , Compensatory ,
then Domain value:
 print "The Other Vacation : domains.VacationRequestType"

```

    - Administrative
    - Educational
    - Family/Personal
    - Jury Duty
    - Military
    - Other Vacation**
    - Overtime
  - In the **then** part of the rule, replace `Compensatory` in the message by typing `Other Vacation`.

**Note**

In the action statement, this term is part of a string.

Your rule now states:

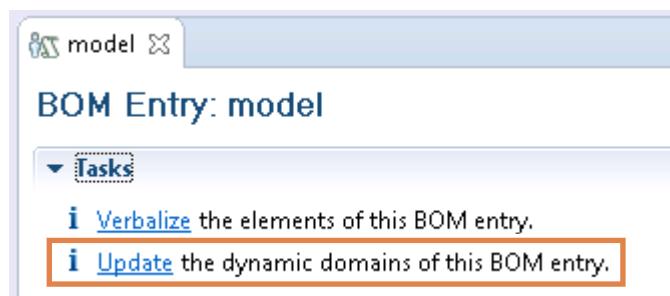
```
definitions
 set 'the driver' to a driver ;
 set 'a vacation request' to a vacation request in the vacation requests of
 'the driver' ;
 if
 the type of 'a vacation request' is one of { Administrative, Other
 Vacation, Educational }
 then
 print "The driver " + the name of 'the driver' + " requested vacation for
 Administrative, Other Vacation, or Educational reason";
```

- \_\_\_ 6. Save the 05\_dynamic\_domain rule.
- \_\_\_ 7. Verify that the first 'Compensatory' warning is no longer present in the **Problems** view:  
    'Compensatory' is deprecated
- \_\_\_ 8. Close the rule.

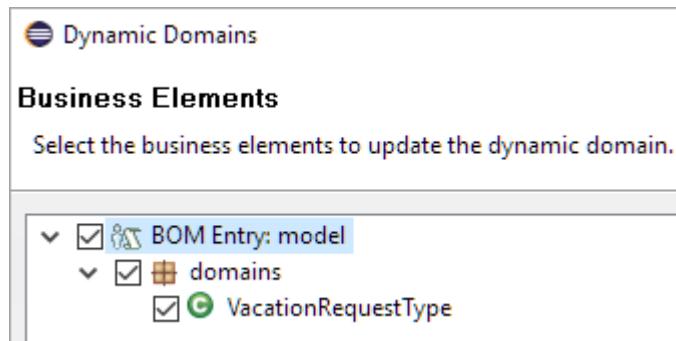
**Information**

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. Go to 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. Click **Finish**.

3. 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.

### 4.1. Identifying the inconsistency problem

First, you identify the inconsistency in the rule project.

- \_\_\_ 1. Reopen the **Problems** view. (If it is closed, you can click **Window > Show View > Other > General > Problems**.)
- \_\_\_ 2. Verify that the Problems view contains the following error:

[B2X] GBREX0021E: 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.

### 4.2. Correcting the inconsistency problem

- \_\_\_ 1. Expand **trucksAndDrivers-xom > src > drivers** and double-click **VacationRequest.java**.
- \_\_\_ 2. Add a private attribute: `private String type;`
- \_\_\_ 3. Create the corresponding setter and getter methods.

```
public void setType(String type) {
 this.type = type;
}
public String getType() {
 return type;
}
```

- \_\_\_ 4. 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.

- \_\_\_ 5. 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 clicking the Windows **Start** menu > **Start Sample Server** shortcut).  
Starting the server might take a few 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:9090/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 **Use Decision Governance**.)
  - e. On the Decision Service Dependent Projects page, make sure that `trucksAndDrivers-bom` and `trucksAndDrivers-rules` are selected, and click **Finish**.



#### Information

If you are prompted about password recovery in a Secure Storage window, click **No**.

- 3. When synchronization completes, no changes are found, so you can 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 Business console as an administrator.
  - \_\_\_ a. Open the Business console from the Windows **Start** menu, by clicking the **Decision Center Business console** shortcut.
  - \_\_\_ b. Sign in by entering `rtsAdmin` in both the **Username** field and the **Password** field.
- \_\_\_ 2. Make sure that you are on the **Library** tab.
- \_\_\_ 3. Open the main branch of the `trucksAndDrivers-tests` decision service, and enable all decision artifacts for viewing in the navigator.
  - \_\_\_ a. In the Decision Services list, find the `trucksAndDrivers-tests` decision service and click the white space to expand the section.
  - \_\_\_ b. Click **main** to open the main branch of `trucksAndDrivers-tests`.



- \_\_\_ c. In the **Decision Artifacts** tab, click **Types (1/7)**, click **All Types**, and click **Apply**.
- \_\_\_ 4. Open the `05_dynamic_domain` action rule in the rule editor.
  - \_\_\_ a. In the **Decision Artifacts** tab navigator pane, expand **trucksAndDrivers-rules** and click **domains**.

- \_\_ b. In the rule list for the **domains** folder, hover the mouse pointer over the **05\_dynamic\_domain** action rule and click the **Edit this rule** icon (pencil).

| Name                        | Last Changed By | Last Changed On   |
|-----------------------------|-----------------|-------------------|
| 01_collection_domain        | rtsAdmin        | March 3, 2017 ... |
| 02_literal_domain           | rtsAdmin        | March 3, 2017 ... |
| 03_static_references_domain | rtsAdmin        | March 3, 2017 ... |
| 04_bounded_domain           | rtsAdmin        | March 3, 2017 ... |
| 05_dynamic_domain           | rtsAdmin        | March 3, 2017 ... |

- \_\_ 5. In the **if** part of the rule, double-click one of the vacation request values, such as **Administrative**, to see the list of domain values.

```

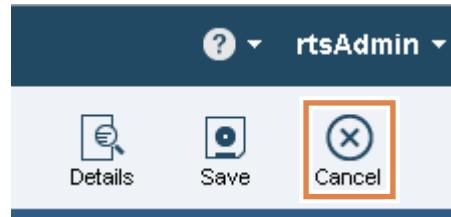
if
 the type of 'a vacation request' is one of { Administrative . Other Vacation . Educa
then
 print "The driver " + the name of 'the drive'

```



The screenshot shows a software interface with a code editor window. The code contains an **if** statement that checks the type of a vacation request. The **then** part of the statement includes a **print** command with a placeholder for the driver's name. Below the code, a dropdown menu is open, listing various vacation request types. The **Administrative** option is highlighted with a blue selection bar, indicating it has been selected or is being viewed. Other options listed include Educational, Family/Personal, Jury Duty, Military, Other Vacation, Overtime, Pregnancy, Recognition, Sick, Strike, and several other categories like @ the type of <a vacation request>, Vacation, Volunteer Fire and Rescue, and Without Pay.

- \_\_\_ 6. Click **Cancel** to exit the rule editor.



## 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 `C:\labfiles\code` directory. You delete the Excel file that is in use and replace it with: `vacationRequestTypes-3.xls`

### 7.1. Viewing and modifying the values of the dynamic domain

- \_\_\_ 1. View the current dynamic domain data on the **Model** tab.
  - \_\_\_ a. On the toolbar, click **Model**. If you do not see the Model tab, click the right arrow to show the other available tabs.
  - \_\_\_ b. On the Dynamic Domains page, expand `domains.VacationRequestType` to see the list of vacation request types from the `vacationRequestTypes.xls` spreadsheet.



### Information

After you modify the `vacationRequestTypes.xls` spreadsheet, you return to the **Model** tab to apply the changes.

- \_\_\_ 2. View and prepare the `vacationRequestTypes-3.xls` file for use in Decision Center.
  - \_\_\_ a. In Windows Explorer, in the `C:\labfiles\code` directory, open the `vacationRequestTypes-3.xls` Excel file to view the modified domain values.  
The first line, which defined the `Administrative` value, is replaced with the `Training` value.
  - \_\_\_ b. Make a copy of the `vacationRequestTypes-3.xls` file and rename it as `vacationRequestTypes.xls`
- \_\_\_ 3. In Decision Center, return to the **Decision Artifacts** tab and delete the existing `vacationRequestTypes.xls` file in the `trucksAndDrivers-bom` project.
  - \_\_\_ a. In the Business console, and click the **Decision Artifacts** tab. (If you cannot see the **Decision Artifacts** tab, click the left arrow on the tab bar.)
  - \_\_\_ b. In the Decision Artifacts navigator pane, expand `trucksAndDrivers-bom` and click the **Resources** folder.

In the Resources pane, you can see the domain Excel spreadsheets that you imported earlier in this exercise.

The screenshot shows the 'Resources' pane with a list of files. At the top are navigation icons (back, forward, search) and a 'Filter:' input field. Below is a table with columns: Name, Last Changed By, and Last Changed On. Two files are listed:

| Name                                       | Last Changed By | Last Changed On   |
|--------------------------------------------|-----------------|-------------------|
| <a href="#">vacationRequestTypes-1.xls</a> | rtsAdmin        | March 3, 2017 ... |
| <a href="#">vacationRequestTypes.xls</a>   | rtsAdmin        | March 3, 2017 ... |

- \_\_\_ c. In the Resources list, hover the mouse pointer over the `vacationRequestTypes.xls` file and click the **Delete this resource** icon (trash can).

A close-up view of the 'vacationRequestTypes.xls' row in the Resources list. The trash can icon in the 'Delete this resource' column is highlighted with a red box.

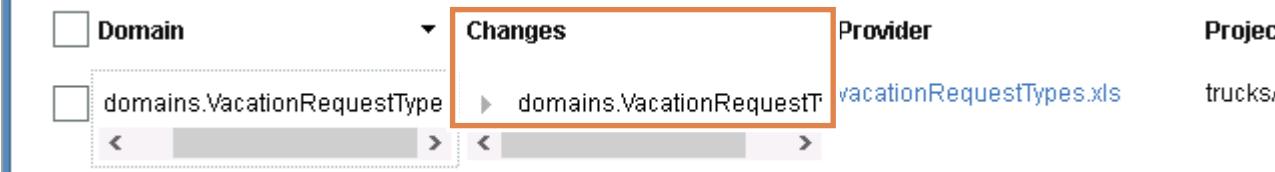
- \_\_\_ d. When prompted to confirm the deletion, click **Yes**.
- \_\_\_ 4. Import the new file to Decision Center.
- \_\_\_ a. In the Resources pane, click **Create an artifact** (the plus [+]) sign), and then click **New Resource**.

The screenshot shows the 'Resources' pane again. The 'Create an artifact' button (plus sign) is highlighted with a red box. Below it, the 'New Resource' button is also highlighted with a red box. The table below shows the same two files as before.

- \_\_\_ b. In the New Resource pane, click **Choose** next to the **File** field.
- \_\_\_ c. In the File Upload window, go to `C:\labfiles\code`, select the `vacationRequestTypes.xls` Excel spreadsheet, and click **Open**.
- \_\_\_ d. In the upper-right area of the New Resource pane, click the **Save changes** icon.

The screenshot shows the 'New Resource' pane. At the top, it says 'Resources > New Resource'. In the center, there's a 'File:' field containing 'vacationRequestTypes.xls (31.5 KB)' with a 'Choose...' button next to it. A red box highlights the 'Choose...' button. Below it is a 'Description' field.

- \_\_\_ e. Verify that the `vacationRequestTypes.xls` file is now visible in the **Resources** list.
- \_\_\_ 5. Apply the changes from the `vacationRequesttypes.xls` file to the BOM.
- \_\_\_ a. Click the **Model** tab.
- \_\_\_ b. Notice that the Changes column lists `domains.VacationRequestType`.

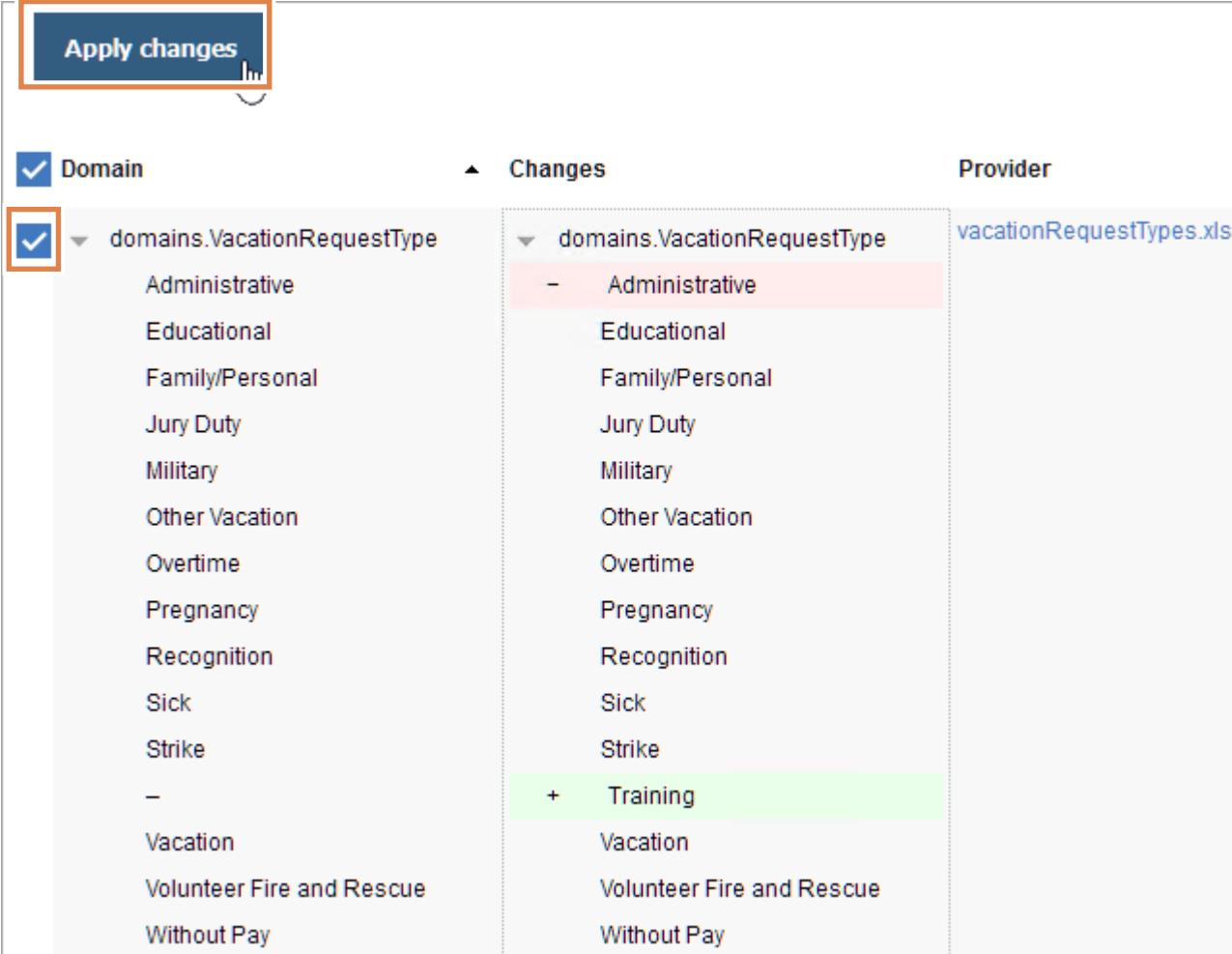


| <input type="checkbox"/> Domain                      | Changes                    | Provider                 | Project |
|------------------------------------------------------|----------------------------|--------------------------|---------|
| <input type="checkbox"/> domains.VacationRequestType | ▶ domains.VacationRequestT | vacationRequestTypes.xls | trucksA |

- \_\_\_ c. Under Changes, expand `domains.VacationRequestType`.

In the Changes list, you can see that the Administrative type was removed and the Training type was added.

- \_\_\_ d. Select the **Domain** check box, and click **Apply changes**.



| <input checked="" type="checkbox"/> Domain                      | Changes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Provider         |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
|-----------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|------------|-------------|----------|-----------------|---------------------------|-----------|-------------|----------|--|----------------|--|----------|--|-----------|--|-------------|--|------|--|--------|--|---|--|----------|--|---------------------------|--|-------------|--|--------------------------|
| <input checked="" type="checkbox"/> domains.VacationRequestType | <table border="1"> <tr> <td>- Administrative</td> <td>+ Training</td> </tr> <tr> <td>Educational</td> <td>Vacation</td> </tr> <tr> <td>Family/Personal</td> <td>Volunteer Fire and Rescue</td> </tr> <tr> <td>Jury Duty</td> <td>Without Pay</td> </tr> <tr> <td>Military</td> <td></td> </tr> <tr> <td>Other Vacation</td> <td></td> </tr> <tr> <td>Overtime</td> <td></td> </tr> <tr> <td>Pregnancy</td> <td></td> </tr> <tr> <td>Recognition</td> <td></td> </tr> <tr> <td>Sick</td> <td></td> </tr> <tr> <td>Strike</td> <td></td> </tr> <tr> <td>-</td> <td></td> </tr> <tr> <td>Vacation</td> <td></td> </tr> <tr> <td>Volunteer Fire and Rescue</td> <td></td> </tr> <tr> <td>Without Pay</td> <td></td> </tr> </table> | - Administrative | + Training | Educational | Vacation | Family/Personal | Volunteer Fire and Rescue | Jury Duty | Without Pay | Military |  | Other Vacation |  | Overtime |  | Pregnancy |  | Recognition |  | Sick |  | Strike |  | - |  | Vacation |  | Volunteer Fire and Rescue |  | Without Pay |  | vacationRequestTypes.xls |
| - Administrative                                                | + Training                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
| Educational                                                     | Vacation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
| Family/Personal                                                 | Volunteer Fire and Rescue                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
| Jury Duty                                                       | Without Pay                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
| Military                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
| Other Vacation                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
| Overtime                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
| Pregnancy                                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
| Recognition                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
| Sick                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
| Strike                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
| -                                                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
| Vacation                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
| Volunteer Fire and Rescue                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |
| Without Pay                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                  |            |             |          |                 |                           |           |             |          |  |                |  |          |  |           |  |             |  |      |  |        |  |   |  |          |  |                           |  |             |  |                          |

- \_\_\_ e. When you are asked to confirm the update, click **Yes**.



## Troubleshooting

If Decision Center hangs when you try to apply the changes, close the browser and reopen the Business console, and then retry this step.

- \_\_\_ 6. Under Domain, expand **domains.VacationRequestType** again to see the updated list.

## 7.2. Modifying the domain value in the rule

In this section, you examine the results of your changes to the BOM in the `trucksAndDrivers-rules` project.

- \_\_\_ 1. Go back to the **Decision Artifacts** tab and reopen the `05_dynamic_domain` rule in the `trucksAndDrivers-rules > domains` rule package.
- \_\_\_ 2. Confirm that the rule has a warning:

`'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" ;
```

**⚠ 'Administrative' is deprecated.**

- \_\_\_ 3. Click **Edit this rule** to edit the **if** and **then** parts of the `05_dynamic_domain` rule.



\_\_\_ 4. Replace Administrative with: Training

**if**

the type of 'a vacation request' is one of { **Administrative**, Other Vacation , Education }

**then**

print "The driver " + the name of 'the driver'

**Administrative**

Other Vacation . Education



## 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-rules - main** project is listed because you added the `05_dynamic_domain` rule in the Business console.

If you expand both rule projects, the blue arrows that point left on the various decision 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).

| Local File                        | Remote File                     |
|-----------------------------------|---------------------------------|
| 28 static <b>Administrative</b> , | 28 static <b>Training,</b>      |
| 29 static <b>Other,</b>           | 29 static <b>Other,</b>         |
| 30 static <b>Educational,</b>     | 30 static <b>Educational,</b>   |
| 31 static <b>FamilyPersonal,</b>  | 31 static <b>FamilyPersonal</b> |
| 32 static <b>JuryDuty,</b>        | 32 static <b>JuryDuty,</b>      |
| 33 static <b>Military,</b>        | 33 static <b>Military,</b>      |
| 34 static <b>Overtime,</b>        | 34 static <b>Overtime,</b>      |
| 35 static <b>Pregnancy,</b>       | 35 static <b>Pregnancy,</b>     |
| 36 static <b>Recognition,</b>     | 36 static <b>Recognition,</b>   |
| 37 static <b>Sick,</b>            | 37 static <b>Sick,</b>          |
| 38 static <b>Strike,</b>          | 38 static <b>Strike,</b>        |

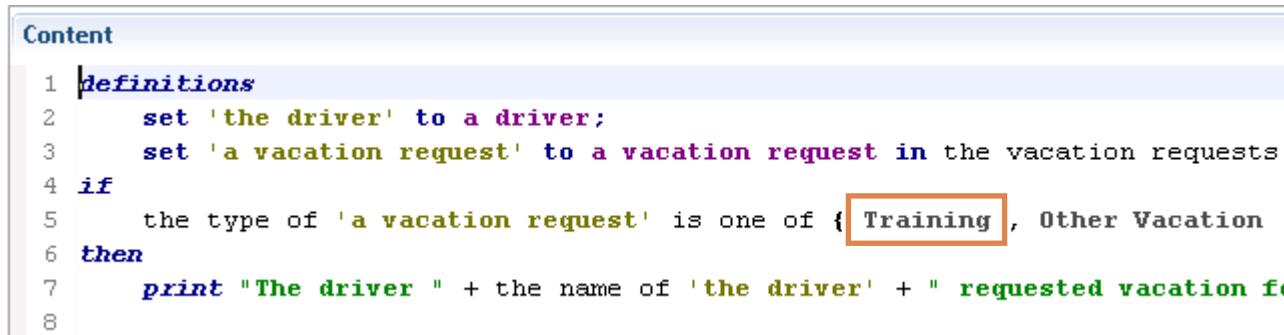
- \_\_\_ d. Expand **trucksAndDrivers-rules - main > rules/domains**.
- \_\_\_ e. Double-click **05\_dynamic\_domain.brl** to open it in the Text Compare view, and notice that Training replaces Administrative.

| Local File                         | Remote File            |
|------------------------------------|------------------------|
| 1                                  | 1                      |
| 2 sion="2.0" xmlns:xmi="http://    | 2 sion="2.0" xml       |
| 3                                  | 3                      |
| 4 uid>                             | 4 uid>                 |
| 5                                  | 5                      |
| 6                                  | 6                      |
| 7                                  | 7                      |
| 8 est in the vacation requests     | 8 est in the vac       |
| 9                                  | 9                      |
| 10 <b>Administrative</b> Other Vac | 10 <b>Training</b> , C |
| 11                                 | 11                     |
| 12 er' + " requested vacation f    | 12 er' + " reques      |
| 13                                 | 13                     |

- \_\_\_ 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.



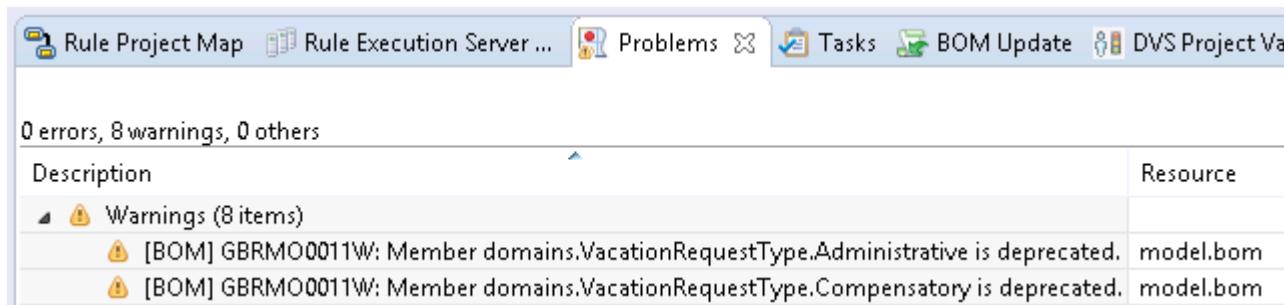
```

Content
1 definitions
2 set 'the driver' to a driver;
3 set 'a vacation request' to a vacation request in the vacation requests
4 if
5 the type of 'a vacation request' is one of { Training, Other Vacation
6 then
7 print "The driver " + the name of 'the driver' + " requested vacation fo
8

```

- \_\_\_ 8. Open the Problems view, and note the deprecation warning for `Administrative`.

`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**, and 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.



### 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.

---

# Exercise 12. Working with searches and queries

## Estimated time

00:30

## Overview

This exercise teaches you how to define queries and rule extractors on rule projects. You also learn how to synchronize queries between Rule Designer and Decision Center.

## Objectives

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
- Synchronize queries between Rule Designer and Decision Center

## 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, and to synchronize queries between Rule Designer and Decision Center.

The exercise involves these tasks:

- [Section 1, "Searching for rule artifacts"](#)
- [Section 2, "Querying rule projects"](#)
- [Section 3, "Working with queries in Decision Center"](#)

## Requirements

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

- Start project: Dev 12 – Queries\start

## 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.

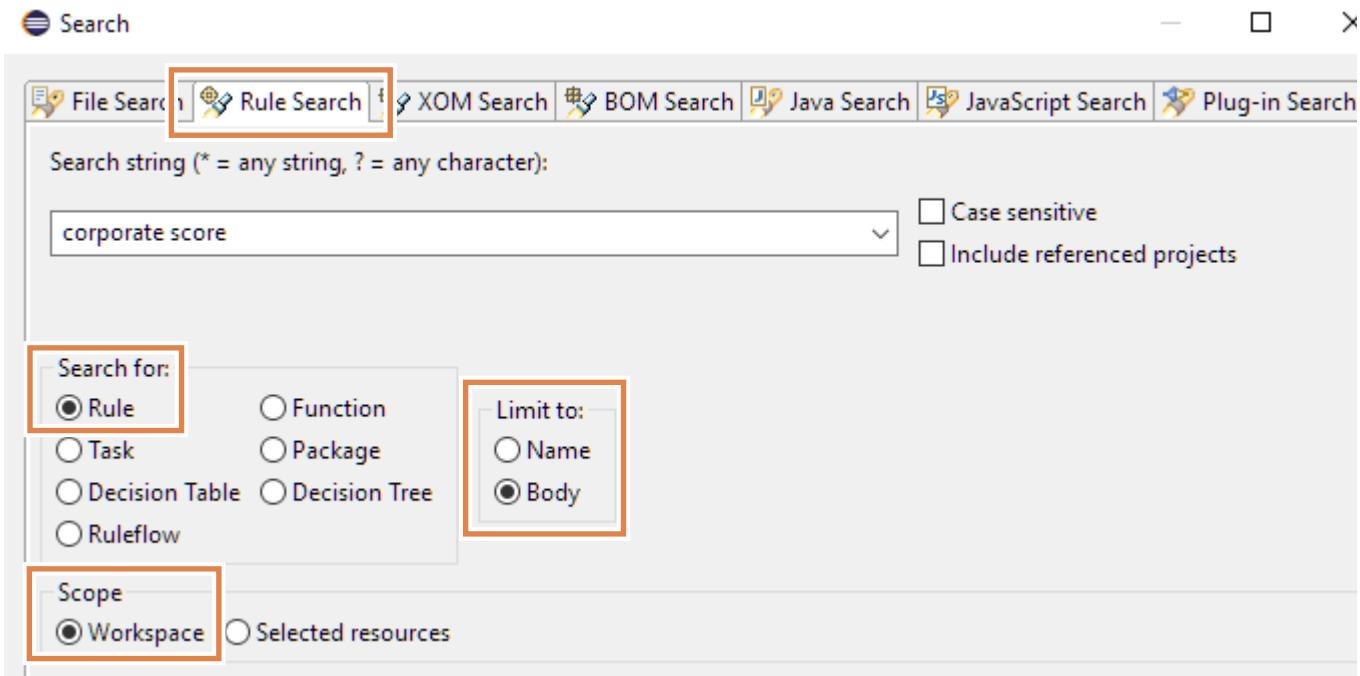
### 1.1. Setting up your environment for this exercise

- \_\_\_ 1. In Rule Designer, switch to a new workspace and name it: `queries`
  - \_\_\_ a. From the **File** menu, click **Switch Workspace > Other**.
  - \_\_\_ b. In the Workspace Launcher window, enter the path:  
`C:\labfiles\workspaces\queries`
  - \_\_\_ c. Click **Launch**.
- \_\_\_ 2. Close the Welcome view.
- \_\_\_ 3. Use the Samples Console to import the exercise start project.
  - \_\_\_ a. Click the **Open Perspective** icon.
  - \_\_\_ b. In the Open Perspective window, click **Samples Console** and click **Open**.
  - \_\_\_ c. In the **Rule Designer** section, expand **Training > Ex 12: Queries > start**, and click **Import projects**.
  - \_\_\_ d. 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.
- \_\_\_ 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. In the **Search for** section, make sure that **Rule** is selected.
  - \_\_\_ c. In the **Limit to** section, select **Body**.

- \_\_\_ d. Make sure that **Workspace** is selected in the **Scope** section.



- \_\_\_ e. Click **Search**.

All the results are listed in the **Search** view.

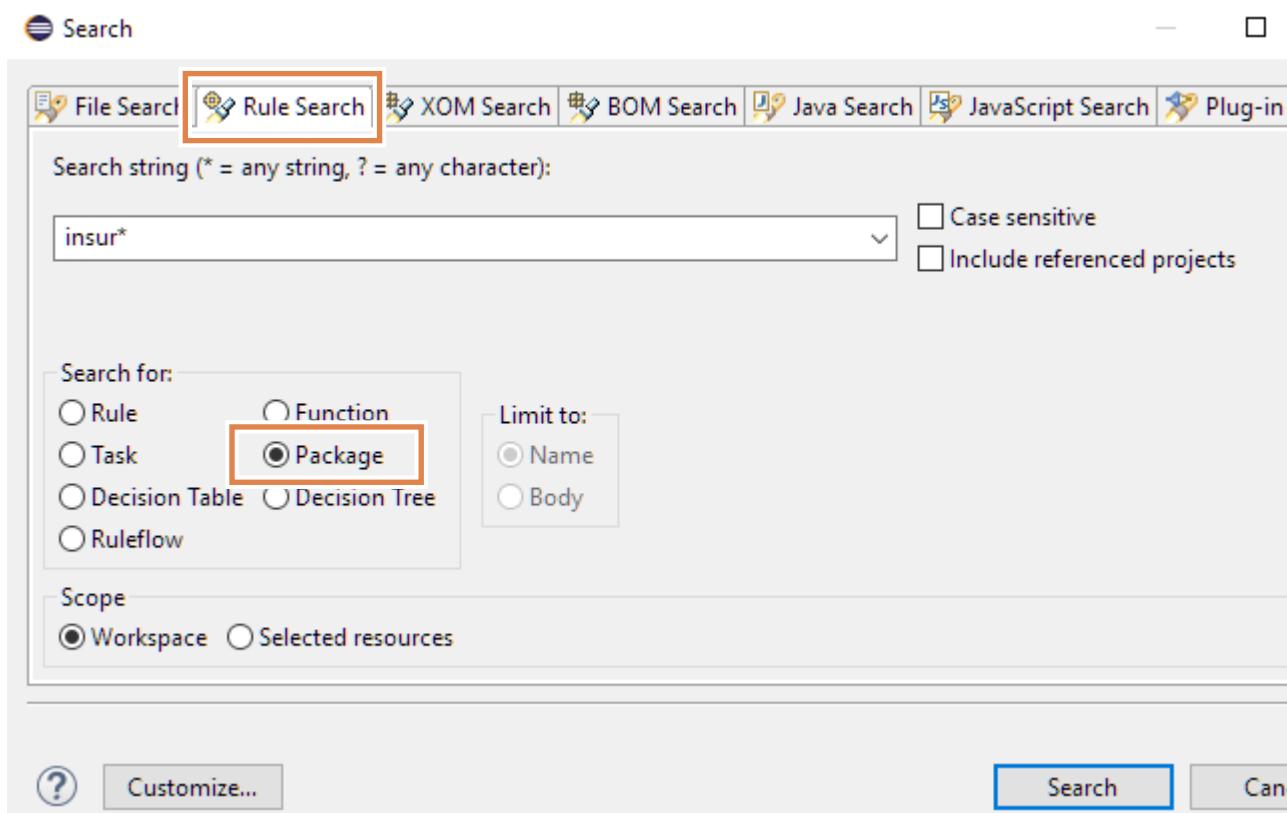
| bankruptcyScore - [loan-rules] - /loan-rules/rules/computation       | (2 matches) |
|----------------------------------------------------------------------|-------------|
| grade - [loan-rules] - /loan-rules/rules/eligibility                 | (2 matches) |
| initialCorporateScore - [loan-rules] - /loan-rules/rules/computation |             |
| neverBankruptcy - [loan-rules] - /loan-rules/rules/computation       |             |
| salary2score - [loan-rules] - /loan-rules/rules/computation          | (2 matches) |

- \_\_\_ 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 on the **Rule Search** tab in all packages in the workspace.



The `insurance` package is returned as a match.

### 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 Rule Explorer, expand `loan-rules > 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.

---

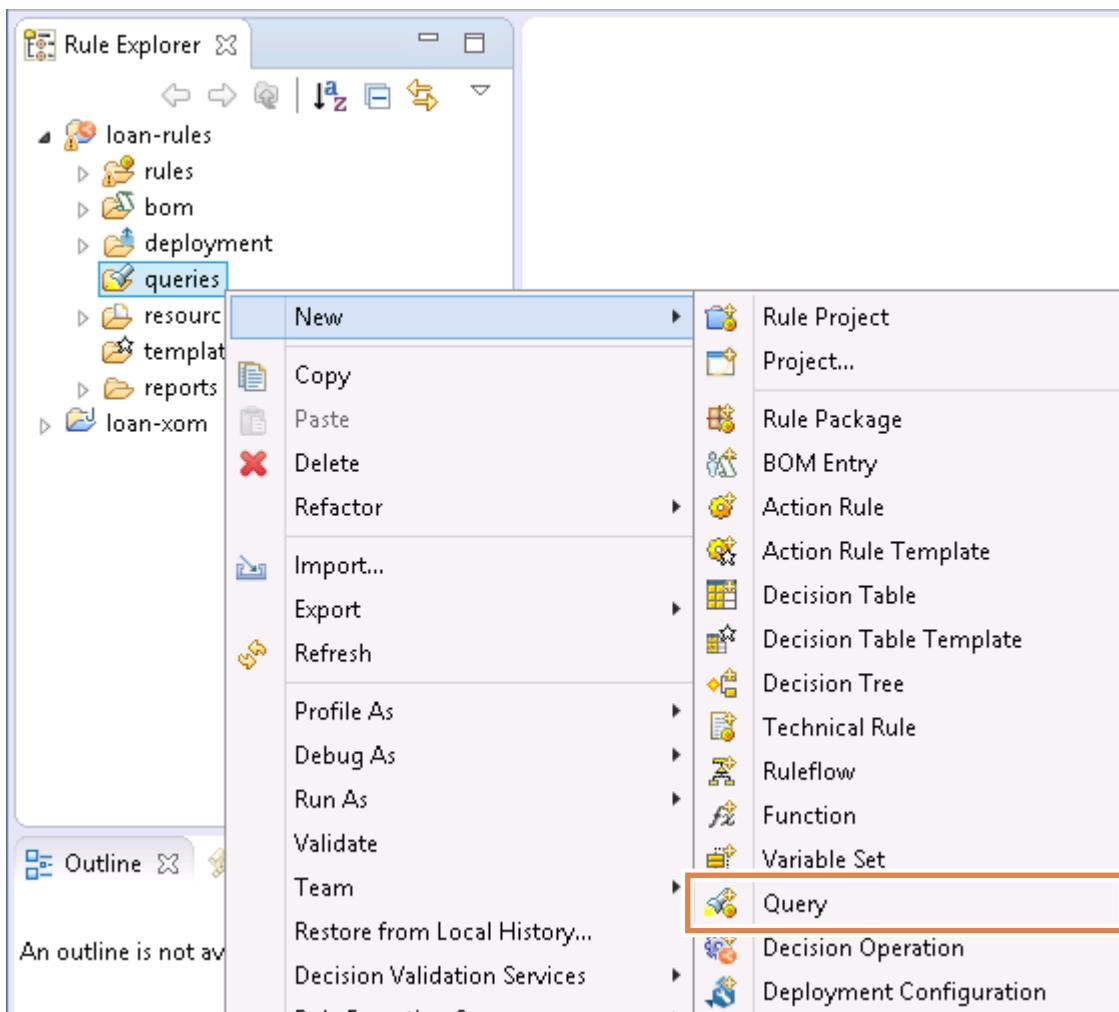
## Section 2. Querying rule projects

In this part of the exercise, you create queries on rule projects to search for rules. You create different queries by using different criteria and 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 ]

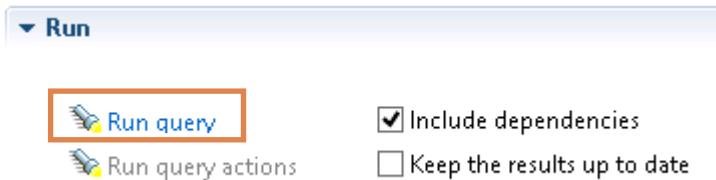
- \_\_\_ c. Save the query.



### Hint

To see the list of choices, press **Ctrl+Space** 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).

Why is the `neverBankruptcy` rule not listed? This 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. Hover the mouse over a row (for example, row 3), and read the complete action rule that corresponds to this row of the insurance decision table.

The screenshot shows the 'insurance' decision table in the Decision Table view. Row 3 is selected, and a tooltip displays the following action rule:

```

if
 (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 ;

```

The decision table has columns for Grade, Amount of loan (Min and Max), Insurance required, and Insurance rate. Rows 1-4 define ranges for the amount of loan, and row 5 contains the conditional logic.

- \_\_\_ 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.Loa`n.`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



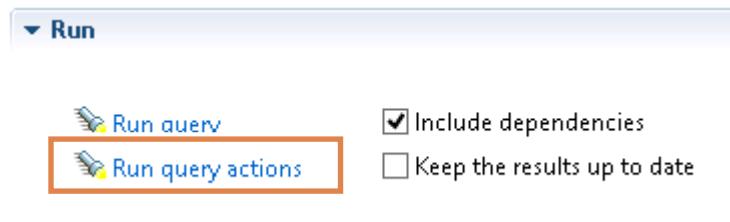
### Questions

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 pane), 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.

## Section 3. Working with queries in Decision Center

In this section, you work with queries in Decision Center. You also learn how to synchronize queries between Rule Designer and Decision Center.

### Publishing and viewing the decision service in Decision Center

- \_\_\_ 1. Publish the `loan-rules` decision service to Decision Center.
  - \_\_\_ a. In the Rule Explorer, right-click `loan-rules` and go to **Decision Center > Connect**.
  - \_\_\_ b. In the Decision Center Configuration window, enter the following connection details and click **Connect**:
    - **URL:** `http://localhost:9090/teamserver`
    - **User name:** `rtsAdmin`
    - **Password:** `rtsAdmin`
  - \_\_\_ c. After the connection is established, click **Finish**.
  - \_\_\_ d. In the Synchronize Complete window, click **OK** to close it.
  - \_\_\_ e. In the Confirm Open Perspective window, click **No**. You do not need to switch to the Team Synchronizing perspective.
- \_\_\_ 2. Log in to Decision Center as: `rtsUser1`
  - \_\_\_ a. From the Windows **Start** menu, click the **Decision Center Business console** shortcut.
  - \_\_\_ b. Enter `rtsUser1` in the **Username** and **Password** fields, and click **Log in**.
- \_\_\_ 3. If you are not on the Library page, click the **Library** tab to open the list of Decision Services.
- \_\_\_ 4. Open the `main` branch of the `loan-rules` decision service that you published.
  - \_\_\_ a. In the Decision Services list, locate `loan-rules`.
  - \_\_\_ b. Click the white space of the `loan-rules` box to expand it.
  - \_\_\_ c. Click the **main** link.



Alternatively, you can click the **loan-rules** link, then click **Branches > main**.

## Working with queries in Decision Center

You can work with queries in the Business console by using the **Queries** tab.

- 1. Click the **Queries** tab.



- 2. In the Queries List, expand **loan-rules** to see the queries that you defined earlier in this exercise:
  - may become applicable
  - may lead to a state
  - the priority of the rules
  - using BOM member

3. Click the `may become applicable` query to view its details in the main pane, and then click Run.

The screenshot shows the 'Query Expression' editor in the IBM Decision Center. The 'Name' field contains 'may become applicable'. The 'Project' dropdown is set to 'loan-rules' and has the 'Include referenced projects' checkbox checked. The 'Description' field is empty. The 'Query Expression' pane displays the following text:

```
Find all business rules
such that each business rule may become applicable when ['a borro
```

Below the expression is a table with columns 'Severity', 'Line', and 'Message'. The 'Severity' column has a dropdown arrow icon. The 'Line' column contains the number '1'. The 'Message' column contains the text 'This query contains constructions that are invalid for this environment.' At the bottom right are three buttons: 'New Query', 'Save', and 'Run', with 'Run' being highlighted by a red border.



## Information

The `using BOM member` query has the following error message:

This query contains constructions that are invalid for this environment.

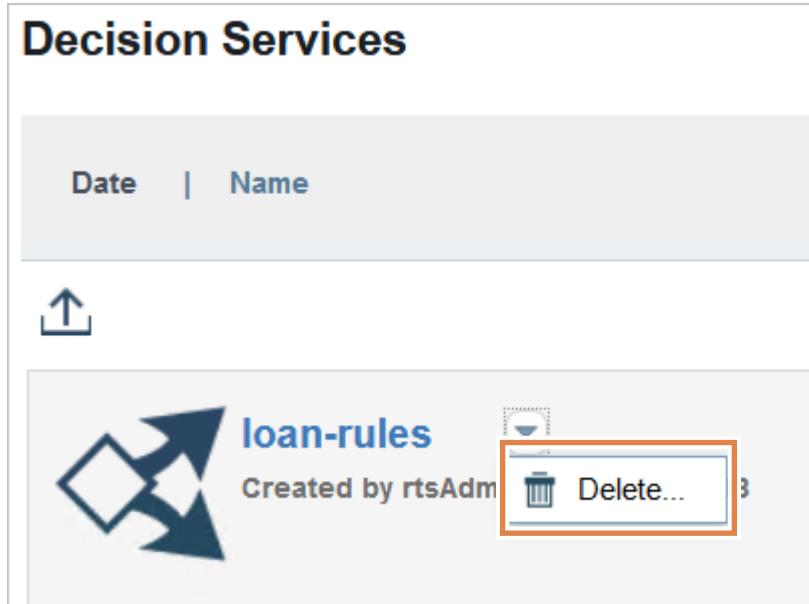
| Severity | Line | Message                                                                  |
|----------|------|--------------------------------------------------------------------------|
| ✖        | 1    | This query contains constructions that are invalid for this environment. |

This error appears in Decision Center because the `using BOM member` construct is specific to the Rule Designer module. This query cannot be run in Decision Center.

If a query must be run in both modules, make sure that you use query constructs that can be used in both Rule Designer and Decision Center.

## Deleting the loan-rules decision service from Decision Center

- \_\_\_ 1. Sign out of Business console, and sign in again with `rtsAdmin` as the user name and password.
- \_\_\_ 2. In the upper left of the window, click the **loan-rules** breadcrumb, then click the **All Decision Services** breadcrumb.
- \_\_\_ 3. Click the arrow beside **loan-rules** and click **Delete**.



- \_\_\_ 4. Click **Delete**.
- \_\_\_ 5. Sign out of the Business console and close the browser window.

**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.

# Exercise 13. Debugging a ruleset

## Estimated time

01:30

## Overview

This exercise walks you through the steps of debugging a ruleset in Rule Designer.

## Objectives

After completing this exercise, you should be able to:

- Use automatic exception handling
- Set breakpoints in rules, decision tables, and ruleflows
- Run a debugging session

## 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, "Setting breakpoints"](#)
- [Section 3, "Debugging a rule"](#)
- [Section 4, "Debugging a decision table"](#)
- [Section 5, "Debugging the ruleflow"](#)

The steps that are described here are based on the *Debugging a decision service* tutorial, which is distributed with the product documentation.

## Requirements

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

- Start project: Dev 13 – Debug\start

**Note****For IBM ODM on Cloud users**

This exercise requires that you use the on-premises version of ODM. For ODM on Cloud, you can use the projects and instructions that are provided at this URL:

<https://github.com/ODMDev/odm-on-cloud-debug>

## 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 and name it: `debug`
  - \_\_\_ a. From the **File** menu, click **Switch Workspace > Other**.
  - \_\_\_ b. In the Workspace Launcher window, enter the path:  
`C:\labfiles\workspaces\debug`
  - \_\_\_ c. Click **Launch**.
- \_\_\_ 2. Close the Welcome view.
- \_\_\_ 3. Use the Samples Console to import the start projects for the exercise.
  - \_\_\_ a. Click the **Open Perspective** icon.
  - \_\_\_ b. In the Open Perspective window, click **Samples Console** and click **Open**.
  - \_\_\_ c. In the **Rule Designer** section, expand **Training > Ex 13: Debugging > start**, and click **Import projects**.
- \_\_\_ 4. Close the Help view.

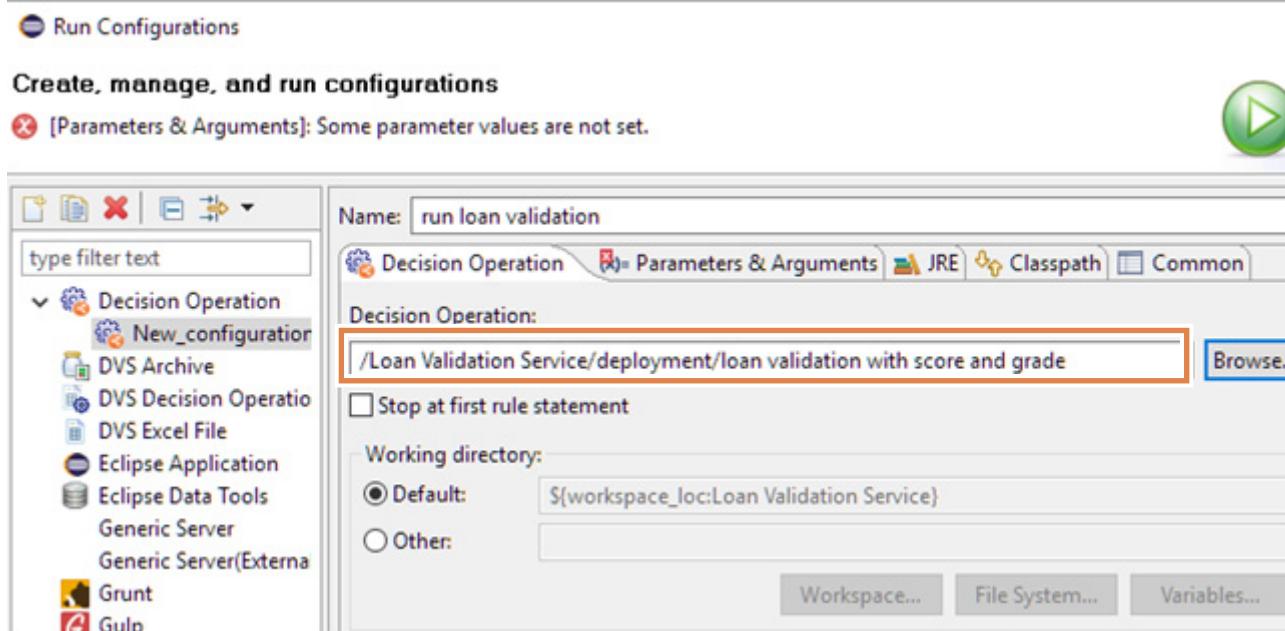
The Rule perspective opens with these projects:

- Loan Validation Base
- Loan Validation Check
- Loan Validation Determination
- Loan Validation Scoring
- Loan Validation Service
- loan-validation-xom

### 1.2. Running the launch configuration

- \_\_\_ 1. From the menu, click **Run > Run Configurations**.
- \_\_\_ 2. Double-click **Decision Operation** to create a new run configuration.
- \_\_\_ 3. In the **Name** field, set the name to: `run loan validation`

4. In the **Decision Operation** field, click **Browse** and select the **loan validation with score and grade** operation.

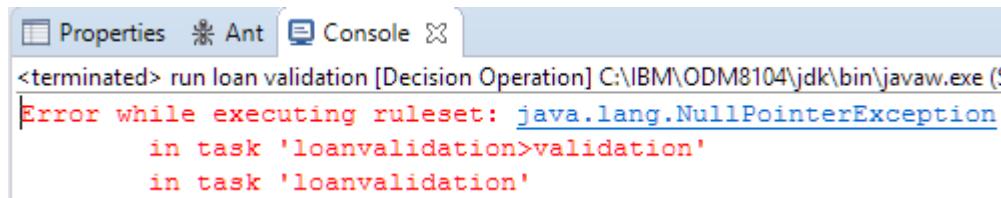


5. On the **Parameters & Arguments** tab, define the **borrower** and **loan** parameters.
- Select the **borrower** parameter in the **Name** column, and click **Edit Value**.
  - Select the **Function body** option and overwrite the initial value with the following code to overwrite and click **OK**:
 

```
java.util.Date birthDate = loan.DateUtil.makeDate(1950,1,1);
loan.Borrower borrower = new loan.Borrower("Smith","John",birthDate,
"123121234");
borrower.creditScore = 200;
borrower.yearlyIncome = 20000;
return borrower;
```
  - Select the **loan** parameter and click **Edit Value**.
  - Select **Function body** and overwrite the initial value with the following code and click **OK**:
 

```
java.util.Date loanDate = new java.util.Date();
loan.LoanRequest loan = new loan.LoanRequest(loanDate,48,100000,1.2);
return loan;
```
6. Click **Apply** to save the changes.
7. Click **Run**.

The Console view shows an exception error.

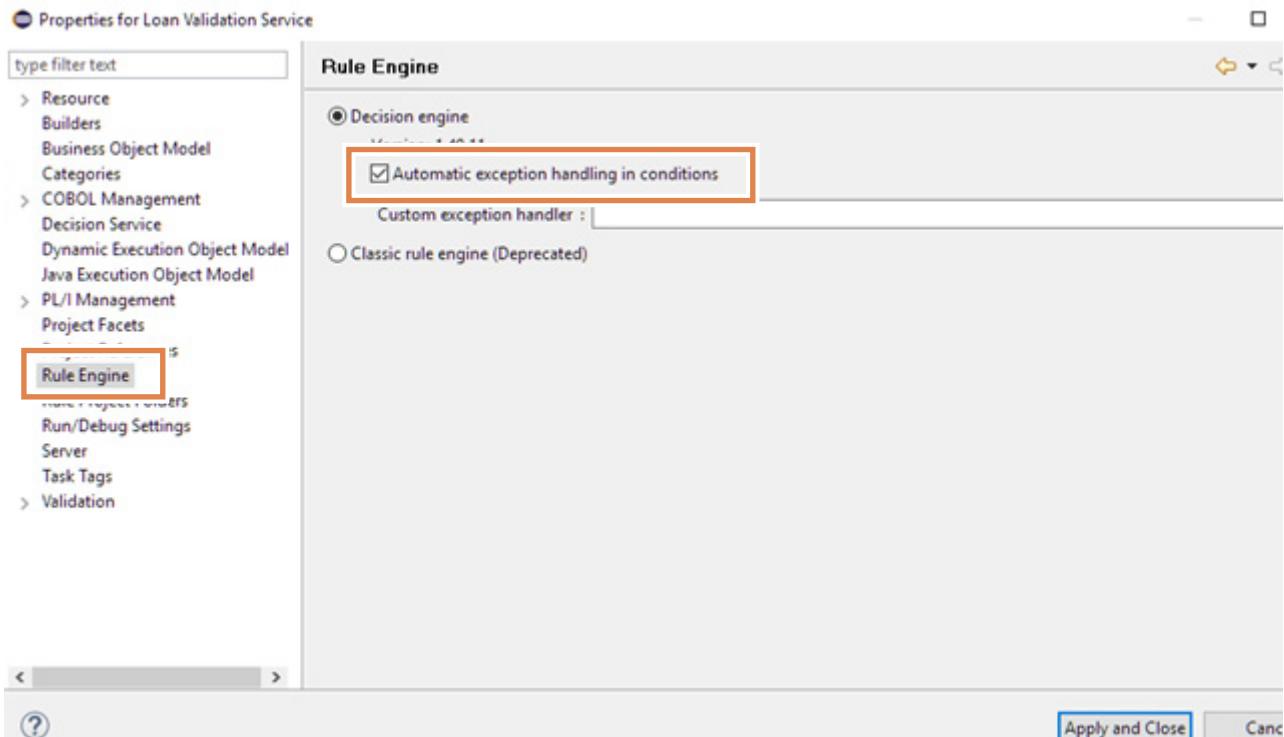


```
<terminated> run loan validation [Decision Operation] C:\IBM\ODM8104\jdk\bin\javaw.exe ([]
Error while executing ruleset: java.lang.NullPointerException
 in task 'loanvalidation>validation'
 in task 'loanvalidation'
```

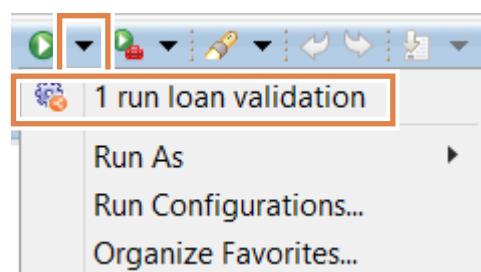
To handle this type of error, you implement automatic exception handling.

### 1.3. Applying automatic exception handling

- \_\_\_ 1. In the Rule Explorer, right-click **Loan Validation Service**, and click **Properties**.
- \_\_\_ 2. In the left column of the Properties dialog, select **Rule Engine**, and select **Automatic exception handling in conditions**.



- \_\_\_ 3. Click **Apply and Close**:
- \_\_\_ 4. Rerun the `run loan validation` launch configuration by clicking the **Run As Run on Server** arrow on the toolbar and click **run loan validation**.



In the Console view, you do not see an exception error so the decision engine automatically handled the exception. You also see that the loan request runs correctly and a decision is returned. However, this is not the correct decision.

## 1.4. Adding exception handing logging

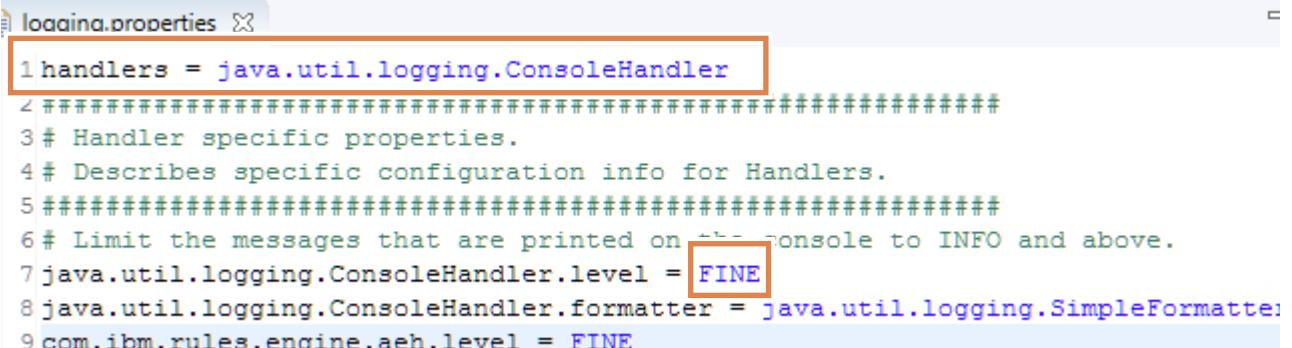
Next, to determine where the exception takes place, you add more logging information to the results.

- 1. In Rule Explorer, expand **Loan Validation Service** and double-click **logging.properties** to open it.

- 2. Notice the line in the file that sets the Console as a logging handler:

```
handlers = java.util.logging.ConsoleHandler
```

- 3. Note also that the log level is set to **FINE**.

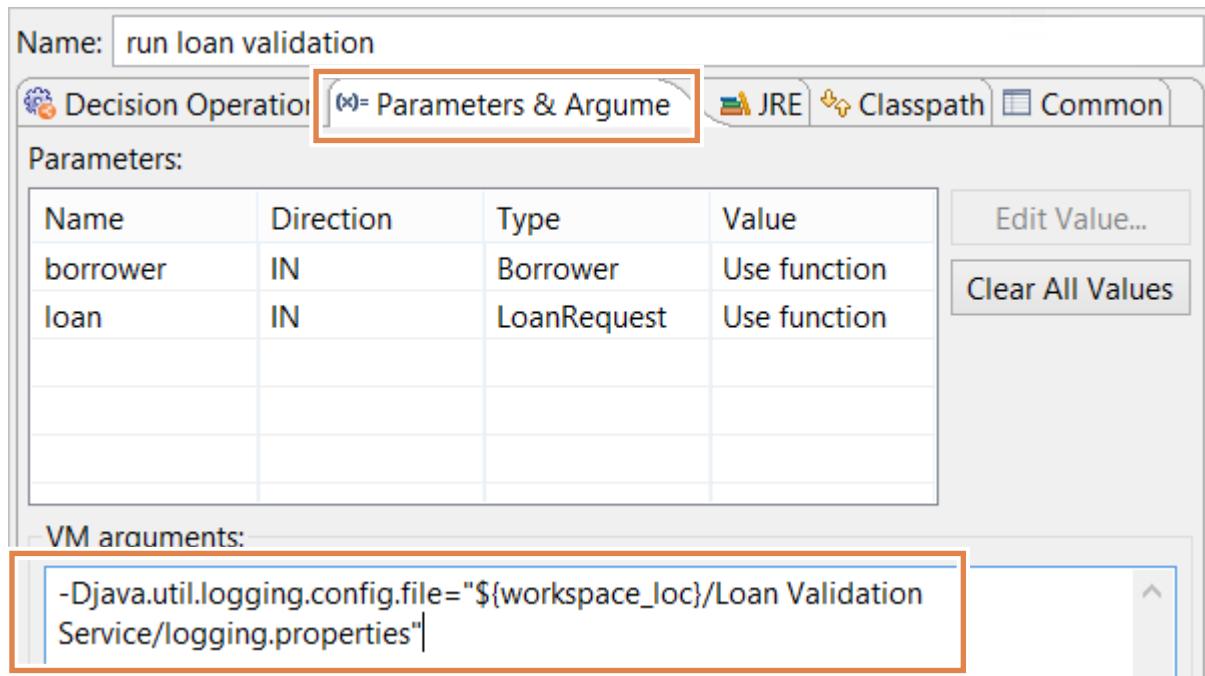


```
loanina.properties ✘
1 handlers = java.util.logging.ConsoleHandler
2 #####
3 # Handler specific properties.
4 # Describes specific configuration info for Handlers.
5 #####
6 # Limit the messages that are printed on the console to INFO and above.
7 java.util.logging.ConsoleHandler.level = FINE
8 java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
9 com.ibm.rules.engine.aeh.level = FINE
```

- 4. Open **Run > Run Configurations**, and select the **run loan validation** configuration.

5. Open the **Parameters and Arguments** tab, and in the **VM arguments** section, add the following argument:

```
-Djava.util.logging.config.file="${workspace_loc}/Loan Validation Service/logging.properties"
```



6. Click **Apply** and click **Run**.

The Console shows the results, which include a log message for the automatic exception handling. The results show that the exception takes place in a condition in the `checkZipcode` rule.

```
<terminated> run loan validation [Decision Operation] C:\IBM\ODM8104\jdk\bin\javaw.exe (Sep 23, 2020, 5:17:10 PM)
Sep 23, 2020 5:17:12 PM com.ibm.rules.generated.ruleflow.loandata$003evali
FINE: AEH_LOG: java.lang.NullPointerException: in the conditions of rule 'checkZ
Sep 23, 2020 5:17:12 PM com.ibm.rules.generated.ruleflow.loandata$003evali
FINE: AEH_LOG: java.lang.NullPointerException: in the conditions of rule 'checkZ
The borrower's age is valid.
The borrower's name is not empty.
```



### Note

The Console shows the `AEH_LOG` message twice. Because the `checkZipcode` rule contains a **then** action and an **else** action, its condition part is evaluated twice, producing two `AEH_LOG` messages.

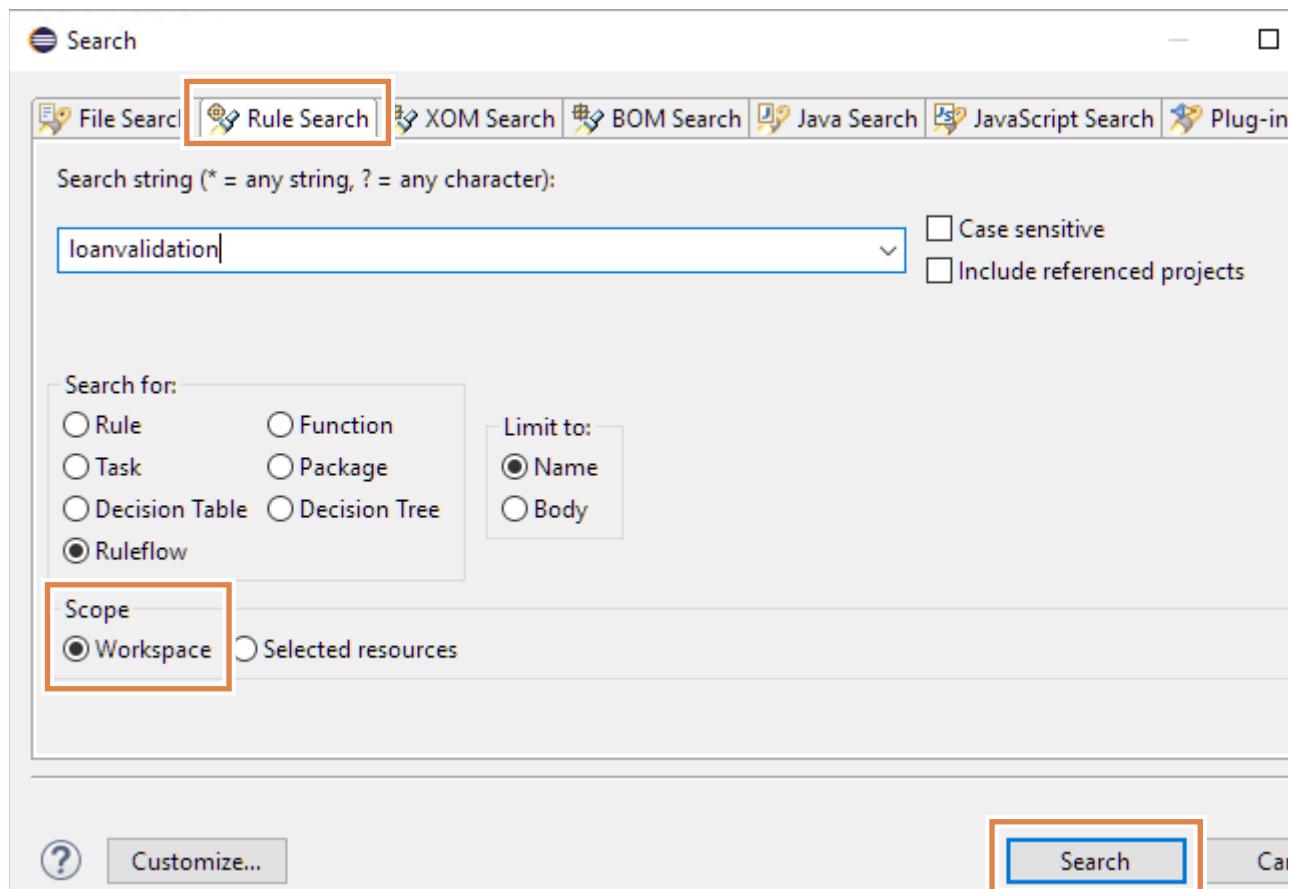
## Section 2. Setting breakpoints

### 2.1. Searching for the source of the problem

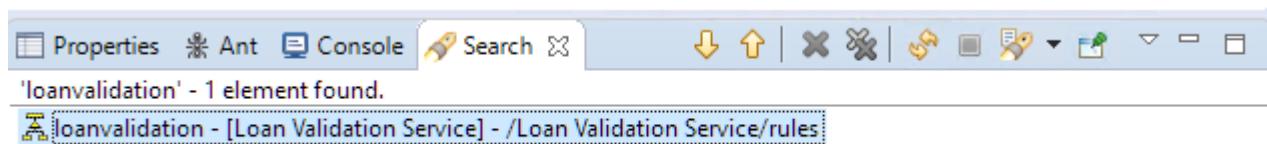
Based on the information in the log, you know that the exception takes place in a condition in the `checkZipcode` rule. The results also point to the validation task in the `loanvalidation` ruleflow:  
`ruleflow.loanvalidation$003validation`

You can set a breakpoint on the ruleflow to search for the source of the error.

- 1. On the **Search** menu, click **Search**.
- 2. On the **Rule Search** tab, in the **Search string** field, enter: `loanvalidation`
- 3. Select **Ruleflow**, and then click **Search**.

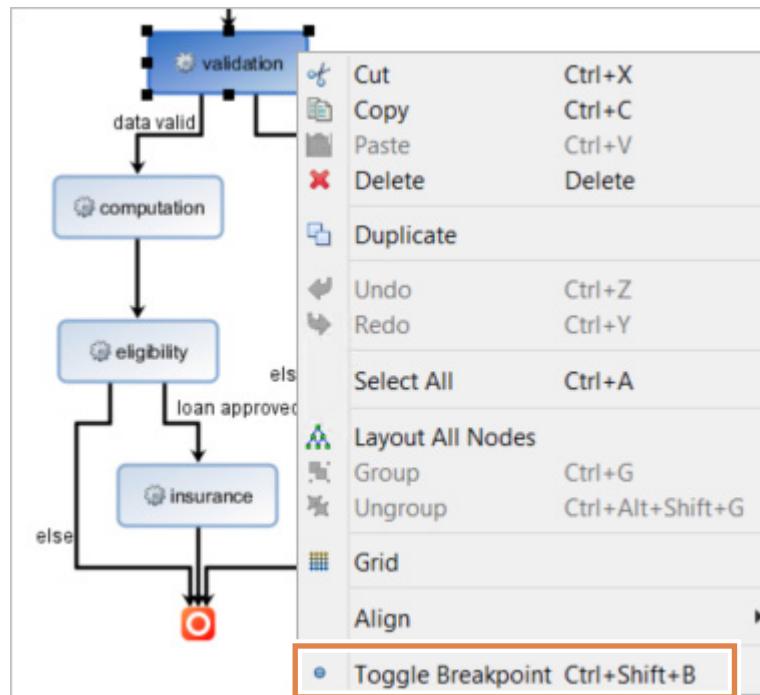


The `loanvalidation` ruleflow is displayed in the Search view.



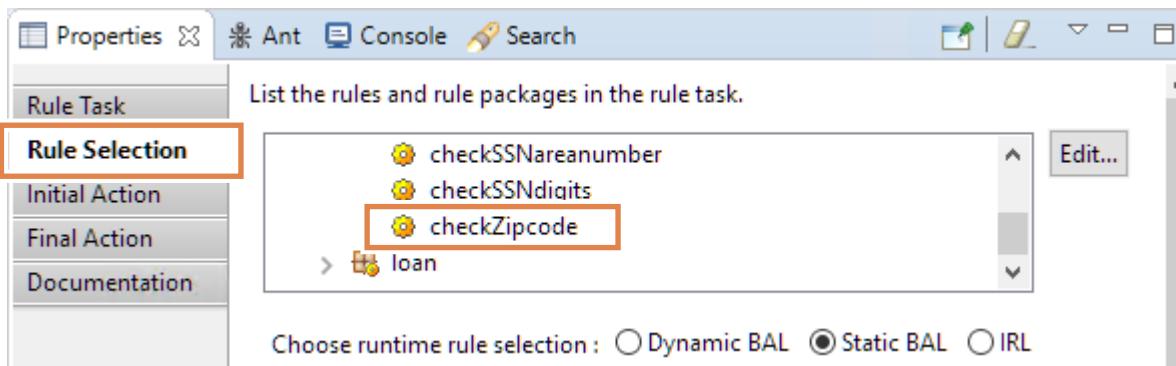
- 4. Double-click the `loanvalidation` ruleflow in the Search view to open it.

- \_\_\_ 5. Click the **validation** task to select it.
- \_\_\_ 6. Right-click the task and click **Toggle Breakpoint**.



## 2.2. Preparing the debug session

- \_\_\_ 1. Double-click the validation task in the loanvalidation ruleflow.
- \_\_\_ 2. In the **Properties** tab, click **Rule Selection** and expand the validation package.
- \_\_\_ 3. Expand borrower, and double-click the checkZipcode action rule to open it in the rule editor.



You see that the condition in the rule uses the length of the zip code. You want to know the value of the zip code, so you add trace information to print the value.

- \_\_\_ 4. Go back to the loanvalidation ruleflow in the Ruleflow editor, and double-click the validation task.
- \_\_\_ 5. In the Properties view, select **Initial Action**.

- \_\_\_ 6. Keep **Use BAL for action** selected and add the following code to display the borrower zip code:
- ```
print "The zip code of the borrower is " + the zip code of the address of  
'the borrower' ;
```
- ___ 7. Save your work.

2.3. Running the debugger on a ruleflow

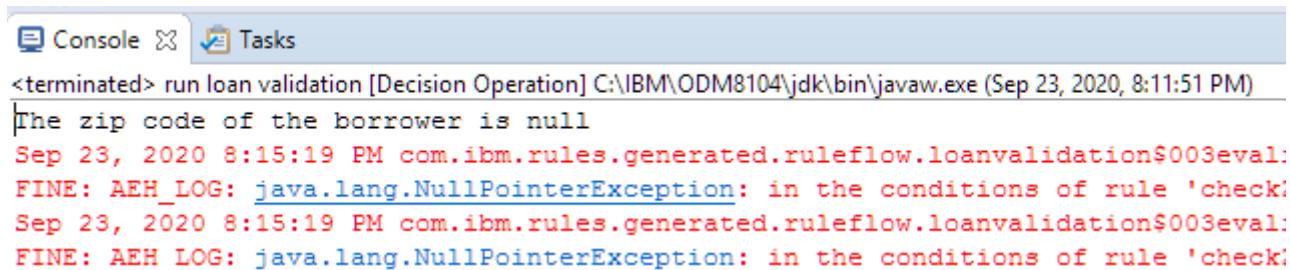
- ___ 1. On the **Run** menu, click **Debug Configurations**.
 - ___ 2. Open the run loan validation configuration and click **Debug**.
 - ___ 3. If the Save and Launch dialog opens, click **OK** to save your changes before running the debugger.
 - ___ 4. When the Confirm Perspective Switch dialog box opens, click **Yes** to open the debug perspective. (Sometimes the dialog box loses focus and is hidden behind the Rule Designer window.)
- The debugger stops at the beginning of the `validation` rule task, at the breakpoint that you set.
- ___ 5. On the **Variables** tab, expand **borrower** and notice the null values for `latestBankruptcy`, `spouse`, and `zipCode`.

Name	Value
↳ this	RuleflowImpl (id=1582)
↳ birthDate	Date (id=1653)
↳ borrower	Borrower (id=1112)
↳ birth	GregorianCalendar (id=1865)
↳ creditScore	200
↳ firstName	\"Smith\" (id=2494)
↳ lastName	\"John\" (id=1332)
↳ latestBankruptcy	null
↳ spouse	null
↳ SSN	Borrower\$SSN (id=1763)
↳ yearlyIncome	20000
↳ zipCode	null

- ___ 6. On the toolbar, click the **Resume** icon.



- ___ 7. In the Console view, scroll to the first lines of the results.



The screenshot shows the Eclipse IDE's Console view. The title bar says "Console" and "Tasks". The content area displays the following text:

```
<terminated> run loan validation [Decision Operation] C:\IBM\ODM8104\jdk\bin\javaw.exe (Sep 23, 2020, 8:11:51 PM)
The zip code of the borrower is null
Sep 23, 2020 8:15:19 PM com.ibm.rules.generated.ruleflow.loanvalidation$003eval:
FINE: AEH_LOG: java.lang.NullPointerException: in the conditions of rule 'check'
Sep 23, 2020 8:15:19 PM com.ibm.rules.generated.ruleflow.loanvalidation$003eval:
FINE: AEH_LOG: java.lang.NullPointerException: in the conditions of rule 'check'
```

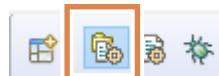
The first line is the trace that you added to the initial action of the validation task:

The zip code of the borrower is null

You see that the automatic exception handling catches the null pointer exception because of the null zip code in the `checkZipcode` rule. The condition part has an unknown status, so the rule is ignored. You confirm with the rule authors about the null zip code for the borrower in the configuration. The rule authors confirm that the zip code rule can be ignored.

Now that you confirmed that the automatic exception handling correctly addresses the problem, you can remove the trace for the logging information that comes from the automatic exception handling.

- ___ 8. Switch back to the Rule perspective.



- ___ 9. On the **Run** menu, click **Run Configurations**, and open the `run loan validation` configuration.
 ___ 10. On the **Parameters and Arguments** tab, delete the VM arguments.
 ___ 11. Click **Apply**, and then **Run**.

The automatic exception handling trace information is no longer shown in the Console view. However, the loan is still not approved, so more errors need to be fixed.

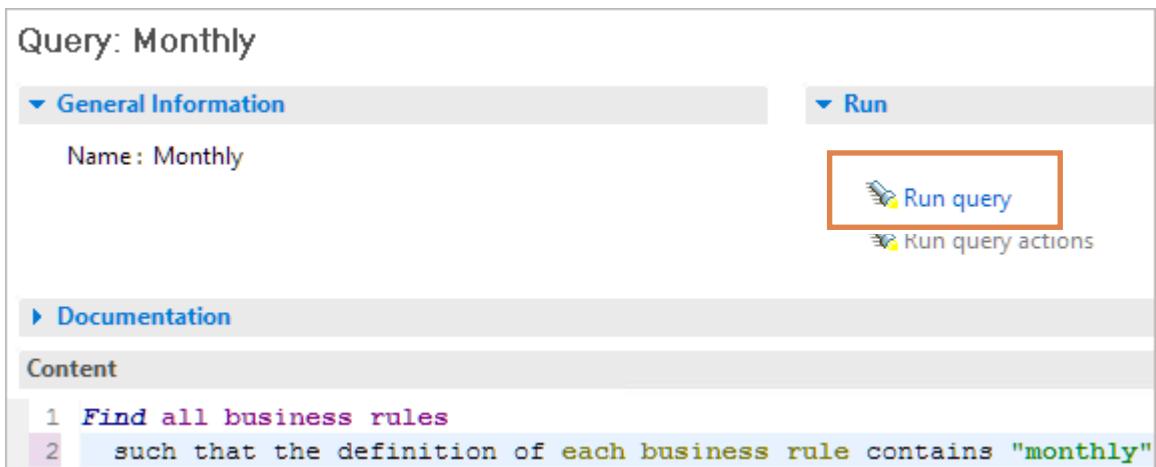
Section 3. Debugging a rule

In this section, you set breakpoints in a rule and run a debug session. When you find an error, you make changes to correct it.

When you run the decision service, it does not generate the correct monthly repayment rate. You want to find the source of this value to determine how to fix it. First, you create a query to find the rule that uses the variable `monthly repayment`. Then, you add a breakpoint to track the rule.

3.1. Setting a breakpoint in a rule

- ___ 1. In the Rule Explorer, expand **Loan Validation Service**.
- ___ 2. Right-click the **queries** folder, and click **New > Query**.
- ___ 3. Set the **Name** field of the new query to `Monthly` and click **Finish**.
- ___ 4. Set the query to state:
 Find all business rules
 such that the definition of each business rule contains "monthly"
- ___ 5. Save your work (Ctrl+S).
- ___ 6. Click **Run query**.



Query: Monthly

General Information

Name: Monthly

Run

Run query (highlighted)

Run query actions

Documentation

Content

```

1 Find all business rules
2 such that the definition of each business rule contains "monthly"
  
```

The query returns the `repayment` rule in the Search view.

- ___ 7. Double-click the `repayment` rule in the Search view to open it in the rule editor.
 You see that the first action, line 7, sets the value of the `monthly repayment` variable.
 set the monthly repayment of 'the loan report' to

8. Click the **ARL** tab to see how `monthlyRepayment` is calculated in the rule.

The screenshot shows the ARL (Advanced Rule Language) tab selected in the Intellirule window. The code in the editor is:

```

}
then
    $EngineData.this.report.monthlyRepayment = loan.LoanUtil.getMonthlyRepayment((double)amount,
}

```

The line `$EngineData.this.report.monthlyRepayment = loan.LoanUtil.getMonthlyRepayment((double)amount,` is highlighted with a red box.

Note that in the **then** statement, the rule calls the `loan.LoanUtil.getMonthlyRepayment` method to calculate `monthlyRepayment`.

```
$EngineData.this.report.monthlyRepayment =
```

```
loan.LoanUtil.getMonthlyRepayment((double)amount, duration, rate);
```

You must step into this method.

9. Reopen the **Intellirule** tab, right-click line 7, and click **Toggle Breakpoint**.

The screenshot shows the Intellirule tab selected. A context menu is open over line 7 of the code. The menu items are:

- Add Bookmark...
- Add Task...
- Show Quick Diff Ctrl+Shift+Q
- Show Revision Information
- Show Line Numbers
- Preferences...

The "Toggle Breakpoint" option is highlighted with a red box.

A breakpoint is displayed next to the line.

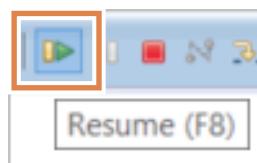
The screenshot shows the Intellirule tab selected. Line 7 of the code has a blue circular breakpoint marker next to it. The code on line 7 is:

```
set the monthly repayment of 'the loan report' to
```

3.2. Debugging the action rule

Next, you run a debugging session. When it stops at breakpoints, you look for reasons for the monthly repayment error.

- 1. From the **Run > Debug Configurations** menu open the `run loan validation` configuration and click **Debug**.
 - 2. Wait for the debug to complete. It takes several moments.
 - 3. When you are prompted to switch to the Debug perspective, click **Yes**. Otherwise, you can switch manually.
- The debugger stops at the validation task in the `loanvalidation` ruleflow.
- 4. On the toolbar, click **Resume**.



The debugger stops at the “set the monthly repayment line” of the `repayment` rule.

- 5. On the **Variables** tab, expand **report**.

The `monthlyRepayment` variable shows `0.0`.

Name	Value
numberOfMonthlyPayments	48
startDate	Date (id=83)
report	Report (id=87)
approved	false
borrower	Borrower (id=81)
insuranceRate	0.0
insuranceRequired	false
loan	LoanRequest (id=84)
messages	ArrayList<E> (id=100)
monthlyRepayment	0.0
validData	true
yearlyInterestRate	0.0

- 6. Click the **Step Into** icon.



The debugger stops at `LoanUtil.java`. The Java code shows the following method at the breakpoint:

```
double i = yearlyRate / MonthsInYear;
```

The function computes a monthly repayment rate by dividing `yearlyRate` by `MonthsInYear`. In the **Variables** view, you see that the value of `MonthsInYear` is 0.

The screenshot shows the IDE interface with the following components:

- Debug View:** Shows a stack trace for a "run loan validation [Decision Operation]" session. The current thread is suspended at line 46 of `LoanUtil.getMonthlyRepayment`.
- Variables View:** A table showing variable values. The row for `MonthsInYear` is highlighted with an orange border, showing its value is 0.
- Code Editor:** An open file named `LoanUtil.java` containing Java code. Line 46 is highlighted:

```
46     double i = yearlyRate / MonthsInYear;
```
- Outline View:** Shows the class structure with methods like `getMonthlyRepayment`, `getMon`, and `formatt`.



Troubleshooting

If the **Variables** view shows an error message instead of the variable, click the **Breakpoints** tab, then click the **Variables** tab again.

The error is that the function is dividing by 0, when it should divide by 12.

- ___ 7. Stop the debugging session by clicking the **Terminate** icon.



3.3. Fixing the error in the rule

You change the value of the `MonthsInYear` variable to 12, and run the decision service to check the results.

- ___ 1. Switch to the Rule perspective.
- ___ 2. In the editing window for `LoanUtil.java`, change the value of the `MonthsInYear` variable to 12.

```
32 public class LoanUtil implements Serializable {
33     private static final long serialVersionUID = 7764736178720624835L;
34     private static final short MonthsInYear= 12;
```

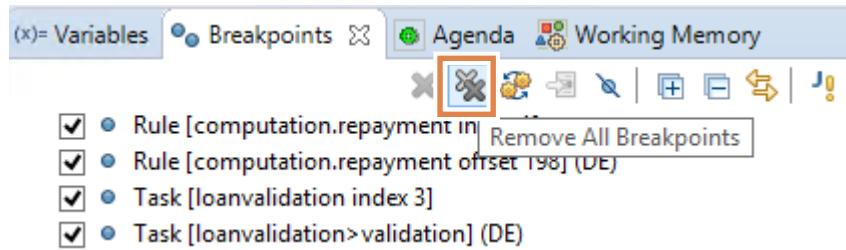
- ___ 3. Save your changes.
- ___ 4. Run the decision service by clicking the **Run** icon on the toolbar.



The Console now shows the correct monthly repayment rate of 570.83.

However, the decision service still does not approve the loan, so you have more work to do.

- 5. Remove all the breakpoints.
 - a. On the **Window** menu, click **Show view > Other**.
 - b. Expand **Debug** and double-click **Breakpoints**.
 - c. Click the **Remove All Breakpoints** icon on the toolbar of the Breakpoints view, and click **Yes** to confirm removal.



Section 4. Debugging a decision table

4.1. Finding the source of the rejection message

- ___ 1. Make sure that you are in the Rule perspective.
- ___ 2. Open the **Run > Run Configurations** menu and click the `run loan validation` configuration to open it.
- ___ 3. Click **Run**.

The results in the Console view show that the loan is low risk, but the decision service still does not approve the loan.

You decide to search for the rule that makes the rejection message by running a query for the message.

- ___ 4. In the Rule Explorer, expand **Loan Validation Service**, right-click the **queries** folder, and click **New > Query**.
- ___ 5. Create a query to find the rule that uses the following rejection message:

Your loan has not been approved

- ___ a. Set the query name to `Approval` and click **Finish**.
- ___ b. Set the query to:

Find all business rules

such that the definition of each business rule contains "Your loan has not been approved"

- ___ c. Save your work.
- ___ d. Click **Run query**.

The query returns the `approval` action rule.

- ___ 6. Double-click the `approval` rule in the Search view to open it in the rule editor.

The action depends on the grade that is given to the loan. If the grade does not equal A, B or C, the loan is rejected:

```

if
  'the grade' is one of { "A" , "B" , "C" }
then
  in 'the loan report', accept the loan with the message "Congratulations!
  Your loan has been approved" ;
else
  in 'the loan report', refuse the loan with the message "We are sorry.
  Your loan has not been approved" ;

```

- ___ 7. Place breakpoints on both actions (lines 4 and 6) in the approval rule.

```

1 if
2   'the grade' is one of { "A" , "B" , "C" }
3 then
4   in 'the loan report', accept the loan with the message "Congratulations"
5   -else
6   in 'the loan report', refuse the loan with the message "We are sorry. Y

```

Now you search for the rule that computes the grade condition for the approval rule.

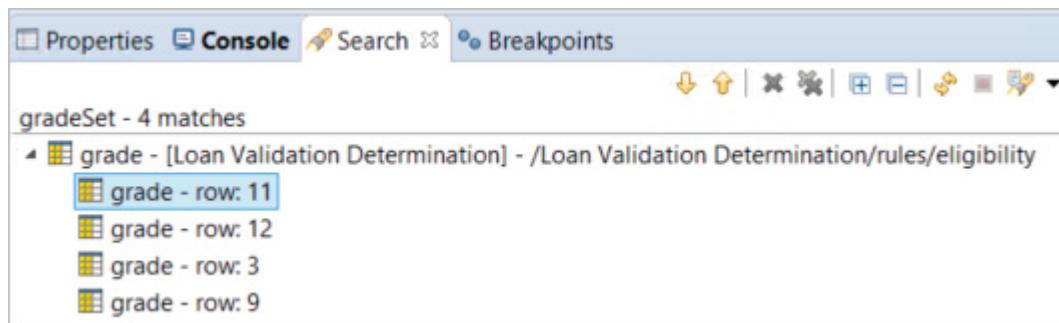
4.2. Setting breakpoints in a decision table

The approval action rule applies a decision that is based on the grade that is given to the loan. You decide to locate the rule that assigns grades to the loans. You create a query that finds a decision table, and then you add breakpoints to the decision table to observe it in a debugging session.

- ___ 1. Create another query on the **Loan Validation Service** project.
- ___ a. Right-click the **Loan Validation Service > queries** folder, and click **New > Query**.
 - ___ b. Name the query `gradeSet` and click **Finish**.
 - ___ c. Set the query to:
- Find all business rules
such that each business rule may lead to a state where ['the grade' is not one of { "A", "B", "C" }]
- ___ d. Save your work.
 - ___ e. Click **Run query**.

The query looks for a rule that can assign a grade that is not A, B or C, and therefore, causes the decision service to reject the loan.

The Search view shows the grade decision table, and lists the rows that do not contain grade A, B, or C.



- ___ 2. Double-click one of the `grade - row` decision table results in the **Search** view to open it in the decision table editor.
- ___ 3. Select the **Grade** column by clicking the column header cell, right-click the A cell in row 1, and click **Toggle Breakpoint**.

Every Grade cell now has a breakpoint.

Grade
A
A
D
A
B
C
B
C
D
C
D
E

4.3. Debugging the decision table

- 1. Click the **Run > Debug Configurations** menu item, open the `run loan validation` configuration, and click **Debug**.
- 2. When you are prompted to switch to the Debug perspective, click **Yes**.
The debugger stops at row 3, which shows grade D and the message `Low risk loan`.
- 3. Click the **Resume** icon.
The debugger stops at the second breakpoint in the `approval` action rule. The loan is rejected because the grade is D. The error is coming from row 3 of the `grade` decision table.
- 4. Click the **Terminate** icon to stop the debugging session.



Information

When you discover this type of error in the business rule, you need to contact the business user to confirm how to proceed. In this case, the grade in row 3 should be changed to B.

- 5. Return to the Rule perspective, and in the `grade` decision table, change the grade in row 3 to B.
- 6. Save your changes.
- 7. Click the **Run > Run Configurations** menu item, open the `run loan validation` configuration, and click **Run**.

The results show that the loan is low risk, but the decision service still rejects the loan.

Section 5. Debugging the ruleflow

The decision service still rejects the loan. The rules run correctly now, so you decide to test the flow of decisions among the rules.

5.1. Debugging the ruleflow

Run a debugging session to determine the source of the error.

- 1. Click the **Run > Debug Configurations** menu item, open the `run loan validation` configuration, and click **Debug**.

The debugger stops at row 3 of the grade decision table.

- 2. In the Debug perspective, open the **Agenda** tab.

The tab shows two rules that are in the same `eligibility` rule task. The first entry points to the rule that is formed by row 3 in the `grade` decision table, and the second entry points to the `approval` action rule.

In the `grade` decision table, you see that an arrow points to the `grade` cell in row 3. The grade is `B`, and the message for the row is `Low risk loan`.

The screenshot shows the IBM Rational Rules Studio interface. At the top, the 'Debug' perspective is selected. In the center, the 'Agenda' tab is open, displaying two entries: 'eligibility.grade_3' and 'eligibility.approval'. Both entries have a green arrow icon next to them. Below the agenda, the 'grade' decision table is visible. Row 3 of the table is highlighted with a blue border. An arrow points from the 'Grade' column header to the 'B' value in the 'Grade' cell of row 3. The table also includes columns for 'Yearly repayment', 'Corporate score', and 'Message'.

	Yearly repayment		Corporate score		Grade	Message
	Min	Max	Min	Max		
1			≥ 900		A	Very low risk
2	0	10,000	600	900	<input checked="" type="radio"/>	Very low risk
3			300	600	<input checked="" type="radio"/>	Low risk loan

- 3. Click the **Resume** icon.

The debugger stops on the **else** line in the `approval` action rule, which rejects the loan.

The screenshot shows the IBM Rules Studio interface. In the top left, there's a tree view of the ruleflow: "run loan validation [Decision Operation]" -> "com.ibm.rules.studio.rve.launch.main.RveRulesetMain at localhost:56428" -> "Thread [main] (Suspended (Rule breakpoint on eligibility.approval at offset 175))". The "Variables" view (top right) shows a table with columns "Name" and "Value". A row for "grade" is selected and highlighted with an orange border, showing its value as "B" (id=1185). Below the variables view is the "Action Rule: approval" section. It has tabs for "General Information" (selected) and "Category Filter". Under "General Information", it says "Name: approval" and "Categories: Any." (with an edit link). The "Content" tab shows the rule code:

```

1 if
2   'the grade' is one of { "A" , "B" , "C" }
3 then
4   in 'the loan report', accept the loan with the message "Congratulations! Your loan is approved"
5 else
6   in 'the loan report', refuse the loan with the message "We are sorry. Your loan has been rejected"

```

The line "6 in 'the loan report', refuse the loan with the message" is highlighted with a green background and an orange border.

Although the `grade` decision table set the `grade` value to B, the action rule still goes to the **else** condition. The approval action rule does not see the `grade` value set by the decision table. This disconnect between the rules indicates a ruleflow problem.

- 4. Click the **Terminate** icon to stop the debugging session.

5.2. Modifying the ruleflow

- 1. Switch to the Rule perspective and open the `loanvalidation` ruleflow.
- 2. Select the `eligibility` rule task.
- 3. In the **Properties** view, open the **Rule Selection** tab, and expand **eligibility**.

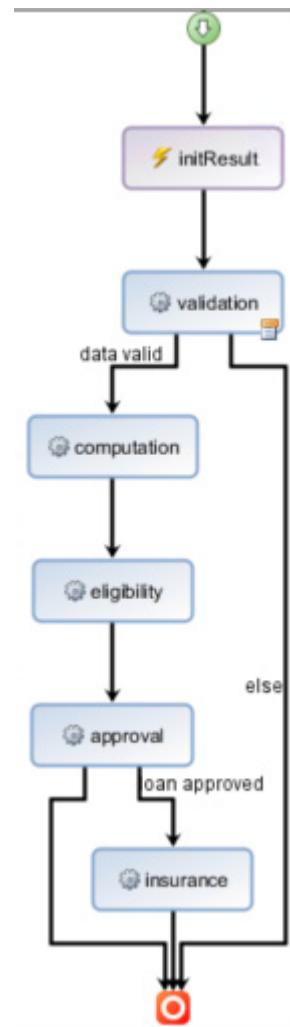
You see that the task contains both the `approval` action rule and the `grade` decision table.

- 4. Open the **Rule Task** tab and note that the Fastpath algorithm is selected.

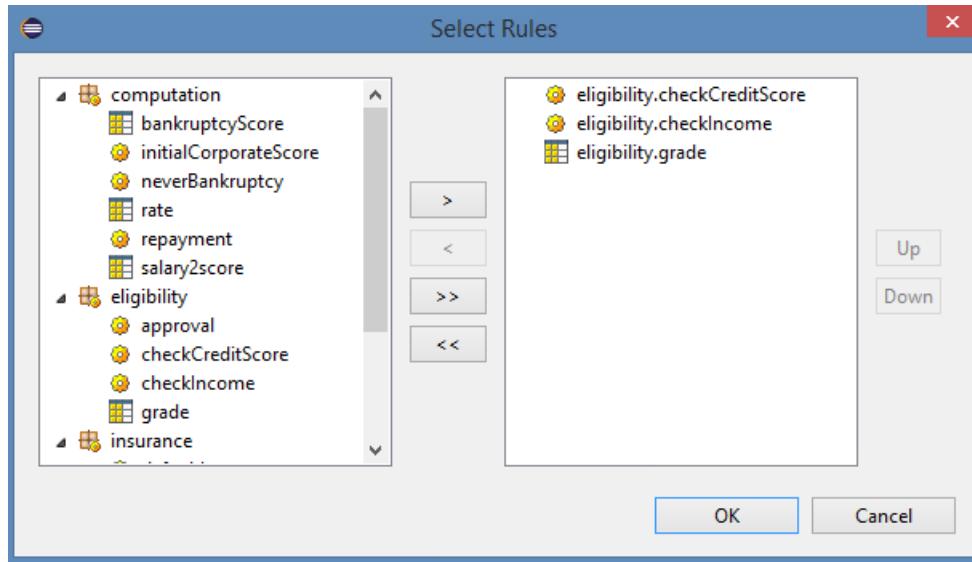
The Fastpath algorithm does not support inference. The agenda is computed once in the rule task, but it is not changed if variables are modified by other rules in the task. This means that the `approval` rule runs with an unset `grade` value, and the value is not changed by the `grade` decision table.

- ___ 5. Create a separate rule task for the `approval` action rule to resolve the error.
 - ___ a. Click the **Create a rule task** icon and then click the ruleflow diagram next to the `insurance` task node to add the task node.
 - ___ b. On the **Rule Task** tab in the Properties view, set the ID to: `approval`
 - ___ c. On the **Rule Selection** tab, click **Edit**, and in the Select Rules window, double-click the `eligibility.approval` rule, and click **OK**.

- ___ 6. Change the transitions to flow between these nodes:
 - `eligibility` to `approval`
 - `approval` to `insurance`
 - `approval` to end node
 - ___ a. Click the transition line from the `eligibility` task to the end node, and reset it to run from the `approval` task to the end node.
 - ___ b. Click the transition line from the `eligibility` task to the `insurance` task, and reset it to run from the `approval` task to the `insurance` task.
 - ___ c. Create a transition line between the `eligibility` task and the `approval` task.



- ___ 7. Edit the `eligibility` task to remove the `approval` action rule from the **Rule Selection** tab.
 - ___ a. Click the `eligibility` task on the ruleflow diagram, and in the Properties view, click the **Rule Selection** tab, and click **Edit**.
 - ___ b. Double-click the **eligibility** package to remove it from the list of selected rules (right side of the Select Rules dialog).
 - ___ c. In the left side of the Select Rules dialog, expand **eligibility**, and double-click each of the `eligibility` rules except the `approval` rule.



- ___ d. Click **OK**.
- ___ 8. Save your work.
- ___ 9. Run the decision service by using the `run loan validation` configuration in the Run Configurations.
Now, the grade value set in the `grade` decision table is set before the `approval` rule is evaluated in the `approval` task, and the loan is approved.
- ___ 10. Remove all the breakpoints.
 - ___ a. Open the Breakpoints view.
 - ___ b. Click the **Remove All Breakpoints** icon on the toolbar of the **Breakpoints** view, and click **Yes** to confirm removal.



- ___ 11. Run the decision service by using the `run loan validation` configuration in the Run Configurations.
The decision service runs correctly, and produces the expected report of an approved loan.

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, a decision table, and in a ruleflow, and you saw how to view variable values 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**.

Exercise 14. Enabling rule validation

Estimated time

01:15

Overview

This exercise teaches you how to set up testing and simulation functionality for business users.

Objectives

After completing this exercise, you should be able to:

- Validate the BOM and generate scenario file templates
- Customize scenario file templates
- 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"](#)
- [Section 2, "Removing columns from the scenario file template"](#)
- [Section 3, "Testing the rules in Rule Designer"](#)
- [Section 4, "Enabling testing from Business console"](#)
- [Section 5, "Testing in Business console"](#)

Requirements

This exercise uses files that are stored in the `C:\labfiles\code` directory. It also uses the following files, which are installed in the `<InstallDir>\studio\training` directory:

- Start project: Dev 14 – Validate\start



Note

For IBM ODM on Cloud users

This exercise uses some features that are not supported in IBM ODM on Cloud.

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 Testing and Simulation BOM 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.



Information

In this exercise, you must write some code. You can find the code snippets to create in the `C:\labfiles\code\enable_testing.txt` file, and you can copy and paste them in Rule Designer.

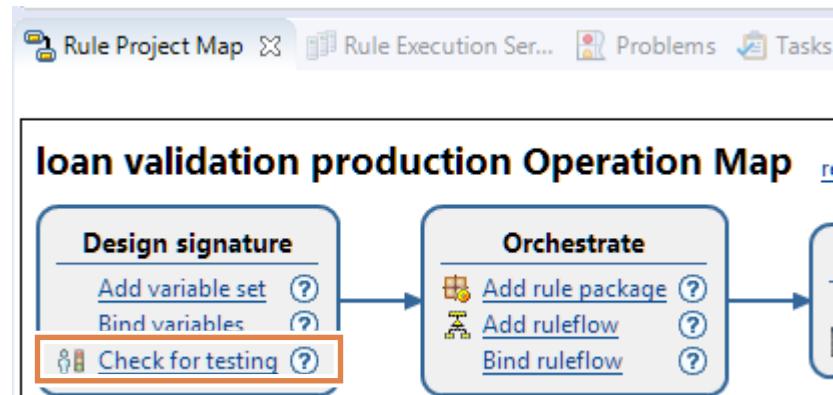
1.1. Import your workspace in Rule Designer

- ___ 1. In Rule Designer, switch to a new workspace and name it: `testing`
 - ___ a. From the **File** menu, click **Switch Workspace > Other**.
 - ___ b. In the Workspace Launcher window, enter the path:
`C:\labfiles\workspaces\testing`
 - ___ c. Click **Launch**.
- ___ 2. Close the Welcome view.
- ___ 3. Use the Samples Console to import the exercise start project.
 - ___ a. Click the **Open Perspective** icon.
 - ___ b. In the Open Perspective window, click **Samples Console** and click **Open**.
 - ___ c. In the **Rule Designer** section, expand **Training > Ex 14: Enabling rule validation > start**, and click **Import projects**.
- ___ 4. Close the Help view.

1.2. Validating the rule project

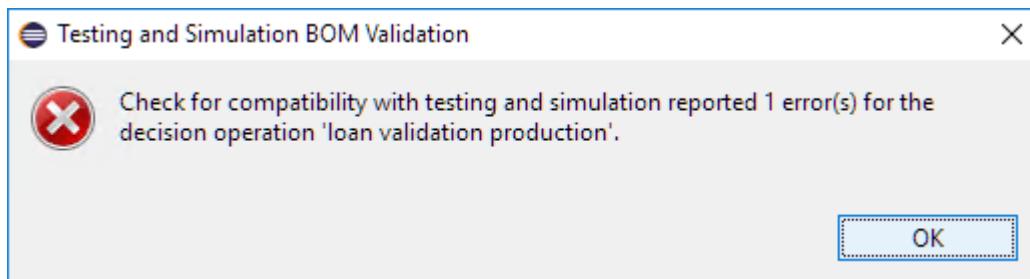
- ___ 1. In Rule Explorer, make sure that the `Loan Service` project is selected and open the Rule Project Map view.
- ___ 2. On the Decision Service Map, in the **Define decision operation** part of the map, click **Go to operation map**, select `loan validation production.dop`, and click **OK**.

3. In the **Design signature** part of the map, click **Check for testing**.



4. When prompted by the Decision Operation Selection window, make sure that only **loan validation production** is selected (clear **eligibility**), and click **Finish**.

The Testing and Simulation BOM Validation window opens and reports a compliance error.



5. Click **OK** to close the Testing and Simulation BOM Validation window.
 6. In the Testing and Simulation BOM Validation view, select the error for `loan.Borrower` to open and read the Solution Description.

ID	BOM Element
GBRTV0003E	loan.Borrower

Solution Description:

input parameter borrower of type loan.Borrower: cannot find a testing and simulation constructor for the business object model class loan.Borrower. If there are no constructors, create one. If there are several constructors in the business object model class, you must specify which one to use by selecting the "Testing and simulation constructor" option in the business object model editor.

An error is raised because no class constructor was defined or specified as the Decision Validation Services constructor.

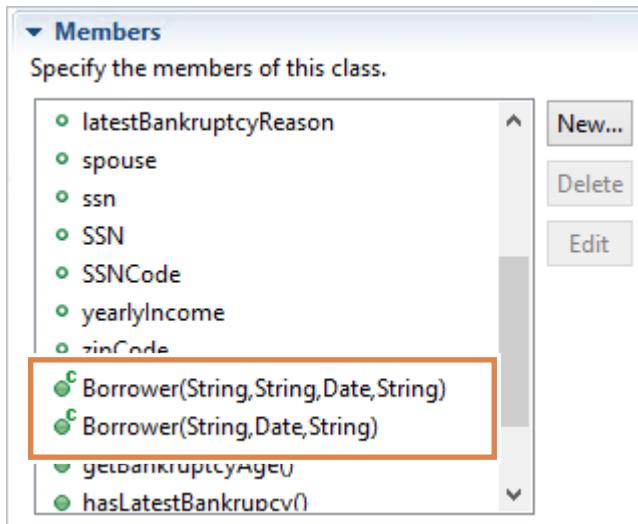
1.3. Choosing a constructor

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.

To choose a constructor:

- 1. In Rule Explorer, expand **Loan BOM > bom > model > 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 validation.

- 2. In the **Members** section of the Borrower class page, double-click the first constructor with four arguments, **Borrower(String, String, Date, String, int)**.
- 3. On the Member page, select the **Testing and simulation constructor** check box.

The screenshot shows the 'General Information' tab for the Borrower member. The 'Name' field contains 'Borrower'. The 'Type' field is empty. The 'Class' field contains 'loan.Borrower'. The 'Deprecated' checkbox is unchecked. The 'Testing and simulation constructor' checkbox is checked and highlighted with a red box.

- 4. Save your work.
- 5. Check the project again for compliance.
 - a. Right-click **Loan Service** and click **Testing and simulation > Check Project**.
 - b. Make sure that only the **loan validation production** decision operation is selected and click **Finish**.

Another compliance error is reported.

- ___ c. Click **OK** to close the error message window.

In the Testing and Simulation Project Validation view, you see a list of errors.

- ___ 6. Click one of the errors to see the description.

ID	BOM Element
GBRTV0021E	loan.Borrower.Borrower(java.lang.String,java.lang.String,java.util.Date,java.lang.String)

Solution Description:

constructor loan.Borrower.Borrower(java.lang.String,java.lang.String,java.util.Date,java.lang.String): argument 'arg1' of
correspond to any attribute in the loan.Borrower class or its superclasses. Rename the arguments by reusing the name
attributes or create virtual attributes that correspond to the arguments in the business object model.

You 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)`

- ___ 7. Double-click the first error message to open the Borrower page in the BOM editor.

Next, you edit the constructor argument names.

Name	Type	Domain
arg1	java.lang.String	
arg2	java.lang.String	
arg3	java.util.Date	
arg4	java.lang.String	

Arguments
Edit the arguments of this member.

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

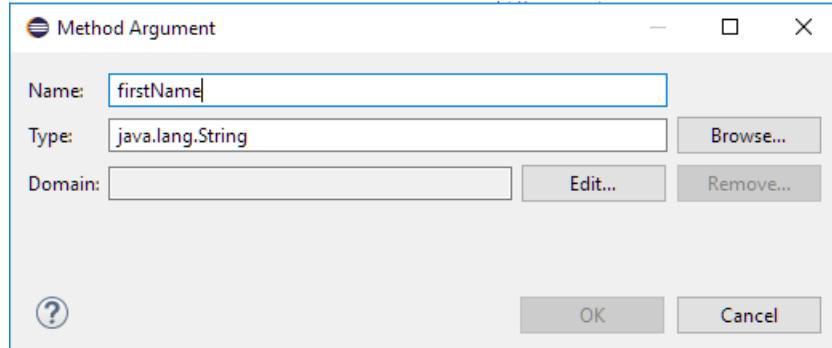
1.4. Editing column headings for the scenario file template

The constructor argument names become the column headings in the scenario file template. You must edit the argument names to match the corresponding BOM member names. For example, if `arg1` sets the value of the `firstName` attribute, then rename `arg1` as: `firstName`

To edit the arguments for the 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**.



Important

The name must match the corresponding attribute name in the `Borrower` class. These names are case-sensitive.

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 values:

- `lastName`
- `birthDate`
- `SSNCode`

▼ Arguments		
Edit the arguments of this member.		
Name	Type	Domain
<code>firstName</code>	<code>java.lang.String</code>	
<code>lastName</code>	<code>java.lang.String</code>	
<code>birthDate</code>	<code>java.util.Date</code>	
<code>SSNCode</code>	<code>java.lang.String</code>	

< >

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

- __ 4. Save your work.
- __ 5. Check the project compliance again.
- __ a. Right-click `Loan Service` and click **Testing and Simulation Services > Check Project**.
- __ b. Make sure that only the **loan validation production** decision operation is selected and click **Finish**.
- You see a message that the project is now compliant.
- __ c. Click **OK** to close the Testing and Simulation BOM Validation window.

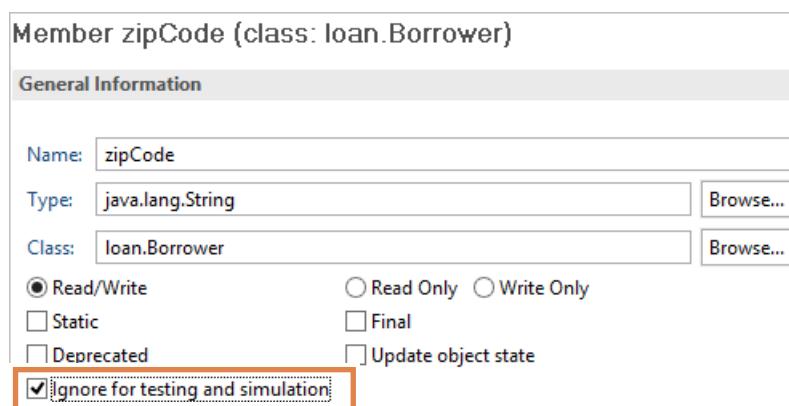
Section 2. Removing columns from the scenario file template

Because of the relationship between the BOM and the scenario file, customizing the scenario file template requires modifying the BOM.

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.

To remove a column:

- 1. Make sure that you are still in the BOM Editor for the Borrower class. (In Rule Explorer, expand **Loan BOM > bom > model > loan**, and double-click **Borrower**.)
 - 2. In the **Members** section of the Class page, double-click `zipCode` to open it on the Member page.
- On the Member page for the `zipCode` attribute, you can see that **Ignore for testing and simulation** is not selected, meaning that this attribute is included in the scenario file template.
- 3. Select **Ignore for testing and simulation**.



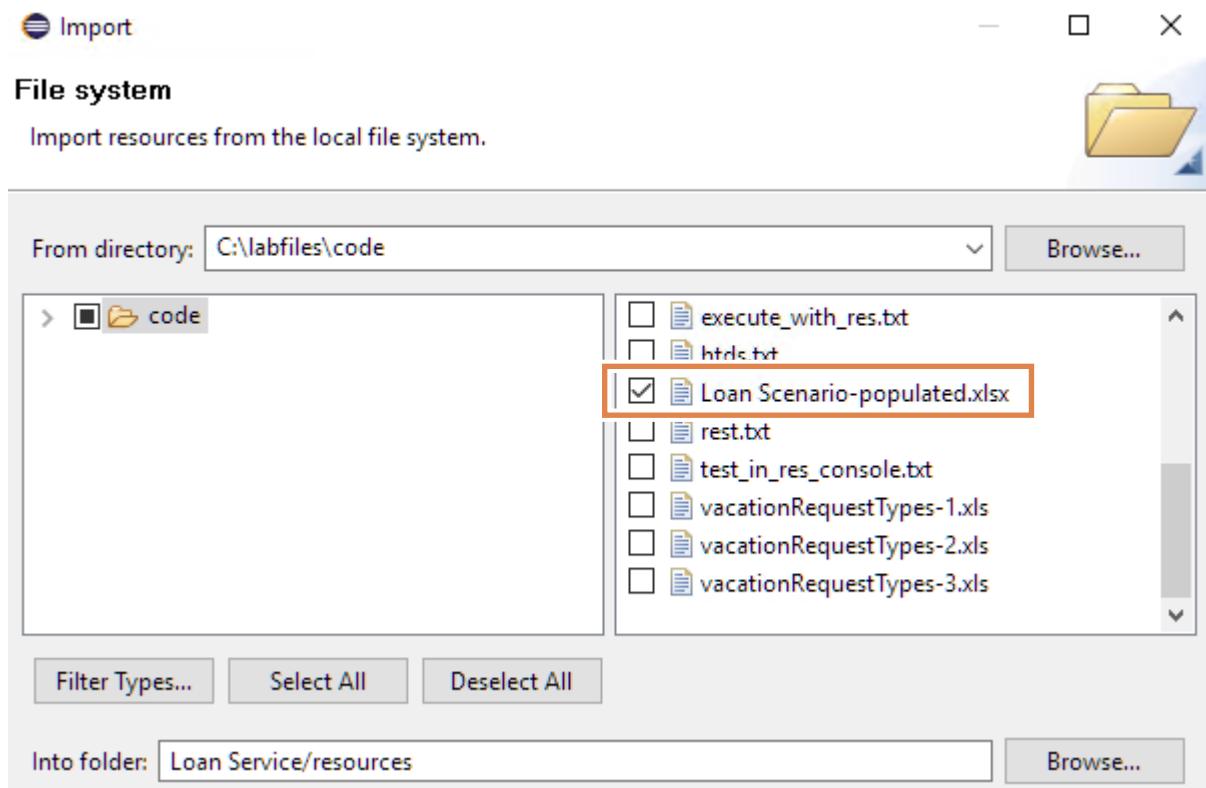
- 4. Save your work.

No input columns will be created for this attribute on 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.

Section 3. Testing the rules in Rule Designer

3.1. Import the scenario file

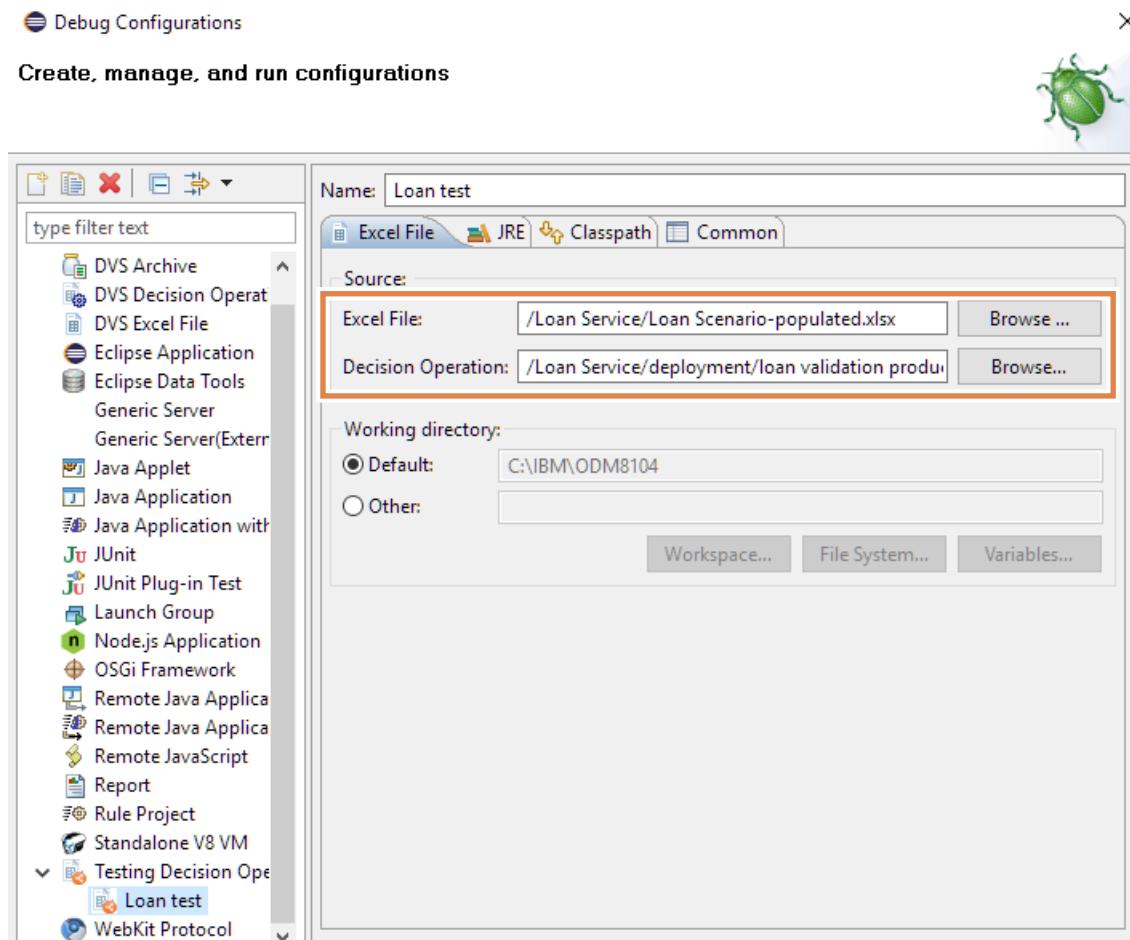
- 1. In Rule Explorer, in the **Loan Service** project, right-click the **resources** folder and click **Import**.
- 2. In the Import wizard, select **General > File System** and click **Next**.
- 3. In the **From directory** field, click **Browse** and select the **C:\labfiles\code** folder.
- 4. In the list of files, select **Loan Scenario-populated.xlsx** and click **Finish**.



3.2. Run the test

- 1. On the **Run** menu, select **Debug configurations**.
- 2. In the left pane, double-click **Testing Decision Operation**.
- 3. Define the testing configuration to use a prepopulated scenario file.
 - a. Set the name of the configuration to: **Loan Test**
 - b. In the **Excel File** field, click **Browse**, expand **Loan Service**, select **Loan Scenario-populated.xlsx** and click **OK**.

- ___ c. In the **Decision Operation** field, click **Browse**, select **loan validation production**, and click **OK**.



- ___ 4. Click **Debug** and wait for the test to complete.

The test results in the Console view show that one test fails, which is expected.

```
Nov 12, 2020 8:13:08 PM com.ibm.rules.res.logging.internal.RESLogger log
INFO: /cdira1605229933789_51463bb0_9ac7_4d5a_af57_fd860799bbfc/1.0/ RuleApp succ
*****
Scenario Number 0 : 'Scenario 1' = Success
Time = 750, Nb Executed Rules = 1, Nb Executed Tasks = 6
*****
Scenario Number 1 : 'Scenario 2' = Failure
Expected result 'the loan report is approved equals ': Failed
Time = 16, Nb Executed Rules = 1, Nb Executed Tasks = 6
*****
Nb Executed 2
Nb Failures 1
Nb Errors 0
End running the JVM for testing
```

Section 4. Enabling testing from Business console

In this section, you publish the updated project to Decision Center so that business users can generate scenario files to run tests and simulations from Business console.

1. Publish the decision service to Decision Center.
-



Reminder

Make sure that the Sample Server is running.

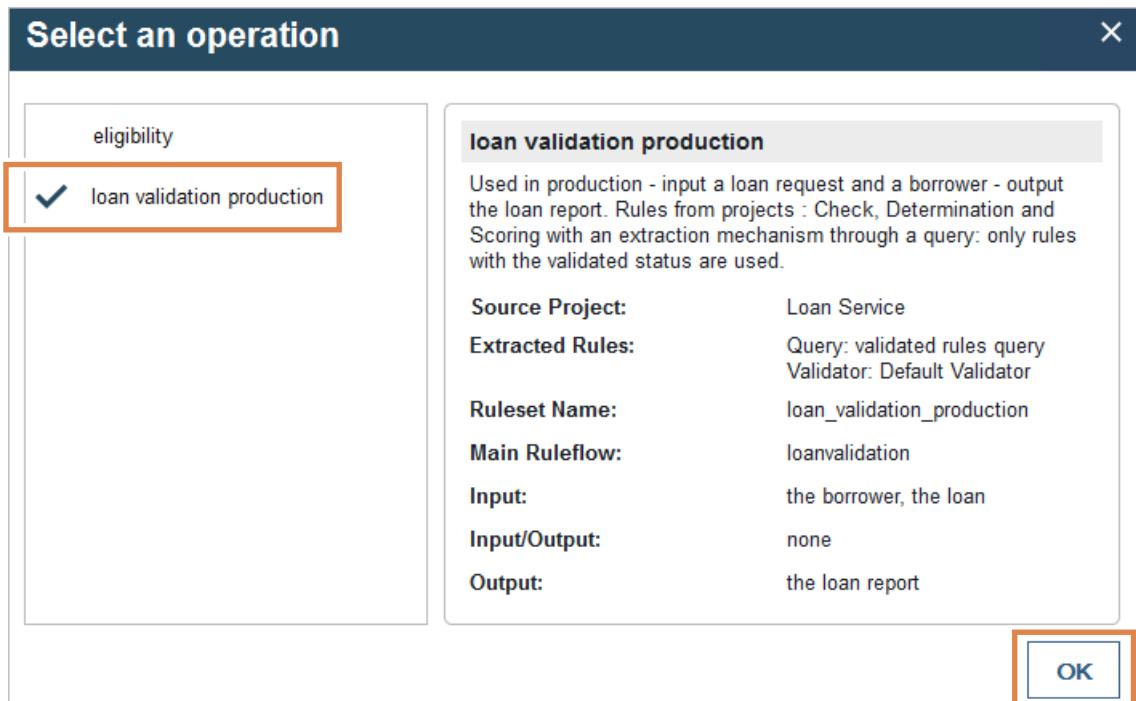
- a. In Rule Explorer, right-click **Loan Service** and click **Decision Center > Connect**.
 - b. Enter the following values in the Decision Center Configuration window fields.
 - **URL:** `http://localhost:9090/teamserver`
 - **User name:** `rtsAdmin`
 - **Password:** `rtsAdmin`
 - **Data source:** (Leave the field empty)
 - c. Click **Connect**.
 - d. Click **Next**.
 - e. On the Synchronization Settings page, click **Next**.
 - f. On the Decision Service Dependent Projects page, keep **Loan BOM** selected and click **Finish**.
2. When prompted to switch to the Synchronizing view, click **No** (because no conflicts exist).

Section 5. Testing in Business console

- ___ 1. Open the Business console by double-clicking the **Decision Center Business console** shortcut on the **Start** menu.
- ___ 2. Sign in with the `rtsAdmin` for the user name and password.
- ___ 3. On the **Library** tab, click **Loan Service**.
- ___ 4. On the **Branches** tab, click **main**.
- ___ 5. Click the **Tests** tab and make sure that you are on the **Test Suites** page.
- ___ 6. Generate a scenario file that is called **My Scenario File**.
 - ___ a. Click the **Generate Scenario File** icon.



- ___ b. In the “Select an operation” window, select **loan validation production**, and click **OK**.



- ___ c. In the **Filename** field, type: **Loan Scenario**

- ___ d. In the **Tests** section, expand **the loan report** and select **approved**.

* **Filename:**
Loan Scenario

Locale:
English (United States)

Scenario file format:
Excel Workbook (.xlsx)

Operation:
loan validation production

Select the tests to include in the scenario file.

* **Tests:**

Field	Operator
<input type="checkbox"/> the loan report	equals
<input type="checkbox"/> borrower	
<input type="checkbox"/> loan	
<input checked="" type="checkbox"/> approved	

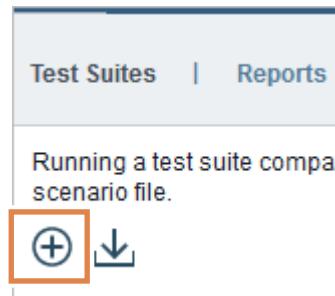
- ___ e. In the upper-right corner, click the **Download** icon and save the file to the default **Downloads** folder.



- ___ f. Go to the **Downloads** folder and view your scenario file in Excel Viewer.

- ___ g. After you review the **Scenarios** tab and **Expected Results** tab, close the file.

- ___ 7. In Business console, click **main** in the **Loan Service > main** breadcrumb to return to the **Tests** tab.
- ___ 8. Create a test suite called: **My Test Suite**.
- ___ a. Click the plus (+) icon to add a new test suite.



- ___ b. When prompted to select a decision operation, select **loan validation production** and click **OK**.
- ___ c. In the **Name** field, change the name to: **My Test Suite**

- ___ d. In the **File to use** field, click **Choose** and browse to the `Loan Scenario-populated.xlsx` file that is provided for you in the `C:\labfiles\code` folder.



You can view the file in Windows Explorer and review it later after you see the results of your test.

- ___ e. In the **Expected execution details to include in report** section, select **The list of rules fired**.

Report

* Report name:
Report

Expected execution details to include in report:

- The number of rules fired
- The number of executed ruleflow tasks
- The number of rules not fired
- The number of not executed ruleflow tasks
- The list of rules not fired
- The list of rules fired
- The list of executed ruleflow tasks
- The duration (in ms) of execution

- ___ f. In the upper-right corner, click **Save and Run**.



- ___ g. When prompted, click **Create New Version** and click **OK** on the Run Test Suite window.

- ___ 9. After execution finishes, click the **Report** link on to view the results. The most recent execution report is at the top of the list.

Decision Artifacts			Queries	Tests	Sim				
Test Suites Reports									
Reports are created by running test suites.									
Total: 1 Selected: 0				1					
Name	Operation	Status							
<input checked="" type="checkbox"/> Report - 2019- 2-20_11-18-0...	loan validation production								

The report shows that the tests ran without errors. The rules were executed correctly and produced a decision for each scenario. In this case, one scenario was successful and one failed.



The 50% success rate means that execution did not produce all the results that the business users expected. In such cases, business users must review the data that is provided in the scenario file to determine whether they should change the scenario or change the rules.

- ___ 10. Click **Close** to close the report
- ___ 11. Close the Business console.

End of exercise

Exercise review and wrap-up

The exercise looked at how to validate decision service projects for business users by preparing the BOM and creating a customized scenario file template. It also demonstrated how business users generate test suites and run tests from Business console.

Exercise 15. Managing deployment

Estimated time

01:00

Overview

This exercise teaches you how to deploy rules and XOMs for managed execution with Rule Execution Server.

Objectives

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
- Build and deploy rulesets in the Decision Center API console

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, "Managing XOM deployment"](#)
- [Section 5, "Exporting deployment server definitions"](#)
- [Section 6, "Managing deployment with the Decision Center API tool"](#)

Requirements

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

- Start project: Dev 15 – Deploy\start

Section 1. Creating deployment configurations

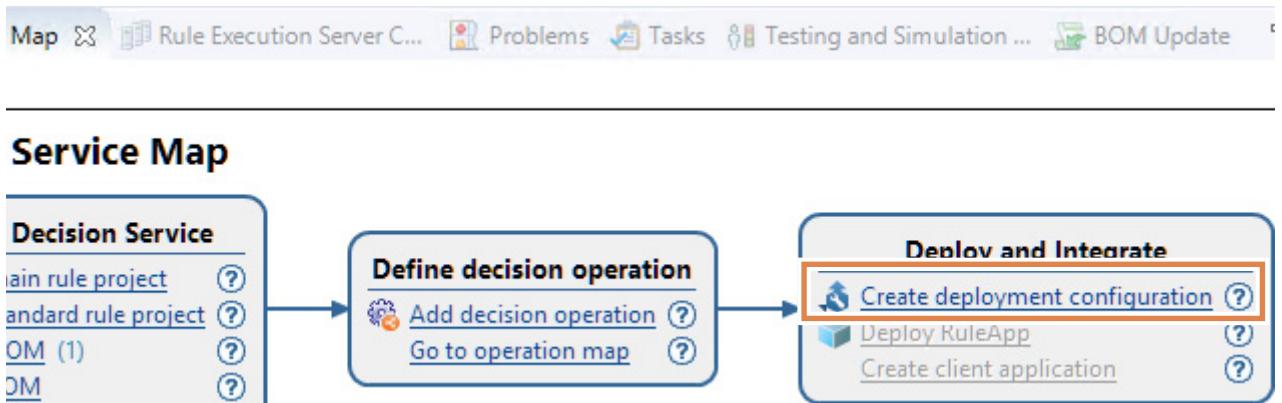
In this part of the exercise, you create a deployment configuration that can be used to deploy the decision service to Rule Execution Server or for a Java SE environment.

1.1. Setting up your environment for this exercise

- 1. If the sample server is not started, start it now by clicking **Start** and then clicking the **Start Sample Server** shortcut.
Starting the server might take several minutes.
- 2. In Rule Designer, switch to a new workspace and name it: `deploy`
 - a. From the **File** menu, click **Switch Workspace > Other**.
 - b. In the **Workspace Launcher** window, enter the path:
`C:\labfiles\workspaces\deploy`
 - c. Click **Launch**.
- 3. Close the Welcome view.
- 4. Use the Samples Console to import the start projects for this exercise.
 - a. Click the **Open Perspective** icon.
 - b. In the Open Perspective window, click **Samples Console** and click **Open**.
 - c. In the **Rule Designer** section, expand **Training > Ex 15: Managing deployment > 01-start**, and click **Import projects**.
 - d. When the workspace finishes building, close the **Help** view.

1.2. Creating a deployment configuration

- 1. Create a deployment configuration named: `loan-deploy`
 - a. In Rule Explorer, select **loanvalidation-rules** with your mouse.
 - b. In the **Rule Project Map** view, in the **Deploy and Integrate** part of the Decision Service Map, click **Create deployment configuration**.



- __ c. Set the deployment configuration name to: loan-deploy
- __ d. Click **Finish**.

The configuration editor opens to the **Overview** tab. The deployment configuration has errors because the configuration settings are not yet defined.

General

Name: loan-deploy

Type: Production Nonproduction Deploy the XOM: Yes No

Description:

RuleApp

Configure the RuleApp to be created upon deployment.

RuleApp Name: loan_deploy

RuleApp Base Version: 1.0

Target Servers

Define target servers...

Overview **Decision Operations** **Target Servers** **Source**

- __ 2. On the **Overview** tab, in the **General** section, in the **Deploy the XOM** option, select **No**.

General

Type: Nonproduction Deploy the XOM: Yes No

Decision Operations

Define decision operations for RuleApp deployment:

Include decision operations

Overview **Decision Operations** **Target Servers** **Source**

- 3. In the **Decision Operations** section, click **Include decision operations**.

The deployment configuration does not reference any active decision operation.

Decision Operations
Define decision operations for RuleApp deployment

+ Include decision operations

The editor switches to the **Decision Operations** tab.



- 4. Add the `loan-operation` decision operation to the configuration.

- a. In the **Configured Decision Operations** section, click the plus sign (+).

Select and configure decision operations for RuleApp deployment:

+ X E R

- b. Choose **Select existing decision operations** and expand `loanvalidation-rules` to choose `loan-operation`.

Select a Decision Operation

Create or select a decision operation

— X

○ Create a decision operation

● Select existing decision operations:

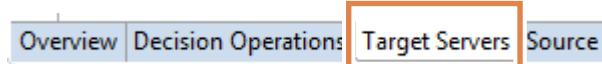
loanvalidation-rules

loan-operation

- c. Click **Finish**.

1.3. Define the target server for the configuration

- 1. Click the **Target Servers** tab and define the target server.



- 2. Click the plus sign (+) to add a server.

- 3. Leave **Create a Rule Execution Server connection** selected and click **Next**.

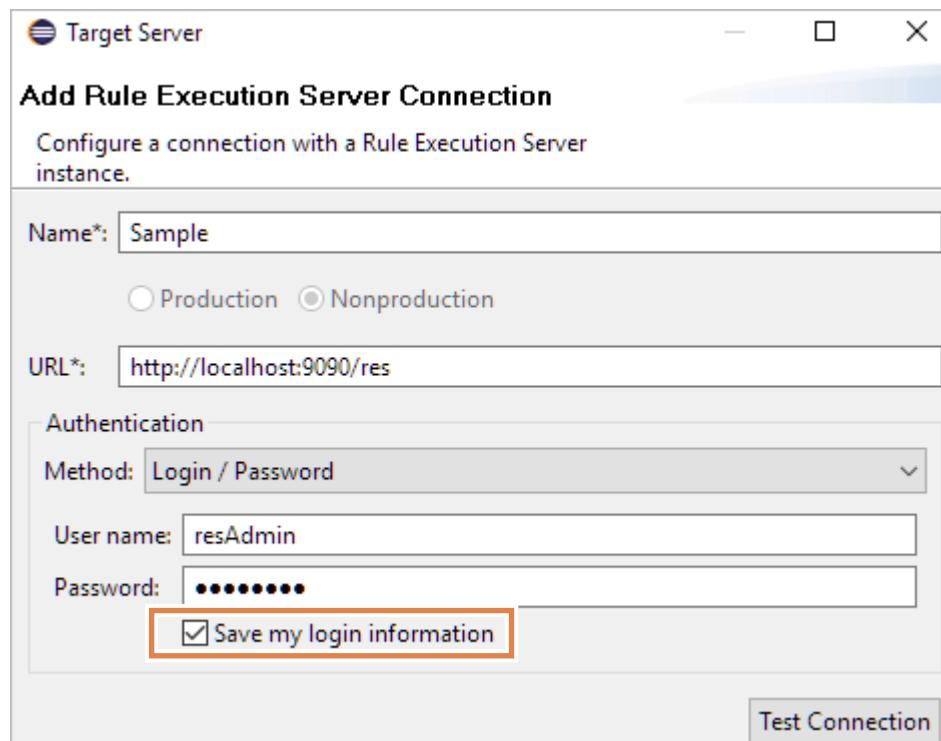
— 4. Define these fields:

- **Name:** Sample
- **URL:** http://localhost:9090/res
- **User name:** resAdmin
- **Password:** resAdmin

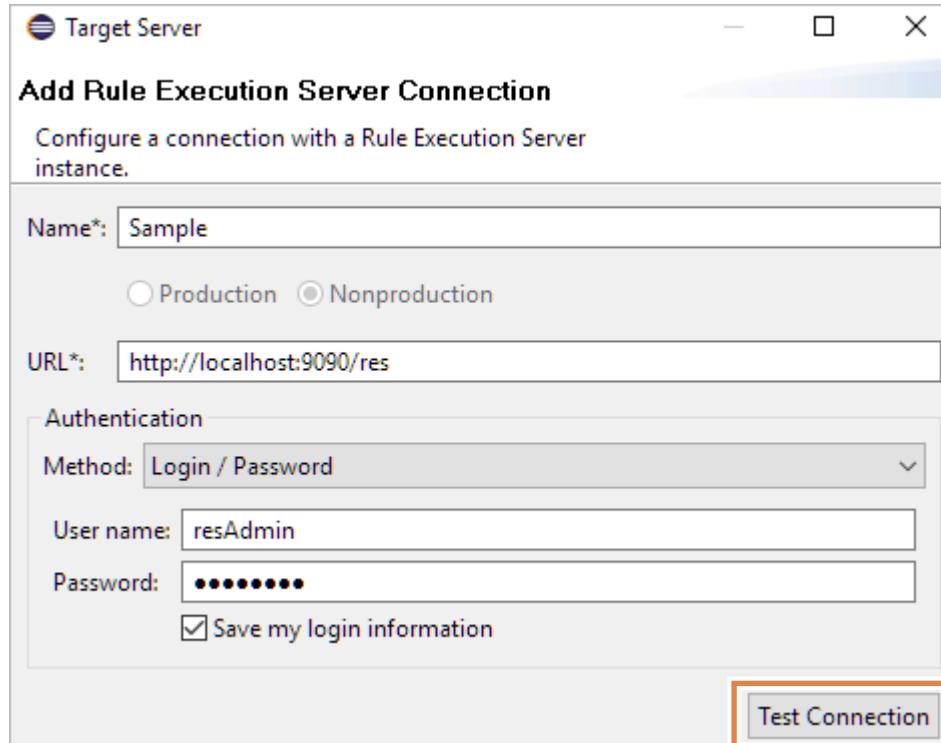
For IBM ODM on Cloud users

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.

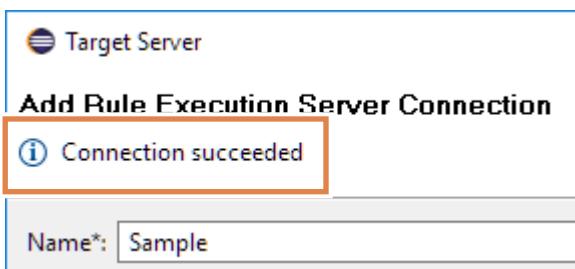
— 5. Select **Save my login information**.



- ___ 6. Click **Test Connection**.



You see the **Connection succeeded** message at the top of the Target Server window.



- ___ 7. Click **Finish**.
___ 8. Save your work (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

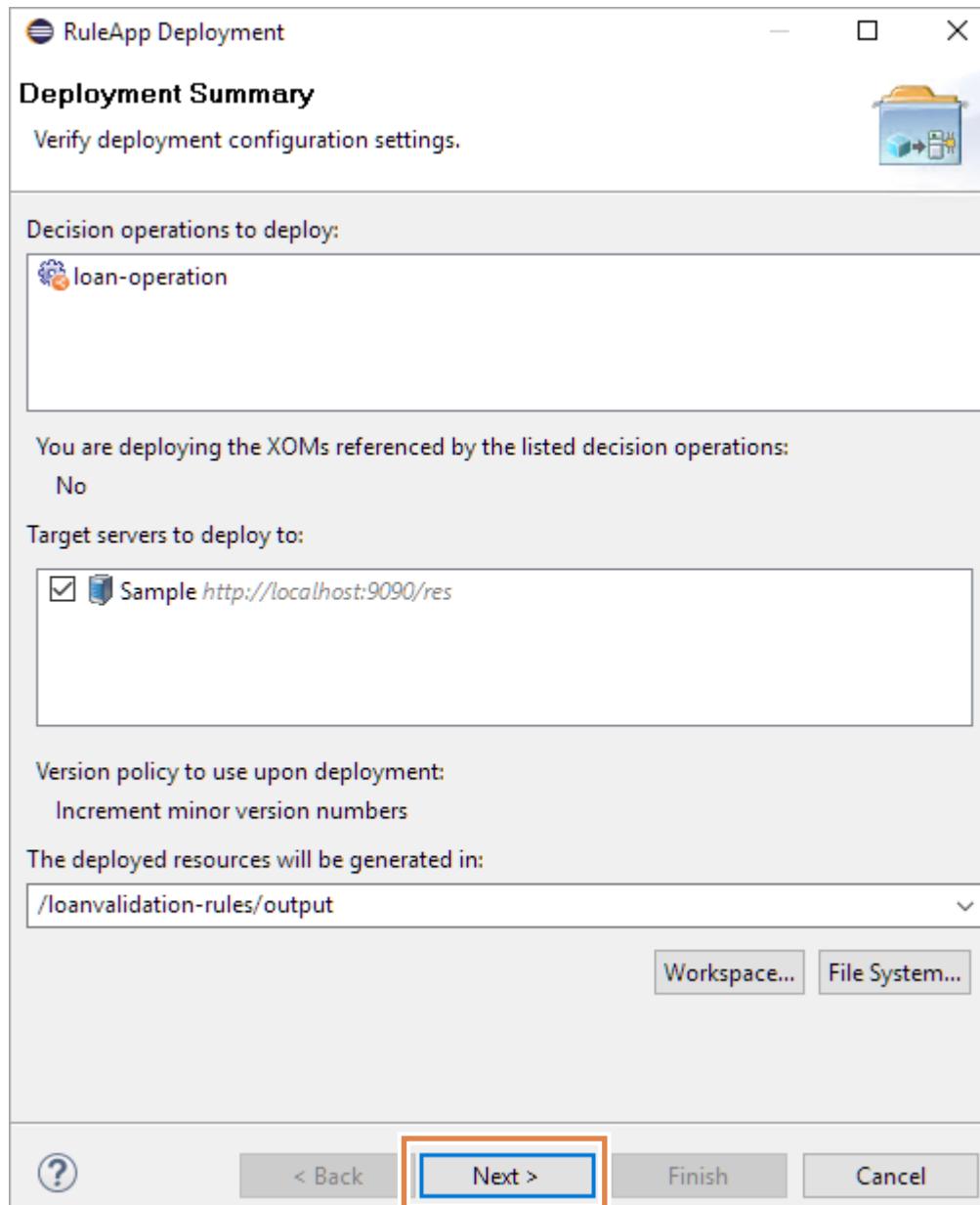
In this section, you deploy the RuleApp.

- ___ 1. Return to the **Overview** tab of the `loan-deploy` deployment configuration editor.
- ___ 2. In the lower right, in the **Deployment** section, click **Proceed to RuleApp deployment**.

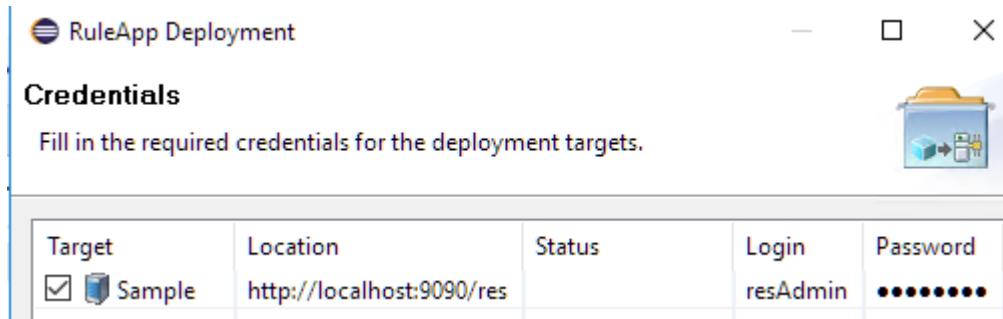
The screenshot shows the **Overview** tab of the deployment configuration editor for the `loan-deploy` RuleApp. The interface is divided into several sections:

- Decision Operations:** A panel on the right containing the text "Define decision operations for RuleApp deployment:" and a link to "loan-operation".
- Deployment:** A large central area with a scroll bar, currently empty.
- Target Servers:** A panel on the right containing the text "Define target servers for RuleApp deployment:" and a link to "Sample".
- Properties:** A section with fields for "Name" (set to `loan_deploy`) and "Base Version" (set to `1.0`).
- Deployment:** A section at the bottom right containing a button labeled "Proceed to RuleApp deployment..." which is highlighted with a red border.

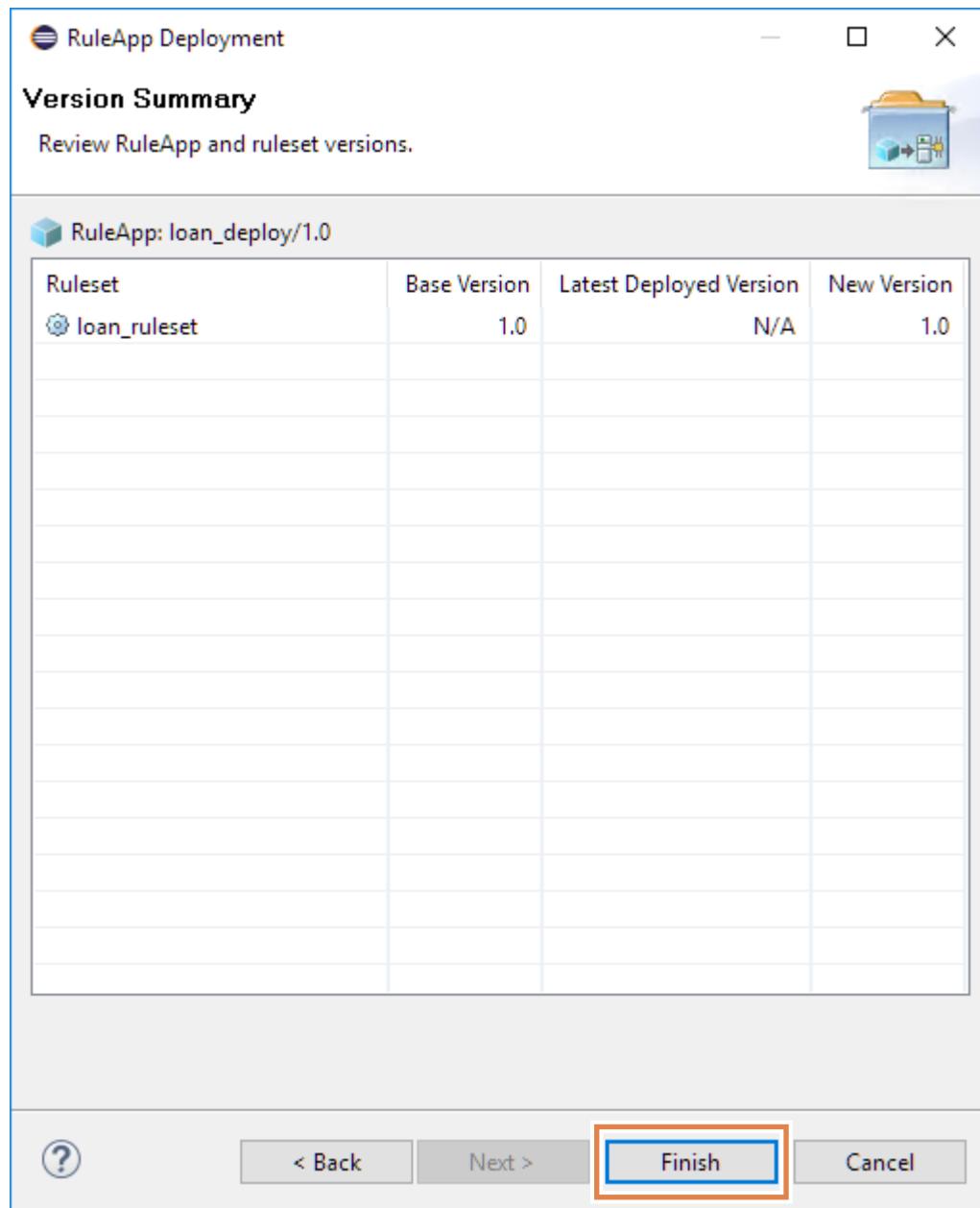
- ___ 3. After reviewing the deployment summary, click **Next**.



- ___ 4. After verifying the login credentials for the server, click **Next**.



5. On the Version Summary page, click **Finish**.



The Deployment Report opens and lists the deployed artifacts.

The screenshot shows the 'Deployment Report - loan-deploy' window. It displays two main sections: 'RuleApp Information - loan-deploy' and 'Ruleset Information - loan-operation'. Both sections contain two items, each preceded by an information icon (i) and labeled 'File name:' and 'Version:'. The 'File name:' for the RuleApp is 'loan_deploy.jar' and for the ruleset it is 'loan_ruleset.dsar'. The 'Version:' for both is '1.0'. These four items are highlighted with a red box. Below these sections are 'RuleApp Build Status' and 'RuleApp Deployment' sections, both containing one item each with an information icon and descriptive text. The final message 'Deployment successful' is displayed at the bottom.

RuleApp Information - loan-deploy

- i File name: loan_deploy.jar
- i Version: 1.0

Ruleset Information - loan-operation

- i File name: loan_ruleset.dsar
- i Version: 1.0

RuleApp Build Status

- i RuleApp file written to: C:\jabfiles\workspaces\intro\loanvalidation-rules\output\loan_deploy.jar
- i RuleApp build finished with no errors.

RuleApp Deployment

- i The RuleApp was successfully deployed to Sample (<http://localhost:9090/res>).

Deployment successful

6. Close the **Deployment Report** tab.



Questions

Do you notice a difference in the name of the deployment configuration? What are the names of the deployed rulesets?

Answer

Empty spaces and hyphens (-) are not authorized characters in the name of a RuleApp or of a ruleset.

The RuleApp defined with as `loan-deploy` with a hyphen (-) is deployed as a `loan_deploy` with underscore (_). The same principle applies to the name of the deployed rulesets that use underscore (_) to replace the empty spaces in the names.



Important

The ruleset paths are:

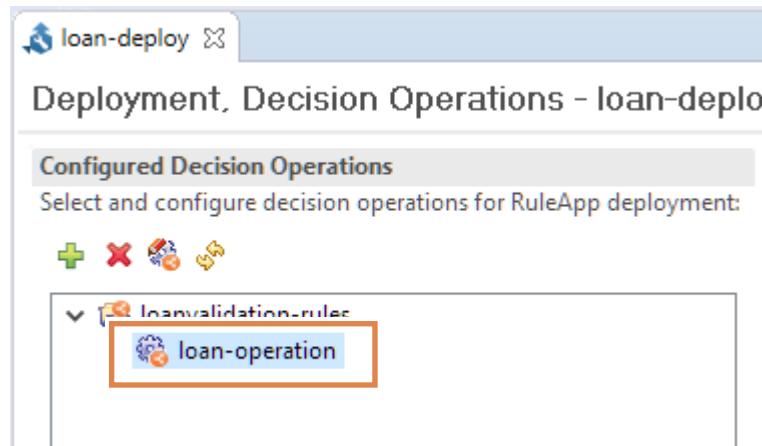
- /loan_deploy/1.0/loan_validation_production/1.0
- /loan_deploy/1.0/loan_validation_with_score_and_grade/1.0

Client applications must use the ruleset path to request the execution of the 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. Open the **Decision Operations** tab of the `loan-deploy` deployment configuration.
- 2. In the **Configured Decision Operations** section, under **loanvalidation-rules**, select **loan-operation**.

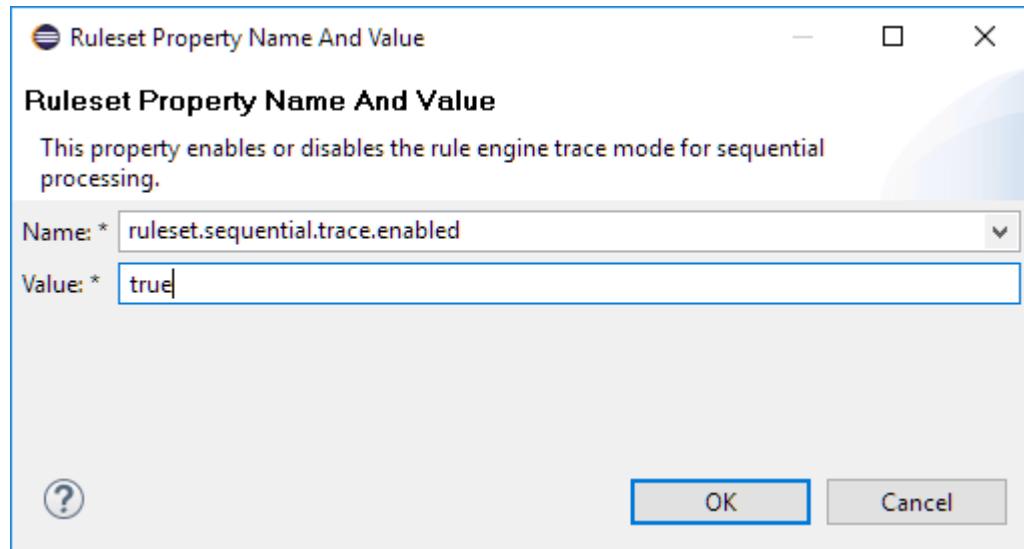


- 3. In the **Ruleset Properties** section, click the plus sign (+) to add a ruleset property.

Name	Value

- 4. Define the ruleset property.
 - a. In the Ruleset Property Name And Value window, in the **Name** field list, type: `rulesetSEQUENTIALTRACEENABLED`

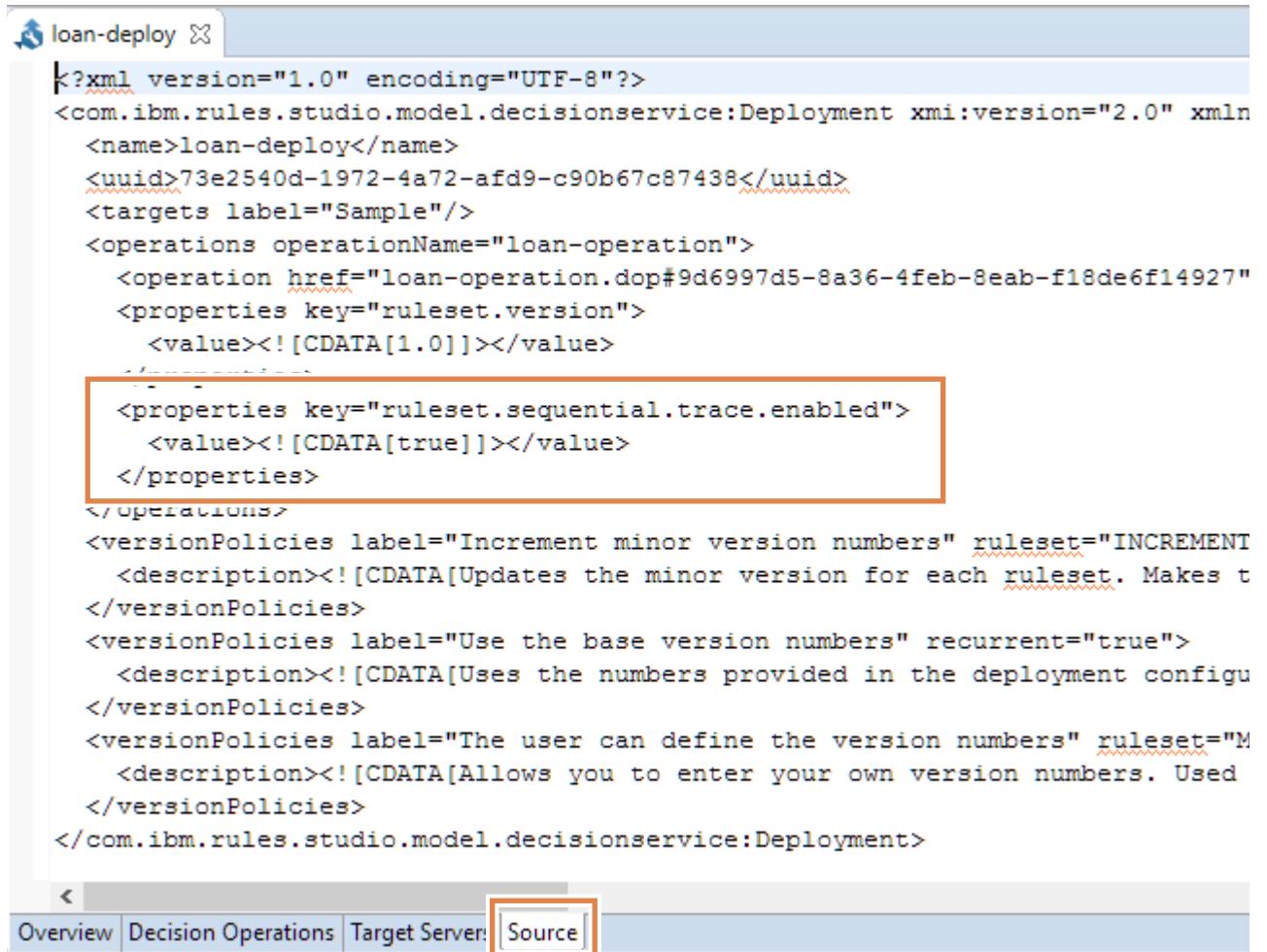
- ___ b. In the **Value** field, type: true



- ___ c. Click **OK** and save your work (Ctrl+S).

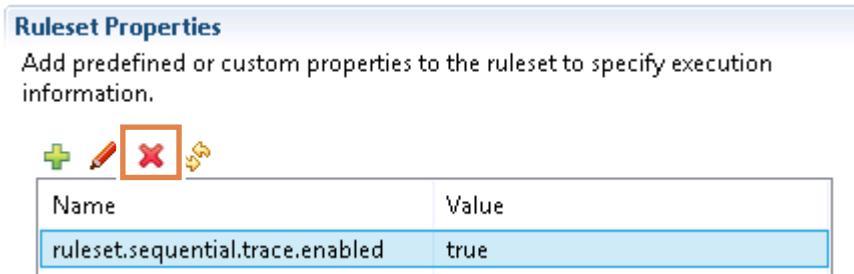
5. Open the **Source** tab to find the added XML code, and verify that the XML definition of the ruleset property is as follows:

```
<properties key="rulesetSEQUENTIALTRACEENABLED">
  <value><! [CDATA[true]]></value>
</properties>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<com.ibm.rules.studio.model.deployment:xmi:version="2.0" xmlns="http://www.eclipse.org/xmi/v2">
  <name>loan-deploy</name>
  <uuid>73e2540d-1972-4a72-afd9-c90b67c87438</uuid>
  <targets label="Sample"/>
  <operations operationName="loan-operation">
    <operation href="loan-operation.dop#9d6997d5-8a36-4feb-8eab-f18de6f14927">
      <properties key="ruleset.version">
        <value><! [CDATA[1.0]]></value>
      </properties>
    </operation>
    <!--
      <properties key="rulesetSEQUENTIALTRACEENABLED">
        <value><! [CDATA[true]]></value>
      </properties>
    -->
  </operations>
  <versionPolicies label="Increment minor version numbers" ruleset="INCREMENT">
    <description><! [CDATA[Updates the minor version for each ruleset. Makes it easier to track changes.]]></description>
  </versionPolicies>
  <versionPolicies label="Use the base version numbers" recurrent="true">
    <description><! [CDATA[Uses the numbers provided in the deployment configuration.]]></description>
  </versionPolicies>
  <versionPolicies label="The user can define the version numbers" ruleset="MANUAL">
    <description><! [CDATA[Allows you to enter your own version numbers. Used for custom builds.]]></description>
  </versionPolicies>
</com.ibm.rules.studio.model.deployment:xmi:version="2.0" />
```

6. Return to the **Decision Operations** tab and delete the rulesetSEQUENTIALTRACEENABLED property by selecting it and clicking the X.



Name	Value
rulesetSEQUENTIALTRACEENABLED	true

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. Managing XOM deployment

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. Include the XOM in the deployment configuration

In this section, you deploy the XOM directly from the rule project.

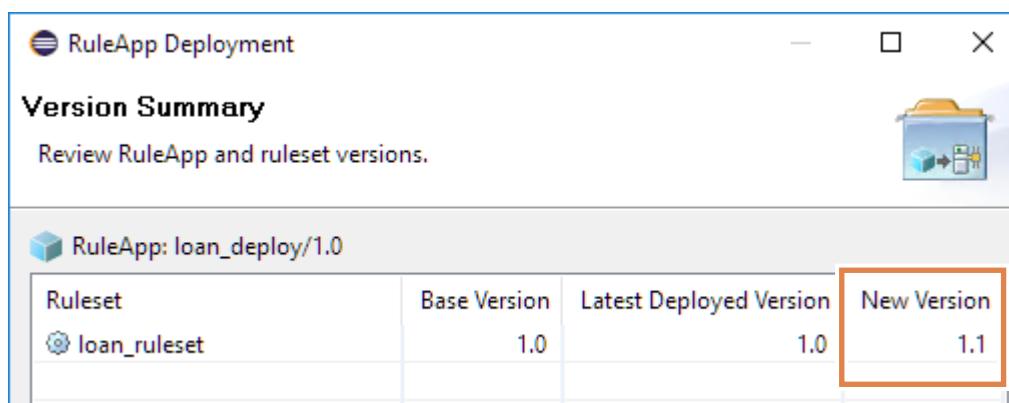
- ___ 1. In Rule Explorer, expand **loanvalidation-rules > resources > xom-libraries** to see the `loanvalidation-xom.zip` file.
- ___ 2. Include the XOM in the project deployment.
 - ___ a. Open the **Overview** tab of the `loan-deploy` deployment configuration.
 - ___ b. In the **General** section, in the **Deploy the XOM** option, choose **Yes**.



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

4.2. Deploy the managed XOM

- ___ 1. In the **Deployment** section, click **Proceed to RuleApp deployment**.
- ___ 2. If you see a “Java version notification” window, select the **Do not show this message again** check box and click **OK**.
- ___ 3. Click **Next** until you reach the Version Summary page, and notice that the minor version number for ruleset is incremented.



- ___ 4. Click **Finish**.

- 5. 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-operation' references the following target Rule project: [loanvalidation-rules].
- i** The following XOM resources were found for Rule project 'loanvalidation-rules': [loanvalidation-xom].
- i** The resource 'loanvalidation-xom.zip' was successfully deployed with version '1.0' on Sample (<http://localhost:9090/res>).
- i** No previous library matching this deployment was found on Sample (<http://localhost:9090/res>): a new library will be created.
- i** The library 'loan_deploy_library' was successfully deployed with version '1.0' on Sample (<http://localhost:9090/res>).



Note

Depending on the order in which you completed the exercises, or 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.

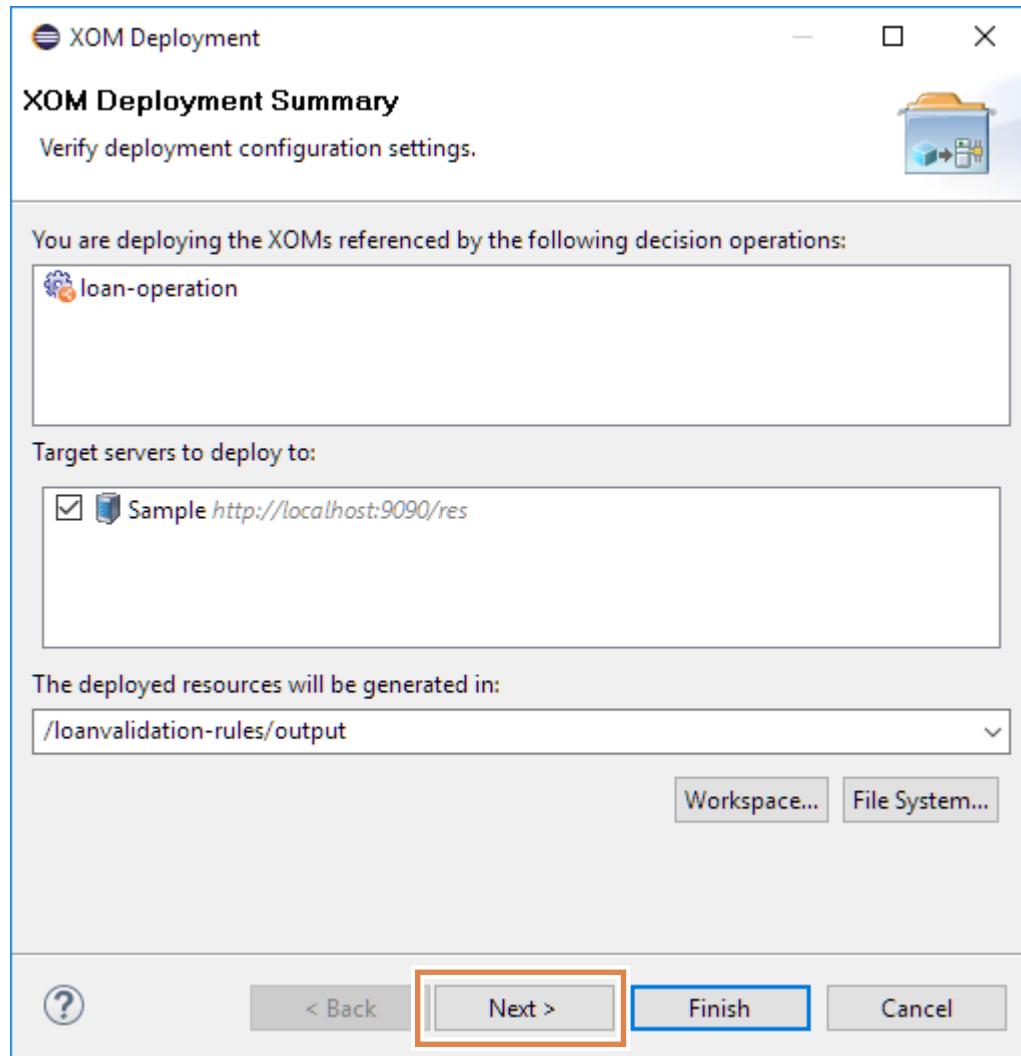
- 6. In Rule Explorer, expand **loanvalidation-rules > resources > xom-libraries** and double-click the `loanvalidation-xom.zip` file to open it.
The `loanvalidation-xom.zip` file opens in Windows Explorer to the **loan** folder, which contains the classes.
- 7. Close the folder and Windows Explorer.

4.3. Modify and redeploy the XOM

In the next steps, you change the XOM, redeploy it, and verify that the version of the deployed XOM is incremented.

- 1. In the Rule Explorer, expand **loanvalidation-xom > src > loan** and double-click the `Test.java` class to open it.
- 2. Edit the `Test.java` class by commenting the final line of the `main` method:
`// System.out.println(r1);`
- 3. Save the `Test` class and wait for the workspace to rebuild.
- 4. Redeploy the XOM from the **loanvalidation-rules** project.
 - a. In Rule Explorer, right-click the `loanvalidation-rules` project and click **Rule Execution Server > Deploy XOM**.

- ___ b. Click Next.



- ___ c. Click **Finish**.

The **XOM Deployment** section of the Deployment Report shows that the `loanvalidation-xom` version is now incremented.

XOM Deployment

- ℹ The decision operation 'loan-operation' references the following target Rule project: `[loanvalidation-rules]`.
- ℹ The following XOM resources were found for Rule project 'loanvalidation-rules': `[loanvalidation-xom]`.
- ℹ The resource 'loanvalidation-xom.zip' was successfully deployed with version '2.0' in Sample (<http://localhost:9090/res>).
- ℹ No previous library matching this deployment was found on Sample (<http://localhost:9090/res>): a new library will be created.
- ℹ The library 'loan_deploy_library' was successfully deployed with version '1.1' on Sample (<http://localhost:9090/res>).

Deployment successful

Because the XOM changed, its checksum also changed, as did the version of the managed XOM in Rule Execution Server.

- ___ 5. Close the deployment report tab.

**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.

4.4. 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](#) on page 15-16.
 - a. In the `loanvalidation-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.
 - c. Wait for your workspace to rebuild.
- 2. Redeploy the XOM.
 - a. In Rule Explorer, right-click the `loanvalidation-rules` project and click **Rule Execution Server > Deploy XOM**.
 - b. Click **Finish**.
 - c. Look at the Deployment Report and look for this line:

The resource '`loanvalidation-xom-zip`' is already deployed with version '`1.0`' on Sample (`http://localhost:9090/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**

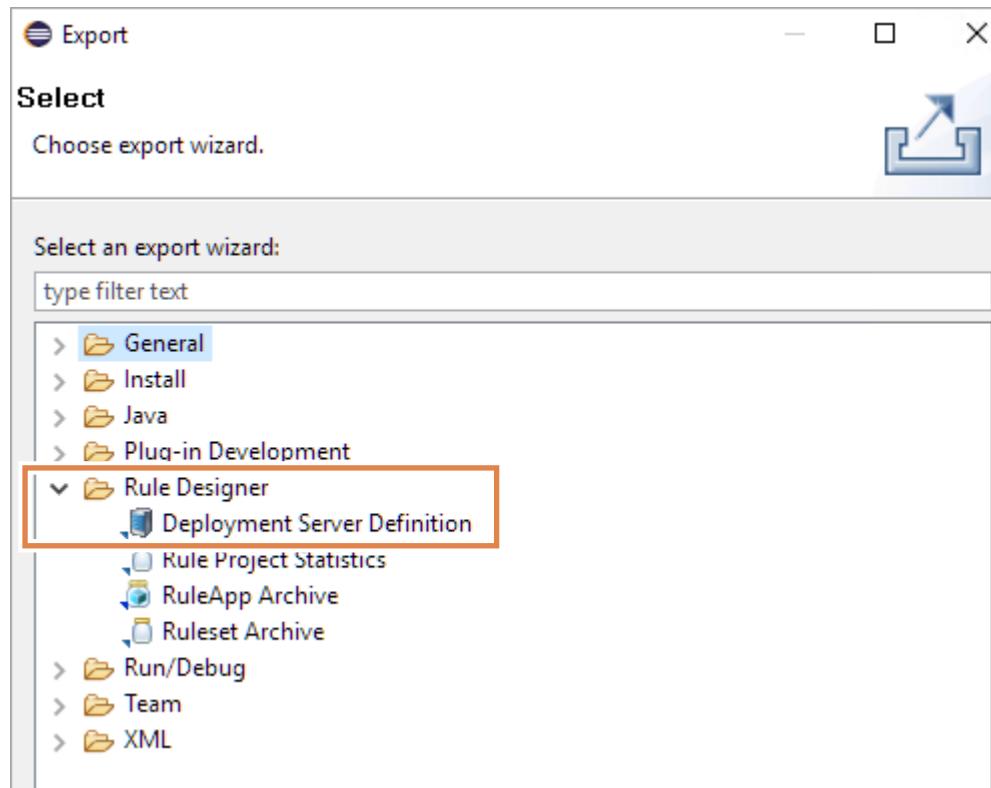
Depending on the number of times you tried deploying the RuleApp during this exercise, you might see a different version number in your results. The main point of this exercise is to recognize how you can use version numbers to manage deployments.

Section 5. Exporting 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. In Rule Designer, from the **File** menu, click **Export**.
- 2. In the Export wizard, expand **Rule Designer**, select **Deployment Server Definition**.



- 3. Click **Next**.
- 4. In the **File** field, click **Browse** and save the `server-list.xml` file to the `C:\labfiles` folder.
- 5. In the Export wizard, click **Finish**.

The file is now available for import to another workspace where you want to use this deployment configuration.

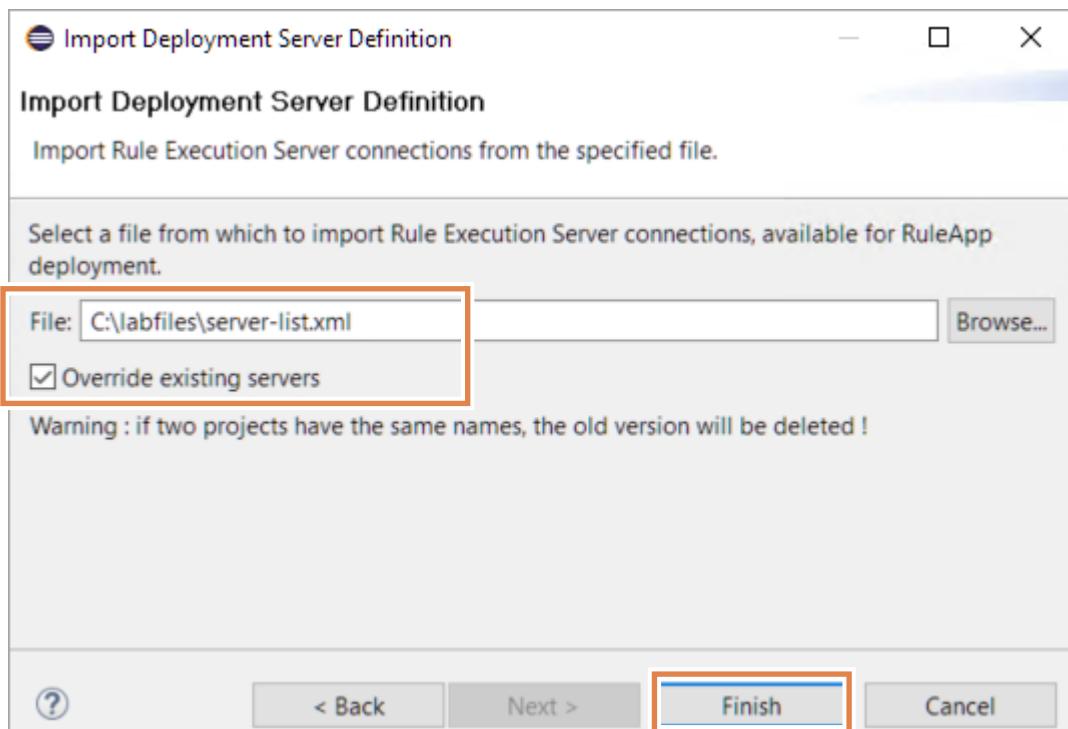
5.2. Importing deployment configurations

To see how to import a deployment server definition, you can switch to a new (empty) workspace.

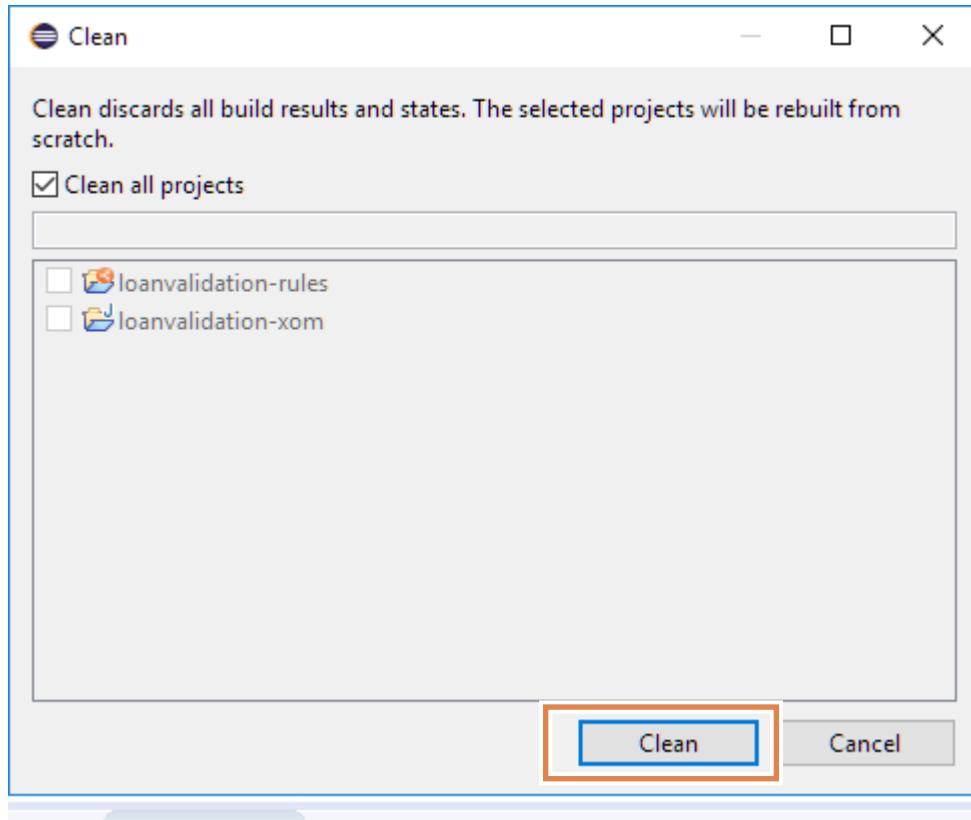
- 1. In Rule Designer, switch to a new workspace with the path:

`C:\labfiles\workspaces\deploy2`

- ___ 2. Use the Samples Console to import the project.
 - ___ a. Click the **Open Perspective** icon.
 - ___ b. In the Open Perspective window, click **Samples Console** and click **Open**.
 - ___ c. In the **Rule Designer** section, expand **Training > Ex 15: Managing deployment > 02-answer**, and click **Import projects**.
- ___ 3. In Rule Explorer, expand **loanvalidation-rules > deployment** and double-click **loan-deploy** to see that the deployment configuration has an error. And in the **Problems** view, you see a message that the target server cannot be retrieved.
- ___ 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**.
 - ___ c. In the Import Deployment Server Definition window, set the **File** field to the **C:\labfiles\server-list.xml** file.
 - ___ d. Select **Override existing servers** and click **Finish**.



- ___ 5. Open the **Project > Clean** menu and click **Clean**.



The errors should no longer be visible.

- ___ 6. Close Rule Designer.

Section 6. Managing deployment with the Decision Center API tool

In this section, you use the Decision Center REST API to explore decision services, build a deployment snapshot, and deploy it to Rule Execution Server.

6.1. Explore deployed decision services

- 1. Open the REST API tool from the Windows Start menu by clicking **IBM Operational Decision Manager > Decision Center API Console**.

The Decision Center API tool opens.

- 2. View the list of deployed decision services.
 - a. Click the **Explore** controller to open it and search for “GET decisionservices” to see the list of decision services.
 - b. Click **GET /v1/decisionservices** and click **Try it out**.

- ___ c. Scroll down and click **Execute**.

The screenshot shows the 'GET /v1/decisionservices' configuration pane. The 'Parameters' section contains the following fields:

- datasource**: string (query) - The JNDI name of the Decision Center data source. If not specified, it defaults to `jdbc/logDataSource`. Description: `datasource - The JNDI name of the Decision`
- page**: integer(\$int32) (query) - Page number. Description: `page - Page number`
- q**: string (query) - A query to filter objects by their properties. For example: `q=name:MyProject`. Description: `q - A query to filter objects by their properties`
- size**: integer(\$int32) (query) - Size of the page. If empty, all elements are returned. Description: `size - Size of the page. If empty, all elements`

At the bottom right of the pane is a blue 'Execute' button.

- ___ d. When prompted for a username and password, enter: `rtsAdmin`
- ___ 3. In the response body, find the ID of the Miniloan Service decision service.
- ___ a. Scroll through the list in **Response body** to find **Miniloan Service**.
- ___ b. Make a note of the ID: **f4440cab-4dca-471c-8a9e-ec05ede7c031**

The screenshot shows the 'Server response' pane. It includes a 'Code' section (status 200) and a 'Details' section. The 'Details' section contains a 'Response body' tab with the following JSON content:

```
{
  "id": "f4440cab-4dca-471c-8a9e-ec05ede7c031",
  "name": "Miniloan Service",
  "buildMode": "DecisionEngine",
  "advancedProperties": null
},
```

A red box highlights the `"id": "f4440cab-4dca-471c-8a9e-ec05ede7c031"` field in the JSON response.

- ___ c. Close the “GET decisionservices” pane by clicking it.
- ___ 4. Get the deployment ID for the Miniloan Service deployment configuration.
- ___ a. Click **GET /v1/decisionservices/{decisionServiceID}/deployments** and click **Try it out**.

- ___ b. In the **decisionServiceId** field, enter the Miniloan Service ID:

f4440cab-4dca-471c-8a9e-ec05ede7c031

The screenshot shows a REST API configuration interface. At the top, there is a blue button labeled "GET" and a URL path: "/v1/decisionservices/{decisionServiceId}/deployments". To the right of the URL, a description reads: "List of the deployment configurations for the given Decision Service". There is also a small lock icon.

Below the URL, there is a "Parameters" section with a "Cancel" button. This section contains a table with two columns: "Name" and "Description".

Name	Description
baselineld string (query)	ID of the target branch, activity or release. If none is specified, defaults to the 'main' branch
baselineld - ID of the target branch, activity o	
datasource string (query)	The JNDI name of the Decision Center datasource. If not specified, defaults to jdbc/ilogDataSource
datasource - The JNDI name of the Decision	
decisionServiceId * required string (path)	decisionServiceId f4440cab-4dca-471c-8a9e-ec05ede7c031

- ___ c. Scroll down and click **Execute**.

- ___ d. Notice the RuleApp is named **my_deployment** and make a note of the deployment ID: **44b9b9ad-f66d-408e-8ef0-61c8fe12f445**

Server response

Code	Details
200	Response body <pre>{ "elements": [{ "id": "44b9b9ad-f66d-408e-8ef0-61c8fe12f445", "name": "Miniloan", "production": false, "description": null, "ruleAppName": "my_deployment", "ruleAppVersion": "1.0", "snapshotMode": "Always" }], "totalCount": 1, "number": 0, "size": 1 }</pre> <div style="text-align: right;">Download</div>

- ___ 5. Scroll back up and click the **GET deployments** panel to collapse it.
 ___ 6. Scroll up and click **Explore** to collapse the controller.

6.2. Build a decision service

Next, you use the deployment ID to build a deployment snapshot.

- ___ 1. Expand the **Build** controller.
- ___ 2. Click **POST /v1/deployments/{deploymentID}/build** and click **Try it out**.
- ___ 3. In the **deployment ID** field, enter the deployment ID from the previous step:
 44b9b9ad-f66d-408e-8ef0-61c8fe12f445
- ___ 4. In the **deploymentSnapshotName** field, enter: miniloanDeployment
- ___ 5. Click **Execute** and wait for the build to complete.

- ___ 6. In the response, note the snapshot is correctly built for this decision service.

Server response

Code	Details
200	<p>Response body</p> <pre>{ "id": "1a1d677b-7d1c-400a-a8af-2ce10428052f", "internalId": "dsm.DSDeploymentReport:2:2", "name": "Report 2020-12-17_01-14-52-795", "status": "COMPLETED", "ruleAppName": "my_deployment", "messages": { "elements": [], "totalCount": 0, "number": 0, "size": 0 }, "snapshot": { "id": "e0338462-41bb-4f2a-b5d2-1425d6eb8d94", "internalId": "dsm.DsDeploymentBsln:130:130", "name": "miniloanDeployment", "parentId": "8ba28be0-8f18-48f7-80e8-993ca89577f8", "documentation": null, "buildMode": "DecisionEngine", "kind": "DeploymentSnapshot" }, "servers": [], "archive": null }</pre>

You now have a snapshot of the decision service that can be deployed.

6.3. Verify your snapshot in the Business console

- ___ 1. Open the Business console from the Windows Start menu. (If you are prompted to sign in, use **rtsAdmin** for the username and password.)
- ___ 2. On the **Library** tab, open the **main** branch of the **Miniloan Service** decision service.

- ___ 3. Go to the **Snapshots** tab and confirm that the **miniloanDeployment** snapshot is listed.

<input type="checkbox"/> Name	Type	Created By
<input type="checkbox"/> Miniloan_2020-10-25T01_57_55Z	Deployment Snapshot	rtsAdmin
<input type="checkbox"/> miniloanDeployment	Deployment Snapshot	rtsAdmin

6.4. Deploy the decision service

In this part of the exercise, you use the deployment ID and the ID for your new deployment snapshot to deploy the decision service.

- ___ 1. Return to the Decision Center API Console.
- ___ 2. In the **Build** controller, search for “POST deploy” to find the deploy operation.
- ___ 3. Click **POST /v1/deployments/{deploymentID}/deploy** and click **Try it out**.
- ___ 4. In the **deploymentId** field, paste the deployment ID from the build step:
44b9b9ad-f66d-408e-8ef0-61c8fe12f445
- ___ 5. In the **deploymentSnapshotName** field, enter the name of your snapshot:
miniloanDeployment
- ___ 6. Click **Execute**.

You see a successful deployment message.

Response body

```
{
  "id": "2378b954-f5a5-4909-912c-16090a0f503b",
  "internalId": "dsm.DSDeploymentReport:6:6",
  "name": "Report 2020-10-25_01-02-37-552",
  "status": "COMPLETED",
  "ruleAppName": "my_deployment",
  "messages": {
    "elements": [
      {
        "message": "The version '1.1' of the ruleset 'my_operation' was successfully deployed to Decision Service Execution (http://localhost:9090/res).",
        "ruleArtifact": null,
        "severity": "INFO"
      },
      {
        "message": "The RuleApp was successfully deployed to Decision Service Execution (http://localhost:9090/res).",
        "ruleArtifact": null,
        "severity": "INFO"
      }
    ],
    "totalCount": 2,
    "number": 0,
    "size": 2
  },
  "snapshot": {
    "Download"
  }
}
```

[Download](#)

6.5. Confirm that the updated Miniloan Service is deployed and accessible to client applications

- ___ 1. Open a browser at this URL: <http://localhost:9090/miniloan-server>
- ___ 2. Click **Execution**.
- ___ 3. Select **Use rules**.
- ___ 4. Set the **Ruleset** field to the canonical ruleset path to use the latest 1.1 version of the ruleset: my_deployment/1.0/my_operation/1.1

5. Click **Validate Loan**.

The screenshot shows a web application interface for a "Sample MiniLoan". The application is running at localhost:9090/miniloan-server/. The interface is divided into three main sections: Borrower, Loan, and Execution.

- Borrower:** Contains fields for Name (Joe), Yearly Income (80000), and Credit Score (600).
- Loan:** Contains fields for Amount (500000), Duration (240), and Yearly Interest Rate (0.05).
- Execution:** Contains fields for Server (http://localhost:9090), User (resAdmin), and Password (*****). It also includes checkboxes for "Use rules" (checked) and "Show trace" (unchecked). The "Ruleset" field is set to "my_deployment/1.0/my_operation/1.1".

The "Validate Loan" button is located at the bottom right of the execution section and is highlighted with a blue border.

The decision service executes correctly and the loan is approved.

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

Estimated time

00:45

Overview

This exercise teaches you how to work with the Rule Execution Server console.

Objectives

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 5, "Exploring the Diagnostics and Server Info tabs"](#)
- [Section 6, "Managing RuleApps in the console"](#)
- [Section 7, "Managing RuleApps with REST API tools"](#)

Requirements

This exercise requires that you complete [Exercise 15, "Managing deployment"](#) before starting this exercise.

Section 1. Exploring the Rule Execution Server console

1.1. Setting up your environment for this exercise

- 1. Make sure that the sample server is started. (Click **Start Sample Server** on the Start menu.)
-

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.

1.2. Exploring the Rule Execution Server console pages

- 1. Open the Rule Execution Server console.
 - a. On the Windows **Start** menu, click the **Rule Execution Server console** shortcut.
 - 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. Click the **Explorer** tab.

The Navigator pane gives access to the elements that you can manage in the Rule Execution Server console.

The screenshot shows the Rule Execution Server console interface. The top navigation bar includes Home, Explorer (which is highlighted with a red box), Decision Warehouse, Diagnostics, and Server. Below the navigation bar, the URL is shown as Explorer > RuleApps. The left side features a Navigator pane with a tree view containing RuleApps (2), Resources (2), Libraries (2), and Service Information (2). The RuleApps node has two children: /loan_deploy/1.0 (2) and /my_deployment/1.0 (2). The /loan_deploy/1.0 node has two children: /loan_ruleset/1.0 and /loan_ruleset/1.1. The right side displays the RuleApps View pane with a title 'RuleApps View' and three buttons: Add RuleApp, Deploy RuleApp Archive, and Update RuleApps. Below these buttons is a section titled 'RuleApps' with the text 'Total Number of RuleApps 2'. A table titled '2 RuleApp(s)' lists two entries: 'loan_deploy' (Version 1.0, Creation Date Dec 17, 2020, 12:40:55 PM) and 'my_deployment' (Version 1.0, Creation Date Dec 16, 2020, 10:34:58 AM). At the bottom of the table, it says 'RuleApp 1 - 2 of 2'.

- 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.

Exploring 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.

RuleApp View

The screenshot shows the 'RuleApp View' interface for the rule app '/loan_deploy/1.0'. At the top, there is a toolbar with four buttons: 'Add Ruleset', 'Add Property', 'Download Archive with All Rulesets', and 'Edit'. Below the toolbar, the rule app name '/loan_deploy/1.0' is displayed in a large blue header. Underneath the header, there is a table showing the following details:

Name	loan_deploy
Version	1.0
Creation Date	Oct 7, 2020, 7:10:00 PM GMT-4
Display Name	
Description	

- ___ 3. In the Navigator pane, click **loan_ruleset/1.0** to open the ruleset on the Ruleset View page.

Ruleset View

/loan_deploy/1.0/loan_ruleset/1.0

Name	loan_ruleset
Version	1.0
Creation Date	Oct 29, 2020, 12:35:23 AM GMT-4
Display Name	loan-operation
Description	
Rule engine	Decision Engine - 1.40.10
Status	✓ enabled
Debug	✗ disabled

Permanent link

- ___ 4. The Ruleset View pane opens, where you can see the `borrower`, `loan`, and `report` parameters that are defined for this ruleset.

Ruleset Parameters			
Direction	Name	Kind	XOM Type
in	borrower	native	loan.Borrower
in	loan	native	loan.Loan
in	report	native	loan.Report

Ruleset Parameters 1 - 3 of 3 prev 10 next 10

- ___ 5. In the Navigator page, click **loan_ruleset/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_library/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 **loanvalidation-xom.zip** link for the first XOM that you deployed in [Exercise 15, "Managing deployment"](#). Depending on the number of times that you deployed, you might have several versions.
- ___ 3. Click **Show references from Rulesets** to see which rulesets reference this XOM. If no rulesets reference this XOM, click **Resources** again in the Navigation pane and click a different XOM link.
- ___ 4. Click **Show references from Libraries** to see which XOM libraries reference this XOM. If no rulesets reference this XOM, click **Resources** again in the Navigation pane and click a different XOM link.

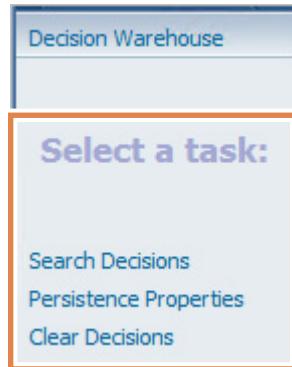
Section 4. Exploring the Decision Warehouse tab

- 1. Click the **Decision Warehouse** tab.

The **Decision Warehouse** tab includes these tasks:

Search Decisions	Define filters on the data source to find only the events or decisions of interest.
Persistence Properties	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	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.



2. On the Search Decisions page, leave all fields empty and click **Search** at the bottom of the **Search Decisions** page.

Search Decisions

Enter information in one or more fields to find stored decisions:

Executed ruleset path:	<input type="text"/>
Decision ID:	<input type="text"/>
Rules Fired and Tasks Executed:	<input type="text"/>
Input parameters:	<input type="text"/>
Output parameters:	<input type="text"/>
From date:	<input type="text"/>
From time:	00:00:00 <input type="button" value="▼"/>
To date:	<input type="text"/>
To time:	00:00:00 <input type="button" value="▼"/>

For decisions to be stored in Decision Warehouse, monitoring properties must be enabled on the ruleset. You work with monitoring in [Exercise 18, "Auditing ruleset execution through Decision Warehouse"](#).

Section 5. 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.



- ___ b. Click **Rerun**.
 - ___ 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.

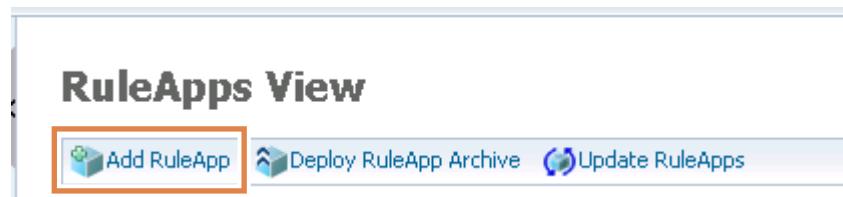
Server Info	
Total number of RuleApps	1
Total number of rulesets	2
Log File	C:\IBM\wlp\usr\servers\odm81040\res-console0.log
Log Level	Info

Section 6. Managing RuleApps in the console

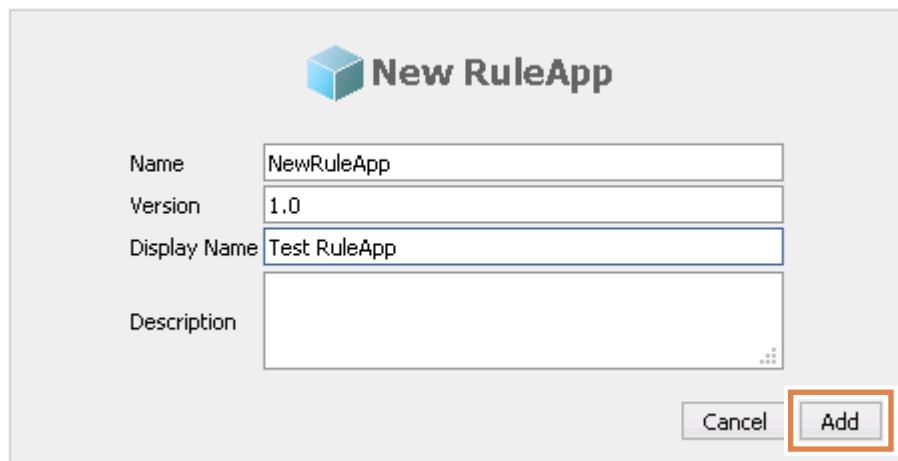
In this section, you see how to manage resources through the Rule Execution Server console.

6.1. Creating a RuleApp

- ___ 1. Click the **Explorer** tab.
- ___ 2. In the RuleApps View, click **Add RuleApp**.



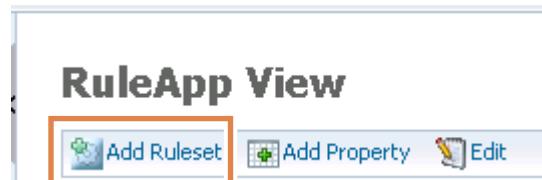
- ___ 3. In New RuleApp pane, leave the **Name** field set to: `NewRuleApp`
- ___ 4. In the **Version** field, keep: `1.0`
- ___ 5. In the **Display Name** field, enter: `Test RuleApp`
- ___ 6. Click **Add**.



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**.



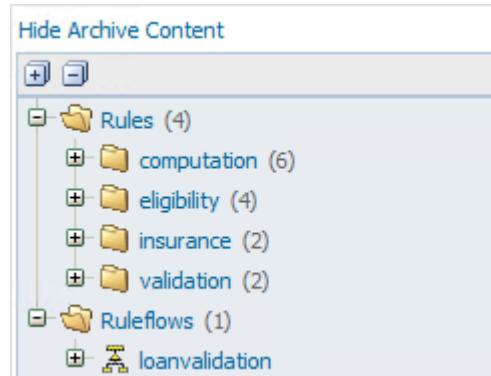
The New Ruleset pane opens.

- ___ 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 `output` folder of the workspace that you used in [Exercise 15, "Managing deployment"](#):
`C:\labfiles\workspaces\deploy\loanvalidation-rules\output`
 - ___ f. Select the `loanvalidation_rulesRuleset.dsar` file, and click **Open**.
 - ___ g. On the New Ruleset page, leave the other fields unchanged, and click **Add**.
The `NewRuleApp/1.0/NewRuleset/1.0` ruleset is now created and available in the Navigator view.
- ___ 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.



Section 7. Managing RuleApps with REST API tools

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.

7.1. Exploring REST API commands

- 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.

REST API tool

You can use this test tool to test the REST API for resource management. To test the REST API for remote ruleset execution, click the Explorer tab.

/ruleapps **/rulesets** **/libraries** **/xoms** **/decisiontraces** **/executionunits** **/utilities**

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 merging and versioning policies passed as parameters. The request body.

7.2. Deploying with REST API

- ___ 1. Deploy a XOM by using the REST API feature.
 - ___ a. On the **REST API** tab, click the **/xoms** tab.

The screenshot shows the REST API tool interface. At the top, there is a navigation bar with tabs: Home, Explorer, Decision Warehouse, Diagnostics, and Services. Below the navigation bar, a sub-menu for 'REST API' is visible. The main area is titled 'REST API tool' and contains the following text: 'You can use this test tool to test the REST API for resource management. To test the REST API for remote resources, click the REST API WADL file: /res/api/auth/v1/DecisionServer.wadl'. Below this text is a horizontal menu bar with several tabs: /ruleapps, /rulesets, /libraries, /xoms (which is highlighted with a red box), /decisiontraces, /executionunits, and /utilities.

- ___ b. Select **POST - deployResource**.

The screenshot shows the details of the POST /xoms/{xomname} endpoint. It includes a green button labeled 'POST' and the URL '/xoms/{xomname}'. Below this, a description box states: 'deployResource Deploys a managed XOM in the repository.'

- ___ c. In the Request Body section, click **Browse** and go to:
C:\labfiles\workspaces\deploy\loanvalidation-rules\output
- ___ d. Select loanvalidation-xom.jar and click **Open**.
- ___ e. In the **xomname** field, type: loanvalidation-xom.jar

___ f. Click Call method.

/xoms/{xomname}

DeployResource Deploys a managed XOM in the repository.

Build a request

/xoms/loanvalidation-xom.jar

Request Body

Request Body loanvalidation-xom.jar

URL Template Parameters

xomname loanvalidation-xom.jar

HTTP Header Parameters

<input checked="" type="checkbox"/> Content-Type	application/octet-stream
<input type="checkbox"/> Accept	application/xml
<input type="checkbox"/> Accept-Language	en
<input type="checkbox"/> X-Method-Override	
<input type="checkbox"/> X-HTTP-Method-Override	

URL Query Parameters

<input type="checkbox"/> versionPolicy	MAJOR_VERSION_POLICY
<input type="checkbox"/> date	iso8601
<input type="checkbox"/> accept	application/xml
<input type="checkbox"/> accept-language	en
<input type="checkbox"/> x-method-override	
<input type="checkbox"/> x-http-method-override	

- 2. In the Response section, make sure that you see HTTP Status 201.

Response

HTTP Status	201
-------------	-----

Headers

```
HTTP content-language: en-US
content-security-policy: frame-ancestors 'self' default-src 'self' 'unsafe-inline' 'unsafe-eval';cache-control:public, max-age=3600
Content-Allow-Credentials:false;Access-Control-Allow-Origin:null
content-type: text/xml; charset=UTF-8
date: Tue, 17 Nov 2020 03:41:28 GMT
transfer-encoding: chunked
x-content-type-options: nosniff
x-frame-options: SAMEORIGIN
x-powered-by: Servlet/3.1
x-xss-protection: 1; mode=block
```

Response Body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<result>
  <resource xsi:type="resource" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <id>loanvalidation-xom.jar/1.0</id>
    <uri>resuri://loanvalidation-xom.jar/1.0</uri>
    <name>loanvalidation-xom.jar</name>
    <version>1.0</version>
    <creationDate>2020-11-13T19:42:39.139Z</creationDate>
    <sha1>95f0a8ae0f73eb6eab768b678a76b4293c04e597</sha1>
  </resource>
  <succeeded>true</succeeded>
</result>
```

- 3. On the **Explorer** tab, in the Navigator pane, expand the **Resources** folder to confirm that the `loanvalidation-xom.jar` is listed as a resource.

7.3. Adding references to the XOM

Both the RuleApp and XOM are deployed, but before you can test ruleset execution, the RuleApp and XOM need to “see” each other. Next, you add references to the URI of the managed XOM resource.

- 1. Go back to the **REST API** tab.
 — 2. Open the `/ruleapps` tab and scroll down to **POST - addRulesetProperty** and click it.

POST `/ruleapps/{ruleappname}/{ruleappversion}/{rulesetname}/{rulesetversion}/properties/{propertyname}`

addRulesetProperty Adds a new, named property to a ruleset, identified by its name and version number, contained in a property value is passed in the request body. If the repository does not contain such a ruleset or the property already exists, the HTTP status 202 is returned.

- 3. Set the request parameters:
- **Request Body:** `resuri://loanvalidation-xom.jar/1.0`
 - **ruleappname:** `NewRuleApp`
 - **ruleappversion:** `1.0`
 - **rulesetname:** `NewRuleset`
 - **rulesetversion:** `1.0`
 - **propertyname:** `ruleset.managedxom.uris`
- 4. Click **Call method.**

- ___ 5. Make sure that the Response section shows HTTP Status 200.

Request	
URL	/api/v1/ruleapps/NewRuleApp/1.0/NewRuleset/1.0/properties/ruleset.managedxom.uris
Method	POST
HTTP Headers	Content-Type: text/plain
Request body	resuri://loanvalidation-xom.jar/1.0

Response	
HTTP Status	200
Headers	content-language: en-US content-security-policy: frame-ancestors 'self' default-src 'self' 'unsafe-inline' 'unsafe-eval';cache-control:public, max-age=14400;Access-Control-Allow-Credentials:false;Access-Control-Allow-Origin:null content-type: text/xml; charset=UTF-8 date: Thu, 29 Oct 2020 05:23:33 GMT transfer-encoding: chunked x-content-type-options: nosniff x-frame-options: SAMEORIGIN x-powered-by: Servlet/3.1 x-xss-protection: 1; mode=block
Response Body	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <result> <succeeded>true</succeeded> </result>

- ___ 6. Go back to the **Explorer** tab, and in the Navigator, expand **RuleApps > NewRuleApp/1.0** and select the **NewRuleset/1.0** ruleset.
 ___ 7. In the Ruleset View, click **Show Managed URIs**. The resource URI is set to the XOM resource.



Note

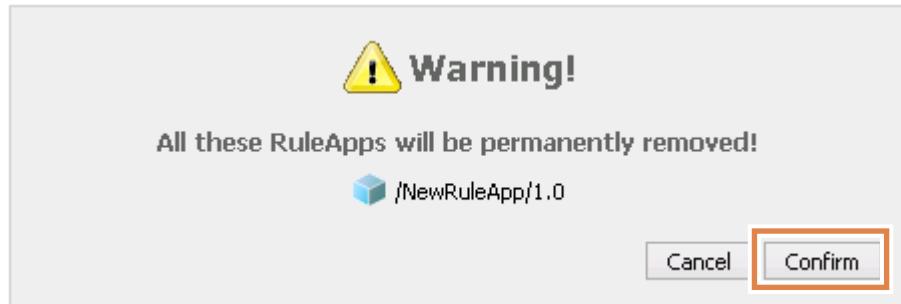
You work more with the REST API in [Exercise 17, "Executing rules using the REST service"](#).

7.4. 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**.

RuleApps					
Total Number of RuleApps 2					
2 RuleApp(s)			Name: <input type="text"/>	<input type="checkbox"/> View only:	<input type="checkbox"/> Latest version Display by: 10
<input type="checkbox"/>	Select All	Name	Version	Creation Date	Number of rulesets
<input type="checkbox"/>		loan_deploy	1.0	Oct 7, 2020, 7:10:00 PM GMT-4	2
<input checked="" type="checkbox"/>		NewRuleApp	1.0	Oct 21, 2020, 7:33:20 PM GMT-4	1
			prev 10 next 10		<input type="button" value="Remove"/>

- ___ 3. When prompted to confirm the removal of the RuleApp, click **Confirm**.



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 using the REST service

Estimated time

00:30

Overview

This exercise teaches you how to retrieve HTDS description files and test ruleset execution by using the REST service.

Objectives

After completing this exercise, you should be able to:

- Retrieve HTDS description files
- Test ruleset execution by using the REST service

Introduction

In this exercise, you review all the steps that are required to use hosted transparent decision services in an SOA environment. 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, "Retrieving the HTDS description and viewing the WSDL"](#)
- [Section 2, "Retrieving and testing the WADL file"](#)
- [Section 3, "Retrieving and testing the OpenAPI description"](#)

Requirements

This exercise requires that you complete [Exercise 15, "Managing deployment"](#) before starting this exercise.

Section 1. Retrieving the HTDS description and viewing the WSDL

In this section, you retrieve the HTDS description file, which is the definition of the web service that is associated with the deployed ruleset.

1.1. View the deployed ruleset

- 1. Open Rule Execution Server console and sign in with `resAdmin` as the user name and password.
- 2. Click the **Explorer** tab and in the list of RuleApps, click the `loan-deploy` RuleApp.
- 3. Click `loan_ruleset/1.1` to open it in the Ruleset View.
The Ruleset View page opens with the details of this ruleset.
- 4. On the Ruleset View page, scroll down and 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.

[Hide HTDS Options](#)

 **Hosted Transparent Decision Services (HTDS) Options**

Location	http://localhost:9090/DecisionService 
Web service endpoint	http://localhost:9090/DecisionService 
Target namespace (SOAP only)	http://www.ibm.com/rules/decisionservice/Loan_deploy/Loan_ruleset 
Parameter target namespace	http://www.ibm.com/rules/decisionservice/Loan_deploy/Loan_ruleset/param 

1.2. View the WSDL file

- 1. On the toolbar of the `loan_ruleset/1.1` Ruleset View page, click **Retrieve HTDS Description File**.

The HTDS description is available in two forms: SOAP and REST. You can retrieve the file to view or to download.

- 2. Click **View**.

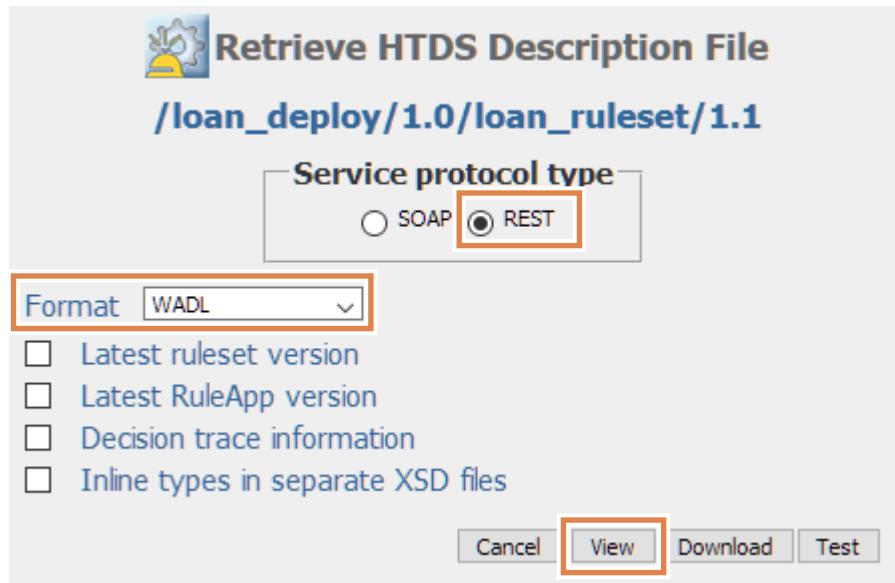
The WSDL opens in a new browser tab and you can examine the XML content.

- 3. Take some time to examine how the parameters are passed and the `soap:address` value, which points to the ruleset path on Rule Execution server.
- 4. Close the browser tab.

Section 2. Retrieving and testing the WADL file

2.1. Retrieve the WADL file

- 1. On the toolbar of the `loan_ruleset/1.1` Ruleset View page, click **Retrieve HTDS Description File**.
- 2. View the WADL description of your HTDS.
 - a. For **Service protocol type**, choose **REST**.
 - b. For the **Format**, choose **WADL**.
 - c. Click **View**.



The WADL file opens in a new browser tab.

- 3. Take time to review the XML description.
- 4. Close the browser tab.

2.2. Test the WADL file

- 1. Return to the Ruleset view to retrieve the WADL again for testing.
 - a. In the `loan_ruleset/1.1` Ruleset View, click **Retrieve HTDS Description File**.
 - b. For **Service protocol type**, choose **REST**.
 - c. For the **Format**, choose **WADL**.
 - d. Click **Test**.

The WADL file opens in the Rest Service browser tab. You can choose to send an execution request in XML or JSON format. For this exercise, you use XML.

Decision Service : /loan_deploy/1.0/loan_ruleset/1.1 REST S

2. Enter the following XML request:

```
<par:Request xmlns:par="http://www.ibm.com/rules/decisionservice/Loan_deploy/Loan_ruleset/param">
  <par:DecisionID>string</par:DecisionID>
  <par:borrower>
    <firstName>Joe</firstName>
    <lastName>Doe</lastName>
    <yearlyIncome>50000</yearlyIncome>
    <creditScore>600</creditScore>
    <latestBankruptcy>
      <date>2020-09-18T13:00:00</date>
      <chapter>3</chapter>
      <reason>Pandemic</reason>
    </latestBankruptcy>
  </par:borrower>
  <par:loan>
    <numberOfMonthlyPayments>72</numberOfMonthlyPayments>
    <startDate>2021-01-10T13:00:00</startDate>
    <amount>100000</amount>
    <loanToValue>0.7</loanToValue>
    <yearlyInterestRate>1.051732E7</yearlyInterestRate>
    <monthlyRepayment>1.051732E7</monthlyRepayment>
  </par:loan>
</par:Request>
```



You can copy and paste the code snippet from the `C:\labfiles\code\htds.txt` file into the **Execution Request** field.

3. Click the **Validate XML** icon to validate that the request is well formatted.

4. Close the **Validate OK** response.

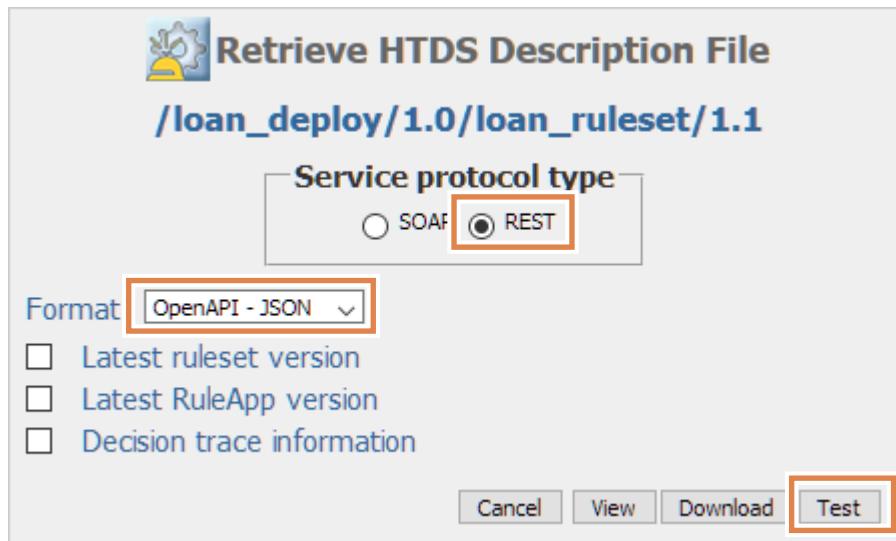
- ___ 5. Click **Execute Request**.
- ___ 6. In the Response body, scroll down to find the <approved>true</approved> **result**.
- ___ 7. Close the browser tab.

Section 3. Retrieving and testing the OpenAPI description

In this section, you test the decision service.

3.1. Retrieve retrieve the OpenAPI description of your HTDS

- 1. On the **Explorer** tab, in the Ruleset View for the `loan_ruleset/1.1` ruleset, click **Retrieve HTDS Description File**.
- 2. For **Service protocol type**, choose **REST**.
- 3. For the **Format**, choose **OpenAPI - JSON**.
- 4. Click **Test**.



3.2. Test execution

- 1. In the REST Service window, make sure that the **Execution Request** field uses JSON and notice that the sample request takes `loan` and `borrower` input.

Decision Service : /loan_deploy/1.0/loan_ruleset/1.1 REST S

Execution Request: **JSON**

```

1 '{
2   "loan": {
3     "numberOfMonthlyPayments": 3,
4     "startDate": "2006-08-19T13:27:14.000-0400",
5     "amount": 3,
6     "loanToValue": 10517320,
7     "yearlyInterestRate": 10517320,
8     "monthlyRepayment": 10517320
9   },
10   "DecisionID": "string",
11   "borrower": {
12     "firstname": "string",
13     "lastName": "string",
14     "birth": "2008-09-28T21:49:45.000-0400",
15     "SSN": {
16       "areaNumber": "string",
17       "groupCode": "string",
18       "serialNumber": "string"
19     },
20     "yearlyIncome": 3,
21     "zipCode": "string",
22     "creditScore": 3,
23     "spouse": {
24       "firstname": "string",
25       "lastName": "string",
26       "birth": "2008-09-28T21:49:45.000-0400",
27       "SSN": {
28         "areaNumber": "string",
29         "groupCode": "string",
30         "serialNumber": "string"
31       }
32     }
33   }
34 }
```

Execute Request

- __ 2. Replace the sample request with the following request:

```
{
  "loan": {
    "numberOfMonthlyPayments": 180,
    "startDate": "2021-01-19",
    "amount": 100000,
    "loanToValue": 0.9
  },
  "__DecisionID__": "test",
  "borrower": {
    "firstName": "Joe",
    "lastName": "Doe",
    "birth": "1987-09-29",
    "SSN": {
      "areaNumber": "424",
      "groupCode": "56",
      "serialNumber": "7942"
    },
    "yearlyIncome": 50000,
    "zipCode": "95372",
    "creditScore": 600,
    "latestBankruptcy": {
      "date": "2020-11-01",
      "chapter": 3,
      "reason": "Pandemic"
    }
  }
}
```



Information

You can copy and paste the code snippet from the `C:\labfiles\code\htds.txt` file into the **Execution Request** field.

- __ 3. Click **Execute Request**.
 __ 4. In the **Server Response** section, verify that the response includes these values:

```
"approved": true,
"messages": [
  "Average risk loan",
  "Congratulations! Your loan has been approved"
],
```

- __ 5. Close the **REST Service** browser tab.

End of exercise

Exercise review and wrap-up

This exercise looked at how you can use hosted transparent decision services by retrieving, examining, and testing the HTDS files.

Exercise 18. Auditing ruleset execution through Decision Warehouse

Estimated time

01:00

Overview

This exercise describes how to enable monitoring of ruleset execution and how to audit execution traces in Decision Warehouse.

Objectives

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.

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 requires that you complete the following exercises before proceeding:

- [Exercise 16, "Exploring the Rule Execution Server console"](#)



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 default, all the information is retrieved, except working memory events.

In this section, you enable monitoring properties on your ruleset. You then execute the ruleset to generate traces. After execution, you review the traces in Decision Warehouse.



Information

The `ruleset.sequential.trace.enabled` property is set when the ruleset contains tasks that use the sequential or the Fastpath execution modes.

1.1. Enabling ruleset execution monitoring

- 1. If the Rule Execution Server console is not already open, sign in as an administrator.
 - a. On the Windows **Start** menu, click the **Rule Execution Server console** shortcut.
 - b. In the **Username** and **Password** fields, enter `resAdmin` and click **Sign In**.
- 2. Open the **Explorer** tab, and in the Navigator page, expand **RuleApps > loan_deploy/1.0**, click **loan_ruleset/1.1** to open the Ruleset View page for your deployed ruleset.

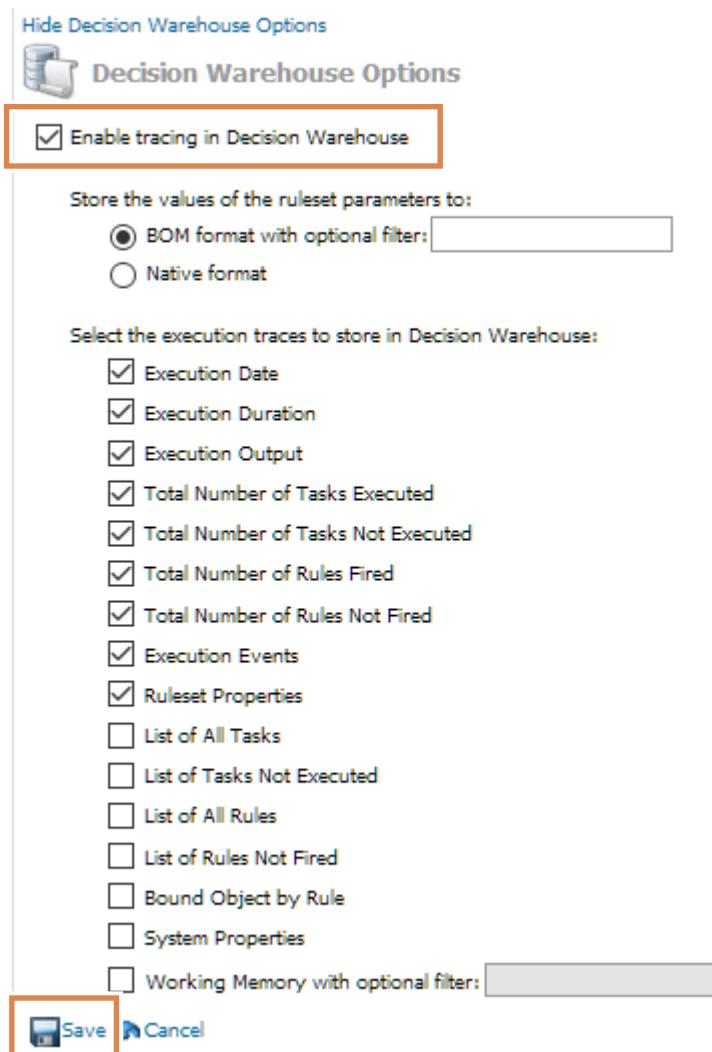
3. In the Ruleset View, click **Show Decision Warehouse Options** and note that tracing is currently disabled for this ruleset.

The screenshot shows the 'Ruleset Parameters' view. At the top, there is a table with columns: Direction, Name, Kind, and XOM Type. The table contains three rows: 'borrower' (native, loan.Borrower), 'loan' (native, loan.Loan), and 'report' (native, loan.Report). Below the table, it says 'Ruleset Parameters 1 - 3 of 3' and has navigation links 'prev 10 next 10'. Below the table, there are several buttons: 'Show Managed URIs (1)', 'Show Properties (0)', 'Show Decision Warehouse Options (tracing currently disabled)' (which is highlighted with an orange border), 'Show HTDS Options', and 'Show Archive Content'.

4. Click the **Edit** icon.

The screenshot shows the 'Decision Warehouse Options' dialog box. It has a 'Hide Decision Warehouse Options' link at the top left. In the center, there is a section with a 'Decision Warehouse Options' icon and a message: 'Tracing in Decision Warehouse is currently disabled'. To the right of the message is an 'Edit' icon, which is highlighted with an orange border.

- ___ 5. Click **Enable tracing in Decision Warehouse** and click **Save**.



You see the list of monitoring properties that can be retrieved in Decision Warehouse.

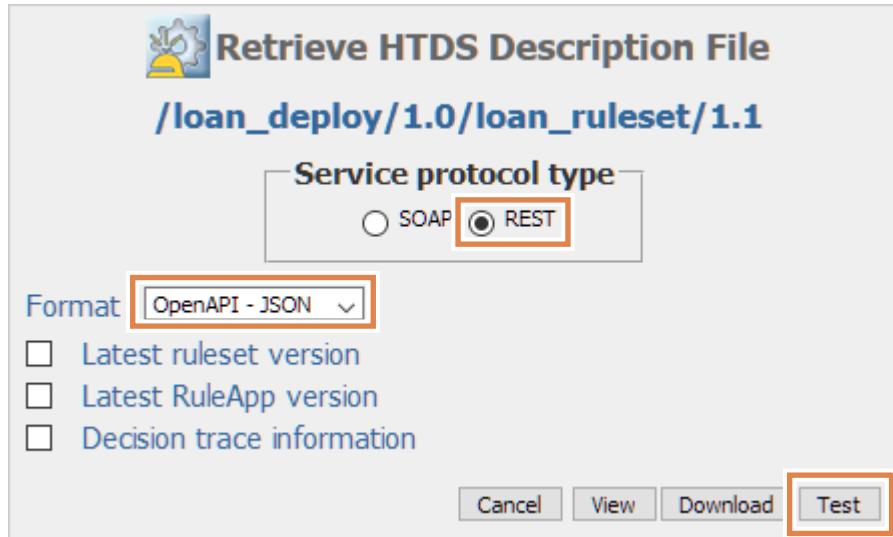
- ___ 6. Click **Show Properties** and note that the properties are related to monitoring, including monitoring.enabled.

<input type="checkbox"/> Select All	Name	Value
<input type="checkbox"/>	monitoring.enabled	true
<input type="checkbox"/>	monitoring.filters	INFO_TOTAL_RULES_FIRED=true,INFO_EXECUTION_DATE=true,INFO_TOTAL_TA
<input type="checkbox"/>	ruleset.bom.enabled	true
<input type="checkbox"/>	rulesetSEQUENTIAL.trace.enabled	true

properties 1 - 4 of 4

1.2. Generate execution traces

- ___ 1. In the Ruleset View, click **Retrieve HTDS Description File**.
- ___ 2. Select the following options, and click **Test**:
 - Service protocol type: **REST**
 - Format: **OpenAPI-JSON**



- ___ 3. On the **REST Service** page, replace the template code with the following request code:

```
{
  "loan": {
    "numberOfMonthlyPayments": 180,
    "startDate": "2021-01-19",
    "amount": 100000,
    "loanToValue": 0.9
  },
  "__DecisionID__": "test",
  "borrower": {
    "firstName": "Joe",
    "lastName": "Doe",
    "birth": "1987-09-29",
    "SSN": {
      "areaNumber": "424",
      "groupCode": "56",
      "serialNumber": "7942"
    },
    "yearlyIncome": 50000,
    "zipCode": "95372",
    "creditScore": 600,
    "latestBankruptcy": {
      "date": "2021-11-01",
      "chapter": 3,
      "reason": "Pandemic"
    }
  }
}
```



Information

You can copy and paste the code snippet from the `C:\labfiles\code\htds.txt` file into the **Execution Request** field.

-
- ___ 4. Click **Execute Request**.
 - ___ 5. Verify that the request was successful by confirming that the Server Response shows that the loan was approved.
 - ___ 6. Click **Execution Request** 2 or 3 more times.
 - ___ 7. Keep the **REST Service** tab open, but go back to the **Rule Execution Server** console tab. You can return to the **REST Service** tab again later to retest the decision service.

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.

2.1. View decision traces

- 1. Return to the Rule Execution Server console window, and click the **Decision Warehouse** tab.
- 2. On the Search Decisions page, leave all fields empty and click **Search** at the bottom of the **Search Decisions** page.
- 3. Scroll down the page to see the results.
- 4. In the **Decision Trace** column, click the **View Decision details** link for one of the recent decisions.

Decision ID	Date	Ruleset Version	Number of rules fired	Decision Trace	Process
765d94f0-653e-4dd7-bf5e-71d8c1245ba80	2020-11-29 20:44:50	/loan_deploy /1.0/loan_ruleset/1.1	9	View Decision details	16
84c17d71-4a83-4fd7-8ef3-73b89b65de8a0	2020-11-29 20:44:48	/loan_deploy /1.0/loan_ruleset/1.1	9	View Decision details	15
5ec23d38-0fae-4b1e-a493-b4cf8ee612bd0	2020-11-29 20:44:46	/loan_deploy /1.0/loan_ruleset/1.1	9	View Decision details	0
deaf6f3f-e7ab-4106-a21b-de941d515c4b0	2020-11-29 20:44:38	/loan_deploy /1.0/loan_ruleset/1.1	9	View Decision details	375

The Decision Trace page opens, and shows the details of the execution.

- ___ 5. Look at the fields in the **Execution Details** section.

Execution Details	
Decision ID:	875abae7-b116-4a64-bc48-b5b5f77a50510
Date:	2020-10-29 14:11:36
Executed ruleset path:	/loan_deploy/1.0/loan_ruleset/1.1
Engine type:	de
Processing Time (ms)	16
Number of rules fired	9
Number of tasks executed	6

Decision Trace

- Ruleflow Tasks (1)
 - + loanvalidation (5)

Input Parameters

```

loan :
<?xml version="1.1" encoding="UTF-8"?>
<loan.Lean xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance">
<startDate>
<date>2020-01-19T12:27:14.000-05:00</date>
</startDate>
<numberOfMonthlyPayments>
<int>70</int>
</numberOfMonthlyPayments>
<amount>
<int>100000</int>
</amount>
<loanToValue>
<double>0.7</double>
</loanToValue>
<duration>
<int>6</int>
</duration>

```

Questions

Do you recognize the information in the Decision Trace window and which monitoring properties this information is related to?

- ___ 6. Close the Decision Trace window.

2.2. Adjust monitoring options

Next, return to Decision Warehouse to change the monitoring options and regenerate decision traces.

- ___ 1. On the **Explorer** tab, open the Ruleset View for `loan_ruleset/1.1` and change the Decision Warehouse Options to include only **Execution Date**.
- ___ a. On the **Explorer** tab, in the **Navigator** pane, click **RuleApps > loan_deploy/1.0 > loan_ruleset/1.1**.

- ___ b. In the **Decision Warehouse Options**, click **Edit**.
 - ___ c. Clear all the options except **Execution Date**.
 - ___ d. Click **Save**.
- ___ 2. Click **Show Properties** and notice the **monitoring.filters** value now shows only one filter set to true.
- ___ 3. Identify the connection between the **Enable tracing in Decision Warehouse** option and the **monitoring.enabled** ruleset property.
- ___ 4. Generate execution traces.
- ___ a. Return to the REST Service window to send another execution request.
 - ___ b. Optional. Change the **numberOfMonthlyPayments** to **12** to produce a “not approved” result.
 - ___ c. Click **Execute Request** 2 or 3 times and verify that the server responded.
- ___ 5. View the Decision Warehouse results.
- ___ a. On the Search Decisions page, leave all fields empty and click **Search** at the bottom of the **Search Decisions** page.
 - ___ b. Scroll down the page to see the results and click **View Decision details** for the most recent decision.



Questions

What differences do you see?

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. Filtering the monitoring options

- ___ 1. Disable tracing for the `loan_ruleset/1.1` ruleset.
 - ___ a. On the **Explorer** tab, in the **Navigator** pane, click **RuleApps > loan_deploy/1.0 > loan_ruleset/1.1**.
 - ___ b. In the **Decision Warehouse Options** section, click **Edit**.
 - ___ 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**.

- ___ 2. Look in the **properties** section and verify that the `monitoring.enabled` ruleset property is now set to: `false`

<input type="checkbox"/> Select All	Name	Value
<input type="checkbox"/>	<code>monitoring.enabled</code>	<code>false</code>

- ___ 3. Enable decision tracing again.
 - ___ a. In the **Decision Warehouse Options** section, click **Edit**.
 - ___ b. Select **Enable tracing in Decision Warehouse** and click **Save**.
 - ___ c. Look in the properties section and verify that `monitoring.enabled` is now set to true and `monitoring.filters` now lists specific execution trace options.

Hide Properties		
	4 properties	
<input type="checkbox"/> Select All	Name	Value
<input type="checkbox"/>	<code>monitoring.enabled</code>	<code>true</code>
<input type="checkbox"/>	<code>monitoring.filters</code>	<code>INFO_TOTAL_RULES_FIRED=true,INFO_EXECUTION_DATE=true,INFO_TOTAL_TASKS_EXECUTED=true</code>
<input type="checkbox"/>	<code>ruleset.bom.enabled</code>	<code>true</code>
<input type="checkbox"/>	<code>rulesetSEQUENTIAL.trace.enabled</code>	<code>true</code>

3.2. Using filters on trace data

To reduce the information that is stored in an execution trace in Decision Warehouse:

- 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.

1. In the **Decision Warehouse Options** section, click **Edit**.
2. In the **Select the execution traces to store in the Decision Warehouse** list, select *all* the remaining execution traces in the list, and click **Save**.
3. 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, which results in storing all execution data. To avoid storing unnecessary data, make sure you apply filters.

3.3. Removing BOM serialization

When the `ruleset.bom.enabled` property is set to `true`, it converts the list of objects in the parameters into a memory buffer by using BOM serialization.

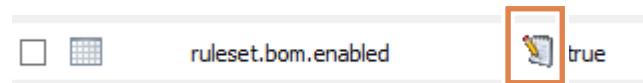


Important

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. In the **Properties** section, click the **Edit** icon for the `ruleset.bom.enabled` ruleset property.



2. Set `ruleset.bom.enabled` to `false` and click the **Save** icon.



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 **Decision Warehouse Options**, click **Edit**.
4. Notice that **Native format** is selected as 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. Click **Cancel**.
- ___ 6. Generate execution traces in the REST Service window.
 - ___ a. Return to the REST Service window.
 - ___ b. Optional. Change the **numberOfMonthlyPayments** back to: 70
 - ___ c. Click **Execute Request** and verify that the server responded.
- ___ 7. In Rule Execution Server console, open the **Decision Warehouse** tab, and click **Search** to get the latest traces.
- ___ 8. Click **View Decision details** for your latest trace and notice the format for the input and output parameters.

Input Parameters	
loan = Loan: Amount 100,000 Starting Jan 19, 2020 Duration 70 month(s) Loan to value 70% borrower = Borrower: First name Joe Last name Doe born Sep 28, 1987 SSN 424-56-7942 - Zipcode 95372 - Yearly income 100,000 - Credit Score 600 - Bankruptcy on Sep 18, 2014 for Summer loss filed under chapter 3	
Output Parameters	
loan = Loan: Amount 100,000 Starting Jan 19, 2020 Duration 70 month(s) Loan to value 70% report = Report: Valid data true Approved true score 750 - Grade B - Insurance required true Insurance rate 2% - Message Low risk loan Congratulations! Your loan has been approved	
Execution output	
Borrower: First name Joe Last name Doe born Sep 28, 1987 SSN 424-56-7942 - Zipcode 95372 - Yearly income 100,000 - Credit Score 600 - Bankruptcy on Sep 18, 2014 for Summer loss filed under chapter 3 Loan: Amount 100,000 Starting Jan 19, 2020 Duration 70 month(s) Loan to value 70% Report: Valid data true Approved true score 750 - Grade B - Insurance required true Insurance rate 2% - Message Low risk loan Congratulations! Your loan has been approved	

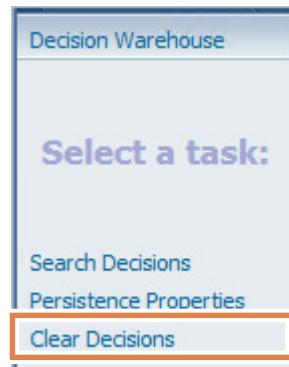
- ___ 9. Click **View Decision details** for the previous trace and take a moment to compare the formats of the input and output parameters between the two traces.
- ___ 10. Close the Decision Trace windows.

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.

4.1. Clearing traces

- 1. In the Rule Execution Server console, on the **Decision Warehouse** tab, click **Search Decisions > Search** and look at the list of decisions to see RuleApp version numbers and dates.
- 2. In the “Select a task” pane, click **Clear Decisions**.



- 3. 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.

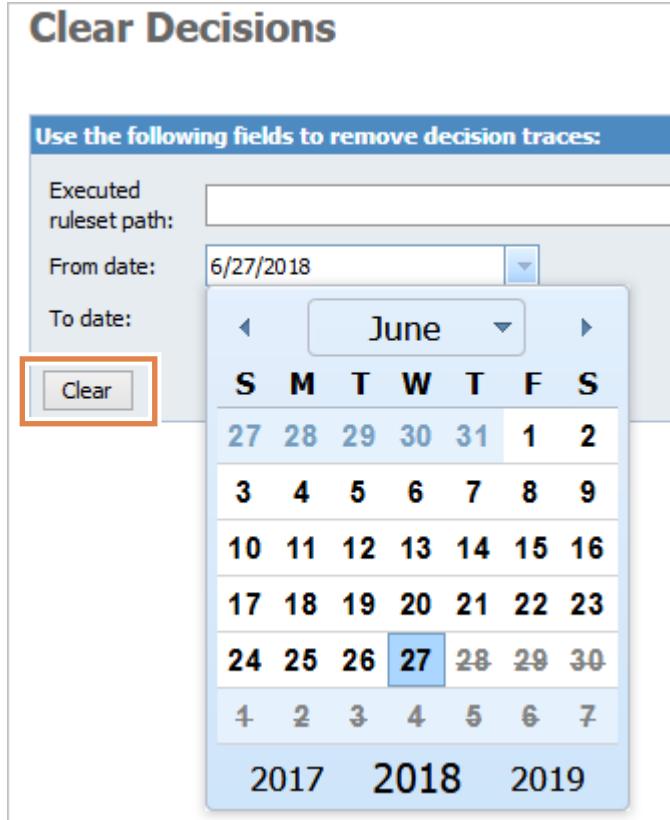


Information

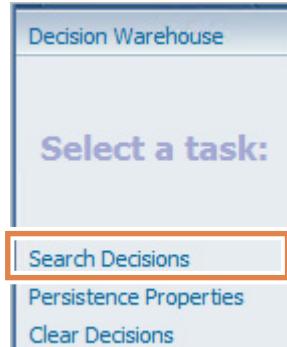
You can leave this field empty to erase all traces. You can also specify a partial or complete ruleset path to limit which traces are erased.

- b. In the **From Date** field, click the arrow beside the field to open the calendar. You can specify a date or leave the field blank.
- c. In the **To Date** field, leave the field blank.

- ___ 4. Click **Clear** to delete the specified traces.



- ___ 5. When prompted to confirm the deletion, click **Confirm**.
___ 6. 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.



IBM Training



© Copyright International Business Machines Corporation 2021.