

Course Exercises Guide

# Managing Decisions in IBM Operational Decision Manager V8.9

Course code WB401 / ZB401 ERC 1.1



## October 2017 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](http://www.ibm.com/legal/copytrade.shtml).

**© Copyright International Business Machines Corporation 2017.**

**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

<b>Trademarks .....</b>	<b>viii</b>
<b>Exercises description .....</b>	<b>ix</b>
<b>Exercise 1. Operational Decision Manager in action .....</b>	<b>1-1</b>
General exercise information .....	3
Section 1. Running the Miniloan web application .....	1-5
1.1. Preparing your environment .....	1-5
1.2. Running the application .....	1-6
Section 2. Managing business rules in Decision Center .....	1-9
2.1. Editing rules in Decision Center Business console .....	1-9
2.2. Reviewing the rule history .....	1-16
2.3. Changing the credit score requirements .....	1-18
Section 3. Validating rule changes .....	1-22
Section 4. Deploying updated rules from Decision Center to Rule Execution Server .....	1-24
4.1. Deploying the decision service .....	1-24
Section 5. Monitoring the deployed rules in Rule Execution Server .....	1-27
Section 6. Viewing the effects in the Miniloan application .....	1-31
Section 7. Exploring the development environments in Rule Designer .....	1-32
7.1. Opening Rule Designer .....	1-32
7.2. Taking a quick tour of Rule Designer .....	1-34
Section 8. Shutting down the modules .....	1-36
<b>Exercise 2. Building the model on paper .....</b>	<b>2-1</b>
Section 1. Discussion: Drawing a class diagram .....	2-3
Draw your class diagram here: .....	5
Draw your class diagram here, continued: .....	6
Section 2. Building the model on paper: Completing the class diagram .....	2-7
<b>Exercise 3. Implementing the model .....</b>	<b>3-1</b>
Section 1. Preparing the rule authoring environment .....	3-2
1.1. Creating a decision service .....	3-2
1.2. Building the BOM .....	3-5
1.3. Verbalizing the classes .....	3-12
1.4. Defining ruleset variables .....	3-15
Section 2. Writing the rules to test the BOM .....	3-17
2.1. Writing new rules .....	3-17
2.2. Creating the minimum and maximum age rules .....	3-19
<b>Exercise 4. Setting up a decision service .....</b>	<b>4-1</b>
Section 1. Creating a standard rule project .....	4-2
1.1. Opening a new workspace in Rule Designer and viewing the Decision Service Map .....	4-2
1.2. Creating a standard rule project with a BOM .....	4-3
Section 2. Importing the XOM and creating the BOM .....	4-6
2.1. Importing the XOM .....	4-6
2.2. Creating the BOM .....	4-8
Section 3. Creating the main rule project for the decision service .....	4-13
Section 4. Creating and defining the decision operation .....	4-15
4.1. Creating the decision operation .....	4-15
4.2. Creating ruleset variables .....	4-16

4.3. Binding the variables to ruleset parameters .....	4-18
Section 5. Orchestrating execution .....	4-20
5.1. Creating rule packages .....	4-20
5.2. Creating a ruleflow .....	4-21
5.3. Creating the ruleflow diagram .....	4-22
5.4. Adding transition conditions .....	4-24
Section 6. Publishing and synchronizing the rule project .....	4-27
6.1. Publishing the rule project to Decision Center .....	4-27
6.2. Viewing the published project in the Decision Center Business console .....	4-30
6.3. Adding a rule .....	4-33
6.4. Updating the rules in Rule Designer .....	4-35
6.5. Deleting the project from Decision Center .....	4-37
<b>Exercise 5. Working with the BOM .....</b>	<b>5-1</b>
Section 1. Reviewing the vocabulary .....	5-2
1.1. Switching to a new workspace and importing the start project .....	5-2
1.2. Reviewing the vocabulary .....	5-3
1.3. Creating a class .....	5-4
Section 2. Setting categories .....	5-7
2.1. Adding a category to the rule project .....	5-7
2.2. Specifying a category for a business element .....	5-8
2.3. Testing the categories .....	5-10
Section 3. Refactoring the vocabulary .....	5-12
3.1. Viewing how vocabulary changes are refactored .....	5-12
Section 4. Creating a domain .....	5-15
4.1. Creating an enumerated domain for marital status .....	5-15
4.2. Testing the new marital status domain in a rule .....	5-18
<b>Exercise 6. Exploring the Decision Center Business console .....</b>	<b>6-1</b>
Section 1. Working with rule projects and assigning tasks .....	6-3
1.1. Logging in to the Business console as Paul .....	6-3
1.2. Viewing the Home tab .....	6-4
1.3. Viewing the Library page .....	6-6
1.4. Working with a decision service .....	6-7
1.5. Posting a comment in the Stream .....	6-12
Section 2. Creating and editing rules .....	6-14
2.1. Logging in as Bea and viewing recent activity .....	6-14
2.2. Modifying an action rule .....	6-14
2.3. Creating an action rule .....	6-16
2.4. Editing a decision table .....	6-18
2.5. Posting a reply to Paul and logging out .....	6-20
Section 3. Working with the Business console management features .....	6-22
3.1. Logging in as Paul and reviewing the changes to my-validation-rules-service .....	6-22
3.2. Working with snapshots .....	6-23
3.3. Viewing the timeline .....	6-27
<b>Exercise 7. Understanding the case study .....</b>	<b>7-1</b>
Section 1. Reading the case study .....	7-3
<b>Exercise 8. Discovering rules .....</b>	<b>8-1</b>
Section 1. Identifying decisions in use cases .....	8-3
Section 2. Creating a decision-points table .....	8-4
Section 3. Formalizing raw decisions as rule statements .....	8-7
<b>Exercise 9. Analyzing rules .....</b>	<b>9-1</b>
Section 1. Making the rules atomic .....	9-2

Section 2. Removing redundancies, overlaps, and inconsistencies .....	9-3
Section 3. Identifying dependencies .....	9-4
<b>Exercise 10. Working with conditions in rules .....</b>	<b>10-1</b>
Section 1. Before starting the exercise .....	10-3
Section 2. BAL condition constructs .....	10-5
2.1. Looking up BAL constructs in the product documentation .....	10-7
Section 3. Identifying conditions .....	10-11
Section 4. Writing comparison conditions .....	10-13
4.1. Creating the “Car age” rule .....	10-13
4.2. Writing the condition statement .....	10-16
4.3. Completing the rule with the action statement .....	10-20
Section 5. Creating a compound condition .....	10-22
5.1. Creating the “Policy type” rule .....	10-22
5.2. Writing the condition statement that tests whether the car is financed or leased .....	10-24
5.3. Completing the rule with the action statement .....	10-26
Section 6. Creating rules with constants and ruleset variables .....	10-28
6.1. Creating the “Accessories value” rule with a constant .....	10-28
6.2. Defining the rule .....	10-29
6.3. Reworking the “Accessories value” rule with a ruleset variable .....	10-32
Section 7. Updating ruleset variables .....	10-33
7.1. Viewing variable sets in the Decision Artifacts tab .....	10-33
7.2. Editing the maxRatio variable .....	10-34
Section 8. Creating a count condition .....	10-35
8.1. Creating the “Number of drivers” rule .....	10-35
8.2. Defining the conditions .....	10-35
8.3. Completing the action statement .....	10-37
Section 9. Creating a count condition with a “where” clause .....	10-39
9.1. Creating the “Number of at-fault accidents” rule .....	10-39
Section 10. Advanced practice .....	10-44
Section 11. Solutions .....	10-46
11.1. Solution for <a href="#">“Creating the “Policy type” rule”</a> .....	10-46
11.2. Solution for <a href="#">“Creating the “Accessories value” rule with a constant”</a> .....	10-46
11.3. Solution for <a href="#">“Creating the “Number of at-fault accidents” rule”</a> .....	10-48
11.4. Solution for “Senior secondary drivers” rule .....	10-48
11.5. Steps for creating the “Senior secondary drivers” rule .....	10-48
<b>Exercise 11. Working with definitions in rules .....</b>	<b>11-1</b>
Section 1. Identifying variables for definitions .....	11-2
Section 2. Using the “from <object>” construct .....	11-4
2.1. Creating the “Collision cover in [min, max]” rule .....	11-4
2.2. Defining a variable .....	11-6
2.3. Defining the condition and action statements .....	11-9
Section 3. Using variables to improve readability .....	11-11
3.1. Rewriting the “Collision cover in [min, max]” rule to improve readability .....	11-11
Section 4. Accessing objects in a collection .....	11-13
4.1. Creating the “Driver age” rule without preconditions .....	11-13
Section 5. Using a precondition .....	11-18
5.1. Rewriting the “Driver age” rule with a precondition .....	11-18
Section 6. Defining a collection variable .....	11-22
6.1. Rewriting the “Driver age” rule with a collection variable .....	11-22
6.2. Reviewing the “Driver age” rules .....	11-24
Section 7. Solutions .....	11-26
7.1. Solution for <a href="#">“Creating the “Collision cover in [min, max]” rule”</a> .....	11-26
7.2. Solution for <a href="#">“Creating the “Driver age” rule without preconditions”</a> .....	11-26
7.3. Solution for <a href="#">“Rewriting the “Driver age” rule with a precondition”</a> .....	11-27

7.4. Solution for "Rewriting the “Driver age” rule with a collection variable" .....	11-27
<b>Exercise 12. Writing complete rules .....</b>	<b>12-1</b>
Section 1. Writing actions that use expressions .....	12-3
1.1. Creating the “Total number of accidents” rule .....	12-3
1.2. Creating the “Number of license withdrawals” rule .....	12-5
Section 2. Writing a computation rule that always executes .....	12-6
2.1. Creating the “Calculate collision global price tax included” rule .....	12-6
Section 3. Writing rules that perform actions on each object in a collection .....	12-8
3.1. Creating the “Calculate collision global premium before tax” rule .....	12-8
3.2. Creating the “Risky co-drivers” rule .....	12-10
Section 4. Solutions .....	12-13
4.1. Solution for "Creating the “Total number of accidents” rule" .....	12-13
4.2. Solution for "Creating the “Number of license withdrawals” rule" .....	12-13
4.3. Solution for "Creating the “Calculate collision global price tax included” rule" .....	12-13
4.4. Solution for "Creating the “Calculate collision global premium before tax” rule" .....	12-14
4.5. Solution for "Creating the “Risky co-drivers” rule" .....	12-14
<b>Exercise 13. Authoring decision tables .....</b>	<b>13-1</b>
Section 1. Creating a decision table .....	13-4
1.1. Creating the “Basic coverage premium” decision table .....	13-5
1.2. Defining the condition column .....	13-6
1.3. Defining the action columns .....	13-8
1.4. Completing the rows of the table .....	13-10
1.5. Handling empty cells .....	13-13
Section 2. Editing conditions in a decision table .....	13-16
Creating the “Senior primary drivers” decision table .....	17
2.1. Creating the rule folder and starting the decision table .....	13-17
2.2. Defining the columns .....	13-19
2.3. Completing the table .....	13-22
Section 3. Solutions .....	13-28
Section 2, "Editing conditions in a decision table" .....	28
3.1. Solution for "Creating the “Senior primary drivers” decision table" .....	13-28
<b>Exercise 14. Authoring rules: Putting it all together .....</b>	<b>14-1</b>
Section 1. Implementing the mandatory constraints about the maximum insurable value of vehicles .....	14-4
Section 2. Implementing maximum collision liability amounts .....	14-7
Section 3. Implementing a decision table to infer the risk for young primary drivers .....	14-11
Section 4. Solutions .....	14-15
<b>Exercise 15. Running tests and simulations in the Business console .....</b>	<b>15-1</b>
Section 1. Preparing the lab environment .....	15-2
1.1. Importing the start project and publishing to Decision Center .....	15-4
Section 2. Running tests on the carinsurance-rules decision service .....	15-5
2.1. Creating a scenario file .....	15-5
2.2. Creating and running a test suite .....	15-9
2.3. Viewing the test results .....	15-10
Section 3. Running a simulation on the car insurance rules .....	15-13
3.1. Defining metrics .....	15-13
3.2. Defining key performance indicators (KPIs) .....	15-15
3.3. Defining the simulation data .....	15-16
3.4. Defining the report format .....	15-17
3.5. Configuring and running the simulation .....	15-19
<b>Exercise 16. Working with management features in Decision Center .....</b>	<b>16-1</b>
Section 1. Preparing the lab environment .....	16-2

Section 2. Importing a decision service project by using the Enterprise console .....	16-4
Section 3. Working with decision service branches in the Business console .....	16-7
Section 4. Exporting and erasing a decision service in the Enterprise console .....	16-14
<b>Exercise 17. Managing user access in Decision Center .....</b>	<b>17-1</b>
Section 1. Viewing permissions for default groups .....	17-3
Section 2. Configuring security on WebSphere Liberty Profile .....	17-4
2.1. Exploring the server.xml file .....	17-4
2.2. Creating custom users through the basic user registry .....	17-7
2.3. Signing in to Decision Center with your custom user .....	17-9
Section 3. Setting up new users on an LDAP server .....	17-10
3.1. Setting up an LDAP connection in Apache Directory Studio .....	17-10
Section 4. Adding users and groups in LDAP .....	17-15
4.1. Creating an organizational unit .....	17-15
4.2. Creating users .....	17-16
4.3. Adding groups .....	17-19
Section 5. Delegating authorization to Decision Center .....	17-24
5.1. Installing the prepared server.xml file .....	17-24
5.2. Viewing the LDAP configuration information .....	17-24
5.3. Checking Decision Center authentication .....	17-26
Section 6. Administering groups and users in the Business console .....	17-28
6.1. Adding an LDAP connection to the Business console .....	17-28
6.2. Importing LDAP groups and users .....	17-29
6.3. Assigning permissions to groups .....	17-31
6.4. Verifying the role assignments and permissions .....	17-32
<b>Exercise 18. Working with the decision governance framework .....</b>	<b>18-1</b>
Section 1. Preparing the lab environment .....	18-3
1.1. Resetting the database .....	18-3
1.2. Importing the carinsurance-rules decision service .....	18-4
1.3. Publishing from Rule Designer to Decision Center .....	18-7
Section 2. Working with the decision service .....	18-10
2.1. Logging in to the Business console as Paul .....	18-10
2.2. Exploring the carinsurance decision service and creating a release .....	18-11
Section 3. Creating a change activity .....	18-15
3.1. Creating a change activity for the Driver age rule .....	18-15
Section 4. Creating a validation activity .....	18-18
Section 5. Updating the rule in the change activity .....	18-20
5.1. Logging in as Bea and finding change activity work .....	18-20
5.2. Updating the rule .....	18-21
5.3. Changing the working status of the change activity .....	18-23
Section 6. Reviewing and approving the change activity .....	18-24
6.1. Reviewing the change activity .....	18-24
6.2. Approving the change activity .....	18-24
Section 7. Working with the validation activity .....	18-26
7.1. Creating and running a test suite .....	18-26
7.2. Completing the validation activity .....	18-30
Section 8. Approving the validation activity, completing the release, and deploying the release .....	18-31
8.1. Viewing the results of the test plan and completing the validation activity .....	18-31
8.2. Completing the release .....	18-32
8.3. Deploying the Training Release .....	18-33

---

# 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:

Bluemix®

ILOG®

Orchestrate®

SPSS®

Express®

Insight™

Rational®

WebSphere®

IBM Bluemix™

Notes®

Redbooks®

z/OS®

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

This course includes the following exercises:

- [Exercise 1, "Operational Decision Manager in action"](#)
- [Exercise 2, "Building the model on paper"](#)
- [Exercise 3, "Implementing the model"](#)
- [Exercise 4, "Setting up a decision service"](#)
- [Exercise 5, "Working with the BOM"](#)
- [Exercise 6, "Exploring the Decision Center Business console"](#)
- [Exercise 7, "Understanding the case study"](#)
- [Exercise 8, "Discovering rules"](#)
- [Exercise 9, "Analyzing rules"](#)
- [Exercise 10, "Working with conditions in rules"](#)
- [Exercise 11, "Working with definitions in rules"](#)
- [Exercise 12, "Writing complete rules"](#)
- [Exercise 13, "Authoring decision tables"](#)
- [Exercise 14, "Authoring rules: Putting it all together"](#)
- [Exercise 15, "Running tests and simulations in the Business console"](#)
- [Exercise 16, "Working with management features in Decision Center"](#)
- [Exercise 17, "Managing user access in Decision Center"](#)
- [Exercise 18, "Working with the decision governance framework"](#)

## Exercise structure

The exercises in this course can be categorized into the following 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 Decision Center.

## *Business rules and decision services*

- [Exercise 2, "Building the model on paper"](#)
- [Exercise 3, "Implementing the model"](#)
- [Exercise 4, "Setting up a decision service"](#)
- [Exercise 5, "Working with the BOM"](#)

These exercises introduce you to business rules and decision services by focusing on rule modeling, rule model implementation, decision service structure, and the BOM.

## *Decision Center*

- [Exercise 6, "Exploring the Decision Center Business console"](#)

In this exercise, you explore the rule authoring environment of the Decision Center Business console.

## *Rule authoring*

- [Exercise 7, "Understanding the case study"](#)
- [Exercise 8, "Discovering rules"](#)
- [Exercise 9, "Analyzing rules"](#)
- [Exercise 10, "Working with conditions in rules"](#)
- [Exercise 11, "Working with definitions in rules"](#)
- [Exercise 12, "Writing complete rules"](#)
- [Exercise 13, "Authoring decision tables"](#)
- [Exercise 14, "Authoring rules: Putting it all together"](#)

These exercises focus on rule authoring concepts and practices, including rule discovery and analysis, rule conditions, rule definitions and variables, rule actions, and decision tables.

## *Decision service management and governance*

- [Exercise 15, "Running tests and simulations in the Business console"](#)
- [Exercise 16, "Working with management features in Decision Center"](#)
- [Exercise 17, "Managing user access in Decision Center"](#)
- [Exercise 18, "Working with the decision governance framework"](#)

These exercises focus on decision service management and governance topics, such as testing and simulation to validate rule changes, Enterprise console administration features, managing users in the Business console, and decision governance.

## **General exercise notes**

Most of the exercises are independent and do not require you to complete a previous exercise.

In the exercise instructions, you can check off the line before each step as you complete it to track your progress.

# Exercise 1. Operational Decision Manager in action

## Estimated time

01:30

## Overview

This exercise provides an overview of the Operational Decision Manager modules. You learn how the Operational Decision Manager modules work together to provide a comprehensive Business Rule Management System (BRMS) across the business and development environments. Both IT and business tasks are included in this exercise to demonstrate the collaboration that occurs between the technical and business teams.

## 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 tools that apply to your role in your organization

## Introduction

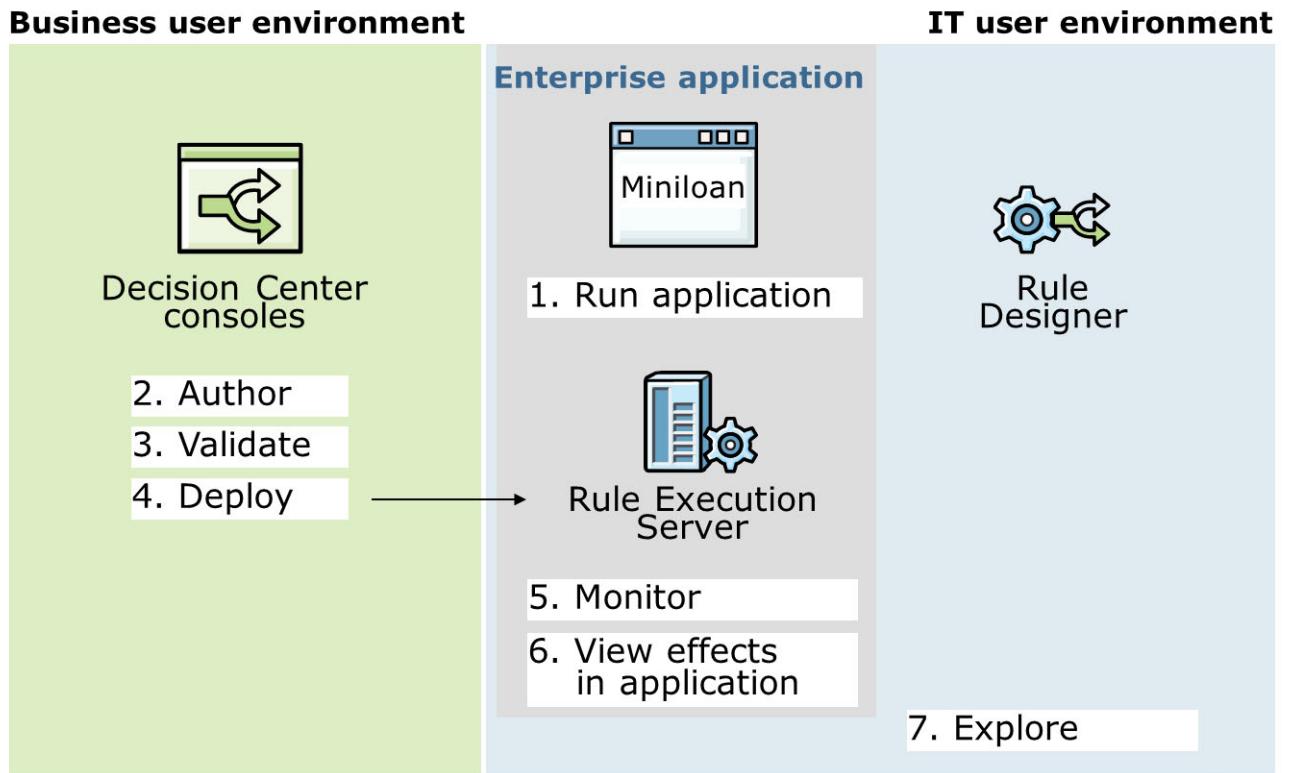
The exercise is based on a fictional financial institution, Miniloan, which provides online quotations for loan requests.

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

- [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, and 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 specific requirements.

## User accounts

**Table 1:**

Type	User ID	Password
Access to VMware image	administrator	passw0rd
Windows Server 2012 R2	administrator	passw0rd
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
Custom user	myUser	myUser
LDAP management user	Jane	Jane
LDAP development user	John	John

## General exercise information



### Cloud

This exercise uses some features that are not supported in IBM ODM on Cloud.

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

For more information about running the Miniloan web application, see the IBM ODM on Cloud documentation.



### Important

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

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

The default directory for `<InstallDir>` is `C:\IBM\ODM89`. If you are not using the provided lab environment for this course, make sure that you know the location of `<InstallDir>`.

Additional lab support files are stored in the `C:\labfiles` directory. In some of the exercises, this directory is referred to as `<labfilesDir>`.

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

---



**Stop**

### ***Course updates and corrections***



A course corrections document might be available for this course.

If you are taking the class with an instructor, the instructor can provide this document to you.

If you are taking the course in a self-paced environment, the course corrections document is provided with the other manuals.

To check whether a course corrections document exists for this course:

1. Go to the following URL: <http://www.ibm.biz/CloudEduCourses>
  2. Find the product category for the course and click the link.
  3. Find your course in the list and click the link.
  4. Click the **Attachments** tab to see whether a course corrections document exists with updated instructions.
  5. To save the file to your computer, click the document link and follow the prompts.
-

## 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. Preparing your environment

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

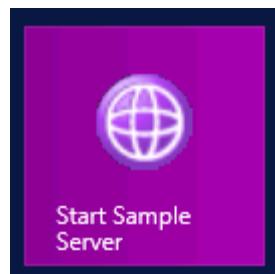


#### Cloud

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

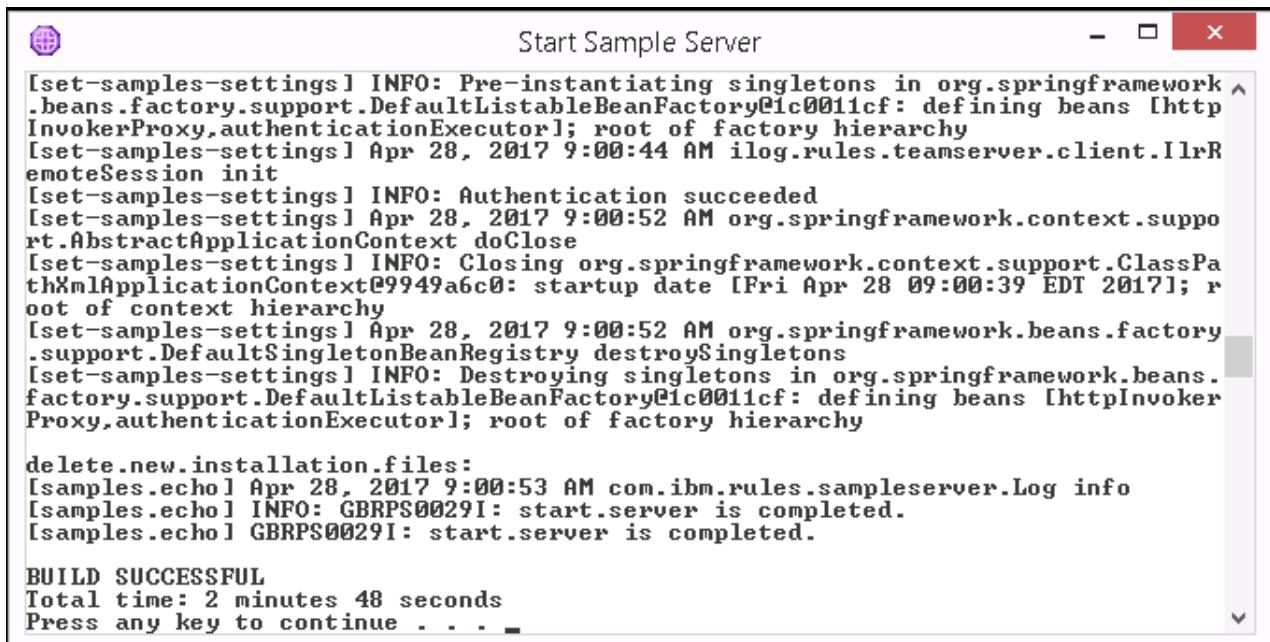
In the “First steps” tutorial, you work with the `MiniloanService` decision service and install the `miniloan-webapp` web application.

- \_\_\_ 1. Start the sample server.
  - \_\_\_ a. Click 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

[set-samples-settings] INFO: Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@1c0011cf: defining beans [httpInvokerProxy,authenticationExecutor]; root of factory hierarchy
[set-samples-settings] Apr 28, 2017 9:00:44 AM ilog.rules.teamserver.client.IlrRemoteSession init
[set-samples-settings] INFO: Authentication succeeded
[set-samples-settings] Apr 28, 2017 9:00:52 AM org.springframework.context.support.AbstractApplicationContext doClose
[set-samples-settings] INFO: Closing org.springframework.context.support.ClassPathXmlApplicationContext@9949a6c0: startup date [Fri Apr 28 09:00:39 EDT 2017]; root of context hierarchy
[set-samples-settings] Apr 28, 2017 9:00:52 AM org.springframework.beans.factory.support.DefaultSingletonBeanRegistry destroySingletons
[set-samples-settings] INFO: Destroying singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@1c0011cf: defining beans [httpInvokerProxy,authenticationExecutor]; root of factory hierarchy

delete.new.installation.files:
[samples.echo] Apr 28, 2017 9:00:53 AM com.ibm.rules.sampleserver.Log info
[samples.echo] INFO: GBRPS0029I: start.server is completed.
[samples.echo] GBRPS0029I: start.server is completed.

BUILD SUCCESSFUL
Total time: 2 minutes 48 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.

# Sample

MiniLoan

---

<b>Borrower</b>	<b>Loan</b>
Name: Joe	Amount: 500000
Yearly Inc... 80000	Duration: 240
Credit Sco... 600	Yearly Int... 0.05

Use rules

**✓ Validate Loan**

---

### Scenario

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

Is Joe eligible for this loan?

- 
2. Click **Validate Loan**.

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

<b>Borrower</b>		<b>Loan</b>	
<b>Name:</b>	Joe	<b>Amount:</b>	500000
<b>Yearly Inc...:</b>	80000	<b>Duration:</b>	240
<b>Credit Sco...:</b>	600	<b>Yearly Int...:</b>	0.05

Use rules

✓ Validate Loan

Rejected

Your loan is rejected  
The debt-to-income ratio is too big.

### Scenario

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

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

As you can see, decisions that are made by an application affect both your business and the lives of ordinary people. For that reason, it is important that experts in business policy, not just the IT team, can see and maintain the rules that are implemented.

Instead of hardcoding the rules into the application, and requesting technical developers to make rule changes in the code, Operational Decision Manager separates the decision logic from the code. It empowers business users to access and manage the rules through Decision Center. Rules are stored in the Decision Center repository, and can be shared among various users through permission and locking mechanisms.

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

## Section 2. Managing business rules in Decision Center

---

### **Scenario**

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

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

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

---

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

### 2.1. Editing rules in Decision Center Business console

- 1. Open the Business console from the **Start** menu, by clicking 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.



## Cloud

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

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

- 
- 2. 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**.

- 
- 3. Sign in as a business user with the following values for the **Username** and **Password** fields.
- **Username:** rtsUser1
  - **Password:** rtsUser1



### Note

These values are case-sensitive.

---



- \_\_\_ 4. Click **Log in**.



#### Note

If you see a message to store your password, you can click **Never Remember Password for This Site**.

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



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



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



## Information

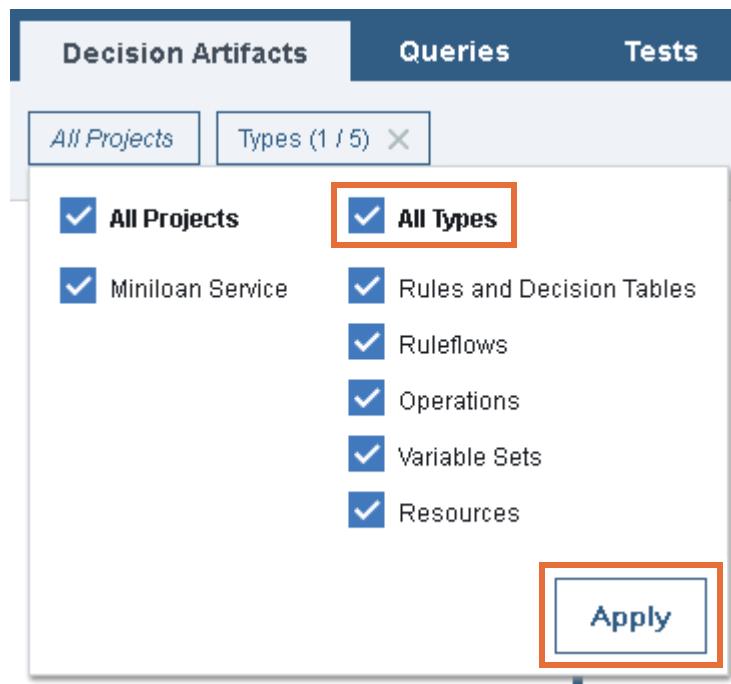
By default, the Business console shows only rule artifacts, such as rules and decision tables. However, you can select various decision artifact types for display in the navigator pane.

— 8. View all of the decision artifacts that are associated with Miniloan Service.

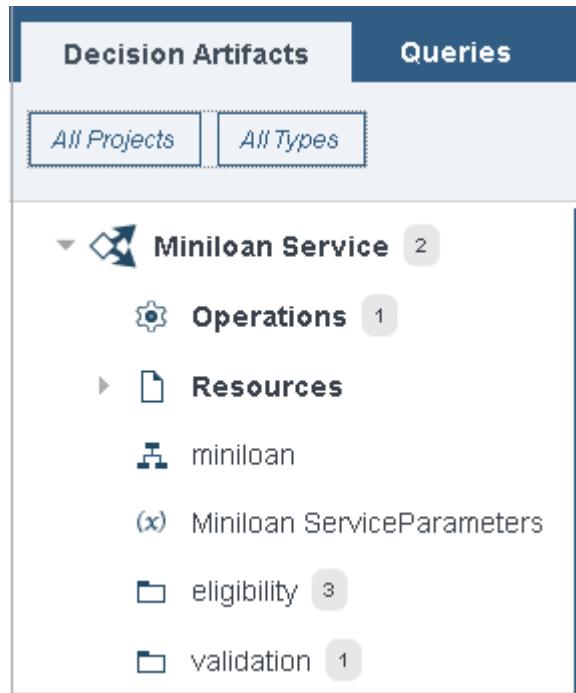
— a. On the **Decision Artifacts** tab, click **Types (1/5)**.



— 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)

- \_\_ 9. View the rules.

- \_\_ a. Click the **eligibility** folder to see the rules that it contains.

- \_\_ b. Click **minimum income** to review the minimum income policy, which is:

```

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

```

```

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.

- \_\_ 10. 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

- 11. 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**

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

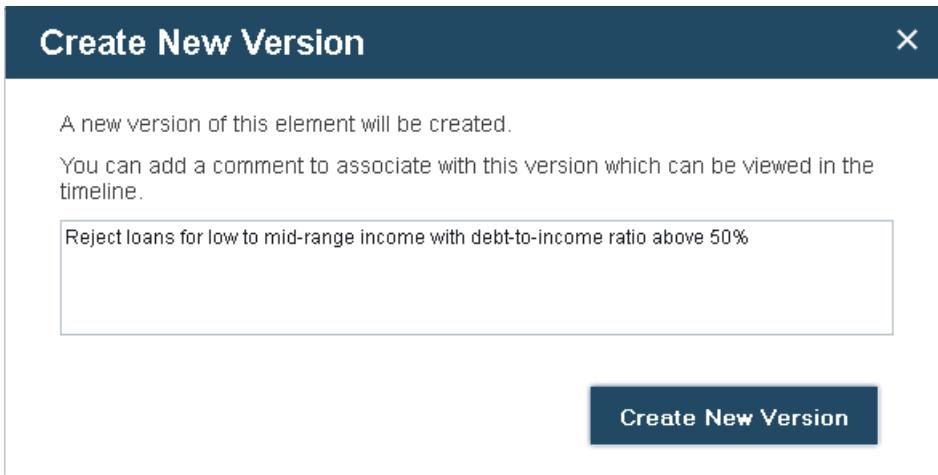
- a. Click **Save**.



- b. In the **Create New Version** field, type a comment, such as:

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

- \_\_ c. Click **Create New Version**.



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

```

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

```

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

## 2.2. Reviewing the rule history

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



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

2. Click **Compare**.

## Compare minimum income Versions X

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

**Current Version (1.1)**   
Created by itsUser1 on Apr 28, 2017  
Reject loans for low to mid-range income with debt-to-income ratio above 50%

**Version 1.0**   
Created by Paul on Jan 23, 2017

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.

Miniloan Service > main

### minimum income ?

You are comparing [Version 1.0](#) with [Current Version \(1.1\)](#) of minimum income.

▼ [Hide summary](#) [Content \(2\)](#) [Properties \(0\)](#)

★ *0.3 was changed to 0.5*

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

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

### version 1.1 (current)

Created by rtsUser1 on Apr 28, 2017

**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 50000

**then**

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

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



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

## 2.3. Changing the credit score requirements

### Scenario

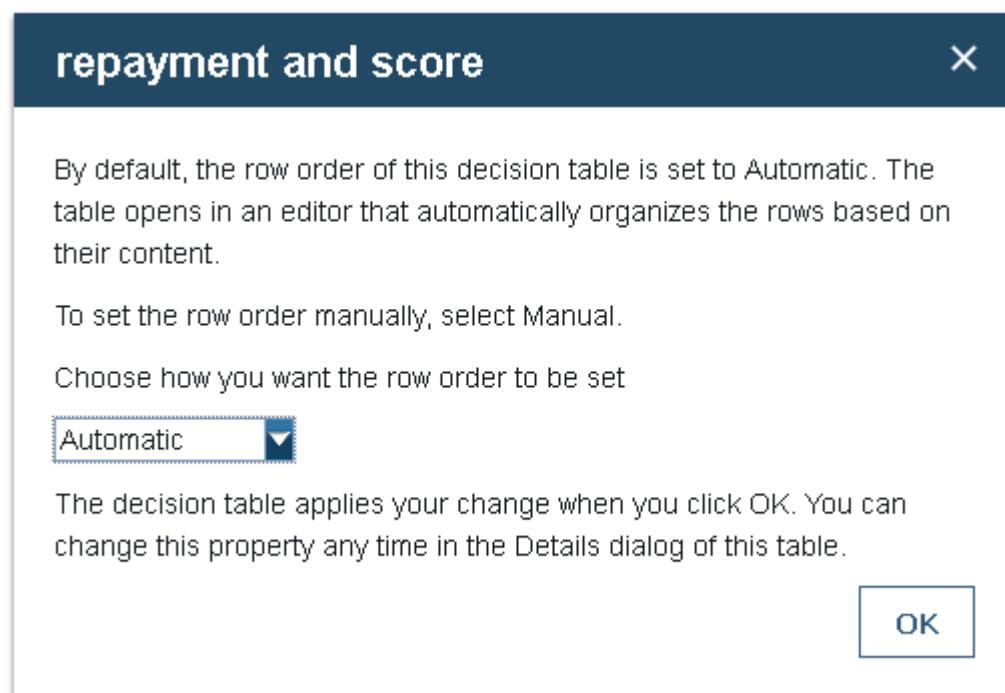
After reviewing the eligibility business policies, Miniloan decided to adjust its credit score requirements. Borrowers with a debt-to-income ratio that falls within the range of 45 and 50 are rejected when their credit score is less than 500.

The credit score policies are implemented in a decision table.

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

The screenshot shows the left navigation pane of the ODM interface. Under the 'Miniloan Service' category, the 'eligibility' folder is selected and highlighted in grey. Inside the 'eligibility' folder, there is a single row named 'repayment and score'. To the right of the table, there is a toolbar with various icons for operations like add, delete, and filter. The 'repayment and score' row itself has columns for 'Name' (containing 'repayment and score'), 'minimum credit score' (containing 'Paul'), 'minimum income' (containing 'rtsUser1'), and a status column with a yellow star icon. The edit icon for this row is highlighted with a red box.

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



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

- \_\_\_ d. Delete 600 and type: 500

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

- \_\_\_ e. Click another area of the table to make sure that the value changed.  
 \_\_\_ f. Click **Save**.  
 \_\_\_ g. In the Create New Version window, add a comment about the update that you made, such as:

Row 5 value changed from 600 to 500

- \_\_\_ h. Click **Create New Version**.

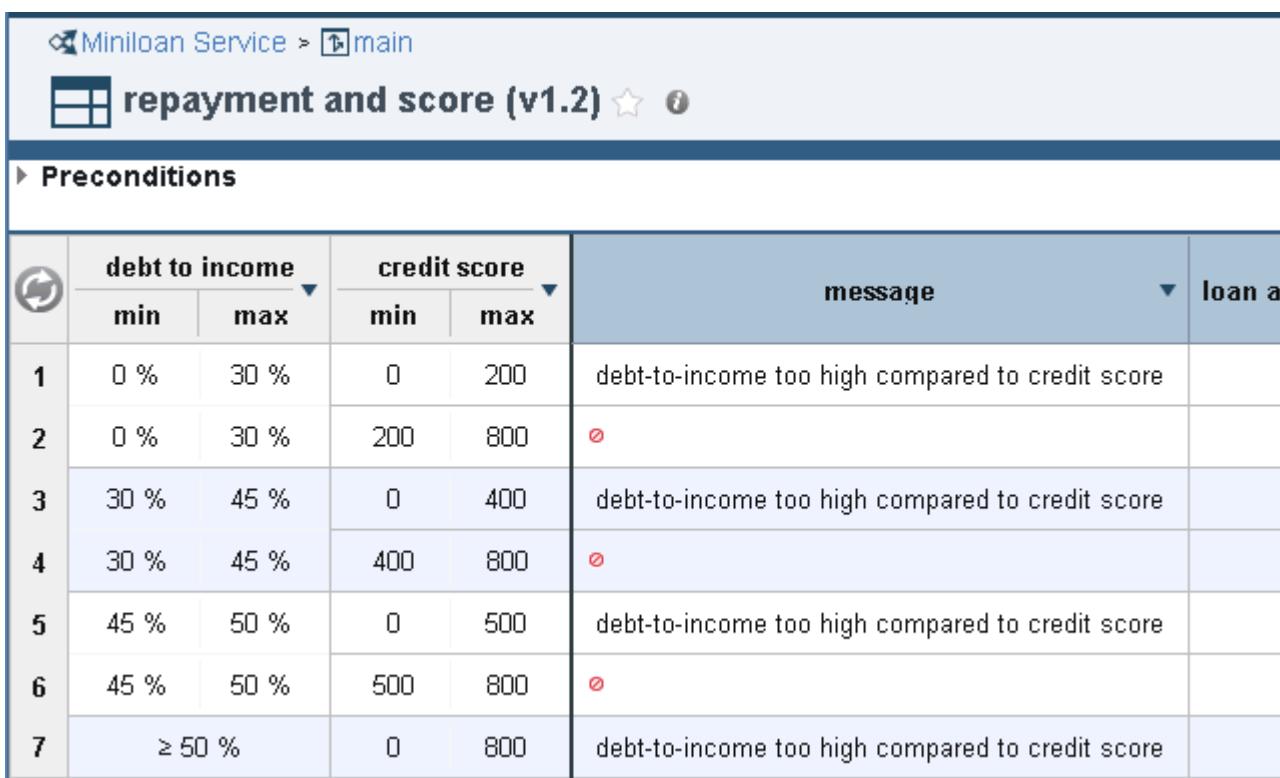
After you create the version, your changes are saved and a gap error is detected in the table. The problem cells are highlighted with a gold triangle in the lower-left corner.

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

**Errors**  
Lines 5 to 6 have gaps - Missing values: [500..600[

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

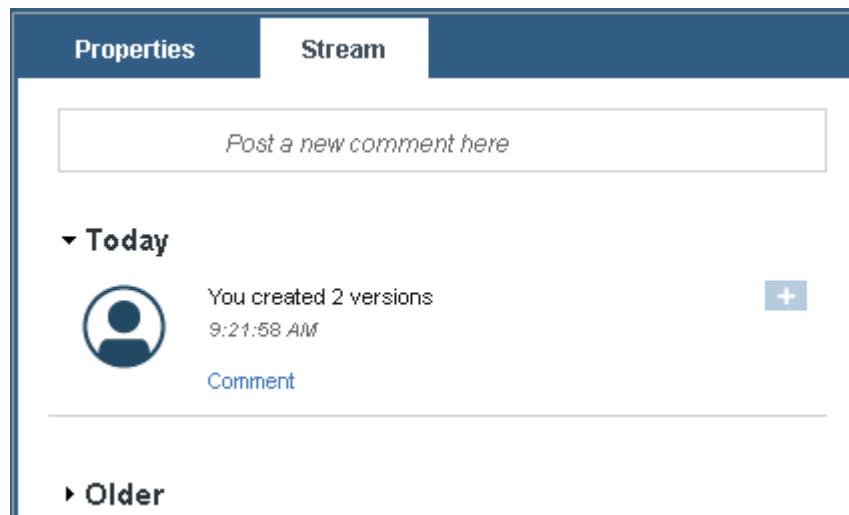
The table no longer shows any errors.



The screenshot shows the 'Miniloan Service > main' interface. Below the header, there is a section titled 'repayment and score (v1.2)' with a star icon and a help icon. Underneath this, a section titled 'Preconditions' is expanded. A table lists seven rows of preconditions:

	debt to income		credit score		message	loan a
	min	max	min	max		
1	0 %	30 %	0	200	debt-to-income too high compared to credit score	
2	0 %	30 %	200	800	∅	
3	30 %	45 %	0	400	debt-to-income too high compared to credit score	
4	30 %	45 %	400	800	∅	
5	45 %	50 %	0	500	debt-to-income too high compared to credit score	
6	45 %	50 %	500	800	∅	
7	≥ 50 %		0	800	debt-to-income too high compared to credit score	

- 3. Click the **Stream** tab on the right view, and note that the rule stream was automatically updated.



The screenshot shows the 'Stream' tab. At the top, there is a text input field with placeholder text 'Post a new comment here'. Below it, a section labeled 'Today' shows a comment from the user: 'You created 2 versions 9:21:58 AM'. There is a blue plus icon (+) to the right of the text. Below this, a 'Comment' button is visible. A link labeled 'Older' is at the bottom left.

- 4. Click the plus icon  to see the details of the version that you created.



The screenshot shows the 'Stream' tab with the same comment from the user. Below it, a section labeled 'Today' shows the same comment again. To the right of the text, there is a minus icon (-). Below the text, a list of changes is shown in a blue box: 'v1.2 Nb Comments 9:21:58 AM' and 'v1.1 Row 5 value changed from 600 to 500 9:21:36 AM'. A 'Comment' button is at the bottom.

## Section 3. Validating rule changes

This section continues with business tasks, and provides an introduction to validating business 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.

By using the testing features in Decision Center, the rule authors can run tests and simulations directly from the Decision Center Business console.

---

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

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

`<InstallDir>\gettingstarted\BusinessConsole`

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

## Running the test

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

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

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

Name	Operation	Recent Report
Miniloan ServiceOperatio	Not run	

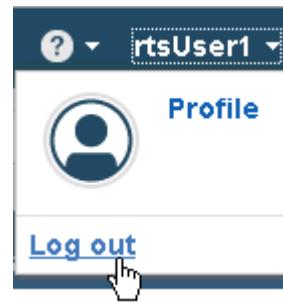
- \_\_\_ 4. In the Run Test Suite window, click **OK** to close the window.  
 \_\_\_ 5. Wait for the test to complete and then click the report link to open the report.  
 Because you changed the rules, you should not expect a 100% success rate.  
 \_\_\_ 6. Click **Close** to close the report.



### Information

Later in this course, you learn how to run tests and simulations.

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



- \_\_\_ c. Close the browser.

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

This section introduces how to use Decision Center to deploy rules to Rule Execution Server. Deployment can be considered to be either a business user task (depending on permissions) or an IT task.

### Scenario

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

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

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



### Information

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

### 4.1. Deploying the decision service

- 1. Sign in to the Business console as a user with administrative privileges:
  - **User name:** rtsAdmin
  - **Password:** rtsAdmin
- 2. Click **Library** and open the **main** branch of the Miniloan Service decision service.
- 3. On the **Decision Artifacts** tab, enable all types of decision artifacts for display in the navigation pane by clicking **Types (1/5)**, selecting **All Types**, and clicking **Apply**.
- 4. In the navigation pane, click **Operations**, and click **Miniloan ServiceOperation**.
- 5. Open the Operation view by clicking the **Open Operation View** icon in the upper-right area of the pane.



- 6. In the upper-right area of the page, click **Edit**.
- 7. Change the value in these fields:
  - **Name:** my\_operation
  - **Ruleset name:** my\_operation

- 8. Save your changes.
  - a. Click **Save** to exit the editor.
  - b. In the Create New Version window, click **Create New Version**.
- 9. Click the **main** breadcrumb to return to the main Decision Service page.
- 10. Click the **Deployments** tab and make sure that you are on the Configurations page.

The screenshot shows the 'Miniloan Service' main page. At the top, there's a breadcrumb trail with a back arrow and the text 'Miniloan Service'. Below it is a navigation bar with tabs: 'Tests', 'Simulations', 'Deployments' (which is highlighted with a red box), 'Snapshots', and 'Model'. Underneath the navigation bar, there are two buttons: 'Configurations' (highlighted with a red box) and 'Reports'. On the right side of the page, there are two buttons: 'Merge Branches' and 'Take Snapshot'.

- 11. Click the **Miniloan** deployment configuration.

The screenshot shows the details of the 'Miniloan' deployment configuration. It includes fields for 'Name' (with a checkbox) and 'Type' (set to 'Nonproduction'). The 'Miniloan' name is highlighted with a red box.

<input type="checkbox"/> Name	▲ Type
<input type="checkbox"/> Miniloan	Nonproduction

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



- 13. On the **General** tab, edit the configuration by changing the **RuleApp name** field to: `my_deployment`
- 14. Click the **Targets** tab and make sure that **Sample** is selected.

The screenshot shows the 'Targets' tab selected. It displays a message: 'Select one or more servers for the deployment. If you...'. Below that is a section for 'Target servers' with two checkboxes: 'Servers' (checked) and 'Sample' (highlighted with a red box).

<input checked="" type="checkbox"/> Servers	Des
<input checked="" type="checkbox"/> Sample	

- 15. Click **Save** and click **Create New Version**.

- \_\_\_ 16. In the toolbar, click the **Deploy** icon.



- \_\_\_ 17. In the “Deploy main” window, click **Deploy**.  
\_\_\_ 18. In the “Deployment status” window, click **OK**.  
\_\_\_ 19. When deployment completes, click the **Report** link.  
\_\_\_ 20. View the report details, and when you are finished, click **Close**.  
\_\_\_ 21. Log out of the Business console.

## Section 5. Monitoring the deployed rules in Rule Execution Server

In the next several parts of this exercise, you work with tools that are in the ODM IT environment: Rule Designer and Rule Execution Server (RES).

The purpose of this part of the exercise is to introduce you to some of the tasks that IT users perform and the ODM modules that are used in the IT environment. Learning about these tasks and ODM modules is important for understanding the role that the IT team plays in the development and management of rule projects.

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

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

### Viewing the deployed RuleApp

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

`http://localhost:9090/res`



#### Information

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



#### Cloud

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

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

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

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

3. Click the **Explorer** tab, and in the Navigator pane, expand **RuleApps > my\_deployment**.

The screenshot shows the IBM Rule Execution Server console interface. The top navigation bar includes tabs for Home, Explorer (which is selected), Decision Warehouse, Diagnostics, and Services. Below the navigation bar is a breadcrumb trail: Explorer > RuleApps. The left sidebar is titled 'Navigator' and contains nodes for RuleApps (1), Resources (1), Libraries (2), and Service Information. The main content area is titled 'RuleApps View'. It features three buttons: 'Add RuleApp', 'Deploy RuleApp Archive', and 'Update RuleApps'. A large 'RuleApps' icon is displayed, followed by the text 'Total Number of RuleApps 1'. Below this, a table lists the deployed RuleApp:

<input type="checkbox"/> Select All	Name	Version	Creation Date
<input type="checkbox"/>	my_deployment	1.0	Feb 20, 2017 2:28:38 PM GMT-1

RuleApp 1 - 1 of 1

You see the ruleset that you deployed.



## Troubleshooting

If you see a Java exception error when you are using Rule Execution Server console, try the following steps:

- Stop the sample server (click **Start**, and then click the **Stop Sample Server** shortcut).
- Restart the sample server (click **Start**, and then click the **Start Sample Server** shortcut).
- Reload the Rule Execution Server console page.

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

The screenshot shows the RuleApp View for the deployment **/my\_deployment/1.0**. The top navigation bar includes links for Add Ruleset, Add Property, Download Archive, and Edit. The deployment details are listed below:

Name	my_deployment
Version	1.0
Creation Date	Apr 28, 2017 9:44:26 AM GMT-04:00
Display Name	
Description	

Below the deployment details is a link to **Show Properties (0)**. The main content area displays a list of rulesets under the heading **1 Ruleset(s)**. A search bar labeled **Name:** is present. The ruleset list table has columns: **Select All**, **Name**, **Version**, and **Ruleset Path**. One ruleset is listed:

Select All	Name	Version	Ruleset Path
<input type="checkbox"/>	my_operation	1.0	/my_deployment/1.0/my_operation/1.0

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

- 5. In the list of rulesets, click **my\_operation** to open it in the Ruleset view.

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

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

Ruleset Parameters

Direction	Name	Kind	XOM Type
	<b>borrower</b>	native	miniloan.Borrower
	<b>loan</b>	native	miniloan.Loan

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

---

Show Managed URIs (1)

---

[Hide Properties](#)

18 properties

<input type="checkbox"/> Select All	Name	Value
<input type="checkbox"/>	decisioncenter.url	http://localhost:9090/decisioncenter?datasource=baselineId=dsm.DsDeploymentBsln%3A1
<input type="checkbox"/>	decisioncenter.version	Decision Center 8.9.0.0
<input type="checkbox"/>	decisionservice.branch.id	dsm.DsDeploymentBsln:130:130
<input type="checkbox"/>	decisionservice.branch.name	Miniloan_2017-02-20T19_28_29Z
<input type="checkbox"/>	decisionservice.branch.type	snapshot
<input type="checkbox"/>	decisionservice.branch.url	http://localhost:9090/decisioncenter/t/library#overviewsnapshot?datasource=baselineId=dsm.DsDeploymentBsln%3A1
<input type="checkbox"/>	decisionservice.deployer.id	rtsAdmin
<input type="checkbox"/>	decisionservice.deployer.name	rtsAdmin
<input type="checkbox"/>	decisionservice.deploymentConfiguration.id	dsm.Deployment:11:33
<input type="checkbox"/>	decisionservice.deploymentConfiguration.name	Miniloan

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. Select **Use Rules** and click **Validate Loan**.

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

The screenshot shows the Miniloan application's user interface. It has two main sections: 'Borrower' and 'Loan'. In the 'Borrower' section, there are three fields: 'Name' (Joe), 'Yearly Inc...' (80000), and 'Credit Sco...'. In the 'Loan' section, there are four fields: 'Amount' (500000), 'Duration' (240), and 'Yearly Int...'. Below these sections is a checkbox labeled 'Use rules' which is checked. A blue button labeled '✓ Validate Loan' is centered below the sections. At the bottom, there is a green box containing the word 'Approved' and a message: 'Your loan is approved with a yearly repayment of 39597'. Another blue box labeled 'Information' contains the text: 'Rule eligibility.repayment and score 6 was executed in rule task miniloan#eligibility.'

Notice that the **Information** section lists the decision table row that you edited as the rule that was used for this loan decision.

- 3. When you are finished viewing the results, close the browser.

## Section 7. Exploring the development environments in Rule Designer

This section continues the introduction to ODM IT tasks and modules by exploring the rule development environment in Rule Designer.

### 7.1. Opening Rule Designer

- \_\_\_ 1. On the **Start** menu, click the **Rule Designer 8.9** shortcut.



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

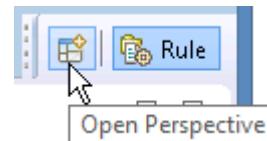
C:\labfiles\workspaces\intro

- \_\_\_ 3. Close the Welcome view.

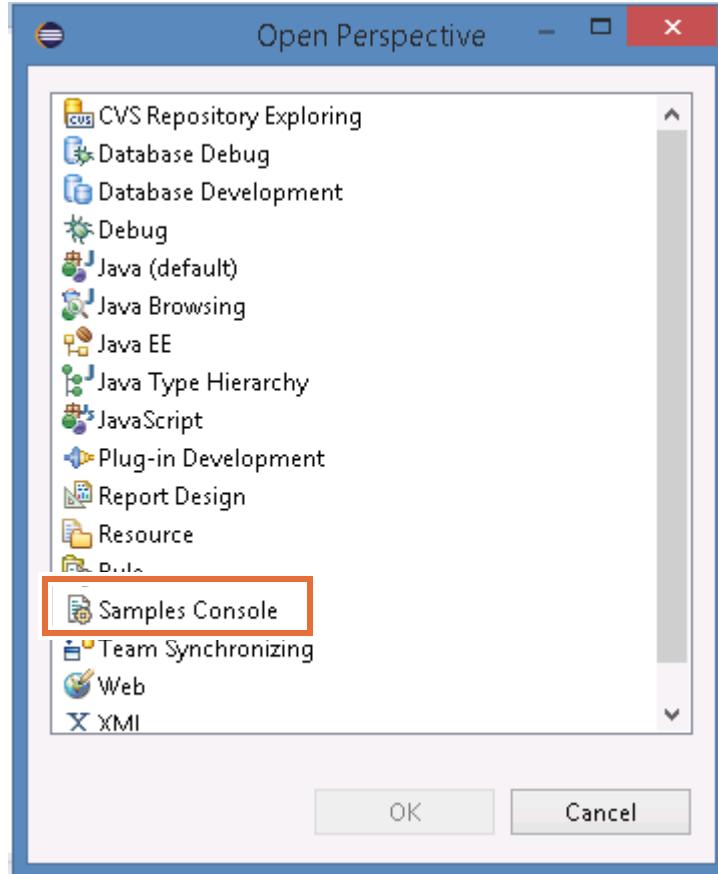
By default, the Rule Designer perspective opens in Eclipse.

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

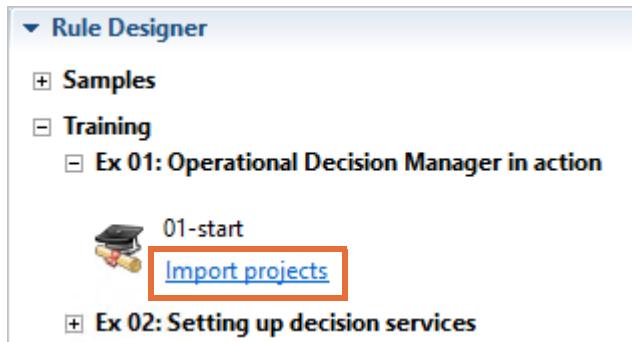
- \_\_\_ a. In the upper-right area of the window, click the **Open Perspective** icon to switch from the Rule perspective.



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



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



The Eclipse perspective automatically switches back to the Rule perspective.

- \_\_ d. Close the Help view by clicking the X.



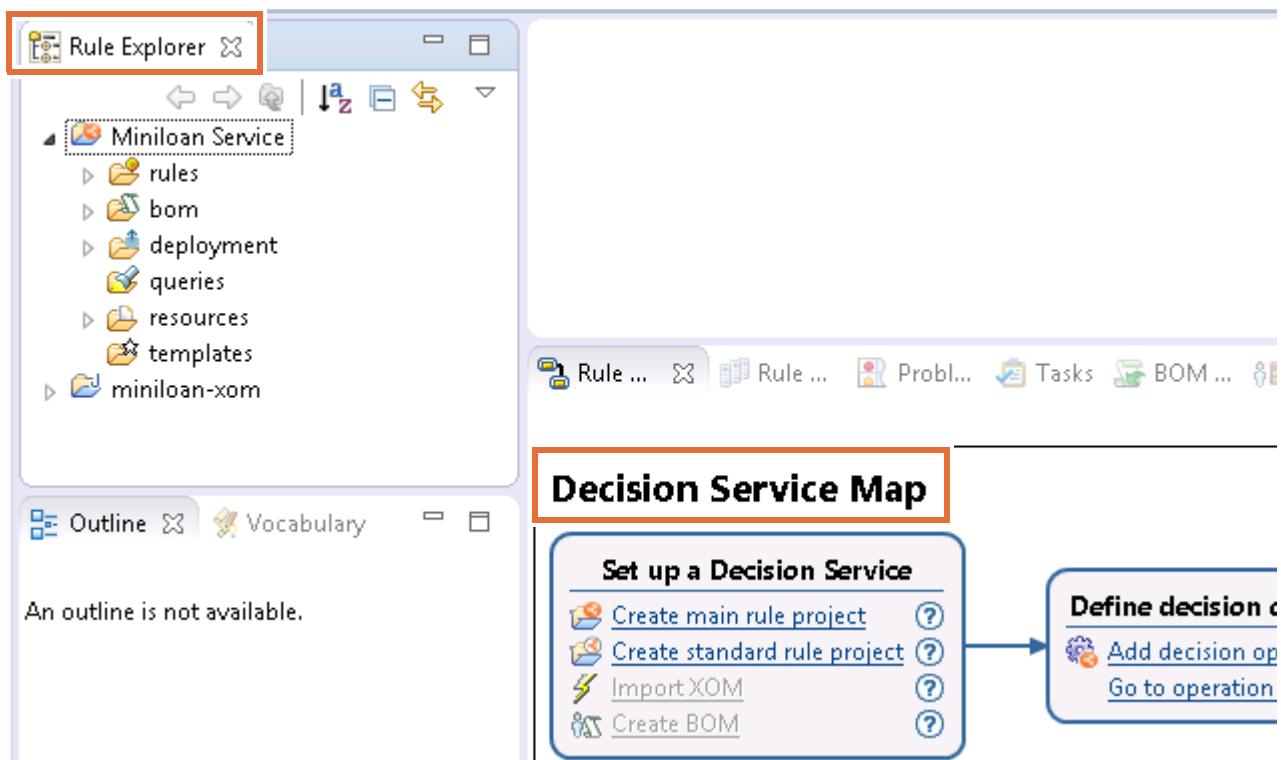
**Hint**

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

## 7.2. Taking a quick tour of Rule Designer

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

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



The Rule Explorer contains these projects:

- **Miniloan Service**: The main rule project that contains Miniloan business rules.
  - **miniloan-xom**: The Java project of the Miniloan application that is composed of the **borrower** and the **loan** Java classes.
- 2. In Rule Explorer, expand **Miniloan Service > rules** to see the rule artifacts.
    - a. Expand the **eligibility** and **validation** folders to see the rules that they contain, and notice that these folders are the same folders that you saw in Decision Center.
    - b. In the **eligibility** folder, double-click **repayment and score** to open the decision table.

- \_\_\_ c. Notice that the values in the **credit score** columns for row 5 and 6 do not reflect the changes that you made in Decision Center, where you changed 600 to 500.
  - \_\_\_ d. Close the editing window for the decision table.
- \_\_\_ 3. Expand the **bom > miniloan > miniloan** folder to see **Borrower** and **Loan**.
- The **bom** folder contains all the vocabulary that is used in the rules.
- \_\_\_ a. Expand **Borrower** and **Loan** to see their members in Rule Explorer.
  - \_\_\_ b. Double-click either **Borrower** or **Loan** to open it in the BOM editor and view its properties and members.
- \_\_\_ 4. Open the **Outline** view (under Rule Explorer), and note the connection between this view and the contents of the **bom** folder.
- The **Outline** view lists all the members of the BOM.
- \_\_\_ a. In the Outline view, expand **Borrower** and **Loan** to see their members.
  - \_\_\_ b. Click the **Vocabulary** tab and expand **borrower** to see the vocabulary expressions that are assigned to the BOM members.
  - \_\_\_ c. Notice the similarities and differences between the expressions that are listed in the **Vocabulary** view and the **Outline** view.
- \_\_\_ 5. In Rule Explorer, go back to the **rules** folder to see how the rules use the vocabulary.
- \_\_\_ a. Go back to the **eligibility** folder and double-click **minimum income** to open this rule in the rule editor.
- 



## Questions

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

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

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

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

---



## Information

This exercise does not cover synchronization between Rule Designer and Decision Center.

You perform synchronization between Rule Designer and Decision Center in [Exercise 4, "Setting up a decision service"](#).

---

## Section 8. Shutting down the modules

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

- 1. Close all browsers for the Miniloan application, Rule Execution Server, and Decision Center consoles.
- 2. Close Rule Designer and click **OK** when prompted to confirm your exit from Eclipse.
- 3. Stop the sample server.
  - a. Click **Start**, click the down arrow, and in the IBM Operational Decision Manager V8.9 section, click **Stop Sample Server**.  
You can also use the **Start** menu **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. Building the model on paper

## Estimated time

01:00

## Overview

This exercise and the next several exercises introduce you to business rule and decision service concepts and tasks, such as modeling business rules, working with rule projects, and BOMs. In this exercise, you learn how to create a UML class diagram that clearly expresses rule vocabulary requirements for implementation by developers.

## Objectives

After completing this exercise, you should be able to:

- Use business policy documents and use cases to create a class diagram that models the business domain
- Identify the connection between the model and the vocabulary that is used to author rules

## Introduction

For this exercise, the use case involves an individual (for example, the driver of the car, an insurance agent, or a car dealer) who submits an application for car insurance coverage by using an online application.

This exercise is split into these sections:

- [Section 1, "Discussion: Drawing a class diagram"](#)

During the first part, your instructor works with you to identify the objects and attributes in the use case. The class discussion with your instructor prepares you for [Exercise 3, "Implementing the model"](#).

- [Section 2, "Building the model on paper: Completing the class diagram"](#)

During the second part, you complete the class diagram, and you read through the supporting business policies and rules that are provided for this exercise.

You start this exercise by identifying objects in the use case and creating a class diagram. In an actual project, vocabulary and rules are discovered together.

For this exercise, the rules are provided, along with more pieces of information, as sources to discover vocabulary. Your goals are to see how the business rules are expressed, and to make sure that the vocabulary in those rules is reflected in your class diagram or object model.

The model in this exercise is simple by design, but your work in this exercise helps prepare you for the next exercise, when you start implementing the model in Operational Decision Manager. You work with more complicated models in subsequent exercises.

## Requirements

This exercise is done on paper and does not require the computer lab environment.

## Section 1. Discussion: Drawing a class diagram

As a class, identify the objects in the use case and discuss how to create the class diagram. You can start the diagram as a class, and complete the class diagram individually in the next exercise activity.

If you are a self-paced student, you can complete this exercise as an independent activity.

<b>Use case name</b>	Handle car insurance request
<b>Actors</b>	Individual who submits the request for the driver (for example, the individual can be the car owner, an insurance agent, or a car dealer) Online car insurance application system
<b>Main scenario</b>	
<ol style="list-style-type: none"> <li>1. The individual accesses the online car insurance application system.</li> <li>2. The online car insurance application system prompts the individual for information about the drivers, the car to be insured, and the amount of insurance coverage.</li> <li>3. The individual enters information about the drivers, the car to be insured, and the type of insurance coverage into the online car insurance application.             <ol style="list-style-type: none"> <li>a. <b>Driver</b> information includes: first name, last name, birth date, gender (male, female), whether this driver is the primary driver.</li> <li>b. <b>Primary driver</b> information includes: marital status (single, married, divorced, widowed), occupation.</li> <li>c. <b>Vehicle</b> information includes: year, value, category (compact, standard, premium, high-end luxury), usage (personal, commercial).</li> <li>d. <b>Policy</b> information includes: amount of coverage, deductible.</li> </ol> </li> <li>4. The online car insurance application validates the information about the drivers and the vehicle.</li> <li>5. The information about the drivers is valid and complete.</li> <li>6. The online car insurance application system determines the eligibility and pricing for the car insurance request, which includes the complete and valid information about the drivers.</li> <li>7. If the request is eligible for insurance coverage, the online insurance quotation system calculates the appropriate premium for the insurance policy.</li> <li>8. The online car insurance application proposes the insurance policy to the individual.</li> </ol>	

### To start the class diagram:

- 1. Circle all the nouns in the use case. Nouns are often objects or attributes.

- How many objects do you see?
  - Can you see at least one or two attributes for each object?
- 2. Draw your class diagram in the blank pages that are provided.
- What kinds of associations and relationships can you identify?
  - Do you see examples of aggregation or composition?
  - Do any of the objects exhibit inheritance?

**Draw your class diagram here:**

## Draw your class diagram here, continued:

If you need more space for your diagram, you can use this page.

## Section 2. Building the model on paper: Completing the class diagram

In this part of the exercise, you complete the object model that you started in [Section 1, "Discussion: Drawing a class diagram,"](#) on page 2-3.

Using the business policies that are provided, you identify vocabulary that is required to formalize the policies into business rules. Later, you implement this model in Rule Designer and write some rules with the vocabulary that you discover here.

### To complete the class diagram:

- 1. Read through the rules in the following **Car insurance coverage rules** table.
- 2. Look for vocabulary that is missing from your model, and add it to the model.

The following table contains excerpts from business documentation that can be used to augment the business vocabulary in the object model.

### Car insurance coverage rules

*Table 1. Car insurance eligibility rules*

Rule name	Rule text	Source	Status
CONS_DRIVER_ AGE	The age of the driver must be in the range of 18 – 80 years of age.	Roberta Lee	Validated
CONS_VEHICLE_ AGE	The vehicle must be under 40 years of age.	Underwriting	Validated
PRIMARY_DRIVER_ MARITAL_STATUS	Primary driver must include marital status.	Underwriting	Validated
PRIMARY_DRIVER_ OCCUPATION	Primary driver must include occupation.	Underwriting	Validated
CONS_VEHICLE_ CATEGORY_AND_ VALUE	The vehicle must not be a luxury vehicle, including “High-end luxury” vehicles, and the value of the vehicle must be under \$120,000 USD.	John Smith Check the vehicle classification guide in doc2765	Validated
CONS_VEHICLE_ USAGE	The vehicle usage must not be commercial.	Underwriting	Validated
CHECK_POLICY_ LIABILITY_AMOUNT	The maximum amount of coverage for liability that is related to damages caused by the primary driver must be in the range \$5,000 - \$200,000 USD.	Underwriting	Validated

For example:

- Is all the vocabulary that is used for the `PrimaryDriver` found in your model?
- Do you have `occupation` as an attribute of `PrimaryDriver`?

**When you are finished with the class diagram:**

- 1. To review a possible solution for this part of the exercise and to continue with Unit 2, see Unit 2, "Modeling for business rules", [Figure 2-48, "Exercise review: A possible solution,"](#) on page 2-56.

**End of exercise**

## Exercise review and wrap-up

The first part of the exercise looked at how to draw a class diagram. In the second part, you used business policy documentation to complete the class diagram.

# Exercise 3. Implementing the model

## Estimated time

01:30

## Overview

This exercise continues the focus on introducing you to business rule and decision service concepts and tasks. It introduces you to developer tasks by showing you how to implement developer models in Rule Designer. You learn what developers need from business users when requirements are handed over to them.

## Objectives

After completing this exercise, you should be able to:

- Implement the business model in Rule Designer
- Write rules against the model to test for completeness

## Introduction

During a regular project in your own organization, you might not be required to set up the authoring environment for rule authors. However, by completing some developer tasks in this exercise, you can better understand how to communicate your requests to the development team for implementation.

This exercise includes these tasks:

- [Section 1, "Preparing the rule authoring environment"](#)
- [Section 2, "Writing the rules to test the BOM"](#)

## Requirements

This exercise is based on [Exercise 2, "Building the model on paper"](#), and requires you to work in Rule Designer on the computer lab environment.

## Section 1. Preparing the rule authoring environment

Before you can begin working on the BOM, you must set up a rule project structure. Rule projects are folders that organize the various artifacts that are used to author rules.

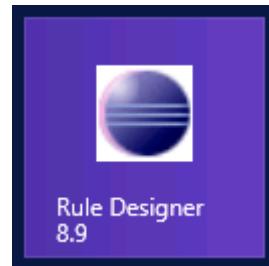


### Note

This exercise gives you the basic instructions to get you started by creating a simple decision service. You learn more details in later units and exercises.

### 1.1. Creating a decision service

- 1. Open Rule Designer and define a workspace.
  - a. Click **Start**, and then click the **Rule Designer 8.9** Start menu shortcut.



You can also open the Apps screen (in the taskbar, click **Start**, and then click the down arrow at the bottom screen) and in the IBM section, click **Rule Designer 8.9**.

- b. When prompted to select a workspace, type:

`<labfilesDir>\workspaces\modeling`



### Reminder

In the computer lab environment that is prepared for this course, `<labfilesDir>` is: `c:\labfiles`

- 2. Close the Welcome pane.



Rule Designer opens in the Rule perspective by default. You can see the Decision Service Map in the lower window. You can use this map to guide you.

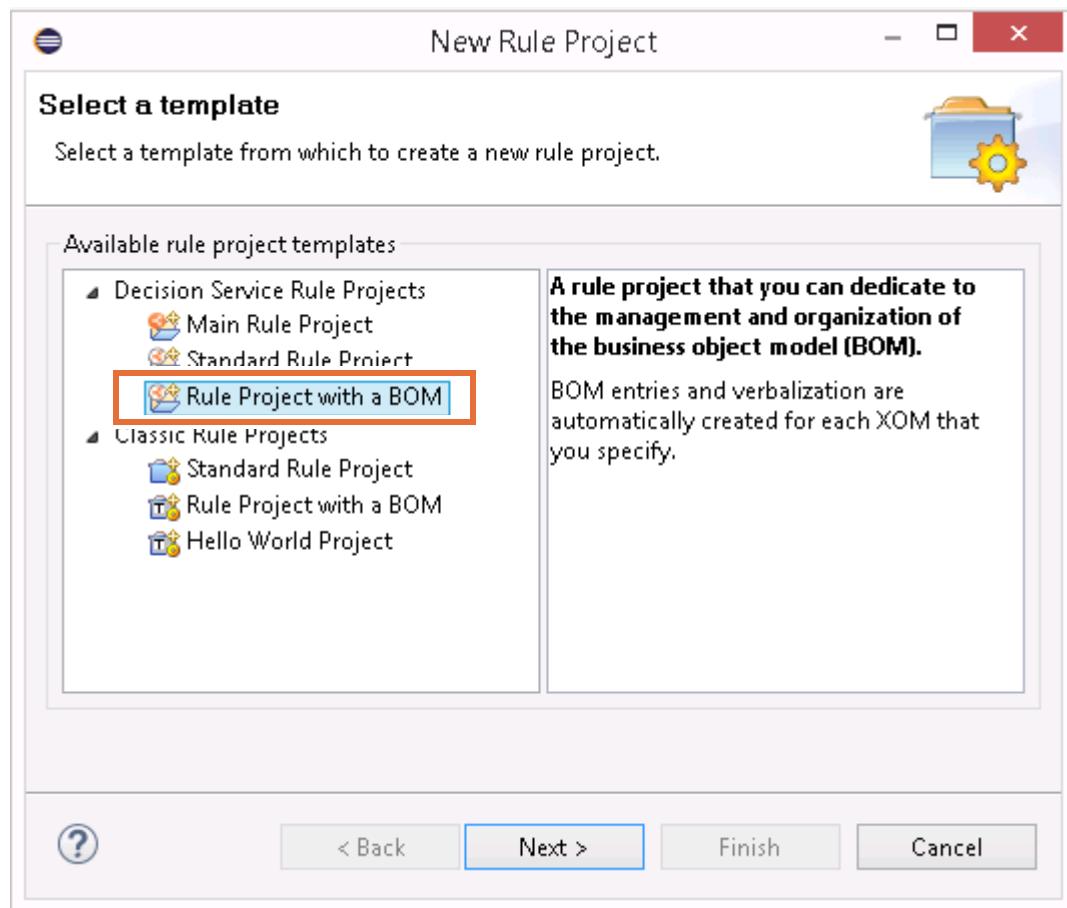
— 3. Create a rule project.

— a. In the Decision Service Map view, click **Create standard rule project**.



The New Rule Project wizard opens.

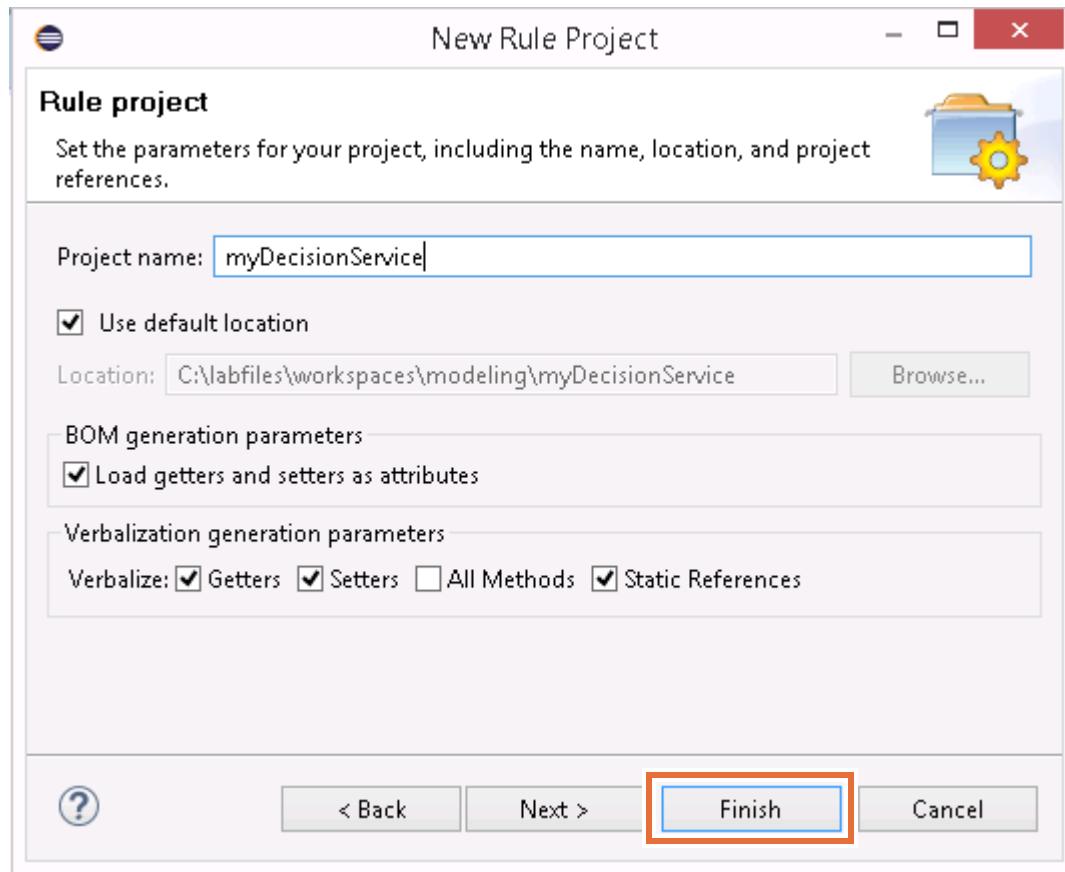
— b. In the Decision Service Rule Projects section of the “Select a template” window, select **Rule Project with a BOM**.



— c. Click **Next**.

— d. In the **Project name** field, type a name for your project, such as: `myDecisionService`

- e. Click **Finish**.

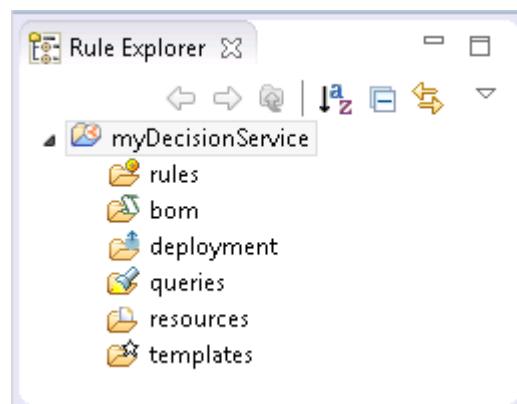


The new project becomes available in the Rule Explorer.

- 4. In the Rule Explorer, expand the folder for your new project to see the contents.

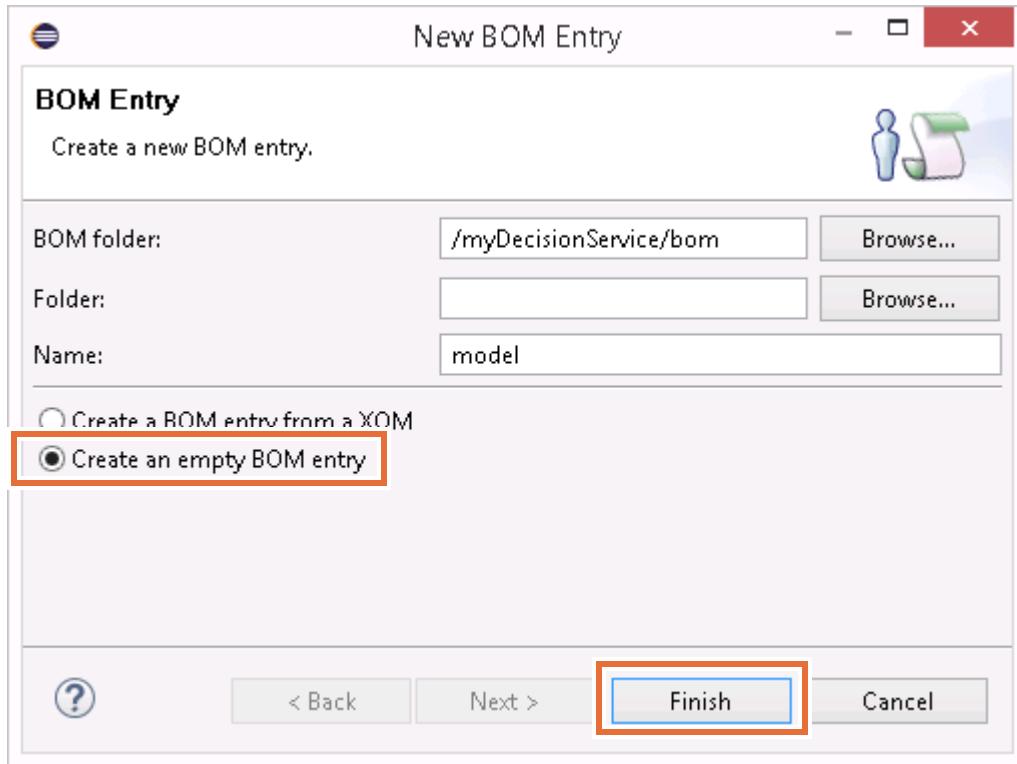
The rule project includes these folders:

- **rules**
- **bom**
- **deployment**
- **queries**
- **resources**
- **templates**



— 5. Create a BOM entry.

- a. Right-click the **bom** folder, and click **New > BOM Entry**.
- b. Leave the name as `model` and select **Create an empty BOM entry**.
- c. Click **Finish**.



You can now start building the BOM to match your class diagram.

## 1.2. Building the BOM

Now that a BOM structure is available, you can complete it by adding classes and verbalizing them to make the classes accessible as vocabulary.

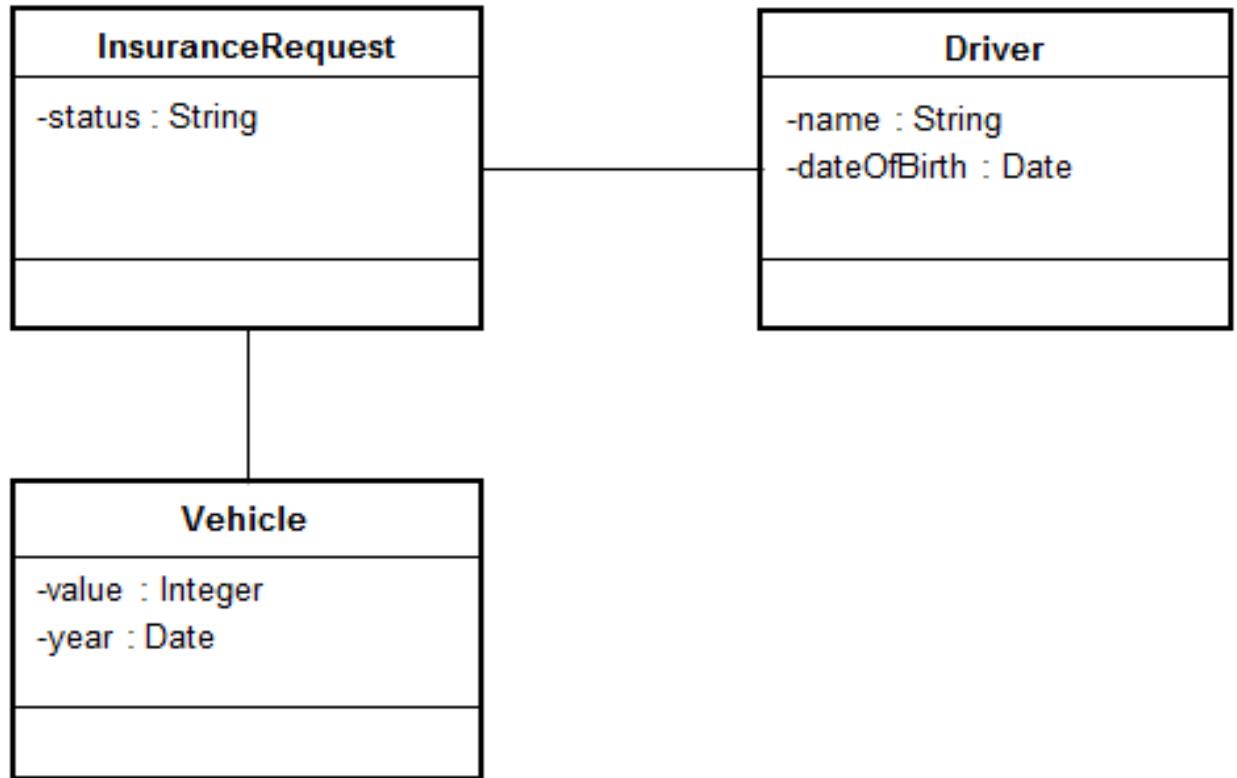
Follow the steps here to create the following classes and some of their attributes from the class diagram.



### Information

This class diagram is based on the diagram from [Exercise 2, "Building the model on paper"](#).

For simplicity, you do not implement the full class diagram that was presented as a possible solution.

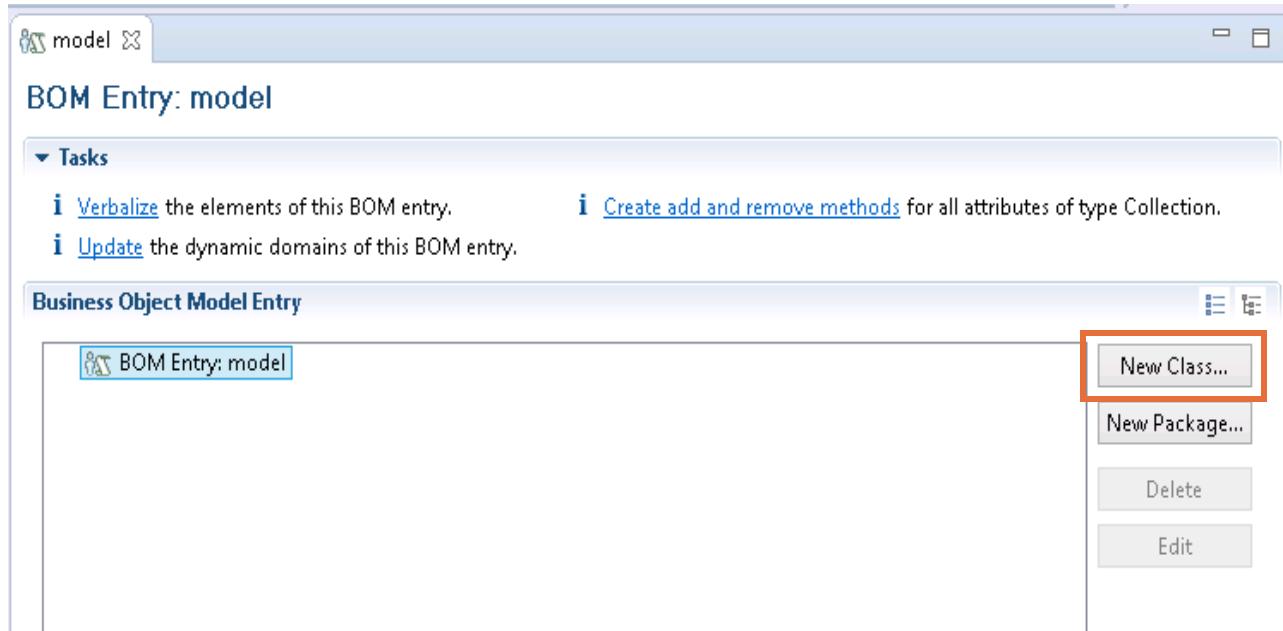
**Note**

The attribute type is shown beside the attribute name. For example, the `name` attribute is of type `String`.

## Creating the InsuranceRequest class

- 1. Create the `InsuranceRequest` class.
- a. Expand the **bom** folder and double-click **model** to open the BOM editor.

- \_\_ b. In the BOM editor, click **New Class**.



- \_\_ c. In the **Name** field, type `InsuranceRequest` as the name for your first class, and click **Finish**.  
 \_\_ d. Save your work. You can press Ctrl+S or use the **File > Save** menu.

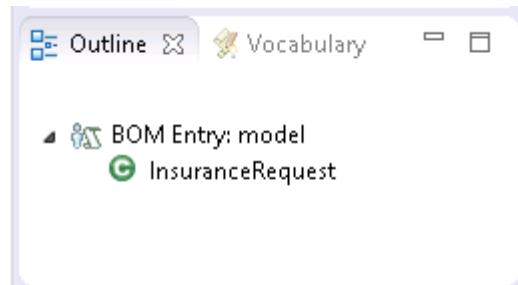


## Troubleshooting

You can ignore errors that are shown after you save your work. Rule Designer checks whether supporting code that corresponds to the new class is in the execution object model (XOM).

Because you create the class without using code, Rule Designer shows warnings.

- \_\_ 2. Add the `status` attribute to the `InsuranceRequest` class.  
 \_\_ a. In the Outline view, double-click `InsuranceRequest` to open it in the BOM editor.



- \_\_ b. In the Members section, click **New** to add an attribute.

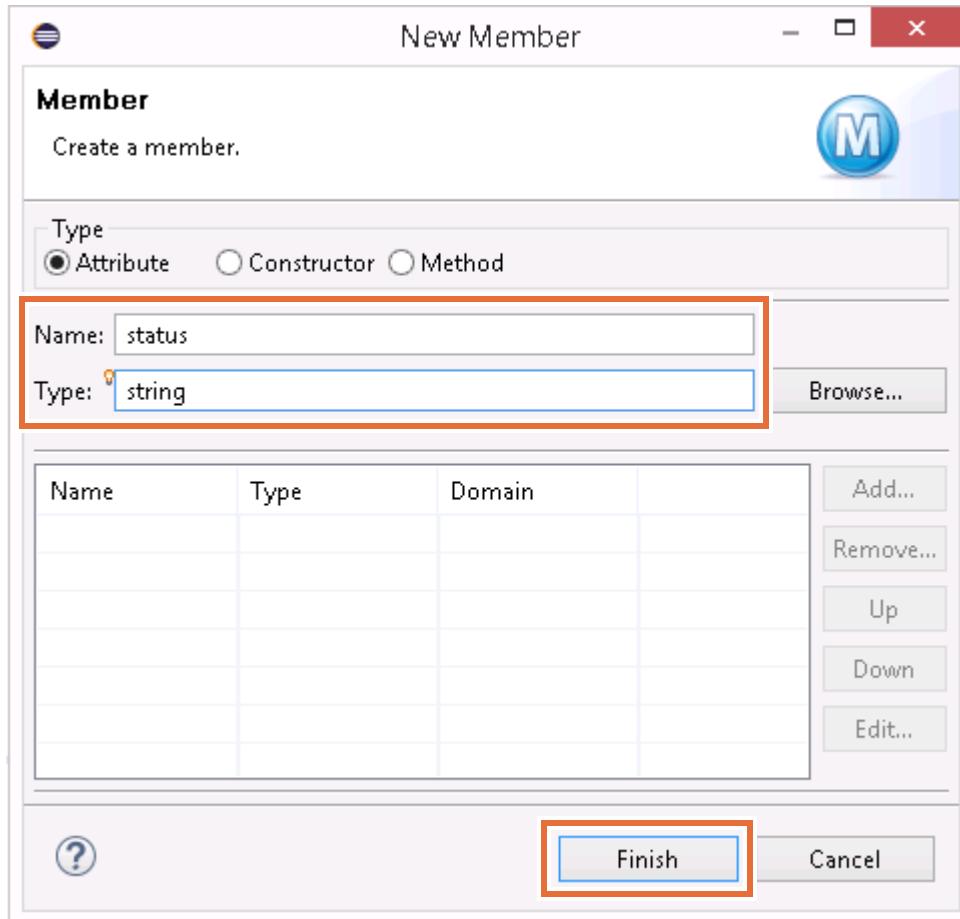
The screenshot shows the 'Class InsuranceRequest (package: default)' properties dialog. The 'General Information' tab is selected, showing fields for Name (InsuranceRequest), Namespace, Superclasses (java.lang.Object), and Interfaces. A 'Deprecated' checkbox is unchecked. The 'Members' tab is also selected, displaying a list area with a 'New...' button highlighted by a red box. Other tabs visible include 'Class Verbalization', 'Domain', and 'Categories'.

## i Information

Attributes are added as *members* of the class.

- \_\_ c. In the **Name** field, type: status

- \_\_ d. In the **Type** field, enter `string` and click **Finish**.



### Information

You can also click **Browse** next to the **Type** field to select a valid type. You can then either type the name of the attribute in the field to filter the list, or scroll through the list to find the attribute that you need.

- \_\_ 3. Save your work.

## Adding the Driver class

- 1. Click the **Package** tab of the BOM editor to add another class.

The screenshot shows the 'BOM Entry: model' editor window. At the top, there's a toolbar with icons for file operations. Below it is a 'Tasks' section with two items: 'Verbalize the elements of this BOM entry.' and 'Create add and remove methods for all attributes of type C'. The main area is titled 'Business Object Model Entry' and contains a tree view. The root node is 'BOM Entry: model', which has a child node 'InsuranceRequest'. To the right of the tree view are four buttons: 'New Class', 'New Pack', 'Delete', and 'Edit'. At the bottom of the editor window, there's a navigation bar with tabs: 'Package' (which is highlighted with a red box), 'Class', 'Member', 'model.bom', 'model.b2x', and 'model\_en\_US.voc'.

- 2. Add the **Driver** class.
- a. Click **New Class**.
  - b. In the **Name** field, type **Driver** and click **Finish**.
  - c. Save your work.
- 3. Double-click **Driver** (either in the BOM Entry list or in the Outline view) to open the **Class** tab.



### Information

Next, you add the following attributes (or members) to the **Driver** class, as shown in the class diagram.

- name of type **string**
- dateOfBirth of type **Date**

- 
- 4. Add the **name** attribute.
- a. In the Members section for the **Driver** class, click **New**.

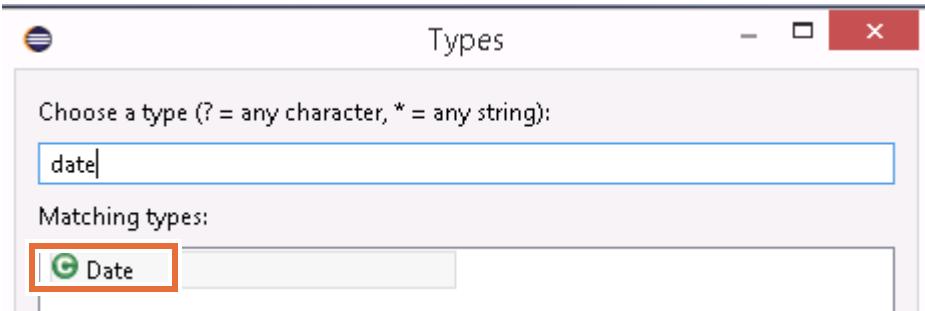
- \_\_\_ b. In the **Name** field, type: name
  - \_\_\_ c. For the **Type** field, enter string and click **Finish**.
  - \_\_\_ 5. Add the `dateOfBirth` attribute.
  - \_\_\_ a. In the Members section, click **New**.
  - \_\_\_ b. In the **Name** field, type: `dateOfBirth`
- 



### Reminder

Remember that attribute names cannot have spaces.

- \_\_\_ c. For the **Type** field, click **Browse**.
- \_\_\_ d. In the **Choose a type** field, enter: date
- \_\_\_ e. In the Matching types list, select **Date** and click **OK**.



- \_\_\_ f. Click **Finish**.



### Troubleshooting

When you manually enter a type in the **Type** field, such as `Date`, you might see the following error message:

You must specify a valid type.

You can use **Browse** to select the type that you need. In the **Browse** menu, enter the type name in the **Choose a type** field. As you enter the name, the list of types is filtered to help you select the correct type.

- \_\_\_ 6. Save your work.

## Adding the Vehicle class

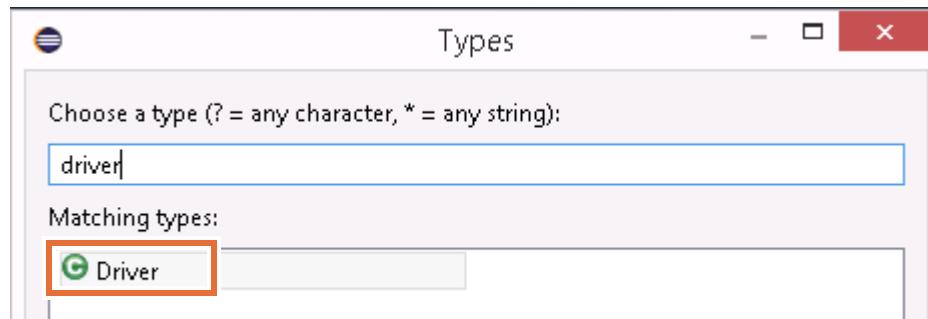
- \_\_\_ 1. Return to the **Package** tab, and add a class that is named: `Vehicle`
- \_\_\_ 2. Add the following attributes to the `Vehicle` class.
  - `value` of type `Integer`
  - `year` of type `Date`

- \_\_\_ 3. Save your work.

## Creating associations between classes

In this section, you create associations between the `Driver` and `Vehicle` classes and the `InsuranceRequest` class by adding attributes of these types. You add a `driver` attribute of type `Driver` and a `vehicle` attribute of type `Vehicle`.

- \_\_\_ 1. Click the **Package** tab, and double-click **InsuranceRequest** to open the **Class** tab.
- \_\_\_ 2. In the Members section for the `InsuranceRequest` class, add the `driver` attribute.
  - \_\_\_ a. Click **New**.
  - \_\_\_ b. In the **Name** field, type: `driver`
  - \_\_\_ c. For the **Type** field, click **Browse**.
  - \_\_\_ d. In the **Choose a type** field, enter: `driver`
  - \_\_\_ e. In the Matching types list, select **Driver** and click **OK**.



- \_\_\_ f. Click **Finish**.
- \_\_\_ 3. Repeat [Step 2](#) to add the `vehicle` attribute.
- \_\_\_ 4. Save your work.

### 1.3. Verbalizing the classes

Verbalization associates a natural-language name or phrase with classes and attributes. Those terms and phrases are used in the rule editors when you write rules, instead of programming language terms that are used in the code.

## Verbalizing your classes

- \_\_\_ 1. If the BOM editor is not open to the `InsuranceRequest` class tab, double-click the `InsuranceRequest` class in the Outline view.

- 2. In the Class Verbalization section, click **Create** to generate a default verbalization.

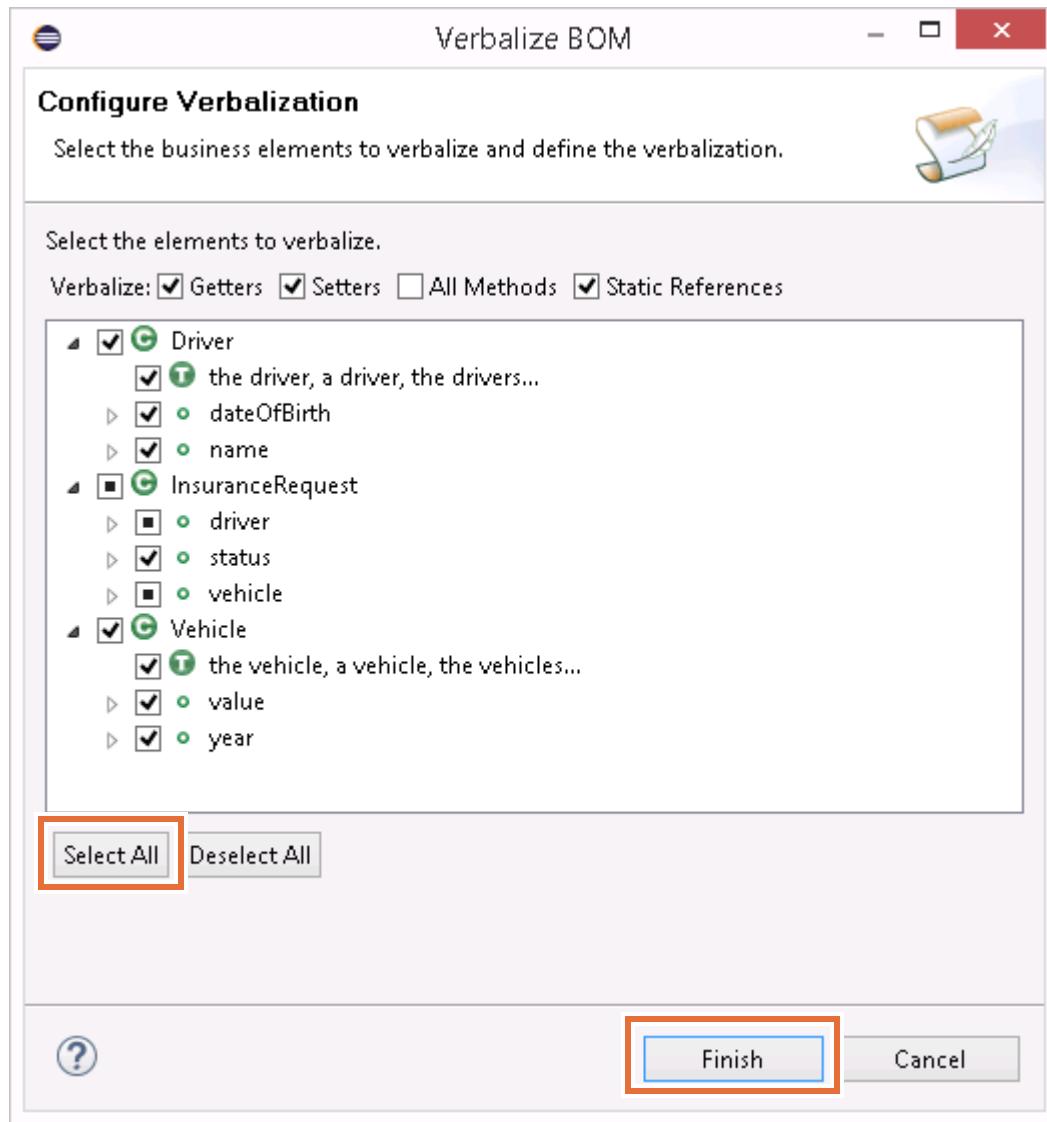
### Information

This step shows you how to manually verbalize a class so that you know how to do it on a one-by-one basis, if required. As you add more classes and attributes, you can generate the verbalization when you first create them.

You can also verbalize all of the elements in the BOM entry on the **Package** tab, as described in the following instructions.

- 3. Verbalize the rest of the classes and attributes in the BOM.
- Click the **Package** tab.
  - Under the Tasks section, click **Verbalize the elements of this BOM entry**.

- \_\_ c. In the Configure Verbalization window, click **Select All**, and then click **Finish**.

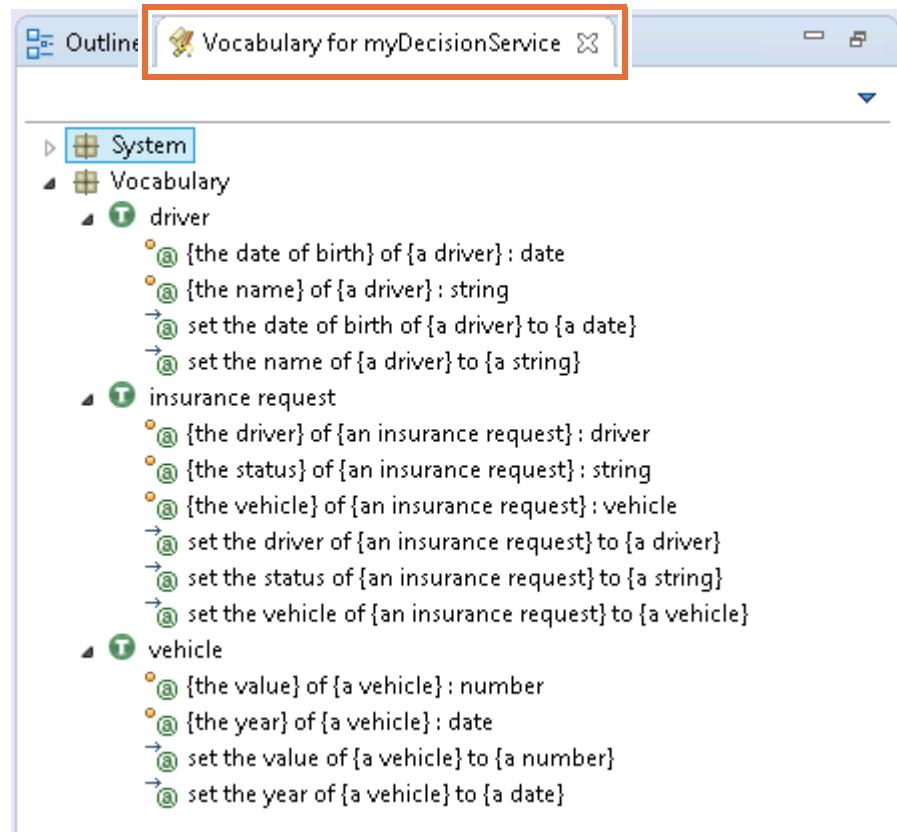


### Note

Ignore the error indicators on the BOM.

- \_\_ 4. Save your work.

- 5. In the Vocabulary view, expand the classes to see the vocabulary list that is now available to use for writing rules.



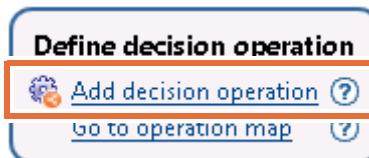
## 1.4. Defining ruleset variables

Before you can start writing rules, you need to have ruleset variables in your vocabulary that represent the data or objects that your rules make decisions about. For example, when John Smith requests car insurance, his request is passed to the decision service by using a ruleset variable.

For this exercise, you create an `insuranceRequest` ruleset variable that is used to pass information to the decision service and return a decision from the decision service.

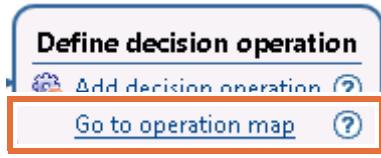
### Defining the variables

- 1. In the Decision Service Map view, click **Add decision operation**.



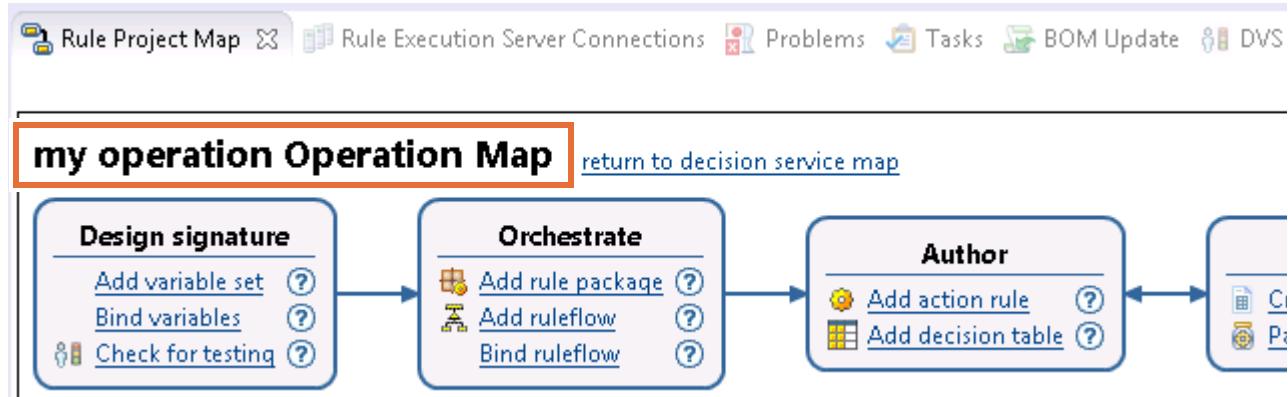
- 2. In the New Decision Operation wizard, in the **Name** field, type: `my operation`  
— 3. Click **Finish**.

- 4. In the Decision Service Map, click **Go to operation map**.

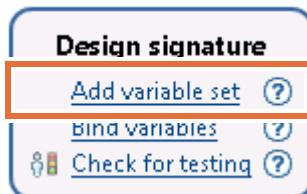


- 5. Select **myDecisionService > deployment > my operation.dop**, and click **OK**.

The Decision Service Map view switches to the Operation Map view.



- 6. In the “Design signature” task, click **Add variable set**.



- 7. In the New Variable Set window, in the **Name** field, type `my parameters` and click **Finish**.

The Variable Set editor opens.

- 8. In the Variable Set editor, add an `insuranceRequest` variable to represent the insurance request.

- a. Click **Add**.

A default row is added, and you can type over the values as described next.

- b. In the **Name** column, type: `insuranceRequest`

- c. In the **Type** column, click the cell, and then click the ellipsis (...) to open the Types window.

- d. Find the **InsuranceRequest** type in the Matching Types list, and double-click to select it.

- e. In the **Verbalization** field, type: the insurance request

The verbalization is what you see in the vocabulary list of the rule editors. In this case, you see the `insurance request` in the list when you start writing rules.

- 9. Press **Ctrl+S** to save your work.

- 10. Close the editing windows for `my parameters`, `my operation`, and `model`.

## Section 2. Writing the rules to test the BOM

In this section, you test the completeness of the BOM by writing rules. Rule authoring is an iterative and collaborative process that involves both the business and technical teams. By writing rules, you can discover what vocabulary needs to be added to the BOM, and you can work with the development team to make sure that the required vocabulary is implemented.

For this exercise, the following business policies must be written as formal rules by using the rule editors:

<b>Policy</b>	The vehicle value must be greater than \$500.
<b>Formal rule</b>	If the value of the vehicle of the insurance request is less than 500 then set the status of the insurance request to REJECTED
<b>Policy</b>	The vehicle value must be less than \$100,000.
<b>Formal rule</b>	If the value of the vehicle of the insurance request is greater than 100000 then set the status of the insurance request to REJECTED
<b>Policy</b>	The driver must be 18 years of age or older.
<b>Formal rule</b>	If ... then ...
<b>Policy</b>	The driver must be 80 years of age or younger.
<b>Formal rule</b>	If ... then ...

### 2.1. Writing new rules

Rule Designer has two rule editors: the default Intellirule editor, and the Guided editor, which is designed to simplify rule authoring. In this exercise, you work with the Guided editor.

#### Creating the minimum vehicle value rule

- \_\_\_ 1. Create a rule in your rule package, named: `min_value`
  - \_\_\_ a. In the Author part of the Operation Map, click **Add action rule**.
  - \_\_\_ b. In the **Name** field, enter `min_value` and click **Finish**.

The new rule opens in the Intellirule editor by default.

- \_\_\_ 2. Switch to the Guided editor to edit the new `min value` rule.
  - \_\_\_ a. Close the `min value` rule.
  - \_\_\_ b. In the Rule Explorer, right-click **min value** and click **Open With > Guided Editor**.
- \_\_\_ 3. Click **<select a condition>** and choose the text that matches the action part of your rule:  
If the value of the vehicle of the insurance request is less than 500
  - \_\_\_ a. In this case, you select:  
**the value of <a vehicle>**
  - \_\_\_ b. Click **<a vehicle>** and select **the vehicle of <an insurance request>**.
  - \_\_\_ c. Click **<an insurance request>** and select **the insurance request**.
  - \_\_\_ d. Click **is** and select **is less than**.
  - \_\_\_ e. Click **<a number>** and type: 500
- \_\_\_ 4. Click **<select an action>** and choose the text that matches the action part of your rule:  
then set the status of the insurance request to REJECTED
  - \_\_\_ a. In this case, you select:  
**set the status of <an insurance request> to <enter a string>**
  - \_\_\_ b. Click **<an insurance request>** and select **the insurance request**.
  - \_\_\_ c. Click **<enter a string>** and type: REJECTED
- \_\_\_ 5. Save your work.

## Creating the maximum vehicle value rule

- \_\_\_ 1. Create a rule in your rule package, named: `max value`
- \_\_\_ 2. Open the new `max value` rule in the Guided editor.
- \_\_\_ 3. Click **<select a condition>** and choose the text that matches the action part of your rule:  
If the value of the vehicle of the insurance request is more than 100000
  - \_\_\_ a. In this case, you select:  
**the value of <a vehicle>**
  - \_\_\_ b. Click **<a vehicle>** and select **the policy of <an insurance request>**.
  - \_\_\_ c. Click **<an insurance request>** and select **the insurance request**.
  - \_\_\_ d. Click **is** and select **is more than**.
  - \_\_\_ e. Click **<a number>** and type: 100000
- \_\_\_ 4. Click **<select an action>** and choose the text that matches the action part of your rule:  
then set the status of the insurance request to REJECTED
  - \_\_\_ a. In this case, you select:

- set the status of <an insurance request> to <enter a string>**
- \_\_\_ b. Click <an insurance request> and select the insurance request.
  - \_\_\_ c. Click <enter a string> and type: REJECTED
  - \_\_\_ 5. Save your work.

## 2.2. Creating the minimum and maximum age rules

In this section, you create the minimum and maximum age rules.

- \_\_\_ 1. Create the min age rule.
  - \_\_\_ a. In the Author part of the Operation Map, click **Add action rule**.
  - \_\_\_ b. Name the rule: min age
  - \_\_\_ c. Close the Intellirule editor and open with the Guided editor.
- \_\_\_ 2. Click <**select a condition**> and look through the list of vocabulary.

Which vocabulary item should you select to express the rule about the age of the driver?



### Hint

As you start writing the rules about driver age, you discover that some of the required vocabulary is missing.

As you write rules in your own organization, you might find missing vocabulary that must be defined in the BOM by Development before you can finish writing the rules.

In this case, you need to have “age” in the vocabulary list before you can author rules about the age of the driver.



### Questions

What steps must you take to include “age” in the vocabulary list for Driver?

### Answers

The following scenario includes the steps that might take place to implement “age” in the vocabulary list:

- The business user realizes that “age” is missing from the vocabulary.
- The business user asks the development team for help in creating vocabulary for “age.”
- Developers update the code so that “age” is included in the vocabulary.
- In this case, the developer:
  - o Writes a function that calculates the age of the driver that is based on the dateOfBirth attribute.

- Updates the vocabulary in the BOM to include the age of the driver.
  - The business user can then author rules that include the age of the driver.
- 

- 3. Close Rule Designer.
  - 4. Go to Unit 2, "Modeling for business rules", [Figure 2-60, "Exercise review \(1 of 2\): A possible solution,"](#) on page 2-71 to see a review of this exercise.
- 



### Important

Remember that defining and implementing the BOM is an iterative process, and often requires frequent collaboration with Development.

Interaction and communication between the business and technical organizations is crucial.

---

## End of exercise

## Exercise review and wrap-up

In this exercise, you used the top-down approach to do the following tasks:

- Model the rule vocabulary by working with use cases and business policies.
- Implement your class diagram as a business object model (BOM) in Rule Designer.
- Author some rules to test whether the BOM is complete.

In an actual project, the next steps would involve the development team who implement the execution object model (XOM) in Java code or XML so that the rules can run in an application.

# Exercise 4. Setting up a decision service

## Estimated time

01:30

## Overview

This exercise continues the focus on introducing you to business rule and decision service concepts and tasks. You learn how to prepare the rule authoring environment in Rule Designer by creating a decision service.

## Objectives

After completing this exercise, you should be able to:

- Set up a decision service for rule authoring
- Create a ruleflow
- Share and synchronize rule projects between the business and development environments

## Introduction

Preparing the rule authoring environment can be a task that technical Operational Decision Manager users, such as developers, perform. However, learning how to use Rule Designer to set up and synchronize rule projects and decision services can help you further understand how ODM modules work together across the business and IT organizations.

This exercise includes these sections:

- [Section 1, "Creating a standard rule project"](#)
- [Section 2, "Importing the XOM and creating the BOM"](#)
- [Section 3, "Creating the main rule project for the decision service"](#)
- [Section 4, "Creating and defining the decision operation"](#)
- [Section 5, "Orchestrating execution"](#)
- [Section 6, "Publishing and synchronizing the rule project"](#)

As you develop your decision service, you can use the Decision Service Map to guide you through the actions.

## Requirements

This exercise requires you to use Rule Designer in the computer lab environment for this course.

## Section 1. Creating a standard rule project

In this section of the exercise, you create an empty standard rule project.

### 1.1. Opening a new workspace in Rule Designer and viewing the Decision Service Map

- 1. In Rule Designer, start a new workspace.
  - a. If Rule Designer is not open, start Rule Designer by clicking **Start** and then clicking the **Rule Designer 8.9** shortcut.
  - If Rule Designer is already open, click **File > Switch Workspace > Other**.
  - b. When prompted for a workspace in the Workspace Launcher, type:  
`<labfilesDir>\workspaces\decision_service`



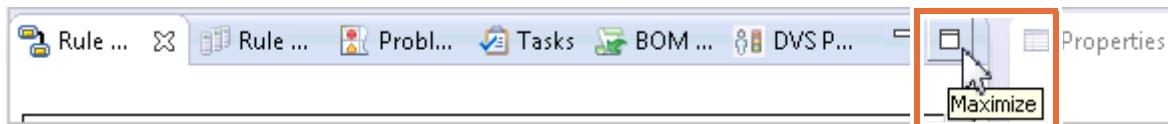
#### Reminder

In the computer lab environment that is prepared for this course, `<labfilesDir>` is: `C:\labfiles`

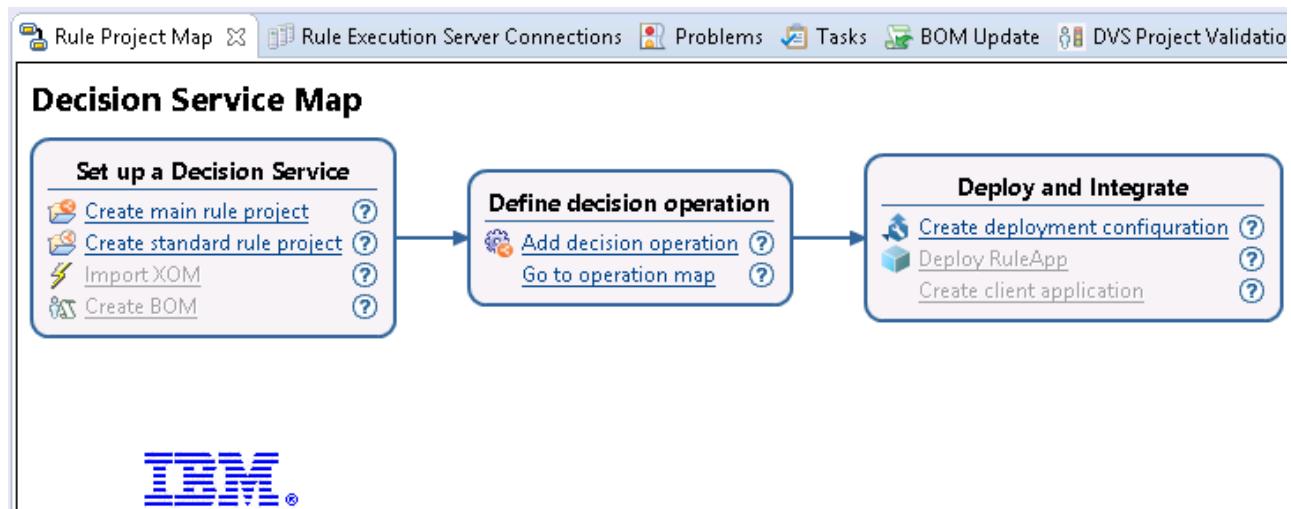
- 2. Close the Welcome pane.



- 3. In the Rule Perspective, maximize the **Rule Project Map** view, which is at the bottom of the Rule Designer window, to see all the **Decision Service Map** tasks.

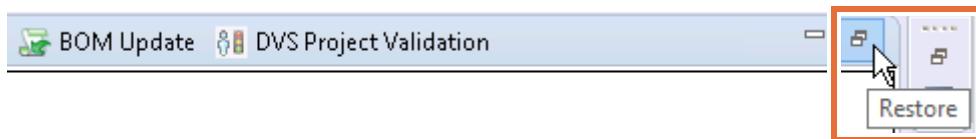


The Decision Service Map guides you through the actions that are involved in setting up the rule projects in a decision service.



The highlighted links indicate which actions you can take next.

- 4. Click **Restore** to return to the original view.



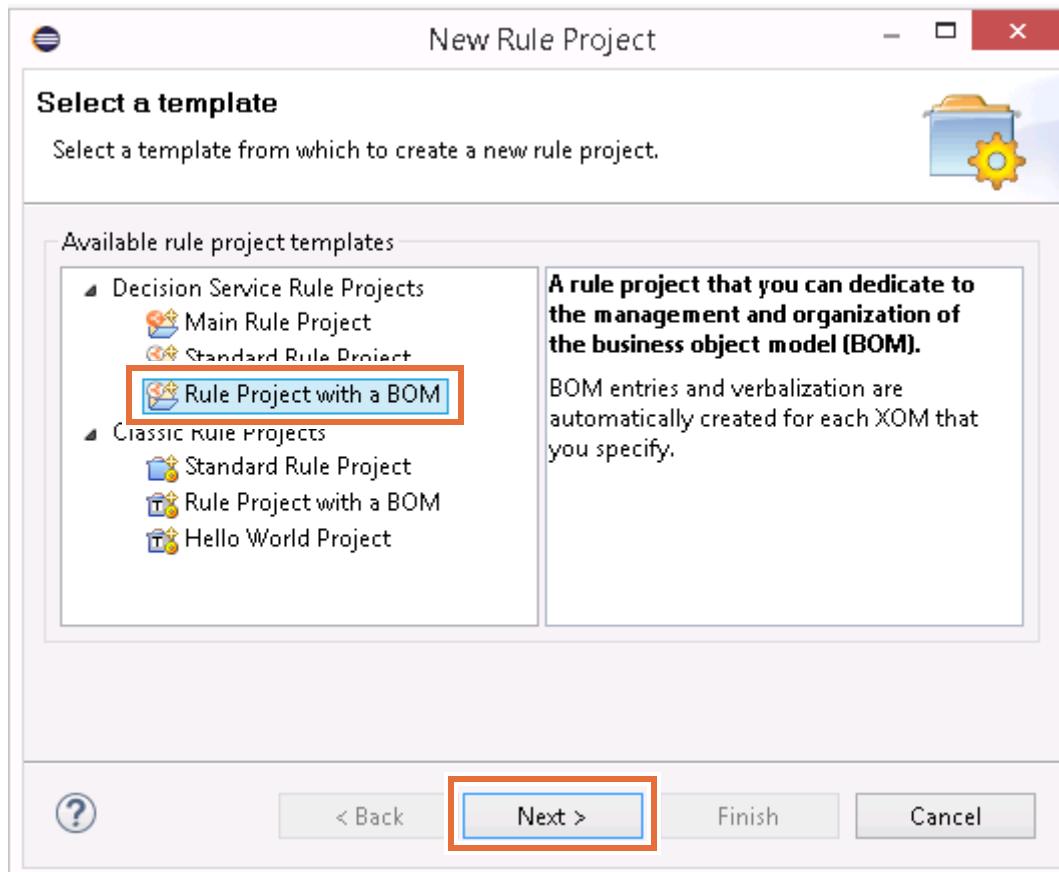
## 1.2. Creating a standard rule project with a BOM

In this section, you create a standard rule project that stores the business object model (BOM) for the decision service.

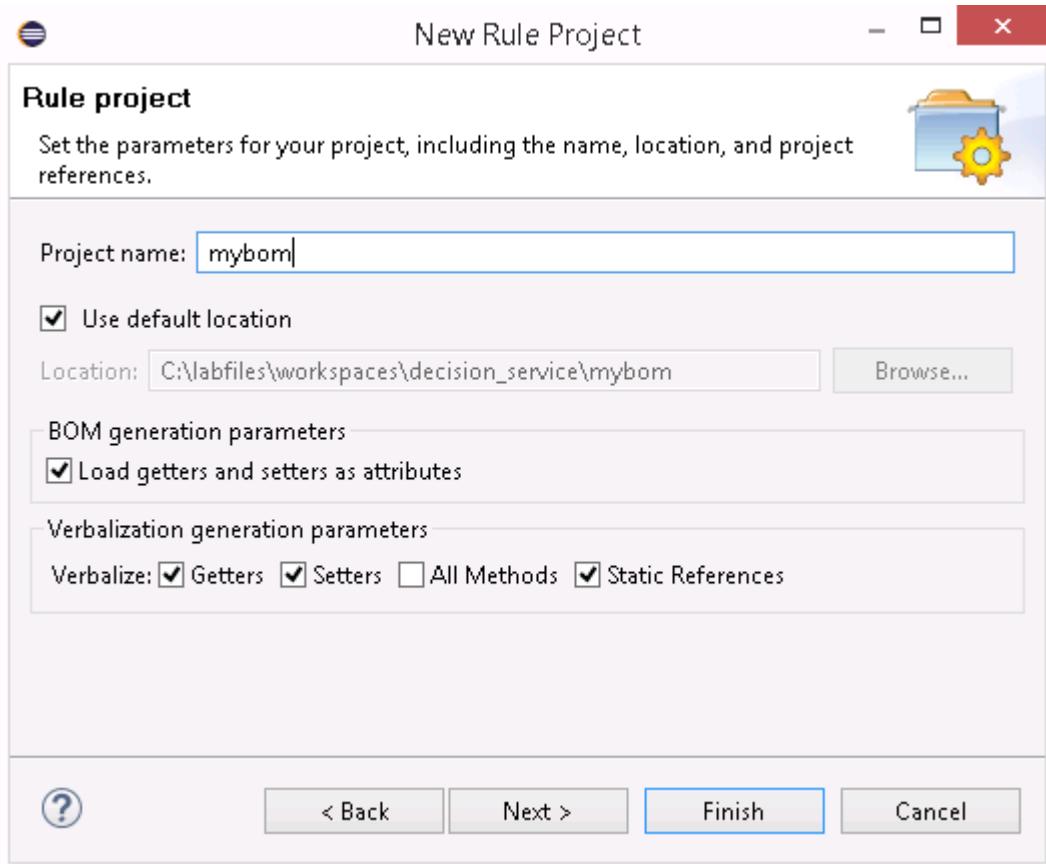
- 1. In the **Rule Project Map** view, go to the “Setup a Decision Service” part of the Decision Service Map, and click **Create standard rule project**.



2. In the New Rule Project window, click **Rule project with a BOM**, and click **Next**.



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



- 4. Click **Finish**.

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



### Information

From the development perspective, some of the first tasks when building a rule project include creating an executable version of the object model that is based on your class diagram and vocabulary requirements.

In this exercise, you create the business object model (BOM) after importing the execution object model (XOM), which is provided in the lab files.

You import the XOM and create the BOM in the next part of the exercise.

## Section 2. Importing the XOM and creating the BOM

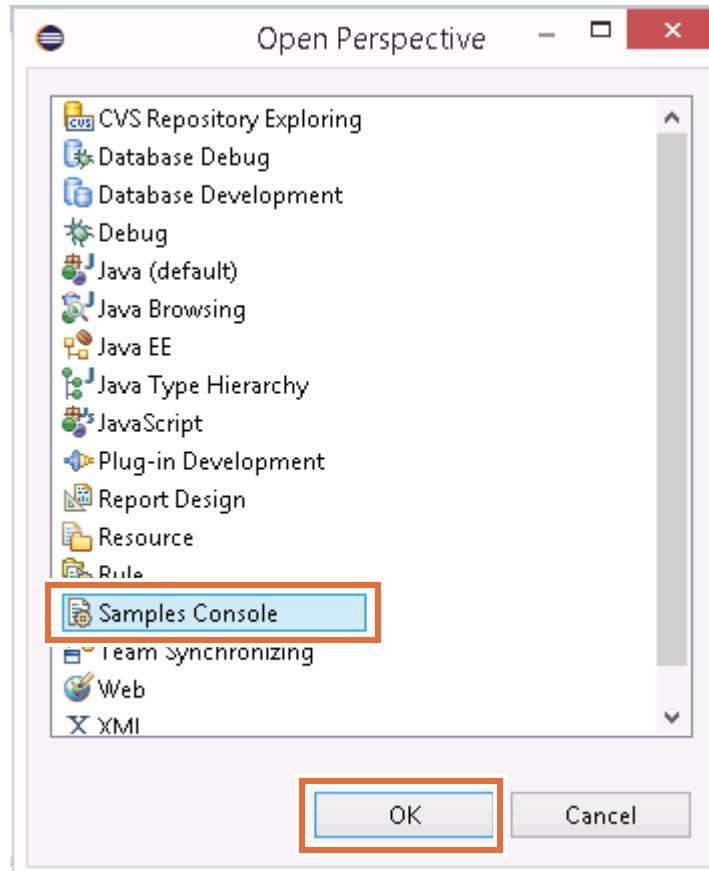
For this exercise, you work with an existing execution object model (or XOM). You import the Miniloan Java project into your workspace.

### 2.1. Importing the XOM

- \_\_\_ 1. Import the XOM project that is provided for this exercise by using the Samples console.
  - \_\_\_ a. In the toolbar, click **Open Perspective**.



- \_\_\_ b. In the Open perspective window, click **Samples Console**, and then click **OK**.



- \_\_\_ c. In the **Samples and Tutorials** tab, expand **Rule Designer > Training > Ex 04: Setting up a decision service**.
- \_\_\_ d. Under **start**, click **Import projects**.

When the import is complete, the **miniloan-xom** folder is in the Rule Explorer pane.

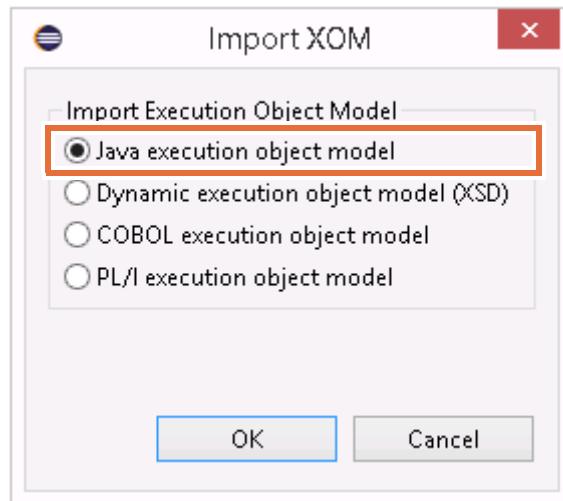
- \_\_\_ 2. Close the **Help** pane by clicking the **X**.



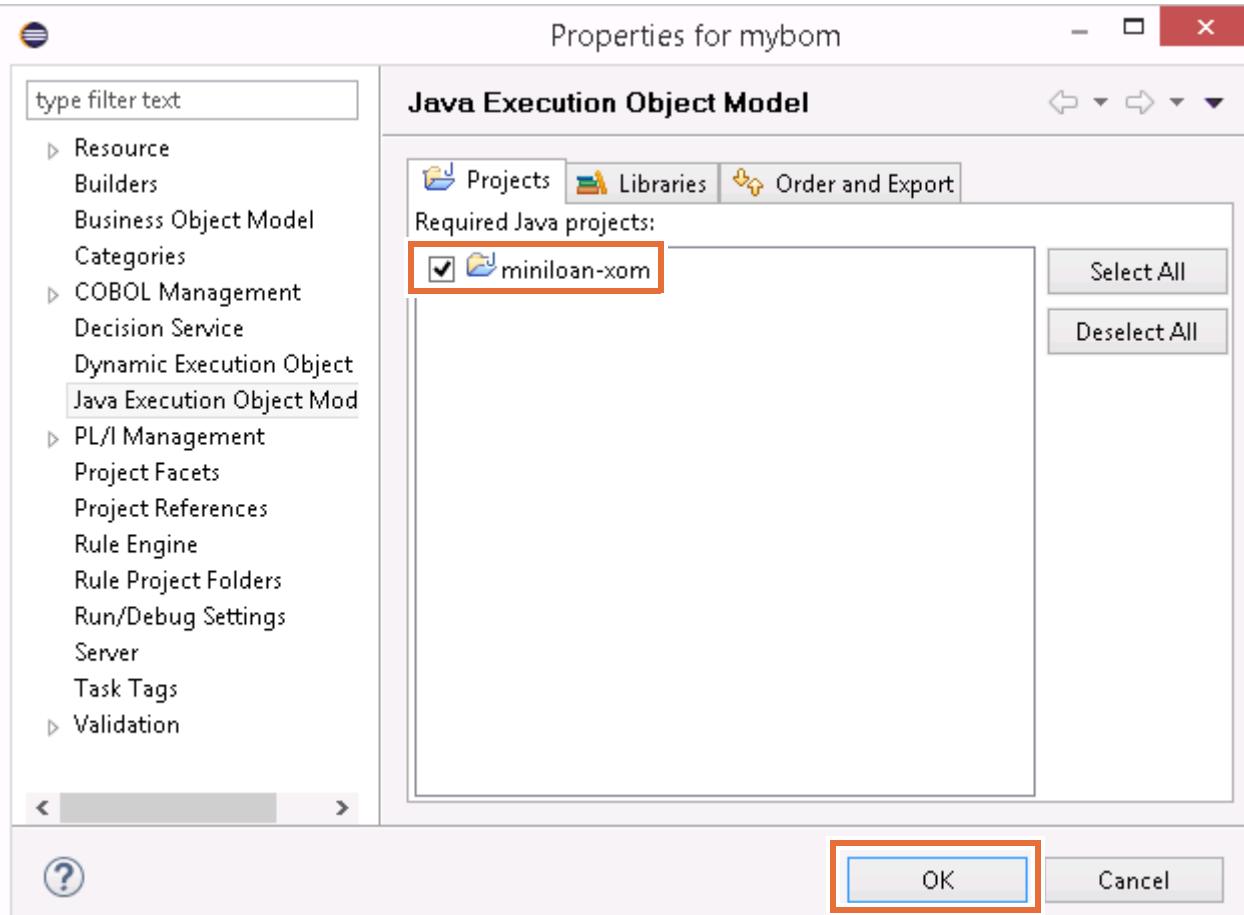
- \_\_\_ 3. In Rule Explorer, expand **miniloan-xom > src > miniloan**, and notice that the two objects, **Borrower** and **Loan**, are implemented in Java.
- \_\_\_ 4. In Rule Explorer, click the **mybom** project, and in the Decision Service Map, click **Import XOM**.



- \_\_\_ 5. In the Import XOM window, make sure that **Java execution object model** is selected, and click **OK**.



- 6. In the Rule Project Properties window, select **miniloan-xom** and click **OK**.



Your rule project can now access the Java code (XOM).

Next, you generate the business view of the XOM by creating the BOM.

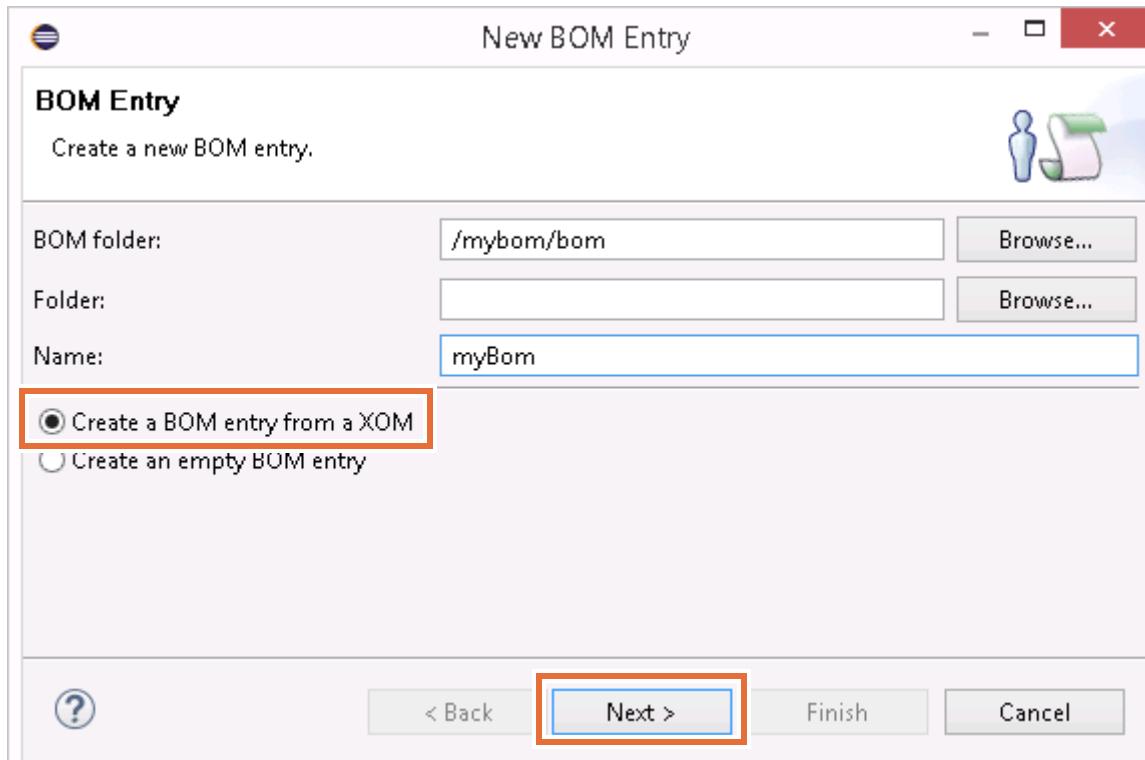
## 2.2. Creating the BOM

- 1. In Rule Explorer, make sure that the **mybom** project is selected.  
— 2. In the “Setup a Decision Service” part of the Decision Service Map, click **Create BOM**.



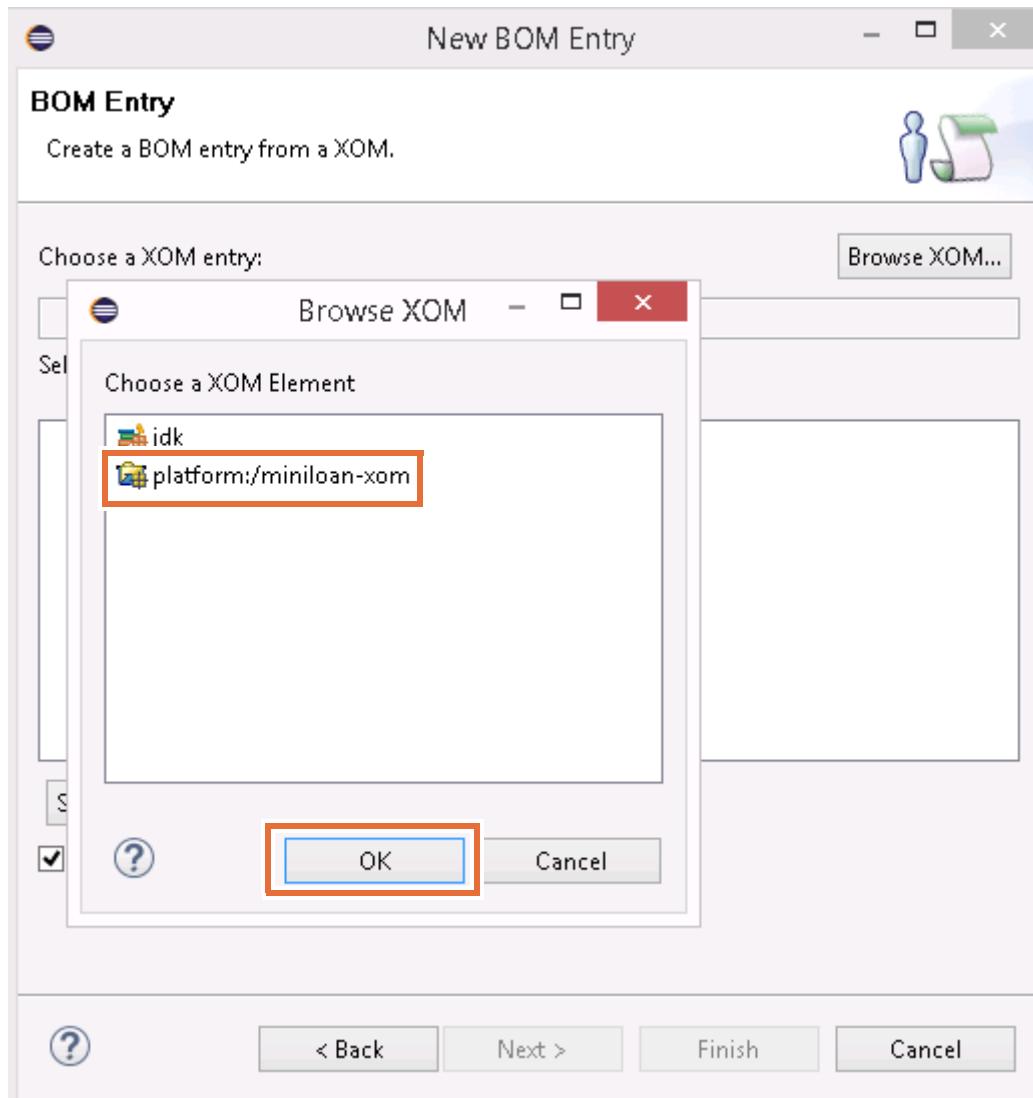
- 3. In the New BOM Entry wizard, complete the following steps.  
— a. In the **Name** field of the New BOM Entry wizard, you can leave the name at the default entry of `model` or type a new name, such as: `myBom`

- \_\_ b. Make sure that **Create a BOM entry from a XOM** is selected, and click **Next**.

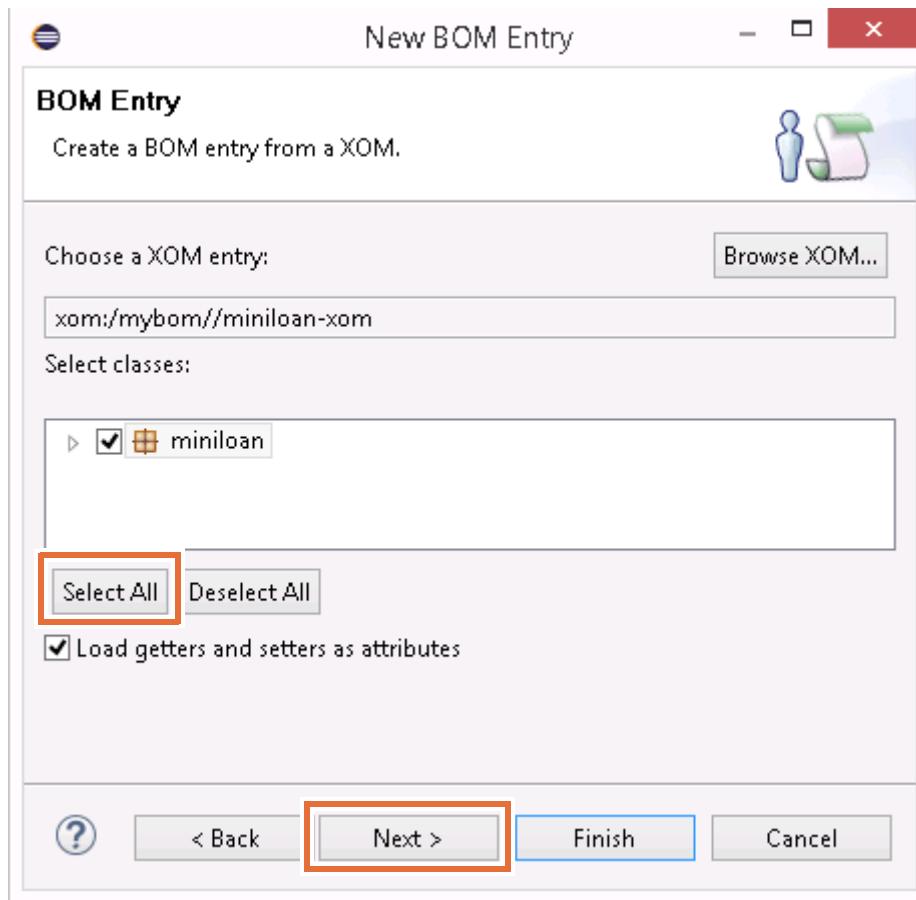


- \_\_ c. In the **Choose a XOM entry** section, click **Browse XOM**.

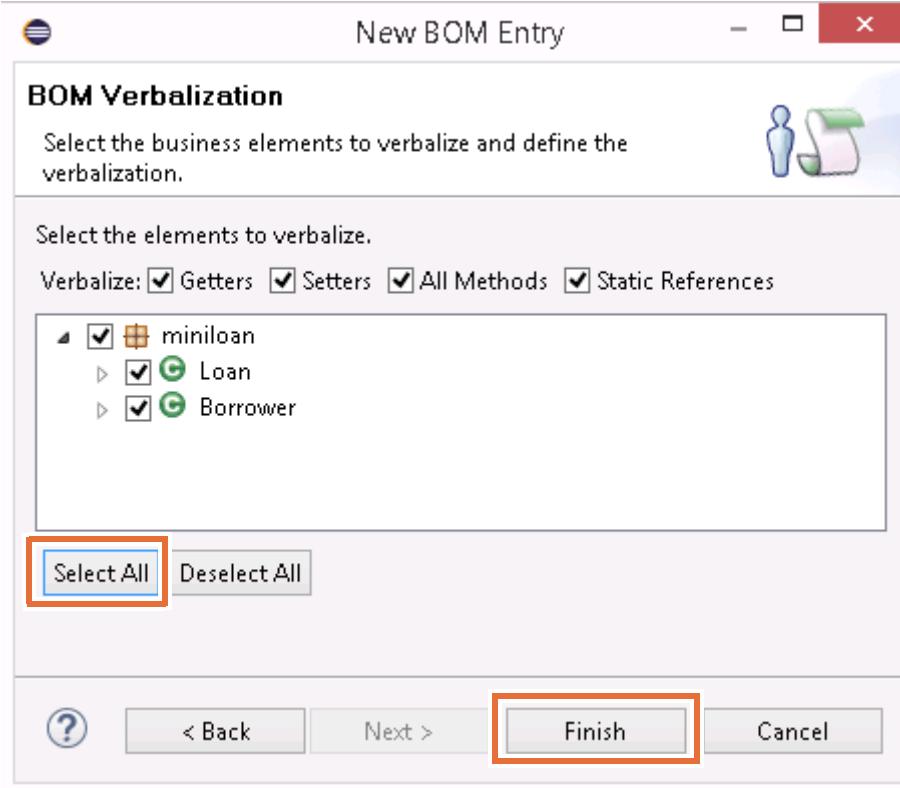
- \_\_ d. In the Browse XOM window, select `platform:/miniloan-xom` and click **OK**.



- \_\_ e. On the BOM Entry page, click **Select All** and click **Next**.

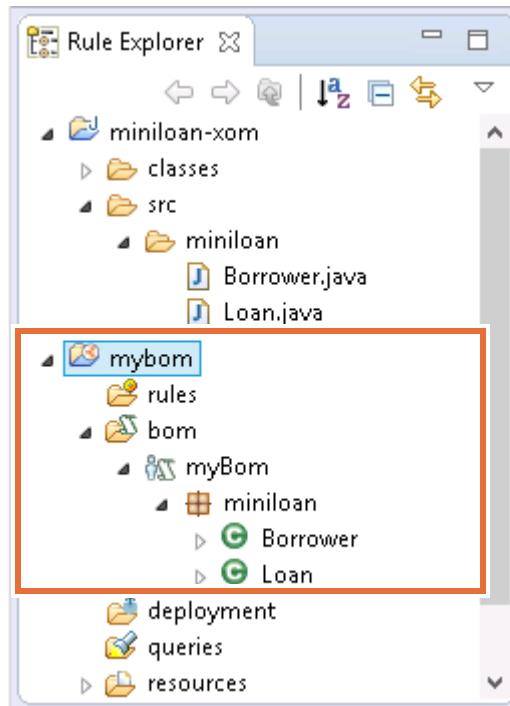


- \_\_ f. On the BOM Verbalization page, click **Select All** and click **Finish**.



The BOM is now created in your rule project.

4. Expand the **bom** folder to see your BOM with the `Borrower` and `Loan` members.



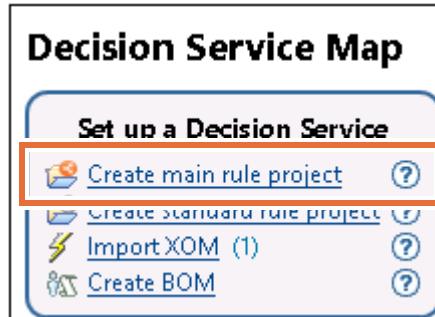
Your BOM includes a `Borrower` and a `Loan` member that correspond to the `Borrower` and `Loan` classes in the XOM.

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

The main rule project serves as the entry point to managing a decision service. It defines the name of the decision service that users see in Decision Center.

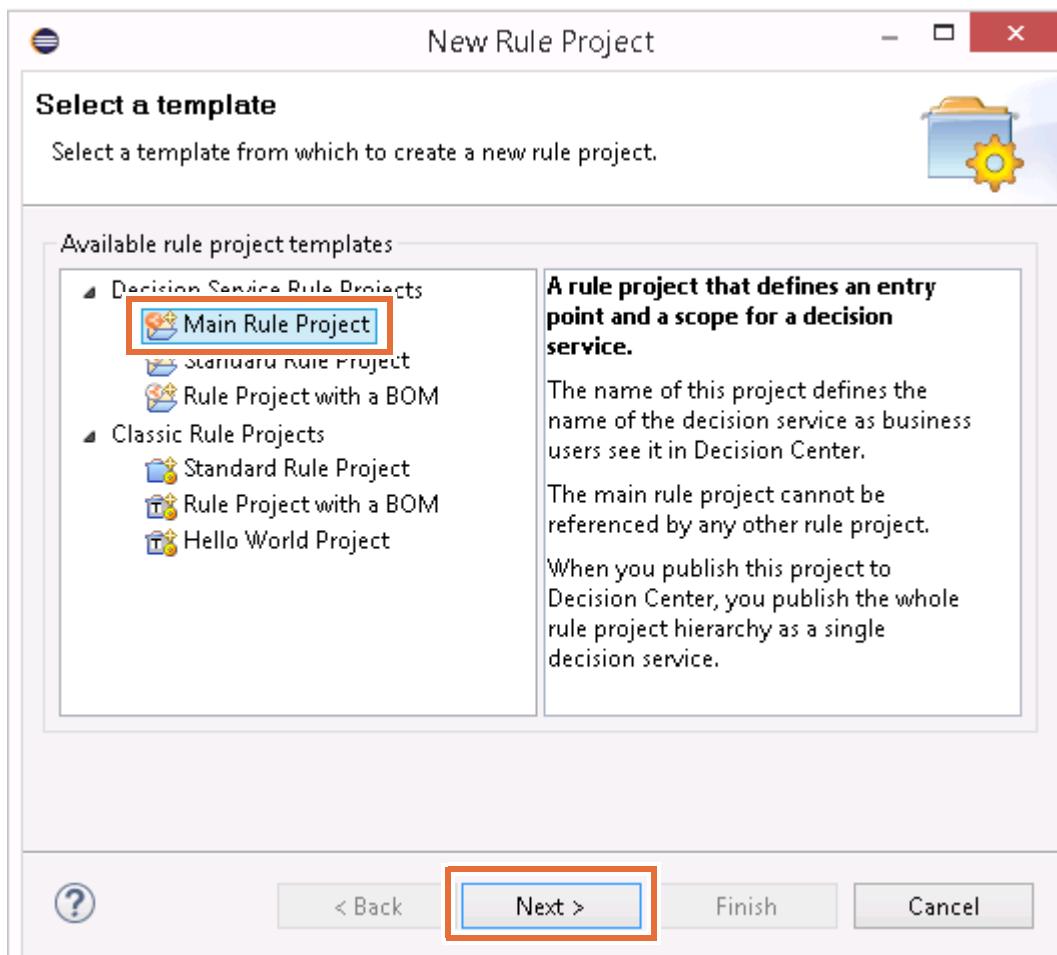
In this section of the exercise, you create a main rule project for your decision service.

- 1. In the “Setup a Decision Service” part of the Decision Service Map, click **Create main rule project**.

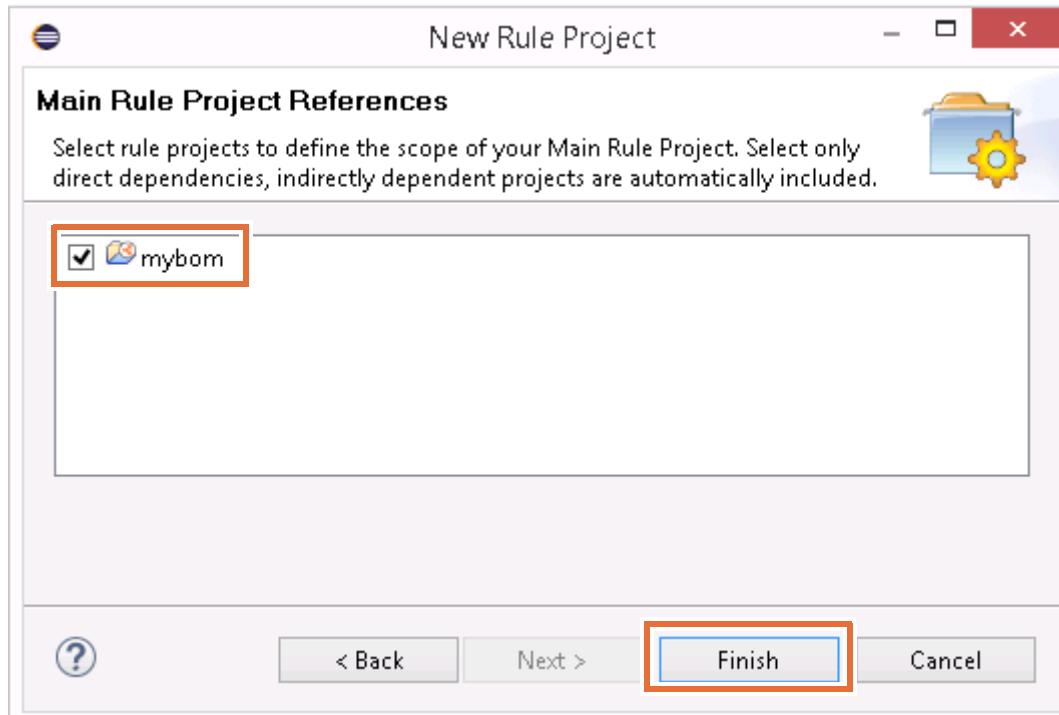


The New Rule Project wizard opens.

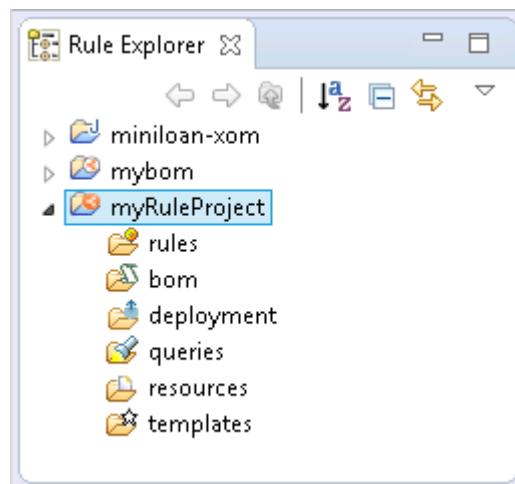
- 2. In the Decision Service Rule Projects section of the “Select a template” page, select **Main Rule Project** and click **Next**.



- \_\_\_ 3. In the **Project name** field, type a name for your project, such as: myRuleProject
- \_\_\_ 4. Click **Next**.
- \_\_\_ 5. On the Main Rule Project References page, select **mybom**, and click **Finish**.



The new project is now listed in the Rule Explorer.

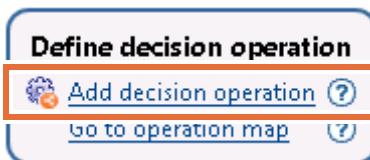


## Section 4. Creating and defining the decision operation

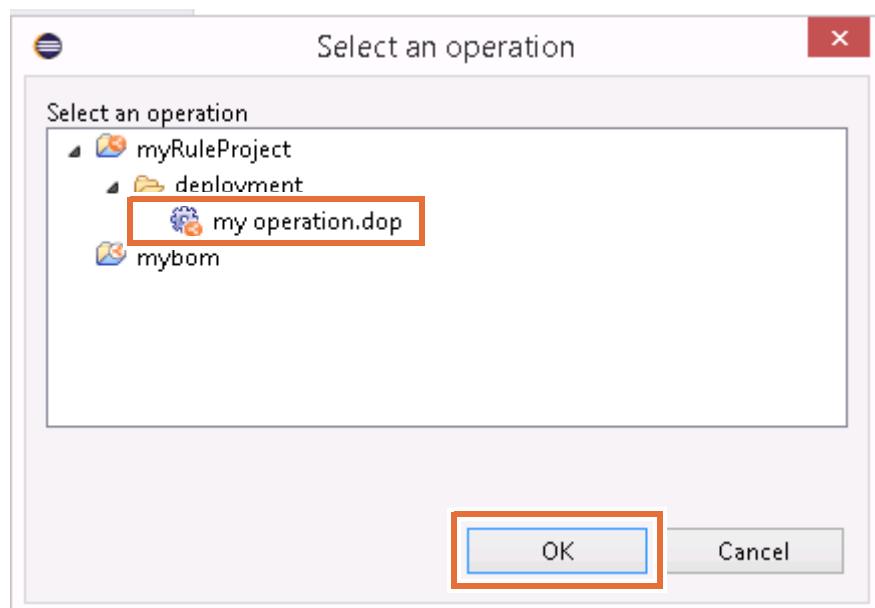
In this section, you work with the “Define decision operation” part of the Decision Service Map.

### 4.1. Creating the decision operation

- 1. Define the decision operation.
    - a. Make sure that the **myRuleProject** folder is selected.
    - b. In the “Define decision operation” part of the Decision Service Map, click **Add decision operation**.
  
  - c. In the New Decision Operation wizard, in the **Name** field, type: **my operation** and click **Next**.
  - d. Ensure **myRuleProject** is selected on the Source Rule Project page, and then click **Finish**.
- 2. In the “Define decision operation” part of the Decision Service Map view, click **Go to operation map**.

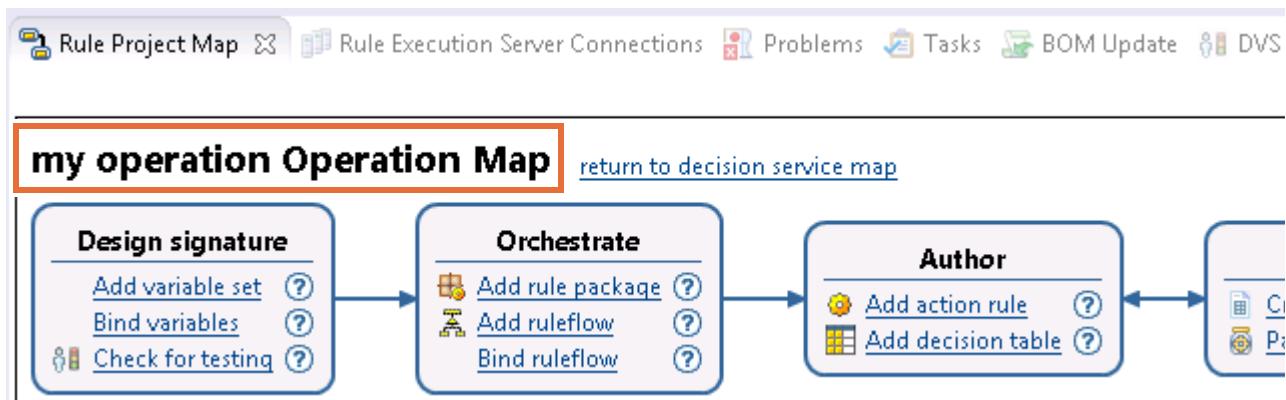


- 3. Select **myRuleProject > deployment > my operation.dop** and click **OK**.



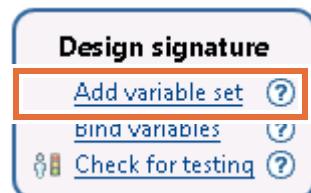
The Decision Service Map view switches to the Operation Map view. The Operation Map has the following parts:

- Design signature
- Orchestrate
- Author
- Test



## 4.2. Creating ruleset variables

- 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 variables`

- 3. Click **Finish**.

The Variable Set editor opens.

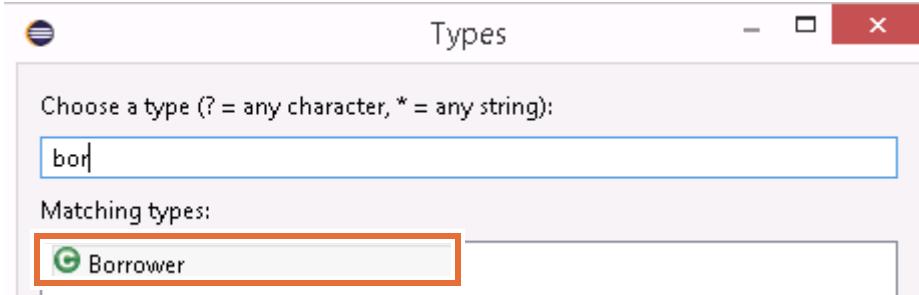
- 4. In the Variable Set editor, define the `borrower` variable.

- a. Click **Add**.

Name	Type	Verbalization	Initial Value
myVar	java.lang.String	myVar	

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

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



### Hint

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

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



### Hint

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

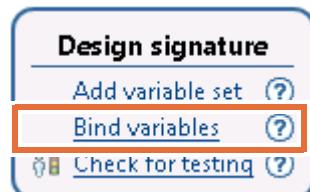
- 
- \_\_\_ 6. Save your work.

- 7. Close the variable set editor by clicking the X for that window.

Name	Type	Verbalization
borrower	miniloan.Borrower	the borrower
loan	miniloan.Loan	the loan

### 4.3. Binding the variables to ruleset parameters

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



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

- 2. Map the variables to the input and output parameters.

- a. Expand **my variables** in the **Eligible variables** section of the Decision Operation Signature editor to see the **borrower** and **loan** variables.
- b. Drag **borrower** to the **Input Parameters** table.
- c. Drag **loan** to the **Input-Output Parameters** table.

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

#### Input Parameters

Define the parameters required to call the execution.

Parameter name	Verbalization	Type
 <code>borrower</code>	the borrower	<code>miniloan.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>miniloan.Loan</code>

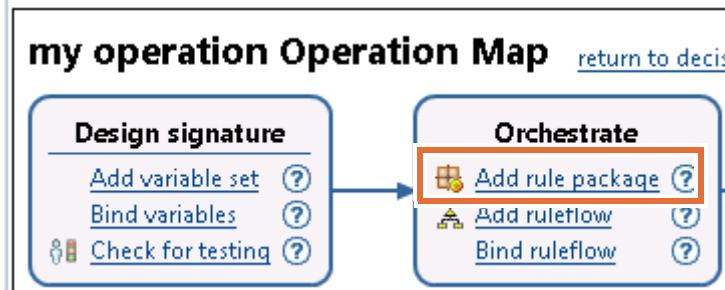
- \_\_\_ 3. Save your work, and close the `my` operation editor.

## Section 5. Orchestrating execution

In this part of the exercise, you create a ruleflow in your rule project.

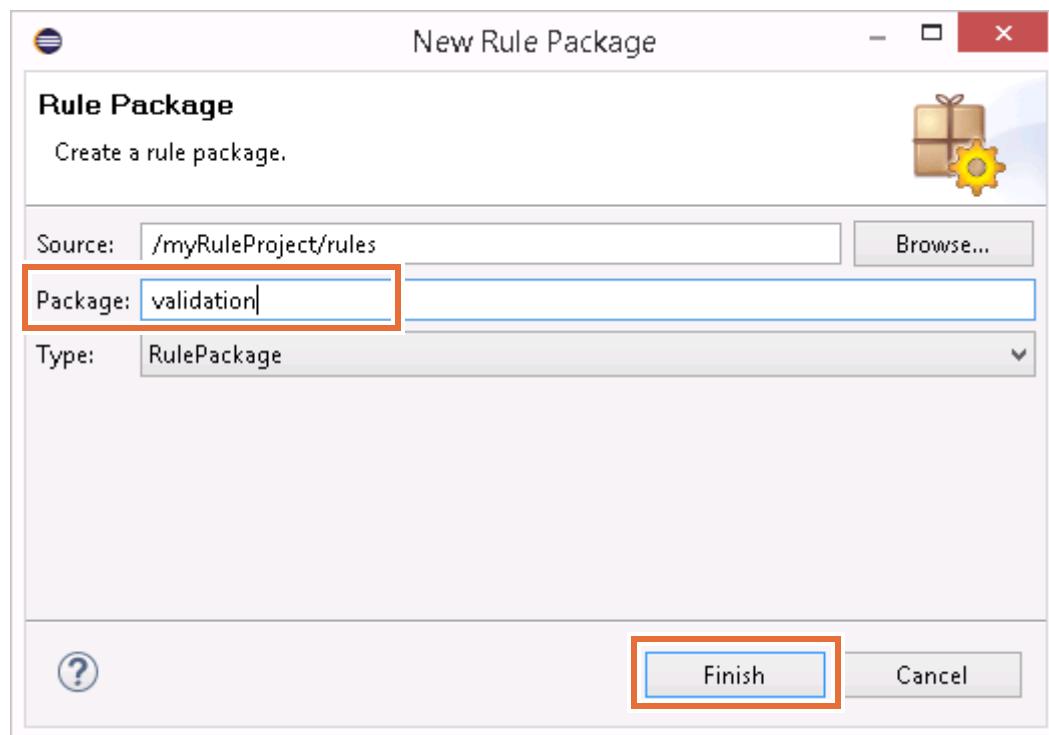
### 5.1. Creating rule packages

- 1. Create a rule package that is named: validation.
- a. In the Orchestrate part of the Operation Map, click **Add rule package**.



The New Rule Package wizard opens.

- b. In the **Package** field, type validation and click **Finish**.

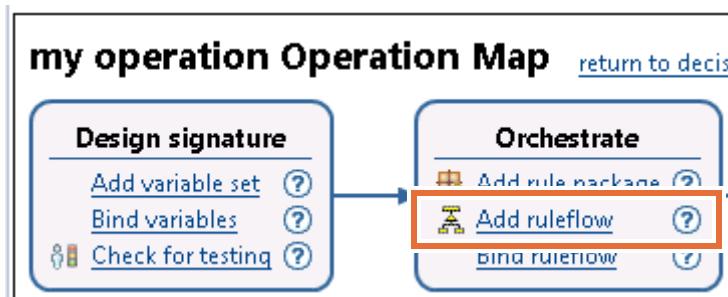


The new validation rule package is now listed in the Rule Explorer under **myRuleProject > rules**.

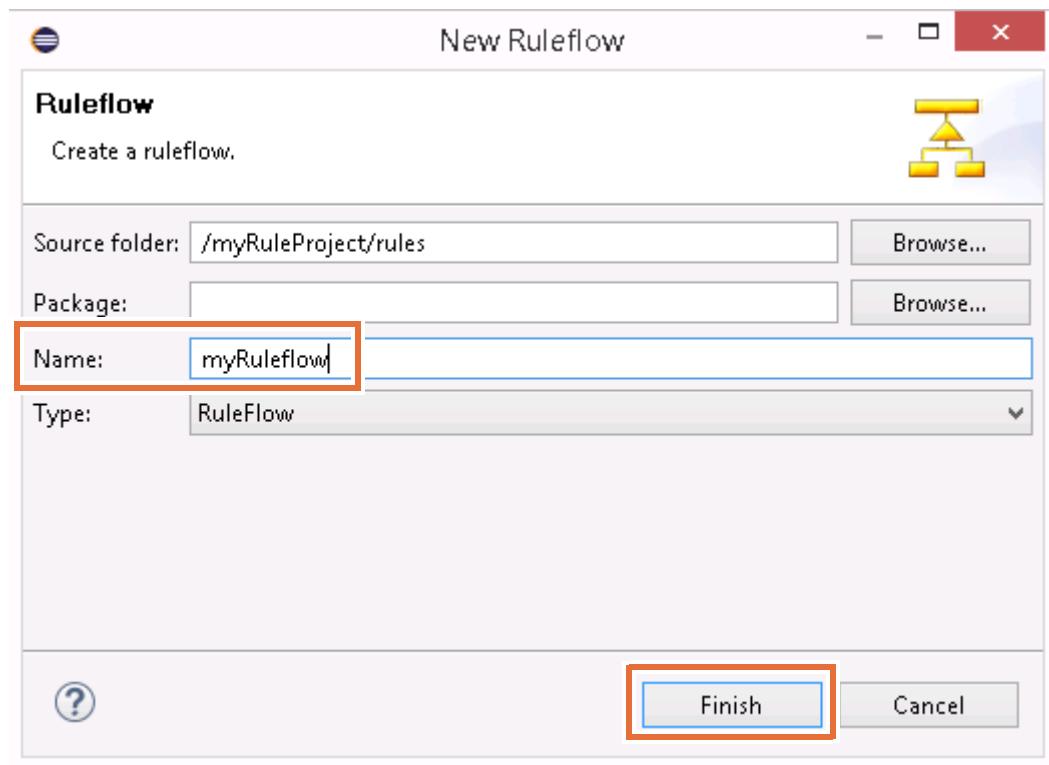
- 2. Repeat [Step 1](#) to create another rule package named: eligibility
- The new eligibility rule package is now listed in the **myRuleProject > rules** folder.
- Your rule project now contains two packages where you can store your rules.

## 5.2. Creating a ruleflow

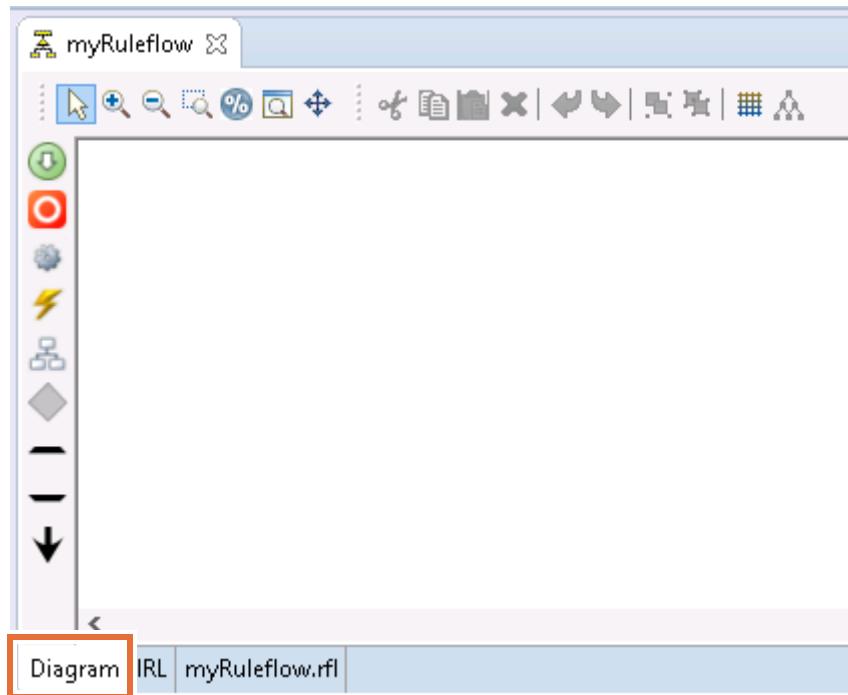
- 1. In the Orchestrate part of the Operation Map, click **Add ruleflow**.



- 2. In the **Name** field of the New Ruleflow wizard, type `myRuleflow` and click **Finish**.



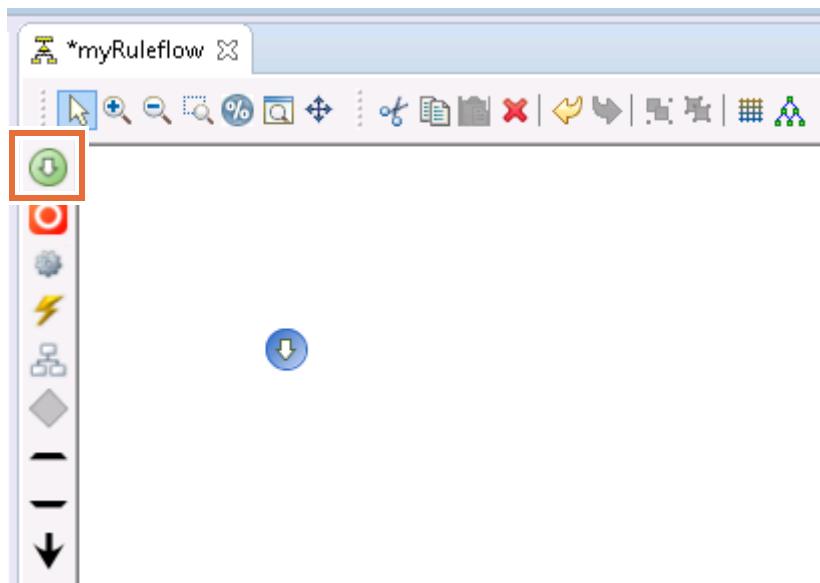
The **Diagram** tab of the Ruleflow editor opens.



In the Diagram view, you can specify graphically how tasks are chained together, and under what conditions they execute.

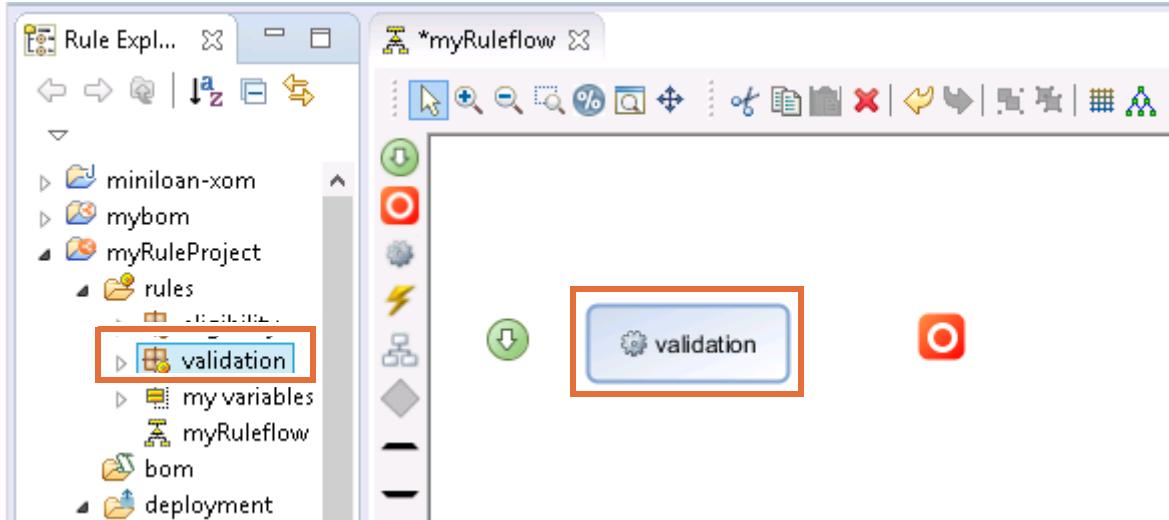
### 5.3. Creating the ruleflow diagram

- 1. Click **Create a start node**, and then click anywhere in the Ruleflow editor.



- 2. Click **Create an end node**, and then click anywhere in the Ruleflow editor.
- 3. Drag the **validation** rule package from the Rule Explorer and drop it into the Ruleflow editor.

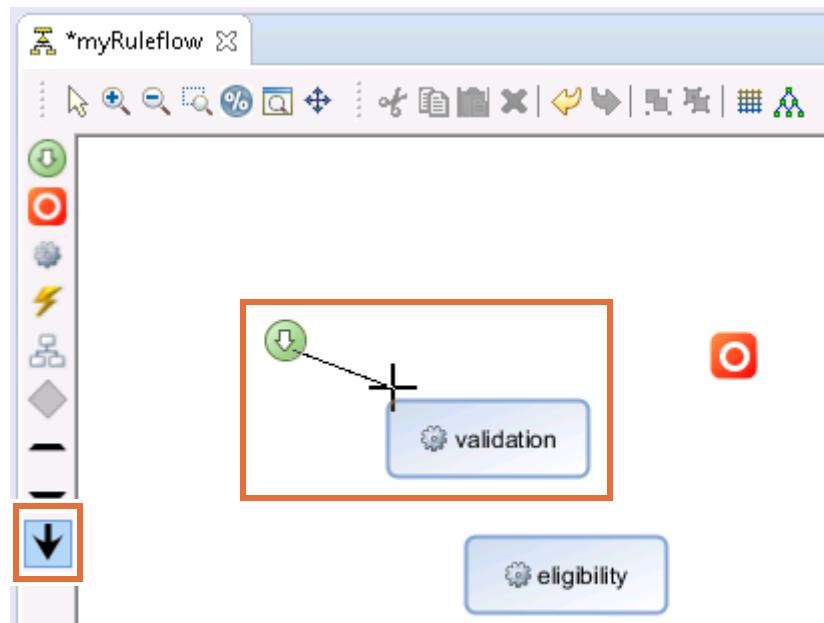
The **validation** rule package becomes a rule task in the ruleflow.



### Note

By dropping this package in the ruleflow, any rule that you create in the package becomes part of the execution, unless you specify otherwise.

- \_\_\_ 4. Drag the **eligibility** rule package into the Ruleflow editor.
- \_\_\_ 5. Click **Create a transition**, and click the start node, and then click the validation task.



- \_\_\_ 6. Create the following transitions (shown as arrows) by clicking the first item and then clicking the second:
  - validation task and eligibility task
  - eligibility task and end node

- validation task and end node

\_\_\_ 7. Click  **Create a transition** again to deactivate the transition tool.

\_\_\_ 8. Click  **Layout All Nodes** to format the ruleflow diagram.

---



### Information

You can ignore the error indicators  on the transitions. The errors on the transitions indicate that the conditions are missing. You add the conditions in the next steps.

---

\_\_\_ 9. Save your work (Ctrl+S).

## 5.4. Adding transition conditions

\_\_\_ 1. Click the transition from **validation** to **eligibility**.

---

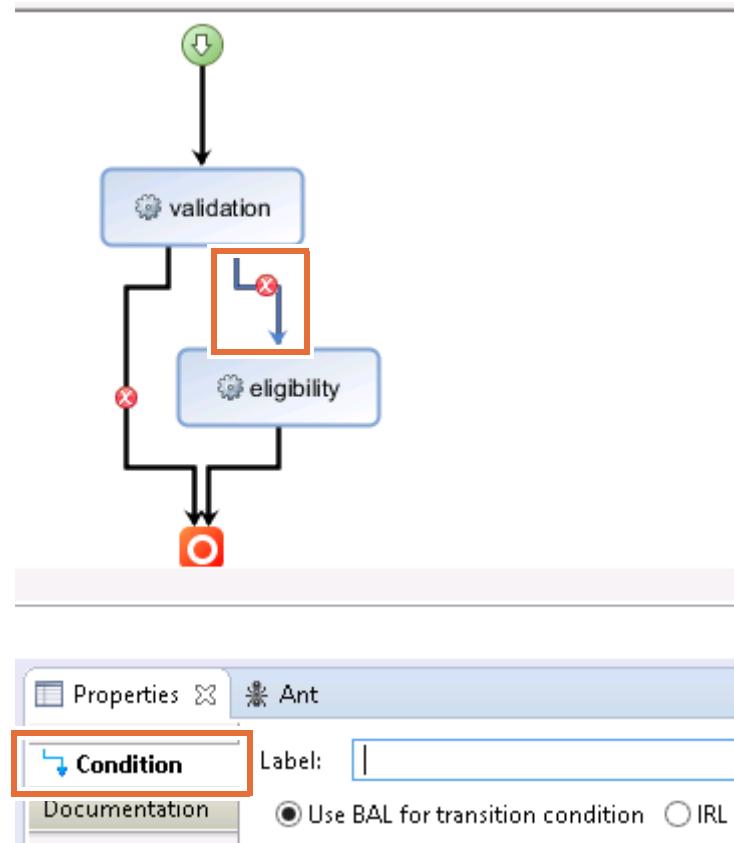


### Hint

If you have trouble with editing the transition, you can click the arrowhead.

---

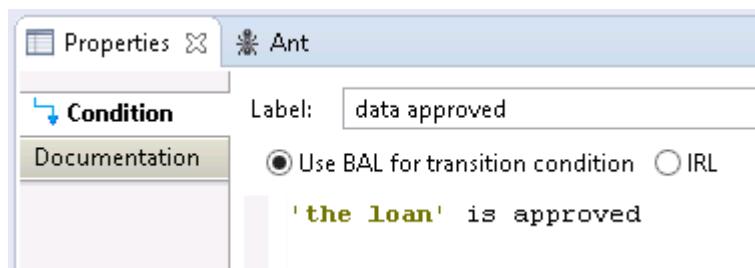
The **Condition** field opens in the Properties view in the lower-right part of Rule Designer.



### Hint

The Properties view is in the lower-right corner of the window. If necessary, you can maximize or expand the Properties view to see more of it.

2. In the Properties view, enter the following information.
  - a. In the **Label** field, type: `data approved`
  - b. Make sure that **Use BAL for transition condition** is selected, and type the following condition (including the single quotation marks):  
`'the loan' is approved`





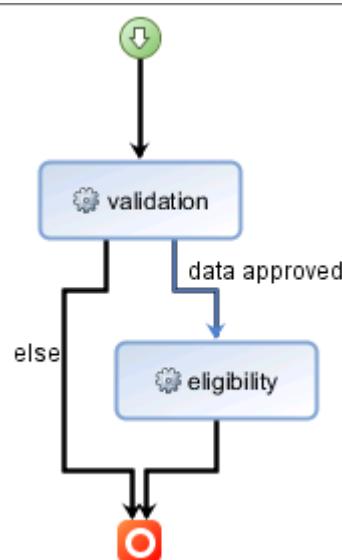
## Information

**BAL** stands for Business Action Language. BAL is a business rule language that uses an intuitive and natural language-like syntax for writing business rules. Using BAL makes it straightforward to author business rules without coding.

The other option for ruleflow properties is **IRL**, or ILOG Rule Language. IRL is an executable rule language that contains a set of keywords, and it has its own syntax to structure each part of the rule. You use IRL in this exercise to define the final action in the end node properties.

Rules that are written in BAL are automatically translated into IRL during execution.

After you set the condition for the transition between **validation** and **eligibility**, the transition from **validation** to the end node is automatically set to **else**.



- \_\_\_ 3. Save your work and close the Ruleflow editor.

## Section 6. Publishing and synchronizing the rule project

To make the rule project available to business users, you connect to Decision Center from Rule Designer, and then publish the project to Decision Center.

After business users work with the rules, they might require updates to rule artifacts that can be made only in Rule Designer, such as changes to the vocabulary or to the ruleflow. Before making updates in Rule Designer, developers first update their version of the rule project with the most recent changes from Decision Center.

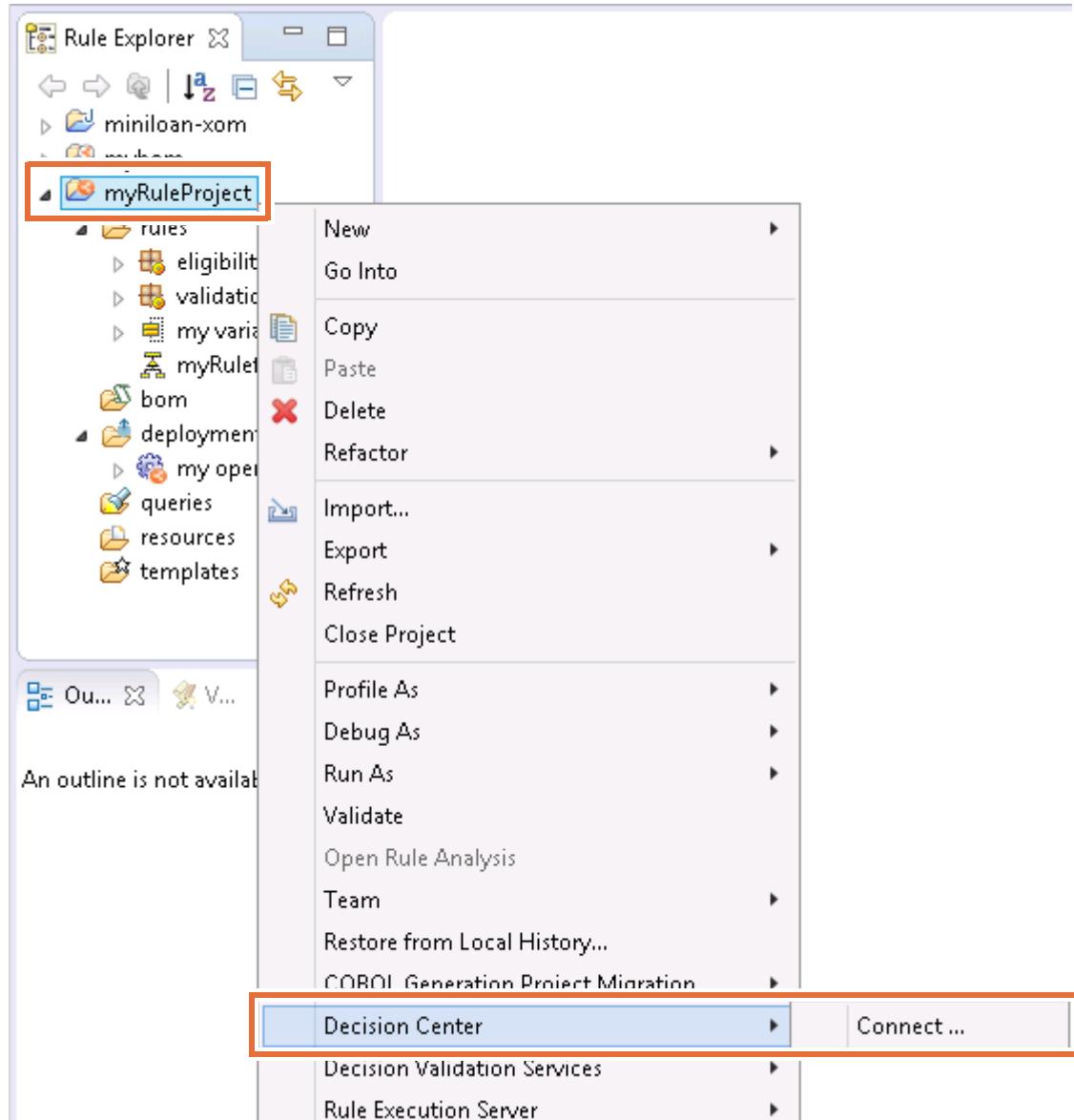
In this exercise, you publish your project from Rule Designer to Decision Center, edit rules in Decision Center, and synchronize those changes back to Rule Designer.

### 6.1. Publishing the rule project to Decision Center

- 1. If the sample server is not running, start it by clicking **Start**, then clicking the **Start Sample Server** shortcut.

You can also start the server from the Apps list (in the taskbar, click **Start**, then click the down arrow) by going to the IBM section and clicking **Start Sample Server**.

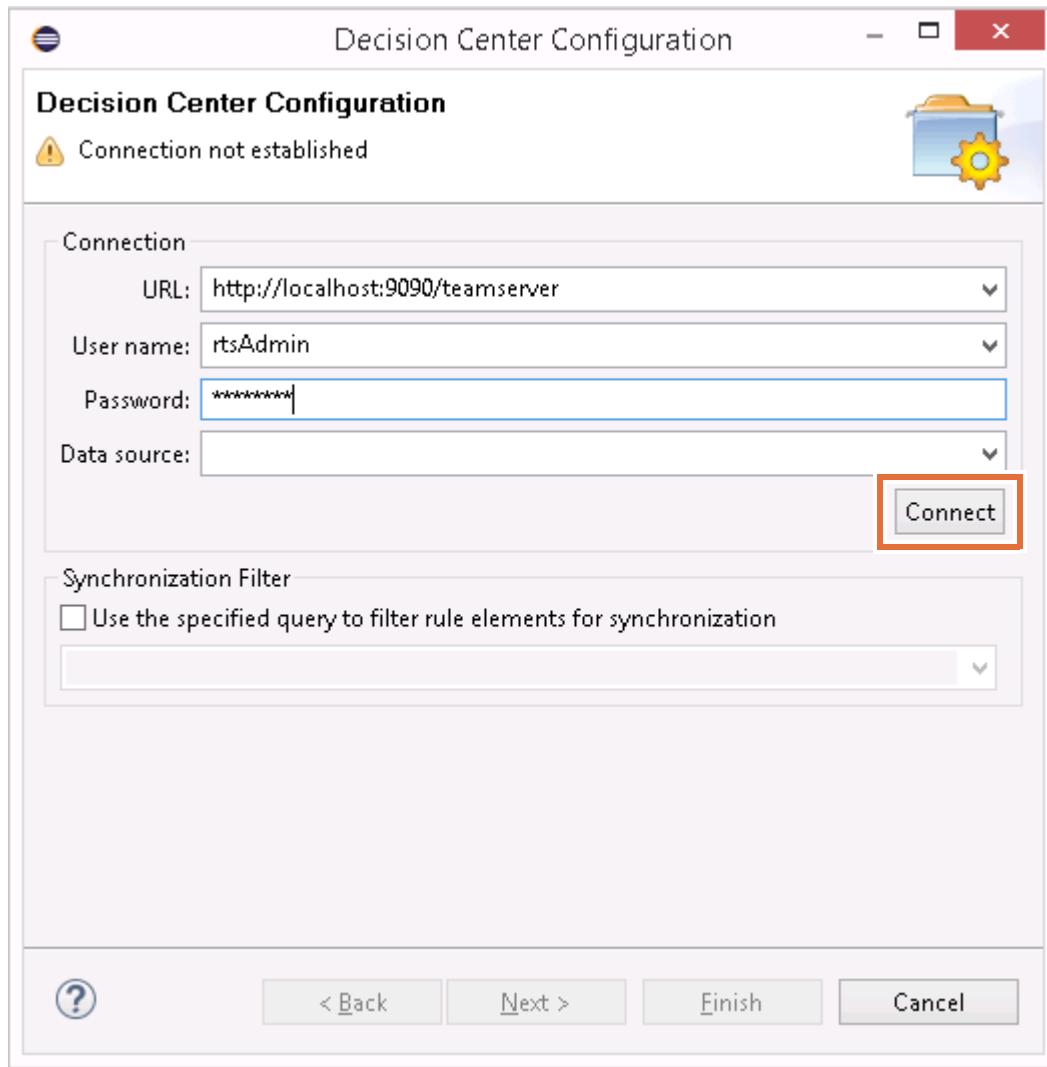
- 2. In Rule Explorer, right-click **myRuleProject** and click **Decision Center > Connect**.



The Decision Center Configuration window opens.

- 3. Complete these Decision Center Configuration page fields with the following information:
- **URL:** `http://localhost:9090/teamserver`
  - **User name:** `rtsAdmin`
  - **Password:** `rtsAdmin`

- 4. Click **Connect**.



### Troubleshooting

If you get a failure to connect message, make sure that the sample server is started as described in step 1 of this section.

- 5. After the connection is established, click **Next** and click **Next** again to get to the Decision Service Dependent Projects page, where you see that **mybom** is selected.
- 6. Click **Finish**.

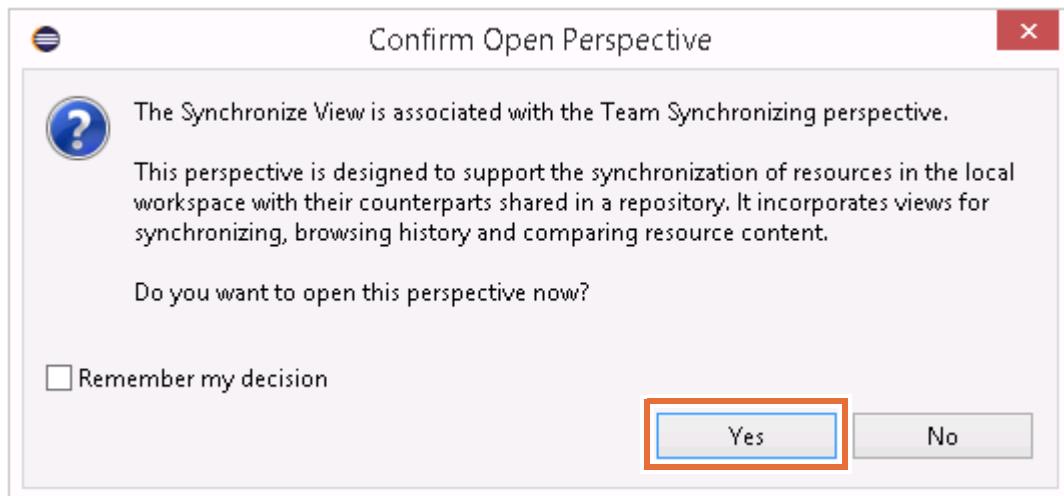


## Troubleshooting

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



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



8. In the Synchronize Complete window, click **OK** to close it.

The Synchronize view opens in Rule Designer and indicates that no changes are detected.

Your rule project is now published to Decision Center as a decision service. Next, you view your rule project in Decision Center.



### Note

Do not close Rule Designer.

## 6.2. Viewing the published project in the Decision Center Business console

1. Open the Decision Center Business console by entering the following URL in a browser, or click **Start** and then click the **Decision Center Business console** shortcut:  
`http://localhost:9090/decisioncenter`

- 2. Sign in to the Business console by using the following details:

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



## Information

You signed in as an administrator (rtsAdmin) when you published the project to Decision Center, but here you sign in as a business user.

- 3. On the Business console **Library** tab, you can see the project that you published.

The screenshot shows the Decision Center interface with the Library tab selected (highlighted with a red box). Below the tabs, the title "Decision Services" is displayed. A search bar contains "Date | Name". The main area shows a project card for "myRuleProject" (marked with a star and a green "NEW" badge). The card includes the creation date "Created by rtsAdmin on May 1, 2017". Below the card, a section titled "Recently updated branches:" lists "main" (also highlighted with a red box), which was also created on May 1, 2017.

- 4. Click the **myRuleProject** tab to expand the project details, and then click **main**.

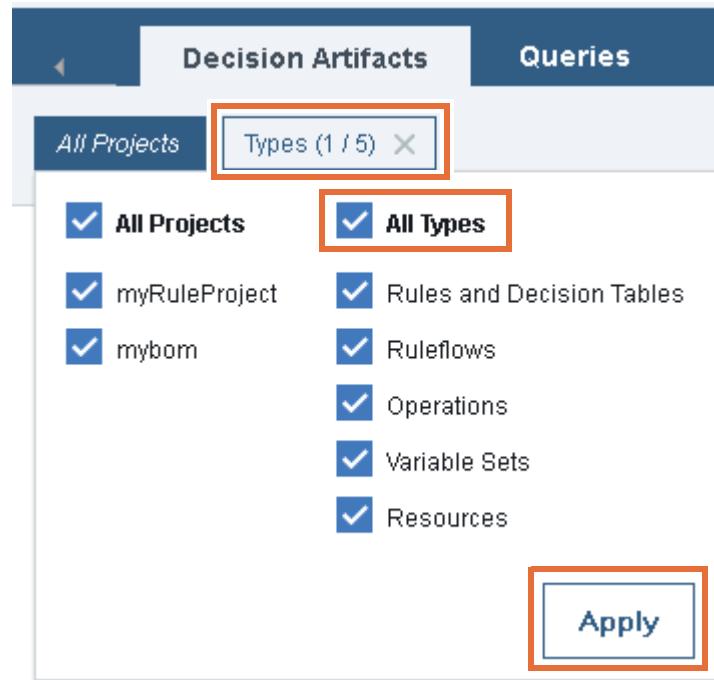
The screenshot shows the expanded project details for "myRuleProject". The "main" branch is selected (highlighted with a red box). The "main" branch was created by rtsAdmin on May 1, 2017. A green "NEW" badge is visible in the top right corner of the project card.

- 5. View the decision artifacts that are associated with your decision service.

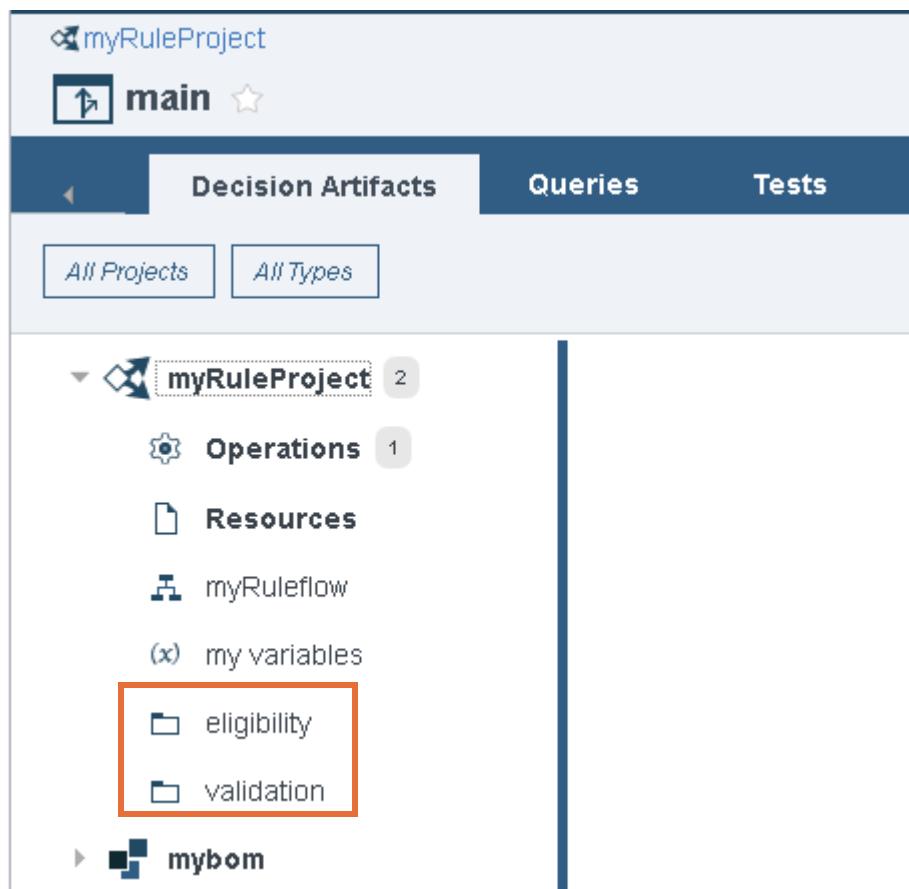
- a. When the project opens, click the **Decision Artifacts** tab.

The screenshot shows the "myRuleProject" project details page with the "Decision Artifacts" tab selected (highlighted with a red box). Other tabs shown include "Queries" and "Tests".

- \_\_ b. View all decision artifact types by clicking the **Types** menu, selecting **All Types**, and clicking **Apply**.



- \_\_ c. Expand **myRuleProject** to see the decision artifacts in your decision service, such as the **eligibility** and **validation** folders.



### 6.3. Adding a rule

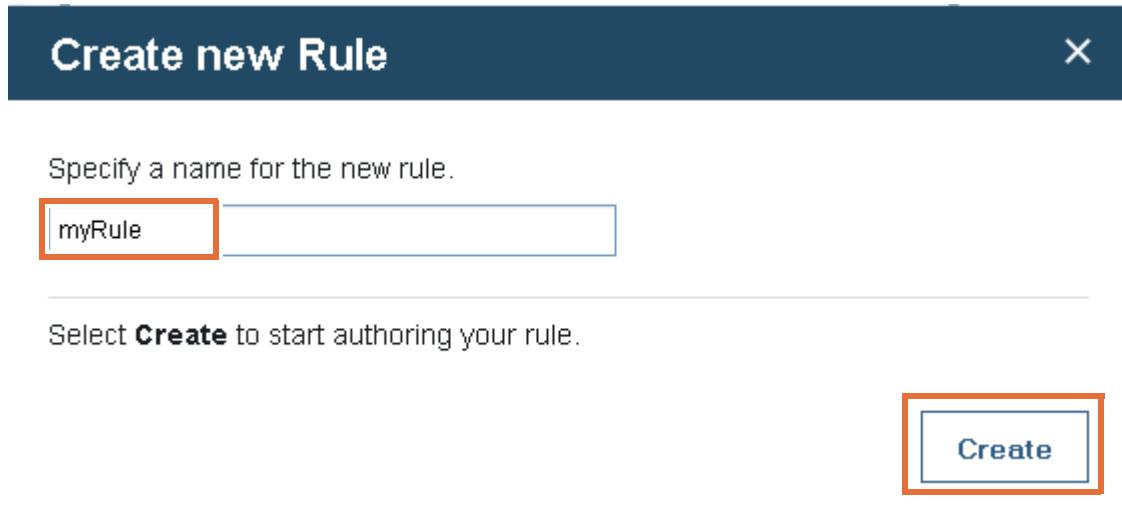
- 1. Click the **eligibility** folder to make it active in the center pane.

The screenshot shows the 'Decision Artifacts' tab selected in the top navigation bar. Below it, two buttons are visible: 'All Projects' and 'All Types'. The main pane displays a project structure under 'myRuleProject'. A folder named 'eligibility' is highlighted with a red box. To the right, a list of items is shown with a header 'Name' and columns for 'Last Changed By' and 'Last Changed On'. A 'Filter' input field is present. The message 'There are no items to display' is centered at the bottom of the list area.

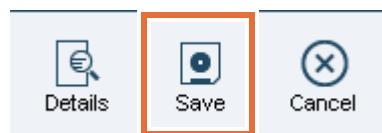
- 2. In the center pane, click the **create an artifact** icon (plus sign), and then click **New Rule**.

This screenshot shows the 'eligibility' center pane. The 'New Rule' option in the toolbar is highlighted with a red box. The same project structure and list view are visible as in the previous screenshot, with the message 'There are no items to display' at the bottom.

- \_\_\_ 3. In the “Create new Rule” window, enter `myRule` as the name, and click **Create**.



- \_\_\_ 4. In the rule editor, click **Save**. (You might have to click **Save** again.)



- \_\_\_ 5. When the Create New Version window opens, click **Create New Version**.



### Note

After saving the rule, you see the error message that the rule is incomplete. For this exercise, you can ignore the error. You work more extensively with Decision Center in later units.

- \_\_\_ 6. Log out of the console by clicking `rtsUser1` in the upper-right corner of the web browser window, and clicking **Log Out**.

- \_\_\_ 7. Close the browser window.

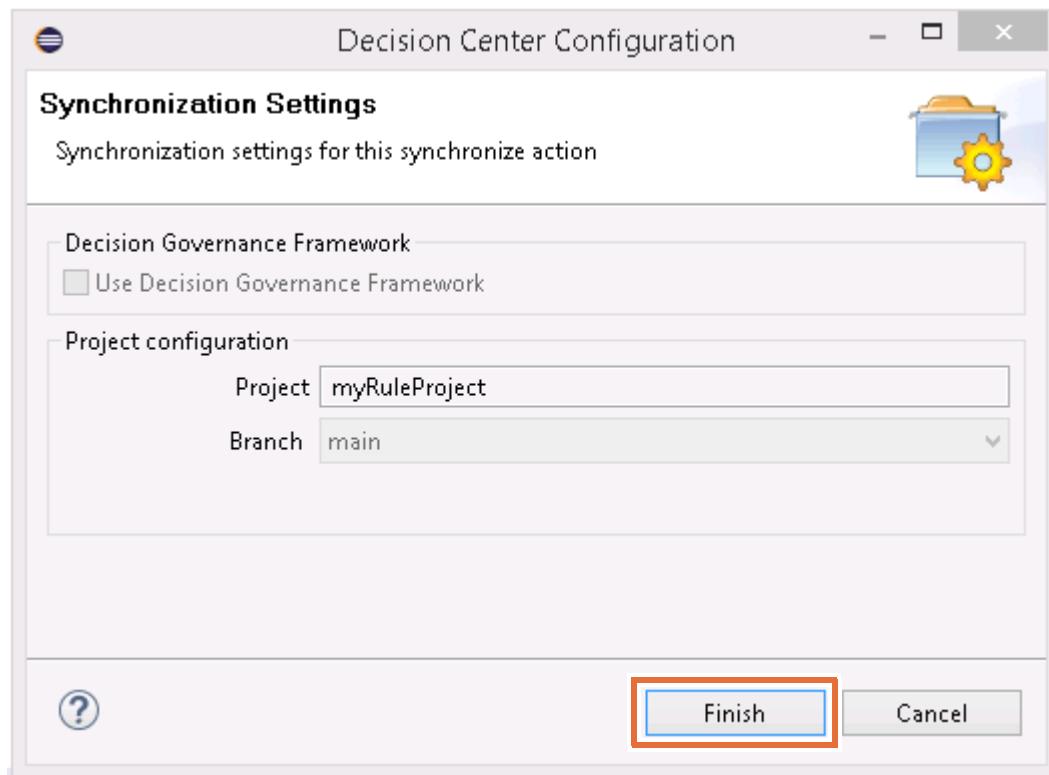
## 6.4. Updating the rules in Rule Designer

- \_\_\_ 1. Back in Rule Designer, switch to the Rule perspective.

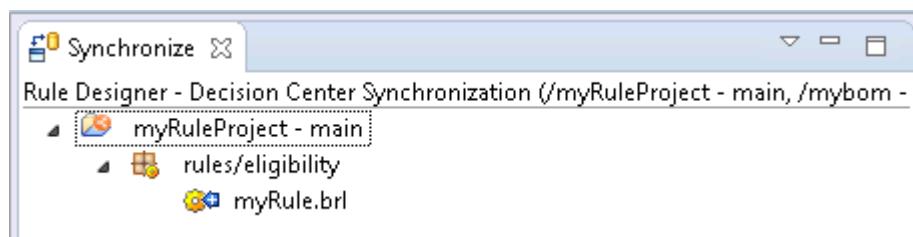


- \_\_\_ 2. Synchronize **myRuleProject** with Decision Center.

- \_\_\_ a. Right-click **myRuleProject** and click **Decision Center > Synchronize with Decision Center**.
- \_\_\_ b. In the Decision Center Configuration window, click **Finish**.

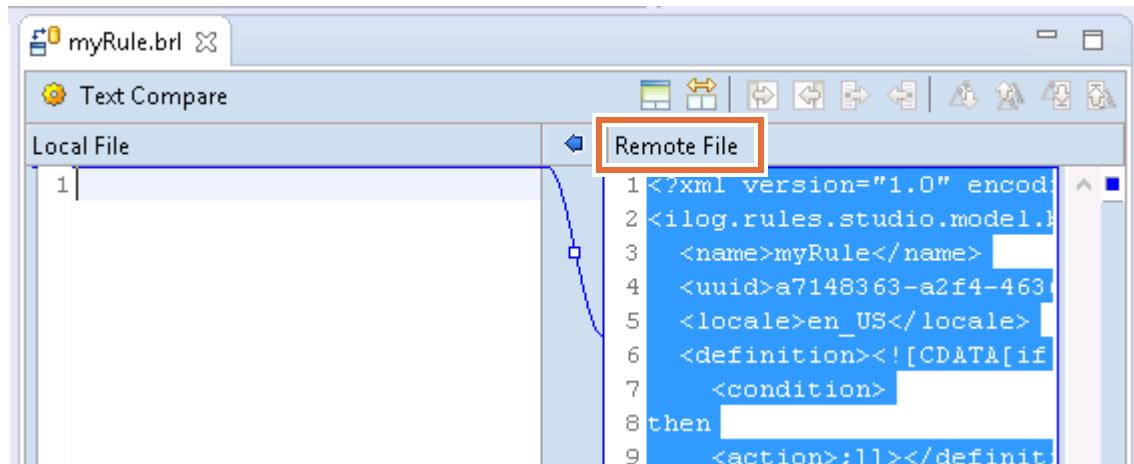


- \_\_\_ c. Click **Yes** when prompted to switch to the Synchronize perspective.
- \_\_\_ 3. In the Synchronize view, expand **myRuleProject - main > rules/eligibility** to see your new rule.



- \_\_\_ 4. Double-click **myRule.brl** in the Synchronize view to open the Compare editor.

Your new rule is shown only in the Remote File section.



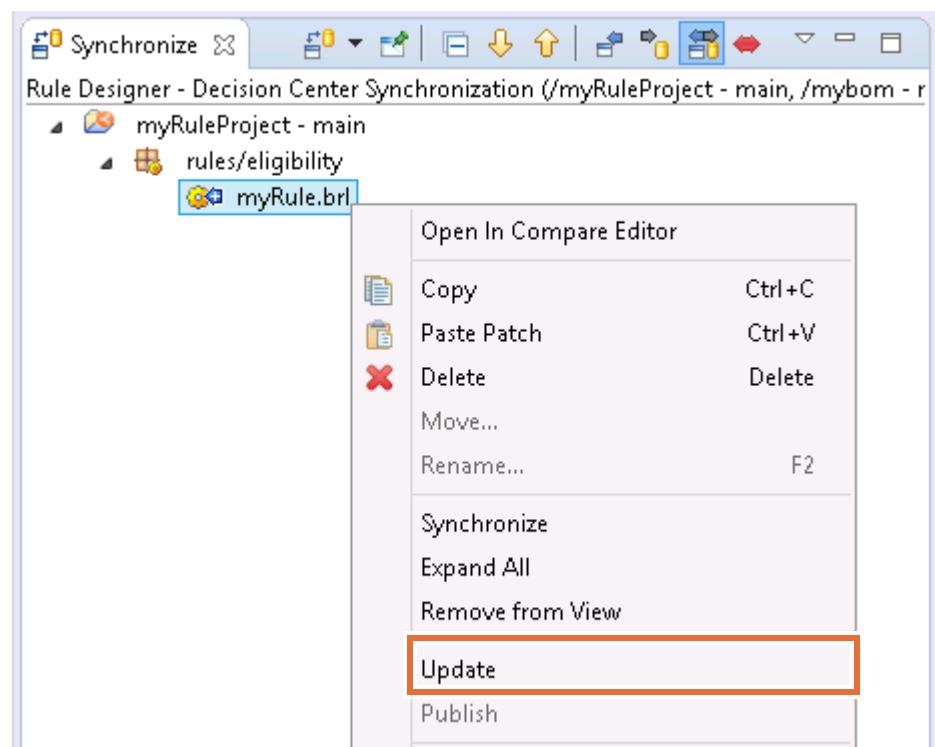
The screenshot shows the 'Text Compare' tool with two tabs: 'Local File' and 'Remote File'. The 'Remote File' tab is highlighted with a red box. The 'Local File' tab has a blue border. The code in the 'Remote File' tab is as follows:

```

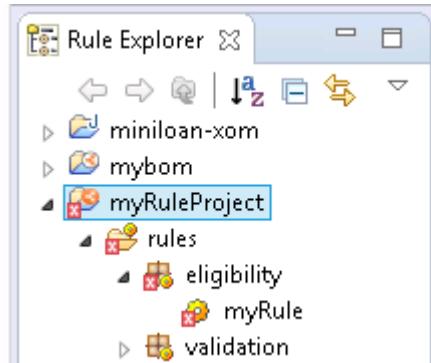
1 <?xml version="1.0" encoding="UTF-8"?
2 <ilog.rules.studio.model.1
3   <name>myRule</name>
4   <uuid>a7148363-a2f4-4631-80d9-1a2a2a2a2a2a</uuid>
5   <locale>en_US</locale>
6   <definition><![CDATA[if
7     <condition>
8       then
9         <action>;]]></definition>

```

- 5. Right-click **myRule.brl** and click **Update** to get the new rule from Decision Center into Rule Designer.



- 6. Return to the Rule Perspective, and notice that **myRule** is now listed in the **eligibility** folder in Rule Explorer.

**Note**

You can ignore the error warnings for **myRule**. You see the error warnings because **myRule** is an empty (incomplete) rule.

## 6.5. Deleting the project from Decision Center

- 1. Open the Decision Center Enterprise console by entering the following URL in a browser, or double-clicking the **Decision Center Enterprise console** desktop shortcut:

`http://localhost:9090/teamserver`

**Cloud**

### For IBM ODM on Cloud users

Make sure that you use the URL that points to your instance of Decision Center Enterprise console instead of `localhost`.

- 2. Sign in to the Decision Center Enterprise console by using the following details:
- **Username:** `rtsAdmin`
  - **Password:** `rtsAdmin`
- 3. On the **Home** tab, make sure that **Work on a decision service** is selected.

- 4. From the **Decision service in use** list, select **myRuleProject** and click the **Configure** tab.

The screenshot shows the IBM Decision Center interface. At the top, there is a navigation bar with tabs: Home, Explore, Compose, Analyze, Project, and Configure. The 'Configure' tab is highlighted with a red box. Below the navigation bar, the text 'Welcome to the Decision Center Home Page' is displayed. There are two main sections for selecting work types:

- Work on a rule project  
Project in use: loanvalidation-rules  
Branch in use: <none>  
Current action: Work on branch
- Work on a decision service  
Decision service in use: myRuleProject  
Branch in use: main  
Current action: Work on branch

The 'Work on a decision service' section is also highlighted with a red box.

- 5. In the **Administration** section, click **Erase Current Decision Service**.  
— 6. Click **Yes** when prompted to confirm deletion.  
— 7. Sign out of the console and close the browser.

**End of exercise**

## Exercise review and wrap-up

This exercise looked at how to set up the authoring environment in Rule Designer. You created a decision service, defined a decision operation, and created a ruleflow. You then published the decision service to Decision Center, and learned how to synchronize Rule Designer with Decision Center. Finally, you learned how to delete a decision service from Decision Center.

# Exercise 5. Working with the BOM

## Estimated time

01:00

## Overview

This exercise continues the focus on introducing you to business rule and decision service concepts and tasks. In this exercise, you review and customize the vocabulary that is attached to the BOM.

## Objectives

After completing this exercise, you should be able to:

- Review and correct verbalization of the BOM
- Refactor changes that you make to the vocabulary
- Define domains for customized rule vocabulary

## Introduction

This exercise involves these tasks:

- [Section 1, "Reviewing the vocabulary"](#)
- [Section 2, "Setting categories"](#)
- [Section 3, "Refactoring the vocabulary"](#)
- [Section 4, "Creating a domain"](#)

## Requirements

This exercise requires you to work in Rule Designer on the computer lab environment.

## Section 1. Reviewing the vocabulary

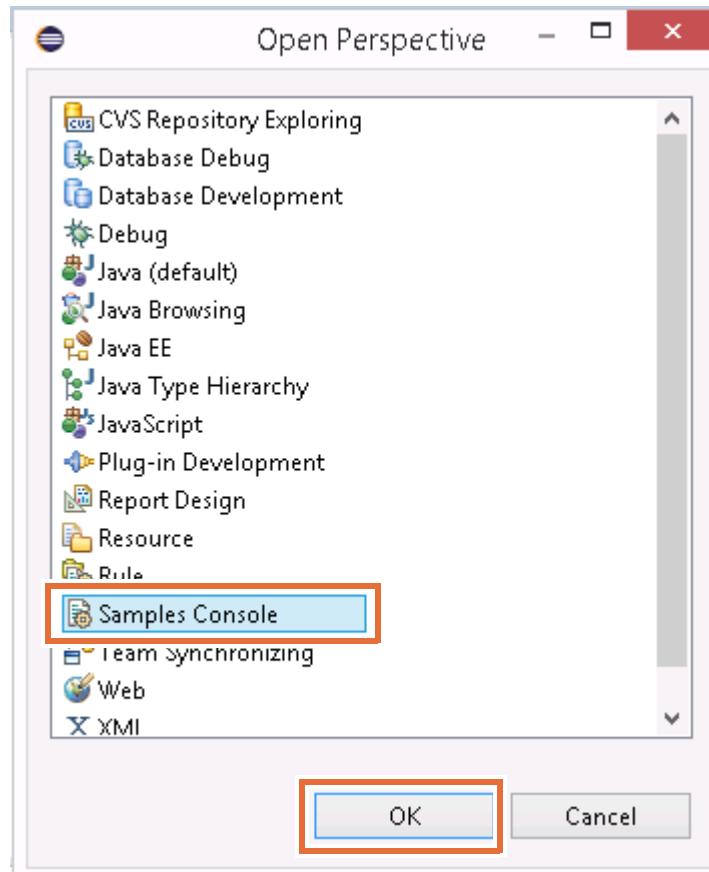
In this part of the exercise, you review the default verbalization of BOM elements for accuracy, and learn how to refactor vocabulary changes.

### 1.1. Switching to a new workspace and importing the start project

- 1. If Rule Designer is still open, you can switch to a new workspace by clicking **File > Switch Workspace > Other**.
- 2. In the Workspace Launcher, type a folder name:  
C:\labfiles\workspaces\bom
- 3. Close the Welcome pane.
- 4. Import the start project that is provided for this exercise by using the Samples console.
  - a. In the toolbar, click **Open Perspective**.



- b. In the Open perspective window, click **Samples Console**, and then click **OK**.

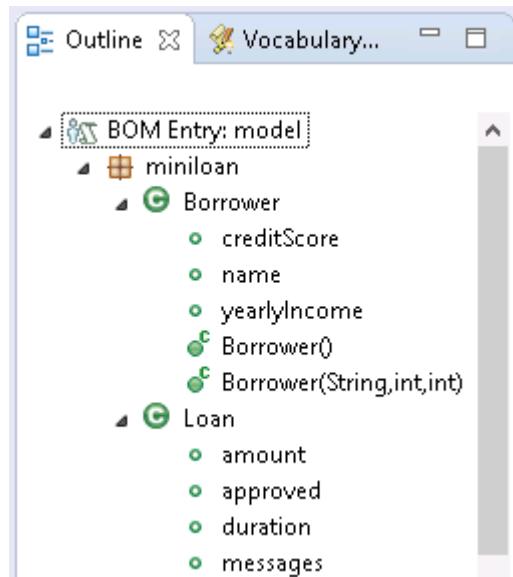


- c. In the **Samples and Tutorials** tab, expand **Rule Designer > Training > Ex 05: Working with the BOM**.

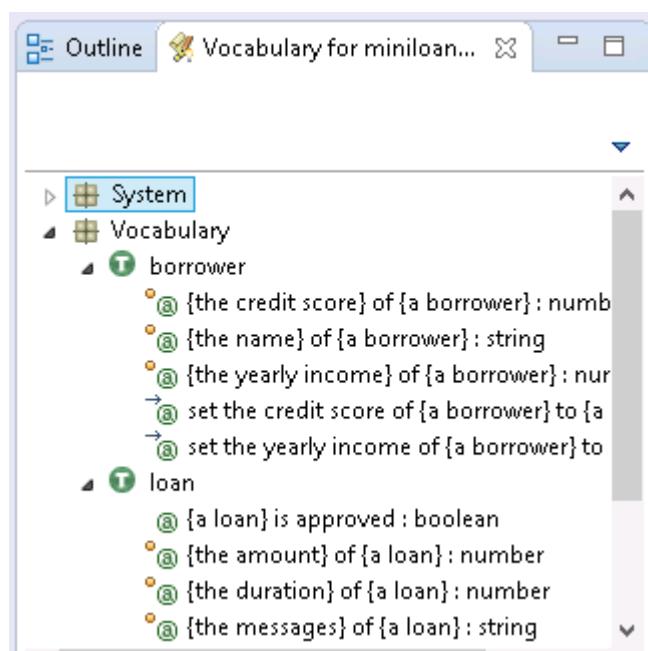
- \_\_\_ d. Under **start**, click **Import projects**.
- \_\_\_ 5. Close the **Help** pane by clicking the X.

## 1.2. Reviewing the vocabulary

- \_\_\_ 1. In Rule Explorer, expand **miniloan-rules > bom > model** and double-click **miniloan**.  
The **miniloan** BOM opens in the BOM editor, and it is also listed in the Outline view.
- \_\_\_ 2. In the Outline view, expand the members of the **miniloan** BOM to see the attributes for the **Borrower** and **Loan** classes.

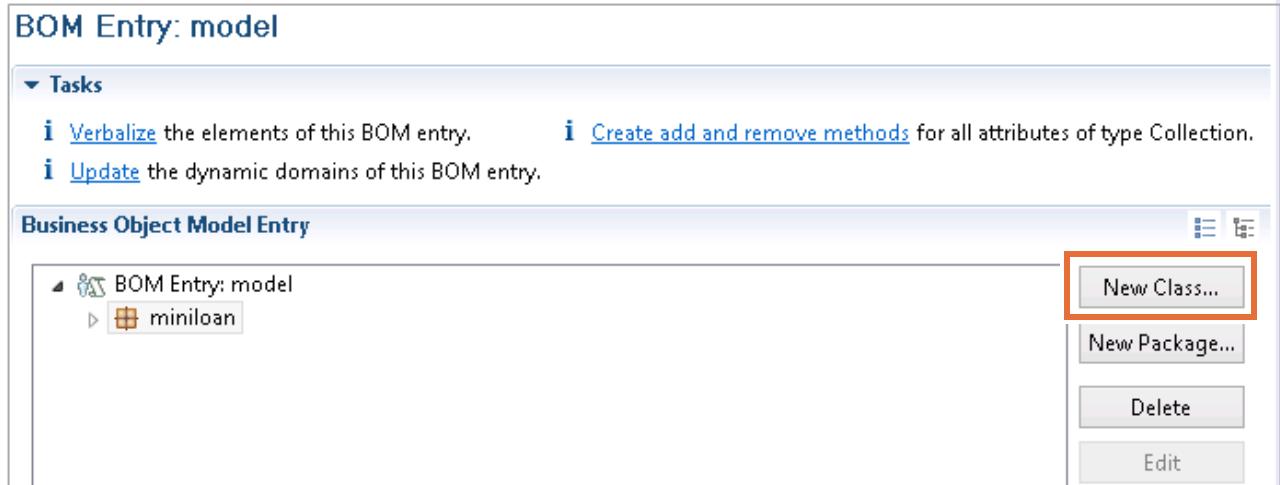


- \_\_\_ 3. Click the **Vocabulary** tab to switch to the Vocabulary view to explore how each of the BOM members is verbalized, and compare to the names used in the Outline view.

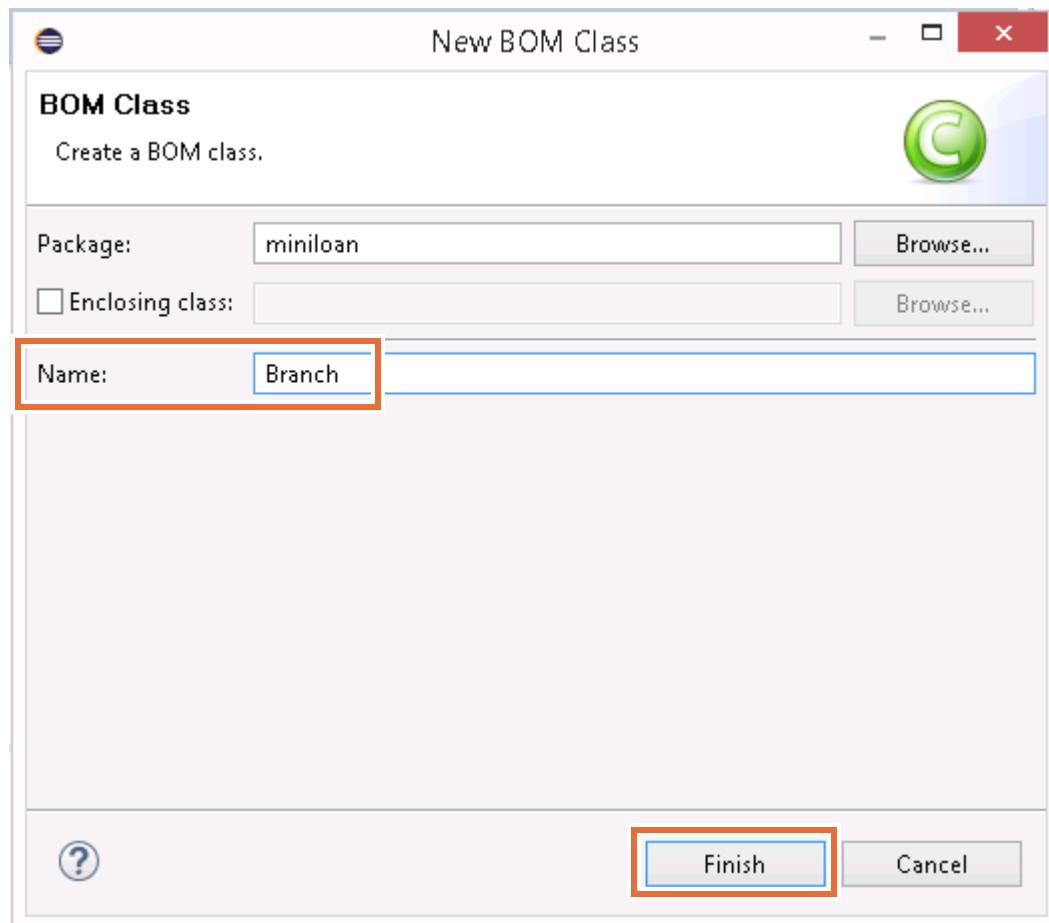


### 1.3. Creating a class

- 1. Go to the BOM editor in the main pane. The miniloan BOM is open on the **Package** tab.
- 2. Add a class to the miniloan BOM that is called **Branch**.
- a. In the Business Object Model Entry section, click **New Class**.



- b. Type **Branch** as the name, and click **Finish**.



The Branch class is now added in the list.

```

Business Object Model Entry

BOM Entry: model
  miniloan
    Borrower
    Branch
    Loan
  
```

- \_\_\_ 3. Save your work.



### Troubleshooting

Because you do not have a corresponding Branch class in the execution object model (XOM), you see errors when you save your work.

After you finish this exercise, you can delete the `Branch` class to eliminate the errors.

- \_\_\_ 4. In the Business Object Model Entry section, double-click **Branch** to open `Branch` in the Class editor.
- \_\_\_ 5. In the **Class Verbalization** section, click **Create**.

**Class Verbalization**

**i** This class is not verbalized. [Create](#) default verbalization.

Generate automatic variable

Notice that the verbalization includes an error for the plural form: **the branchs**.

**Class Verbalization**

**x** [Remove](#) the verbalization. [Edit](#) the documentation.

Generate automatic variable

Term:  [Edit term.](#)

**i** the branch, a branch **the branchs...**

\_\_ 6. Edit and correct the plural form of branch from `branchs` to `branches`.

\_\_ a. Click **Edit** next to the **Term** field.

**▼ Class Verbalization**

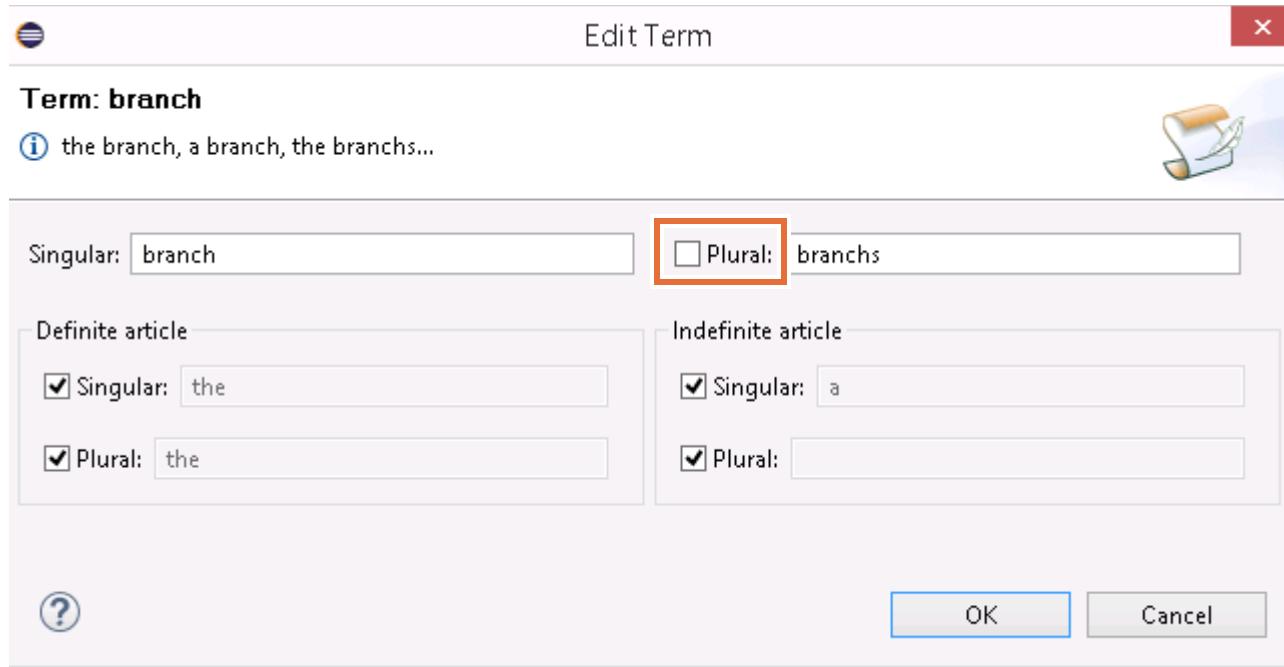
[Remove](#) the verbalization. [Edit](#) the documentation.

Generate automatic variable

Term: `branch`

`i` the branch, a branch, the branchs....

\_\_ b. In the Edit Term window, clear the **Plural** check box.



\_\_ c. Correct the error by changing `branchs` to `branches` and click **OK**.

The verbalization is now correct.

\_\_ 7. Save your work.



### Note

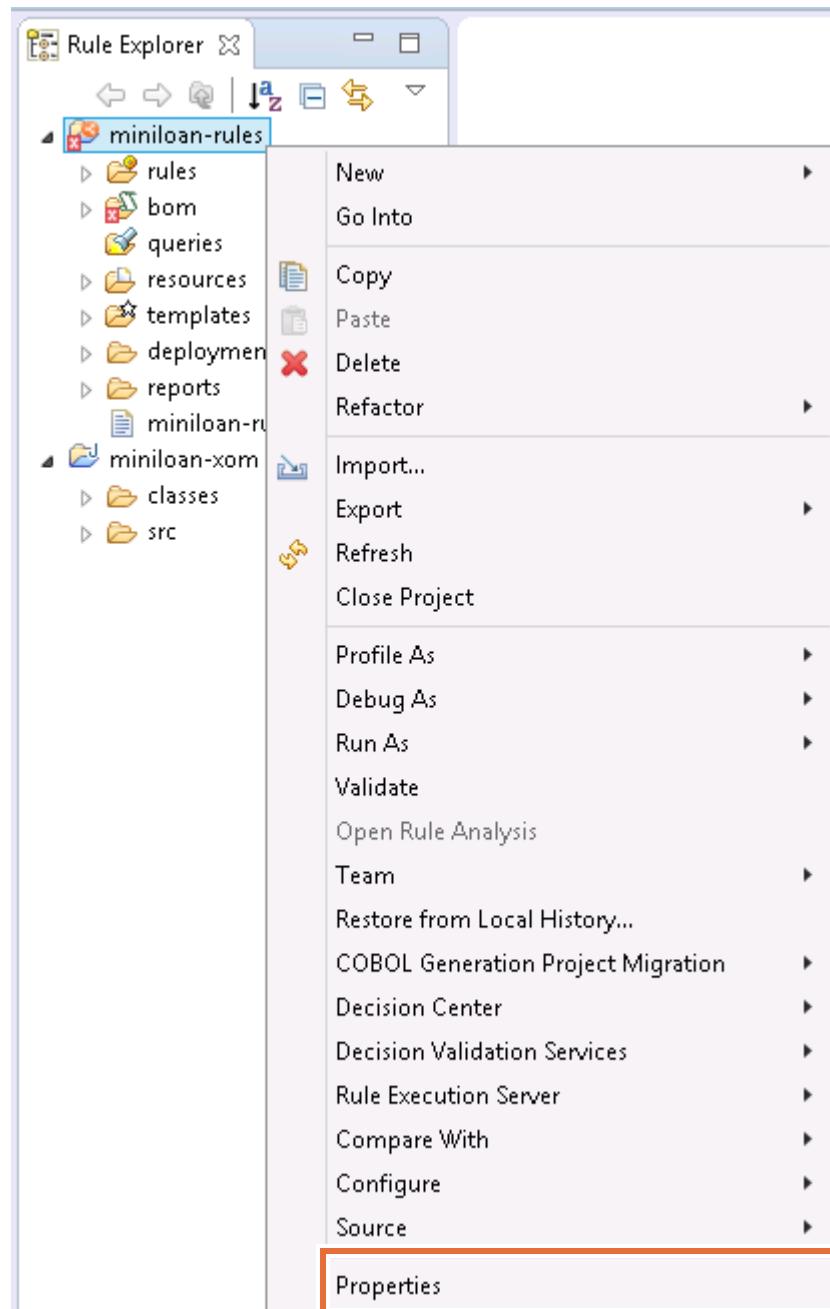
You can continue to ignore the errors on the `Branch` class.

## Section 2. Setting categories

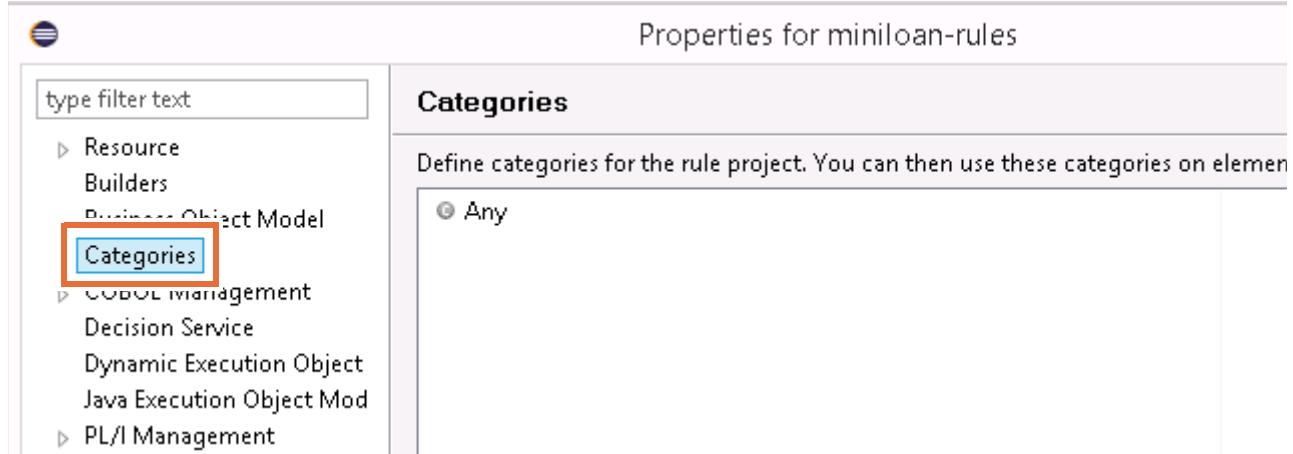
Categories help you author rules by filtering the vocabulary that is available when you write rules. To be able to set categories on business elements, you must first create them at the rule project level.

### 2.1. Adding a category to the rule project

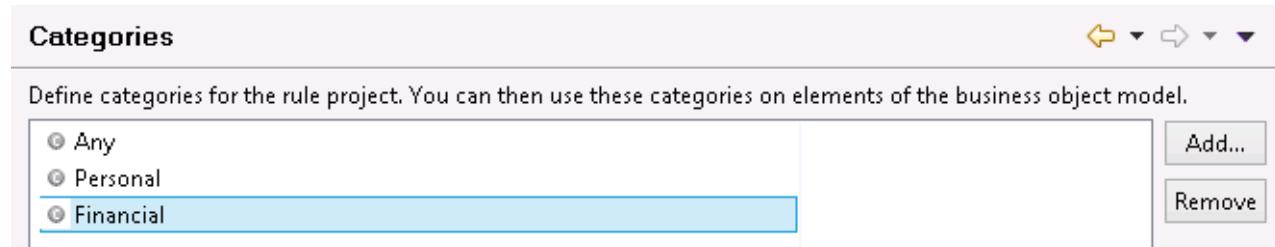
1. In Rule Explorer, right-click **miniloan-rules** and click **Properties**.



- 2. In the left pane of the Properties window, select **Categories**.



- 3. Add a category called **Personal**.
- a. On the Categories page, click **Add**.
  - b. In the New Category window, type **Personal** as a name for the category and click **OK**.  
The new category is shown in the list in the Categories page.
- 4. Add another category that is named **Financial** and click **OK**.



- 5. Click **OK** to close the Properties window.  
Now that the new categories are added to the rule project, you can assign them to business elements.

## 2.2. Specifying a category for a business element

- 1. In Rule Explorer, expand **miniloan-rules > bom > model > miniloan** to view the **Borrower** and **Loan** BOM elements.

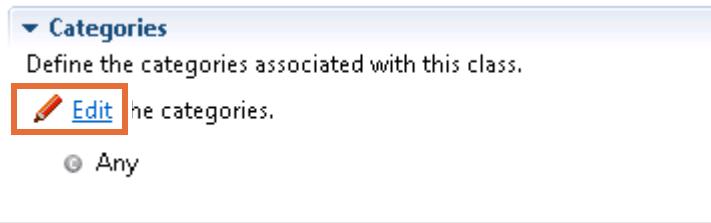


### Note

You can ignore the **Branch** BOM element.

- 2. Double-click **Borrower** to open it in the BOM editor.

- 3. In the Categories section of the BOM editor, click **Edit**.

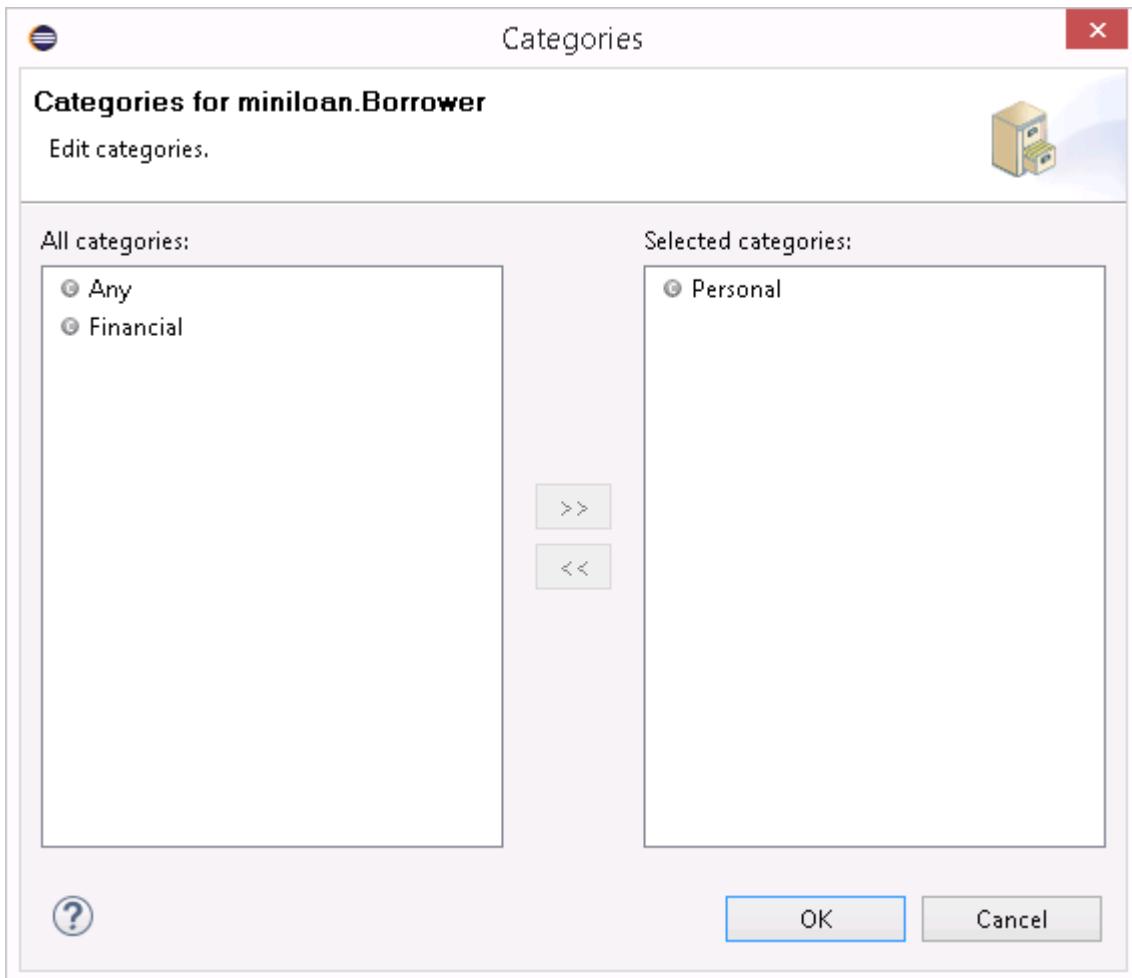


The Categories window opens.

- 4. In the **All categories** field, set the category to **Personal**.

- a. Double-click **Personal** to move it to **Selected categories**.

Alternatively, you can select **Personal** and click move right (**>>**) to move it to the **Selected categories** field.



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

In the BOM editor, the category for `Borrower` is now listed as **Personal**.

The screenshot shows the 'Categories' section of the BOM editor. A blue header bar says 'Categories'. Below it is a descriptive text: 'Define the categories associated with this class.' To the left of the text is a pencil icon. To the right is a link 'Edit the categories.' Underneath is a radio button labeled 'Personal' which is selected.



### Note

The `creditScore`, `name`, and `yearlyIncome` members are automatically assigned the **Personal** category when you set the category for `Borrower`.

- \_\_\_ 5. Set the category for `Loan` to **Financial**.



### Note

The `amount`, `approved`, `duration`, `messages`, `yearlyInterestRate`, and `yearlyRepayment` members are automatically assigned the **Financial** category when you set the category for `Loan`.

- \_\_\_ 6. Save your work.

## 2.3. Testing the categories

- \_\_\_ 1. In Rule Explorer, expand the `miniloan-rules > rules > validation` folder and open the `maximum amount` rule.
- \_\_\_ 2. In the Rule editor, in the Category Filter section, click **Edit**.

The screenshot shows the 'Action Rule: maximum amount' dialog in the Rule Editor. It has two main sections: 'General Information' and 'Category Filter'. The 'General Information' section shows the rule name as 'maximum amount'. The 'Category Filter' section shows the 'Categories' field set to 'Any' with a pencil icon next to it, which is highlighted with a red box. Below these sections are tabs for 'Documentation' and 'Content'. The 'Content' tab displays the rule logic in a script-like syntax:

```

1 if
2   the amount of 'the loan' is more than 1,000,000
3 then
4   add "The loan cannot exceed 1,000,000" to the messages of 'the loan';
5   reject 'the loan';

```

The Rule Category Filter opens.

- \_\_\_ 3. Set the category for the rule to **Financial**, and click **OK**.

- 4. In the Rule editor, select **amount of ‘the loan’** and notice that the items in the list are also categorized as **Financial**.

```

Content
1 if
2 the amount of 'the loan' is more than 1,000,000
3 then
4 add
5 reject
6

```

The screenshot shows a context menu for the phrase "amount of 'the loan'". The menu is filtered to show only items from the "Financial" category. The visible items are:

- amount of <a loan>
- duration of <a loan>
- length of <a string>
- messages of <a loan>
- number of <term>
- number of <term> where <test>,
- number of elements in <objects>
- yearly interest rate of <a loan>
- yearly repayment of <a loan>



### Hint

Put your mouse on **amount** and double-click to select the phrase and see the vocabulary list.

- 
- 5. Switch the category for the rule from the **Financial** category to the **Any** category, and notice the changes to the items available in the vocabulary list.
- 6. Switch from the **Any** category to the **Personal** category.
- 



### Questions

If you switch the category from **Financial** to **Personal**, why are warnings shown in the rule?

---



---

### Answer

The rule contains vocabulary that is from a different category. When you set a category on the rule, only vocabulary from that category can be used in the rule.

---

- 7. Switch the category for the rule back to **Any** and save your work.
- 8. Close the `maximum amount` rule.

## Section 3. Refactoring the vocabulary

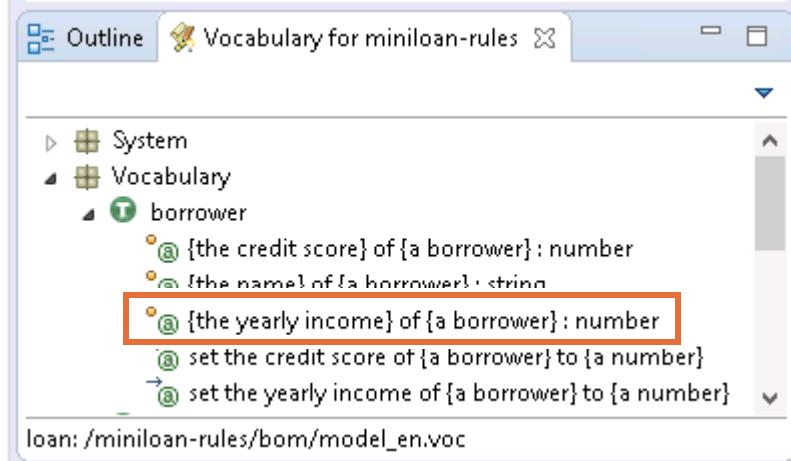
In this section, you update the project in Rule Designer, modify the verbalization of a business element, and refactor the changes.

### 3.1. Viewing how vocabulary changes are refactored

- 1. In the Rule Explorer, open the `minimum income` rule in the `eligibility` package, and note the wording.

```
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' ;
```

- 2. Expand **miniloan-rules > bom** and double-click **model** to open the Vocabulary view.
- 3. In the Vocabulary view, expand **borrower** and double-click **{the yearly income} of {a
 borrower}** to open it in the BOM editor.



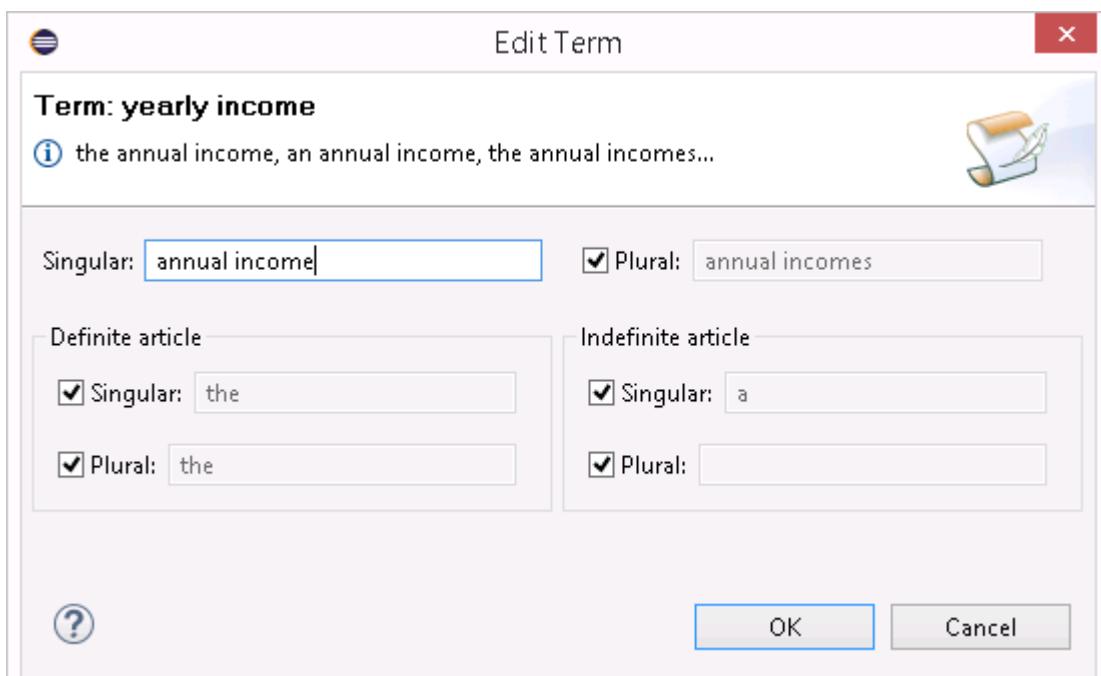
- 4. Under Member Verbalization, click **Edit**.

The screenshot shows the 'Member Verbalization' section of a software interface. It includes options to remove or create navigation and action phrases, and a specific option to edit the subject used in phrases, which is highlighted with a red rectangle. Below this are sections for 'Navigation' and 'Action', each with a template input field and a 'Remove' button.

- Navigation : "the yearly income of a borrower"**
- Action : "set the yearly income of a borrower to a number"**

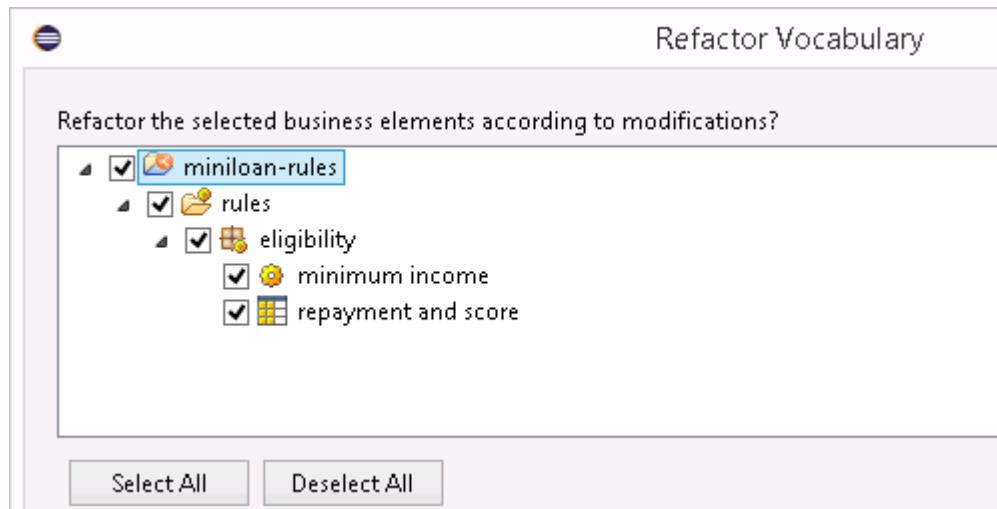
The Edit Term window opens.

- 5. Replace **yearly income** by typing **annual income** and click **OK**.



- 6. Save your work.

The Refactor Vocabulary window opens, and lists the rules that are affected, including the minimum income rule.



- \_\_\_ 7. Click **Yes** to allow the updates.
- \_\_\_ 8. Reopen the `minimum income` rule and note the wording of the condition statement, which now uses "annual income" instead of "yearly income."

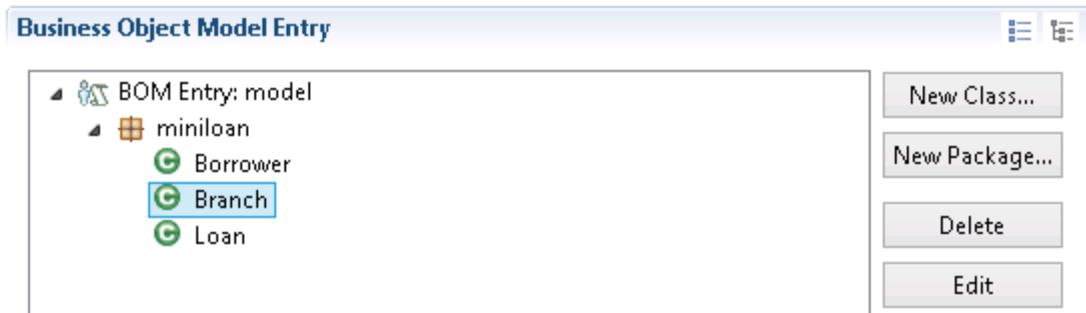
```
if
    the yearly repayment of 'the loan' is more than the annual income of 'the
    borrower' * 0.3
```



### Note

As mentioned in a previous note, the project shows errors because the execution object model (XOM) does not include a Branch class to correspond to the Branch class that you added to the BOM in ["Creating a class"](#) on page 5-4. To eliminate the errors from the project, you can delete the Branch class.

- \_\_\_ 1. Expand **miniloan-rules > bom > model** and double-click **miniloan** to open the BOM editor.



- \_\_\_ 2. Select **Branch** and click **Delete**.
- \_\_\_ 3. In the Remove Element window, click **Yes** to confirm the deletion.
- \_\_\_ 4. Save your work.

## Section 4. Creating a domain

In this exercise, you extend the BOM by adding a domain for marital status.

### 4.1. Creating an enumerated domain for marital status

- 1. In Rule Designer, continue working in the same workspace as used in the previous task.
- 2. In Rule Explorer, expand **miniloan-rules > bom > model** and double-click **miniloan** to open the **miniloan** BOM in the Outline view.
- 3. In the Outline view, expand **miniloan**, and double-click **Borrower** to open the Borrower class in the BOM editor.

Notice that this page is the **Class** tab of the editor.

**Class Borrower (package: miniloan)**

**General Information**

Name:  [Change...](#)

Namespace:  [Change...](#)

Superclasses:  [Change...](#)

Interfaces:  [Change...](#)

Deprecated

**Class Verbalization**

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

Generate automatic variable

Term:  [Edit](#) term.

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

**Members**

Specify the members of this class.

<input checked="" type="radio"/> creditScore	<a href="#">New...</a>
<input checked="" type="radio"/> name	<a href="#">Delete</a>
<input checked="" type="radio"/> yearlyIncome	<a href="#">Edit</a>
<input checked="" type="radio"/> Borrower()	
<input checked="" type="radio"/> Borrower(String,int,int)	

**Domain**

Create and edit a domain for this class.

[Create](#) a domain.

**Categories**

Define the categories associated with this class.

[Edit](#) the categories.

Personal

**Custom Properties**

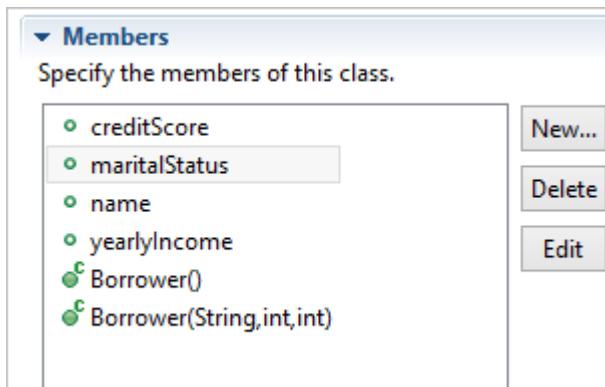
  

**BOM to XOM Mapping**

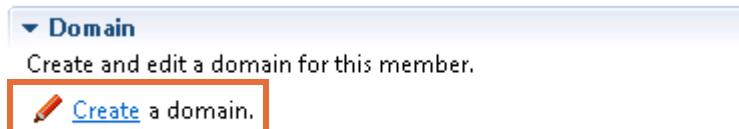
Package	<b>Class</b>	Member	model.bom	model.b2xa	model_en.voc
---------	--------------	--------	-----------	------------	--------------

- \_\_\_ 4. In the Members section, click **New** to create an attribute.
  - \_\_\_ a. For **Type**, select **Attribute**.
  - \_\_\_ b. In the **Name** field, enter: `maritalStatus`
  - \_\_\_ c. In the **Type** field, enter: `string`
  - \_\_\_ d. Click **Finish**, and save your work.

The new attribute is listed as a member of the class.

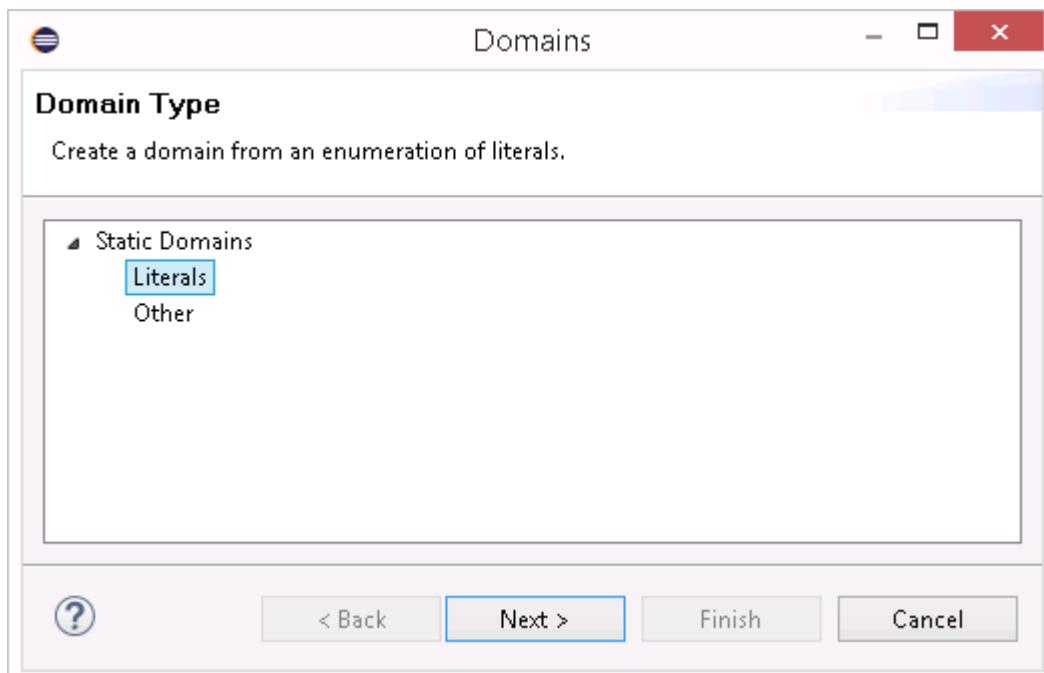


- \_\_\_ 5. In Rule Explorer, double-click **miniloan-rules > bom > model > miniloan > Borrower** to open **Borrower** again in the BOM editor.
- \_\_\_ 6. Double-click the new `maritalStatus` attribute to open the **Member** tab of the editor.
- \_\_\_ 7. In the Domain section of the Member page, click **Create**.

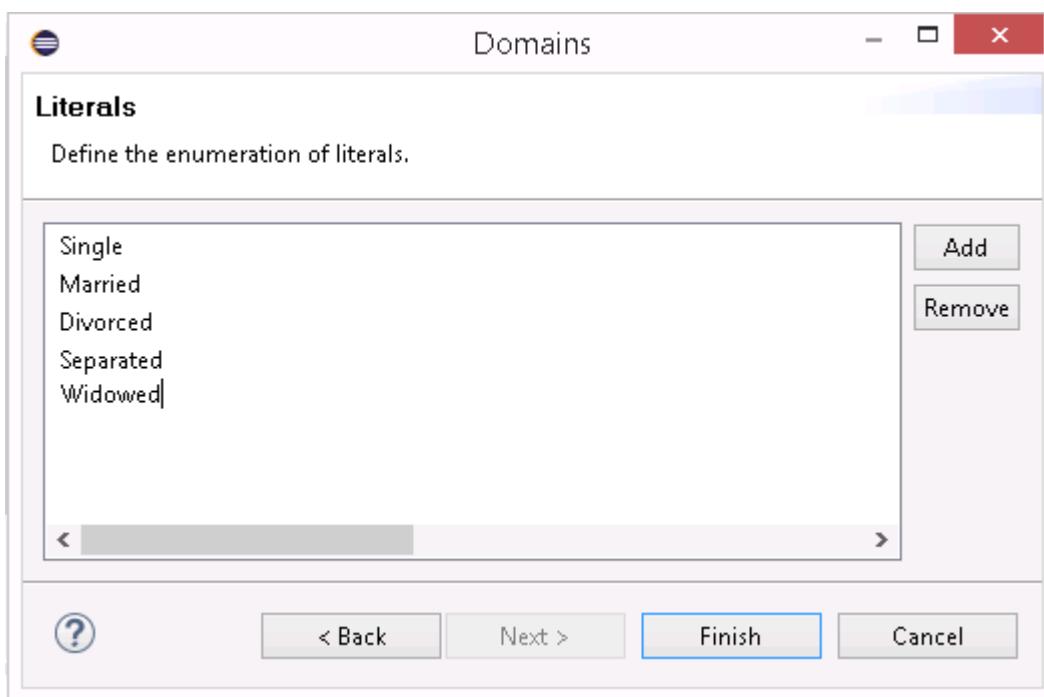


The Domains window opens.

- 8. Define the domain values.
- a. For Domain Type, select **Literals**, and click **Next**.



- b. In the Literals page, click **Add**.  
The **newValue** field is listed in the Literals field. You can type directly on **newValue**.
- c. Replace **newValue** with: Single
- d. Click **Add** to add the values Married, Divorced, Separated, and Widowed to the list of literals.



- e. Click **Finish**.

The values in the domain are now listed in the Domain section.

**Domain**  
Create and edit a domain for this member.  
✎ [Edit](#) the domain.  
✖ [Remove](#) the domain.

**Domain type: Literals**

- Divorced
- Married
- Separated
- Single
- Widowed

- \_\_\_ 9. Save your work.



#### Note

Ignore the errors that are shown in the project. You see an error because the execution object model (XOM) does not include a `maritalStatus` attribute.

You can now write rules specific to marital status.

## 4.2. Testing the new marital status domain in a rule

- \_\_\_ 1. In Rule Explorer, expand the `miniloan-rules > rules`, right-click the **validation** package, and click **New > Action Rule**.
- \_\_\_ 2. In the **Name** field, type `single borrower test rule` and click **Finish**.  
The new rule opens in the Intellirule editor.
- \_\_\_ 3. Switch to the Guided editor.
- \_\_\_ a. Close the rule in the Intellirule editor.



- \_\_\_ b. In Rule Explorer, right-click the rule, and click **Open With > Guided editor**.
- \_\_\_ 4. Click **<select a condition>** and from the vocabulary list, look for **the marital status of <a borrower>**.



## Questions

Is the **marital status** available in the vocabulary list? Why not?

---



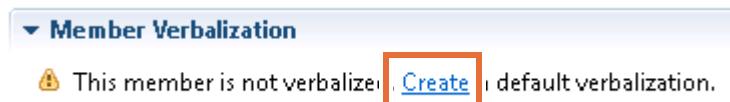
---

## Answer

You must verbalize the **marital status** so that you can use it in the rules.

---

- 5. Double-click **miniloan-rules > bom > model > miniloan > Borrower > maritalStatus** in Rule Explorer. In the Member Verbalization section, click **Create**.

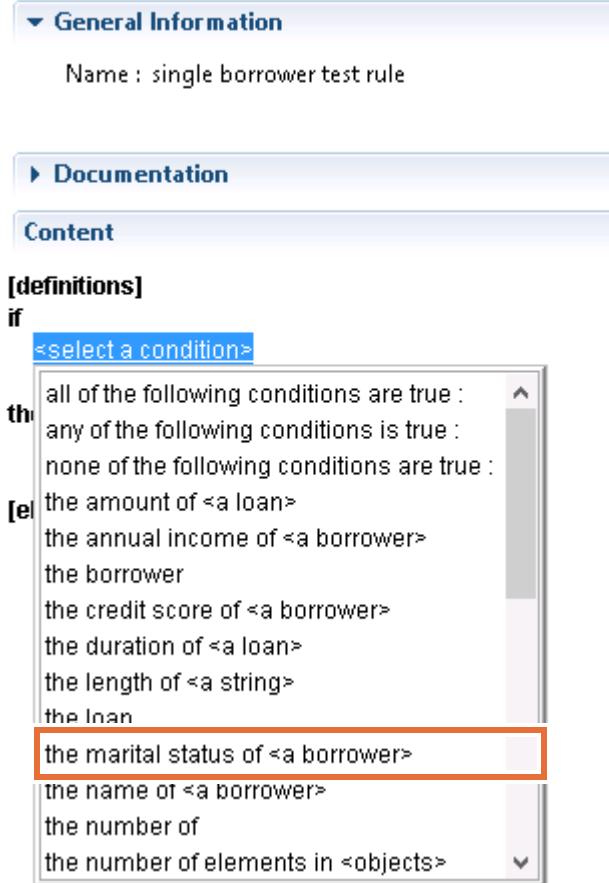


**Member Verbalization**

⚠ This member is not verbalized. [Create](#) default verbalization.

- 6. Press Ctrl+Shift+S to save all.
- 7. Try [Step 4](#) again to select the **marital status of <a borrower>**.

### Action Rule: single borrower test rule



**General Information**

Name : single borrower test rule

**Documentation**

**Content**

[definitions]

if

<select a condition>

th

all of the following conditions are true :

any of the following conditions is true :

none of the following conditions are true :

[el]

the amount of <a loan>

the annual income of <a borrower>

the borrower

the credit score of <a borrower>

the duration of <a loan>

the length of <a string>

the loan

the marital status of <a borrower> **highlighted with a red box**

the name of <a borrower>

the number of

the number of elements in <objects>

- a. Click **<a borrower>** and select **the borrower**.

- \_\_ b. Click <a string> and select **Single**.

The screenshot shows the BOM interface with the following code structure:

```
[definitions]
if
  the marital status of the borrower [x] is <a string> [x]
    ↴
  then
    <select an action>
    ↴
  [else]
```

A context menu is open at the position of the first '<a string>' placeholder, listing several options: Divorced, Married, Separated, Single, and others. The option 'Single' is highlighted with a red border.

- \_\_ 8. Click <select an action> and from the vocabulary list, select **print <a string>**.  
 \_\_ 9. Click <enter a string> and type: The borrower is single

The screenshot shows the BOM interface with the following completed code:

```
[definitions]
if
  the marital status of the borrower [x] is Single [x]
    ↴
  then
    print ▼ The borrower is single [x]
    ↴
  [else]
```

- \_\_ 10. Save your work.

**End of exercise**

## Exercise review and wrap-up

This exercise looked at how to customize the vocabulary that is used in rules by working with the BOM.

---

# Exercise 6. Exploring the Decision Center Business console

## Estimated time

01:30

## Overview

This exercise focuses on the Decision Center Business console, which is the main rule authoring environment in Decision Center. You learn how to author and edit rules, manage changes, and collaborate with other users.

## Objectives

After completing this exercise, you should be able to:

- Use the Business console rule editors to author rules
- Collaborate with other users to modify rules
- Manage changes by using timelines and snapshots
- Compare versions of rules

## Introduction

In this exercise, you use the Business console to edit and create rules, and collaborate with other users. You also learn how to manage changes and work with snapshots.

The exercise includes these sections:

- [Section 1, "Working with rule projects and assigning tasks"](#)
- [Section 2, "Creating and editing rules"](#)
- [Section 3, "Working with the Business console management features"](#)

## Requirements

This exercise requires you to work in the Decision Center Business console on the computer lab environment.

The sample server must be running in this exercise.

## Exercise overview

**Scenario:** This exercise is based on a fictitious rule project of Miniloan, an online lender. The rule project determines whether a customer is eligible for a loan.

You log in to the Business console as the following two users:

- Paul is a manager who initiates and reviews changes to rules.
- Bea is a rule author who implements the changes.

# Section 1. Working with rule projects and assigning tasks

---



## Requirements

In this part of the exercise, you work as Paul. You recently received a request from the underwriting department to change the loan eligibility rules. The changes to the policy are outlined in an external document.

In the original policy, the input data is checked to ensure that the age is in the range 0 - 150. The loan amount is unlimited, and the credit score of the borrower increases according to income.

The new policy requires the following changes to the rules:

- More checks on the input data are required:
    - The borrower must be at least 18 to ask for a loan
    - The loan cannot exceed \$1000000
  - The score that is assigned to borrowers who have a yearly income of \$170000 - \$200000 must increase by 280 points.
- 

### 1.1. Logging in to the Business console as Paul

- 1. If it is not already running, start the sample server by clicking **Start**, and then clicking the **Start Sample Server** shortcut.
  - 2. Sign in to the Business console as Paul.
    - a. Go to Start, and then click the **Decision Center Business console** shortcut.
- 



## Information

You can also start the Business console by entering the following URL in a web browser window:

`http://localhost:9090/decisioncenter`

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

---



## Cloud

If you use IBM ODM on Cloud, you start the Business console by logging in to the User Portal, then clicking **Launch** under **Decision Center Business console**.

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

---

- \_\_\_ b. At the login page, enter **Paul** in the **Username** and **Password** fields, and click **Log in**.

The image shows the IBM Decision Center Business Console login page. The header features the IBM logo and the text "Decision Center | Business Console". Below the header is a form with the following fields:

- Username:** A text input field containing the value "Paul".
- Password:** A password input field showing four dots ("••••") as the password.
- Keep me logged in:** A checkbox labeled "Keep me logged in" which is unchecked.
- Log in:** A blue rectangular button with the text "Log in" in white.

At the bottom of the page, there is a small note: "Licensed Materials - Property of IBM. © Copyright IBM Corporation 2000, 2017".

## 1.2. Viewing the Home tab

The Decision Center Business console displays the **Home** tab when you log in, and it is designed so that you can quickly access your main areas of interest.

- \_\_\_ 1. Note the **top banner**, which is always visible. It shows:
- The main tab that you are on (**Home**, **Library**, **Work**, or **Administration**).

### Information

The tabs that you see depend on your user role. Paul can see the **Administration** tab because he is in a manager user and has administrative privileges.

Later, you see that Bea, who is a rule author user, does not have access to the **Administration** tab.

- 
- Your user name.
    - From the user name menu, you can access your profile and log out.

The screenshot shows the Decision Center Business Console interface with the following elements:

- Top Banner:** The tabs are labeled **HOME**, **LIBRARY**, **WORK** (with a checked checkbox), and **ADMINISTRATION**.
- User Profile:** A dropdown menu for "Paul" is open, showing a profile picture of a man, the name "Profile", a "Follow" button, and a "Log out" link.
- Content Area:** The "What's New" section is active, displaying a "New Rules" item.
- Bottom Footer:** The footer contains the copyright notice "© Copyright IBM Corp. 2017" and the statement "Course materials may not be reproduced in whole or in part without the prior written permission of IBM."

- 2. Note the right column, which displays two permanent sections:
- Followed Rules
  - Rules Recently Worked On

The screenshot shows two sections side-by-side. The left section is titled 'Followed Rules' and contains the message 'You have no followed rules.' with a downward arrow icon. The right section is titled 'Rules Recently Worked On' and contains the message 'You have not worked on any rules yet.'

The most recent items in these feeds are shown first. You do not see any new items now.

- 3. Note the **What's New** tab. It displays the following feeds:
- New Rules
  - New Updates on Followed Rules
  - New Comments in Activity Stream

The screenshot shows the 'What's New' tab selected. It displays three sections: 'New Rules' (No new rules have been created.), 'New Updates on Followed Rules' (There have been no new updates on followed rules.), and 'New Comments in Activity Stream' (There have been no new comments in the activity stream.).

The most recent items in these feeds are shown first. You do not see any new items now.

- 4. Click the **Stream** tab, which displays a chronological feed of activities and comments, with the most recent first.
- You do not see any new items now.

Notice that you can post a comment directly into the feed by entering it into the **Post a new comment here** field.

### 1.3. Viewing the Library page

The Library page lists the decision services that you can work on.

- 1. At the top of the page, click **Library**.

The **Library** tab lists decision services that are available to you.



#### Decision Services

Date | Name

Filter:

You can quickly find a decision service by changing the view (**Date** or **Name**) or by using the **Filter** function. These display functions can be helpful if you work with many projects.

In addition, the **NEW** flag alerts you to new decision services that you can access.



By default, the decision services are listed by the date they were last modified, with the newest date first.

- 2. Click **Name**.

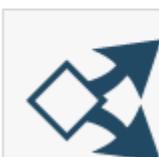
The Name view organizes the projects in alphabetical order.

- 3. Enter `my` into the **Filter** field.

#### Decision Services

Date | Name

Filter: my



[my-validation-rules-service](#)

Created by rtsAdmin on Apr 21, 2017

NEW

The filter automatically shows you the decision service with the text `my` in its name: **my-validation-rules-service**.

- 4. Click the mouse anywhere in the blank space of the **my-validation-rules-service** decision service.

Note the following features:

- The project box expands to show a list of recently updated branches.



- Hovering your mouse pointer over the decision service also displays a star to the right of the link, which you can click to follow the decision service or project.



## 1.4. Working with a decision service

In this section, you open **my-validation-rules-service** from the Library so that you can review its current business rules. You then assign Bea to implement the changes to the rules.

- 1. In the Library, click the **my-validation-rules-service** link.



### Note

By default, decision services open on the **Releases** tab. Because this decision service is not governed, you cannot create a release for it. However, you can work on branches in an ungoverned decision service.

You work with releases and governed decision services in the governance exercise.

- 2. Click the **Branches** tab, and then click **main**.

You see the list of decision artifacts in the decision service, including four folders to organize business rules:

- **computation**: These rules make the preliminary checks for loan approval.
- **eligibility**: These rules determine whether the loan can be approved.
- **insurance**: These rules manage the insurance policy.
- **validation**: These rules make the preliminary checks that determine whether data is rejected immediately.



### Note

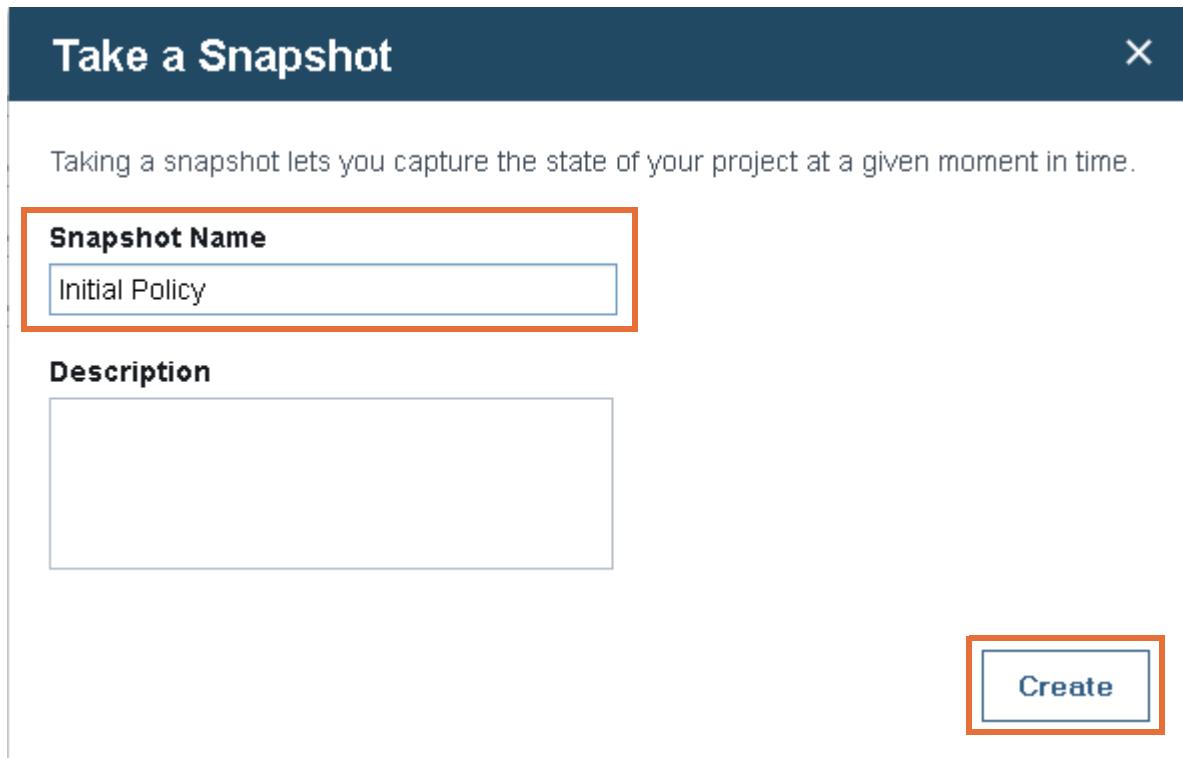
The changes to the policy that Paul received from the underwriting department affect the rules in the **validation** folder. The changes also affect a rule in the **computation** folder.

- 3. Create a snapshot for the main branch to capture the initial state of the decision service.

Snapshots capture the state of a project or branch at a specific moment. You can easily take snapshots, and if you have the appropriate rights, you can restore projects to a former state. You can also rename and delete snapshots. Finally, you can compare snapshots to compare the state of the rules at different times.

- a. In the main toolbar, click **Take Snapshot**.

- \_\_ b. Name your snapshot **Initial Policy** and click **Create**.



Snapshots that you create can be accessed from the **Snapshots** tab.

- \_\_ c. Click the **Snapshots** tab to see that the **Initial Policy** snapshot is listed on the page.

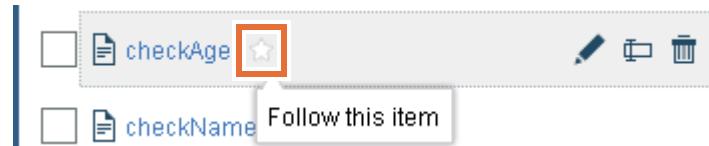


### Information

You might need to click the left and right arrows in the tab bar to access the other Decision Center tabs.

- \_\_ 4. Click the **Decision Artifacts** tab to return to the list of rules in the decision service.

- \_\_\_ 5. *Optional.* Enable all types of decision artifacts for viewing in Decision Center.
  - \_\_\_ a. On the **Decision Artifacts** tab, click **Types**.
  - \_\_\_ b. Select the **All Types** check box, and click **Apply**.
- \_\_\_ 6. Click the **validation** folder to open it and view the list of rules.
- \_\_\_ 7. Follow the **checkAge** rule.
  - \_\_\_ a. Hover the mouse pointer over the **checkAge** link.
  - \_\_\_ b. Click the **Follow this item** icon (the white star).



- \_\_\_ c. The star turns gold, which indicates that you are following this rule.
- Because you are following the **checkAge** rule, you are updated on events that happen to the rule.

- \_\_\_ 8. Click **checkAge** and review the rule:

```

definitions
    set 'minAge' to 0 ;
    set 'maxAge' to 150 ;
if
    it is not true that the age of 'the borrower' is between minAge and maxAge
then
    in 'the loan report', reject the data with the message "The borrower's age is
    not valid.";
```



## Questions

Based on the policy changes that the underwriting department requested, what changes must you make in this rule?

## Answers

Recall that the minimum age must be changed from 0 to 18.

- \_\_\_ 9. Because the requested changes also involve income and credit score, look at the `salary2score` decision table.
  - \_\_\_ a. Click the **computation** folder.
  - \_\_\_ b. In the **computation** folder, click **salary2score** to review the decision table.

	Yearly income		Add to customized credit score
	Min	Max	
1	< 10,000		21
2	10,000	20,000	50
3	20,000	30,000	80
4	30,000	50,000	120
5	50,000	80,000	150
6	80,000	120,000	200
7	120,000	200,000	250
8	≥ 200,000		300

Decision tables provide a way of viewing and managing sets of similar business rules. A decision table contains condition columns and action columns.

In the `salary2score` decision table:

- The left column, **Yearly income**, is the condition column.
- The right column, **Add to customized credit score**, is the action column.

The decision table adds a predetermined amount to a borrower's credit score according to the borrower's income.



## Questions

Based on the requested policy changes from the underwriting department, what changes must you make to the `salary2score` decision table?

## Answers

Recall the new requirements for a salary range and number to add to the borrower's credit score.

- \_\_\_ 10. Follow the `salary2score` table to be notified of changes to the decision table.
  - \_\_\_ a. Click the **computation** folder to see the list of rules again.
  - \_\_\_ b. Select the `salary2score` table and click **follow the selected items** (the first star icon) in the toolbar to follow the decision table.

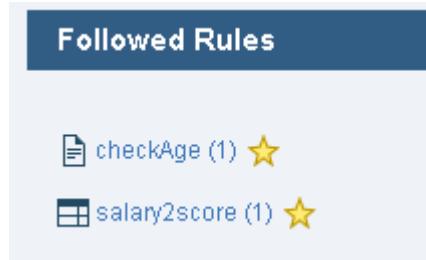
Name	Created By	Last Changed On
<code>bankruptcyScore</code>	rtsAdmin	April 21, 201...
<code>initialCustomizedCreditScore</code>	rtsAdmin	April 21, 201...
<code>neverBankruptcy</code>	rtsAdmin	April 21, 201...
<code>rate</code>	rtsAdmin	April 21, 201...
<code>repayment</code>	rtsAdmin	April 21, 201...
<code>salary2score</code>	rtsAdmin	April 21, 201...

## 1.5. Posting a comment in the Stream

Now that you reviewed the business rules that require changes, you are ready to assign work on the changes to your colleague Bea, who is a rule author. You can ask Bea to implement the changes by posting a comment in the Stream.

- \_\_\_ 1. Go back to the **Home** tab.
  - \_\_\_ a. In the Decision Center top banner, click **Home**.

- \_\_\_ b. On the **Home** tab, notice that the `checkAge` rule and the `salary2score` table are now listed in the Followed Rules section.



- \_\_\_ 2. In the **Home** tab, click **Stream**.  
 \_\_\_ 3. Post a comment to Bea with information about the changes.  
   \_\_\_ a. Click **Post a new comment here** to activate the comment field.  
   \_\_\_ b. Enter a message to Bea:

Hi, Bea. There is a new policy. Can you implement it by Friday? Thanks.

- \_\_\_ c. Click **Post** to post the message to the Stream.  
 \_\_\_ 4. Log out as Paul.  
   \_\_\_ a. Click **Paul**.  
   \_\_\_ b. Click **Log out**.



## Section 2. Creating and editing rules

In this part of the exercise, you work as Bea, a rule author. As Bea, you respond to the request from Paul to update some of the business rules. You learn how to modify a rule and a decision table, and you also learn how to create a rule in the Decision Center Business console.

### 2.1. Logging in as Bea and viewing recent activity

- \_\_\_ 1. Log in to the Business console by entering `Bea` in the **Username** and **Password** fields.
- \_\_\_ 2. Notice that for Bea, the top banner has the following tabs: **Home**, **Library**, and **Work**.



#### Information

As a rule author user, Bea does not have access to the **Administration** tab like Paul did.

- \_\_\_ 3. Notice the post from Paul in the “New Comments in Activity Stream” section.

The screenshot shows a user interface for the "New Comments in Activity Stream". At the top, there is a header with a downward arrow and the text "New Comments in Activity Stream". To the right of the header are two small icons: a blue circle with a minus sign and a blue rectangle with the number "1". Below the header, there is a comment card. On the left side of the card is a small circular profile picture of a man. Next to the picture, the name "Paul" is listed, followed by the text "created a new post" and an envelope icon. Below this, the comment text reads "Hi, Bea. There is a new policy. Can you implement it by Friday? Thanks." and the timestamp "May 3, 2017 at 12:43 PM".

### 2.2. Modifying an action rule

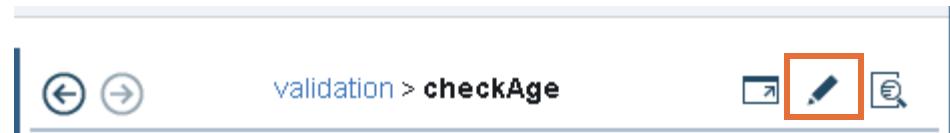
In this part of the exercise, you implement the requirement for a minimum age of 18. Bea knows to work with the `checkAge` action rule.

- \_\_\_ 1. Open the `checkAge` rule in `my-validation-rules-service`.
  - \_\_\_ a. Click **Library** and click the white space in the **my-validation-rules-service** section.
  - \_\_\_ b. Click **main** to open the main branch.

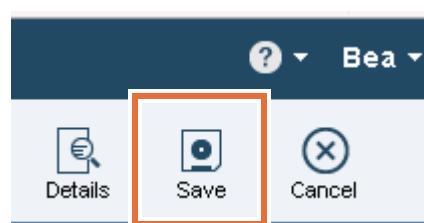
The screenshot shows a library interface for "my-validation-rules-service". At the top, there is a logo consisting of two overlapping arrows pointing upwards and to the right. Next to the logo, the service name "my-validation-rules-service" is displayed in blue text, followed by the text "Created by rtsAdmin on Apr 21, 2017". In the top right corner, there is a green ribbon-like graphic with the word "NEW" in white. Below the service information, there is a section titled "Recently updated branches:" with a list. The first item in the list is a button labeled "main" with an upward arrow icon, which is highlighted with a red box. Below this button, the text "Created by rtsAdmin on Apr 21, 2017" is visible.

- \_\_\_ c. Make sure that you are on the **Decision Artifacts** tab.
- \_\_\_ d. *Optional.* Enable all decision artifact types by clicking **Types**, selecting **All Types**, and clicking **Apply**.

- \_\_\_ e. Click the **validation** folder and click **checkAge**.
- \_\_\_ 2. Edit the `checkAge` rule to reflect the new minimum age of 18.
- \_\_\_ a. Click the **Edit** icon.

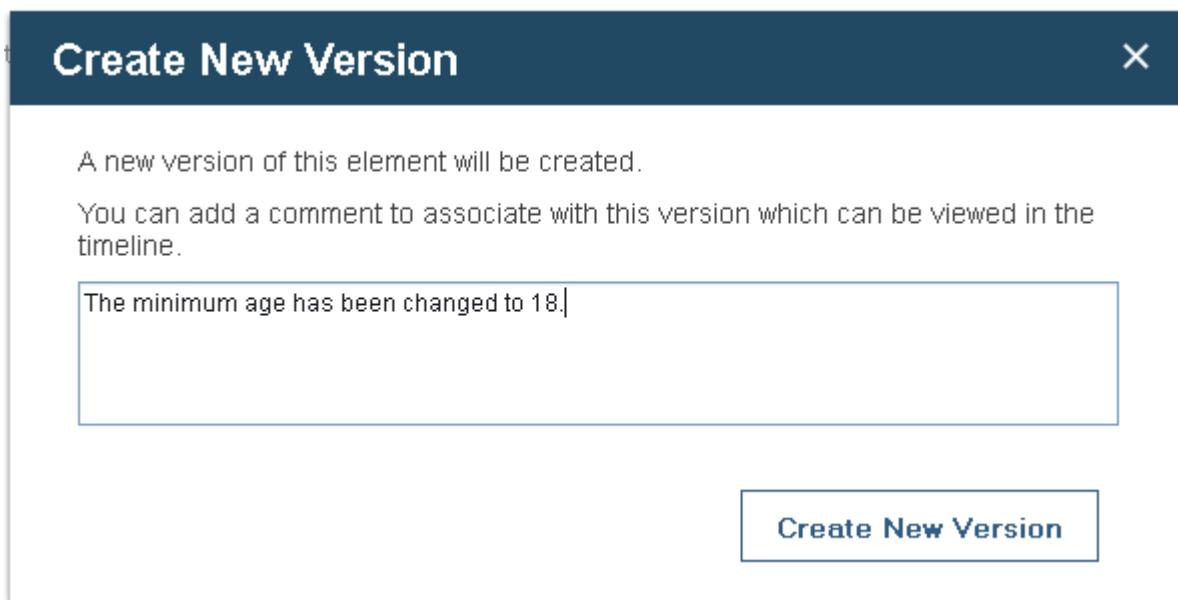


- \_\_\_ b. In the definitions part of the rule, delete the minimum age value **0**, and type **18**.
- \_\_\_ 3. Finish editing the rule and add a comment that describes the change.
- \_\_\_ a. Click **Save** to save your changes.



- \_\_\_ b. In the Create New Version window, enter the following comment, and click **Create New Version**.

The minimum age has been changed to 18



The `checkAge` rule now displays **v1.1** to indicate that it is a new version.



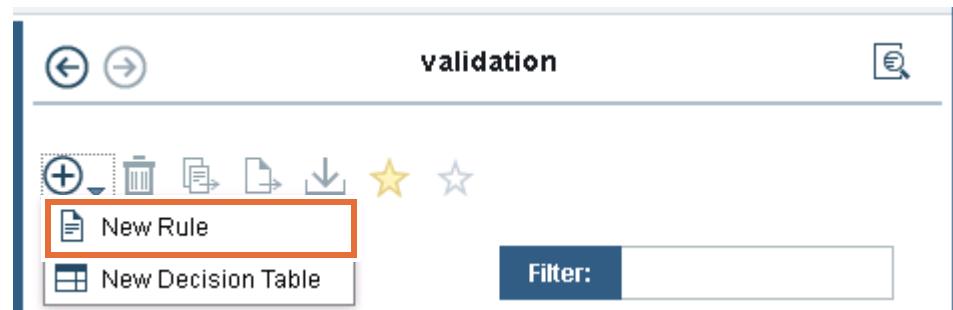
- 4. Click the **main** breadcrumb to return to the main branch page for my-validation-rules-service.



## 2.3. Creating an action rule

In this step, you implement the change to the policy that loans cannot exceed \$1000000. The policy currently does not check for the loan amount, so you must create a validation rule.

- 1. Create a rule called `checkAmount` in the **validation** folder.
  - a. Make sure that you are in **my-validation-rules-service > main**, and that you are on the **Decision Artifacts** tab.
  - b. In the left explorer pane, click the **validation** folder.
  - c. On the **validation** folder toolbar, click **create an artifact** (the plus [+]) and click **New Rule**.



- d. In the “Create new Rule” window, type the name `checkAmount` and click **Create**.

The screenshot shows the 'Create new Rule' dialog box. It has a title bar with a close button ('X'). The main area contains a text input field with the placeholder 'Specify a name for the new rule.' and the value 'checkAmount'. Below the input field is a note: 'Select **Create** to start authoring your rule.' At the bottom right is a blue 'Create' button.

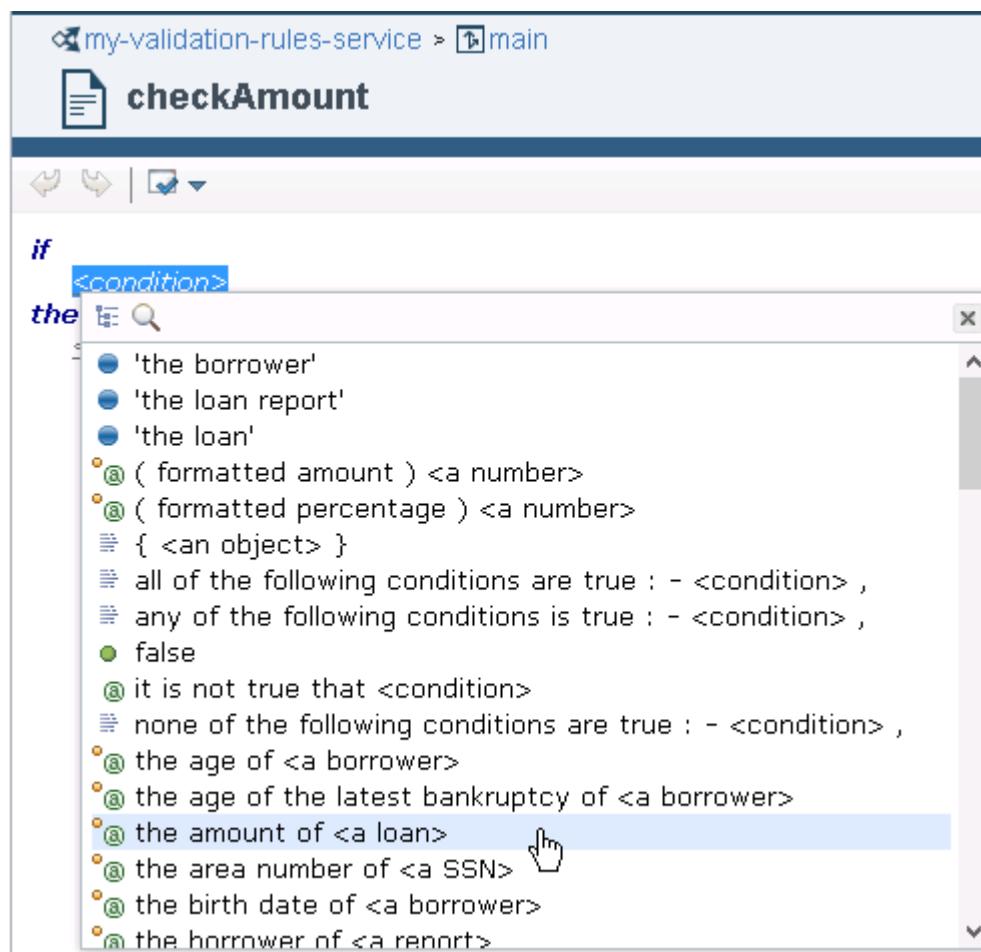
The empty rule opens in the rule editor. Next, you use the completion menu to build the following rule:

```
if
    the amount of 'the loan' is more than 1000000

    then
        in 'the loan report', reject the data with the message "The loan cannot
        exceed 1000000";
```

- 2. Build the condition statement in the “if” section.

- a. Click <condition>, select the amount of <a loan> , and then select ‘the loan’.

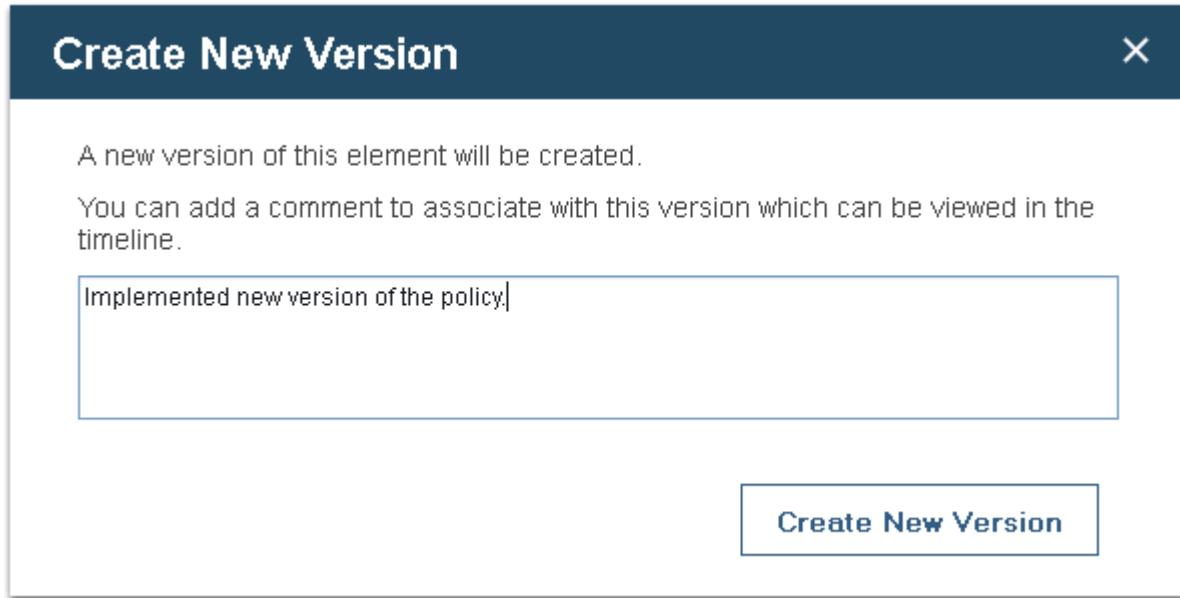


### Information

The substitution of the word “the” for “a” might not seem to be a significant difference. By making this selection, you specify that this rule applies to “the loan” for which data was entered.

The placeholders (such as <a loan>) represent generic objects, while “the loan” represents a specific object or set of information that is provided to the rule engine. The information is passed through either a variable or a ruleset parameter.

- \_\_\_ b. Press the Space bar and select **is more than <a number>** and select **<number>**.
- \_\_\_ c. Delete **0** and enter: **1000000**
- \_\_\_ 3. Build the action statement in the “then” section.
  - \_\_\_ a. Click **<action>** and select **in <a report>, reject the data with the message <a string>**, and then select ‘**the loan report**’.
  - \_\_\_ b. Click **<a string>** and select **<string>**.
  - \_\_\_ c. Enter the following message between the quotation marks (""):  
The loan cannot exceed 1000000
- \_\_\_ 4. Finish creating the rule and provide a comment that describes your work.
  - \_\_\_ a. Click **Save**.
  - \_\_\_ b. In the Create New Version window, enter the following message, and click **Create New Version**:  
Implemented new version of the policy.



- \_\_\_ 5. Click **main** to return to the main branch page for `my-validation-rules-service`.

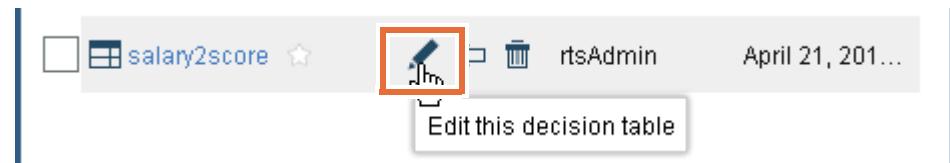
## 2.4. Editing a decision table

You now must implement the final major change to the policy: the credit score for borrowers that have a yearly income of \$170000 - \$200000 must be increased by 280 points.

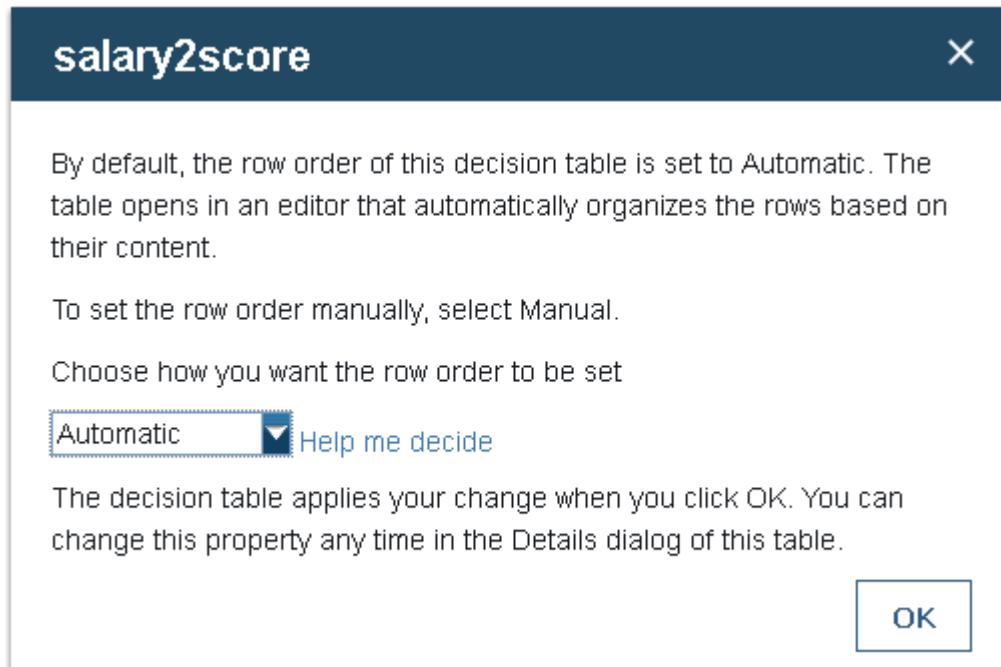
In the initial policy, which is reflected in the `salary2score` decision table, that income range adds 250 points. The initial policy also does not have a separate row for incomes of \$170000 - \$200000.

- \_\_\_ 1. Open the `salary2score` decision table and add a row in the rule editor.
  - \_\_\_ a. In the left explorer pane, click the **computation** folder.

- \_\_\_ b. In the list of rules, hover the mouse over the **salary2score** rule and click the **Edit this decision table** (pencil) icon.



- \_\_\_ c. In the “salary2score” window that opens, keep the default settings and click **OK**.



This window is used to choose the editor and specify how rows are sorted.

- \_\_\_ 2. Add a row in the rule editor, by right-clicking row 7 and clicking **Insert Row > Below**.

7	120,000	200,000	
8		$\geq 200,000$	
9			
10			
11			<div style="border: 1px solid #ccc; padding: 5px; display: inline-block;"> <span>Insert row</span> ▾         </div>
12			Above
13			Below
14			

- \_\_\_ 3. Edit the condition to reflect the business policy requirements.

- \_\_\_ a. Double-click the **Min** column for the new row 8, and enter: 170000

- \_\_\_ b. Click the **Max** column for the new row 8, and enter: 200000
- \_\_\_ 4. Edit the action cell.
  - \_\_\_ a. Double-click the action cell, which is in the “Add to customized credit score” column.
  - \_\_\_ b. Enter: 280
- The new row 8 is now complete, but the table now has a range overlap in the condition cell in row 7.
- \_\_\_ 5. Edit the condition cell in row 7 to remove the range overlap.
  - \_\_\_ a. Double-click the **Max** condition cell of row 7.
  - \_\_\_ b. Change the value of **200000** to: 170000

This change eliminates the overlap of the ranges in rows 7 and 8.

	Yearly income	
	Min	Max
1	< 10,000	
2	10,000	20,000
3	20,000	30,000
4	30,000	50,000
5	50,000	80,000
6	80,000	120,000
7	120,000	170,000
8	170,000	200,000
9	≥ 200,000	

- \_\_\_ 6. Finish editing the `salary2score` decision table and provide a comment that describes your work.
  - \_\_\_ a. Click **Save**.
  - \_\_\_ b. In the Create New Version window, type the following comment.  
Added a new income range of 170000 to 200000 to add 280 points.
  - \_\_\_ c. Click **Create New Version**.

You are now finished updating the action rules to implement the new business policies.

## 2.5. Posting a reply to Paul and logging out

- \_\_\_ 1. Post a message to Paul stating that you completed implementing the rule changes.
  - \_\_\_ a. In the top banner, click **Home**.

- \_\_\_ b. In the **Home** tab, click **Stream**.
- \_\_\_ c. Find the message from Paul and click **Comment** to open the comment field.

▼ Today

 **Paul** created a new post  
Hi, Bea. There is a new policy. Can you implement it by Friday? Thanks.  
12:43:18 PM

**Comment**

- \_\_\_ d. Enter a message to Paul stating that you updated the rules and click **Post**.

Hi, Paul. I completed the changes to the rules according to the new policy.

Your comment appears in the Stream.

▼ Today

 **Paul** created a new post  
Hi, Bea. There is a new policy. Can you implement it by Friday? Thanks.  
12:43:18 PM

**Comment**

 You added a comment  
Hi, Paul. I completed the changes to the rules according to the new policy.  
4:10:19 PM

*Post a new comment here*

- \_\_\_ 2. Log out as Bea.

## Section 3. Working with the Business console management features

In this part of the exercise, you learn about the various management activities that you can do in the Decision Center Business console, including viewing a timeline and creating a snapshot.

You work as Paul to create and compare a snapshot, and review the timeline of events for an action rule.

### 3.1. Logging in as Paul and reviewing the changes to my-validation-rules-service

- \_\_\_ 1. Log in to the Business console as Paul.
  - \_\_\_ a. Enter Paul into the **Username** and **Password** fields, and click **Log in**.
  - \_\_\_ b. Make sure that you are on the **Home** tab.
  - \_\_\_ c. In the **What's New** tab, notice the message from Bea in the Activity Stream notifying you of the changes to the rules in `my-validation-rules-service`.

The screenshot shows a section titled "New Comments in Activity Stream". It contains one item from Bea, dated May 3, 2017, at 4:10 PM. The message reads: "Hi, Paul. I completed the changes to the rules according to the new policy." There is a small envelope icon next to the message.

- \_\_\_ 2. Review the rule changes in `my-validation-rules-service`.
  - \_\_\_ a. In the **What's New** tab, go to the "New Updates on Followed Rules" section.
  - \_\_\_ b. Hover the mouse pointer over the `salary2score` decision table to see Bea's comment.

The screenshot shows a section titled "New Updates on Followed Rules". It lists an update for the `salary2score` decision table, updated by Bea in the `main` branch on May 3, 2017, at 4:10 PM. The message states: "Added a new income range of 170000 to 200000 to add 280 points." Below this, there is another entry for the `checkAge` decision table.

- \_\_\_ c. Click the **salary2score** link to open the decision table and verify the change.
- \_\_\_ d. Click the **main** breadcrumb to return to the **Decision Artifacts** tab.

- \_\_ e. In the left explorer pane, click the **validation** folder and note the rules that were last changed by Bea.

Name	Last Changed By
checkAge	Bea
checkAmount	Bea
checkName	rtsAdmin
checkSSNareanumber	rtsAdmin
checkSSNdigits	rtsAdmin
checkZipcode	rtsAdmin

### 3.2. Working with snapshots

Paul determines that now is a good time to create another snapshot of the decision service.

- \_\_ 1. Create a snapshot of the `my-validation-rules` project called `NewPolicy`.
- \_\_ a. On the toolbar, click **Take Snapshot**.

- \_\_\_ b. In the “Take a Snapshot” window, type **New Policy** in the **Snapshot Name** field, enter the following description:

Includes Bea's updates for the policy changes.

The screenshot shows a modal dialog titled "Take a Snapshot". Inside, there is a "Snapshot Name" field containing "New Policy" and a "Description" field containing "Includes Bea's updates for the policy changes." At the bottom right of the dialog is a blue "Create" button.

- \_\_\_ c. Click **Create**.

## Viewing snapshots

- \_\_\_ 1. To look at the different snapshots for `my-validation-rules-service`, click the **Snapshots** tab.

The screenshot shows the main interface of the Decision Center. At the top, there is a header with a project name and three icons: "Merge Branches", "Take Snapshot", and "Timeline". Below the header, there is a navigation bar with tabs: "Schemas" (selected), "Simulations", "Deployments", "Snapshots" (which has a red box around it), and "Model". A reminder icon (a blue circle with a white bell and document) is on the left, and a right-pointing arrow is on the far right.



You can use the right arrow to access the other tabs in Decision Center.

- 2. In the list of snapshots, you see the New Policy snapshot and the Initial Policy snapshot.

The screenshot shows a navigation bar with tabs: Projects, Simulations, Deployments, Snapshots (which is highlighted), and Model. Below the navigation bar is a search bar labeled 'Filter:' with a magnifying glass icon. The main content area displays a list of snapshots. The first snapshot is 'Initial Policy', indicated by a folder icon, with the text 'Created by Paul on May 3, 2017'. The second snapshot is 'New Policy', also indicated by a folder icon, with the same creation date. Both entries have a small blue square icon to their left.

- 3. Click New Policy to open the snapshot.

In the snapshot page, you can:

- Compare the snapshot with another snapshot
- Restore the project to the state captured in the snapshot
- View details of the snapshot, including who created the snapshot and linked projects

## Comparing snapshots

You can compare snapshots to see the details between different project states side by side. By comparing snapshots, you can ensure that the right version of each rule is included so that the snapshot can be deployed. You can also note the date and time when changes are made.

- 1. Compare the New Policy snapshot with the InitialPolicy snapshot.

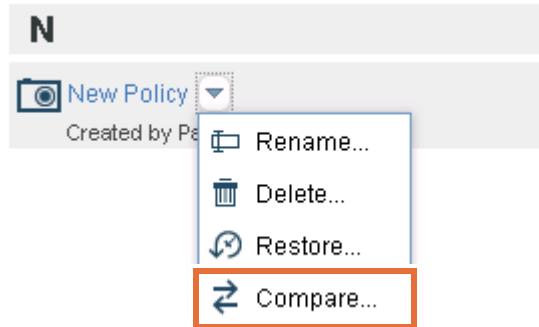
- a. In the New Policy snapshot page, click **Compare**.





## Information

You can also compare snapshots from the main **Snapshots** tab. Hover the mouse pointer over the snapshot name, click the action menu, and click **Compare**.



- b. In the Compare Snapshots window, click **Initial Policy** to select it, and click **Compare**.

**Compare Snapshots**

Select the two items to compare

Filter:

**Current State of the Project**

**Initial Policy**  
Created by Paul on May 3, 2017 at 11:23 AM

**New Policy**  
Created by Paul on May 3, 2017 at 4:28 PM

**Compare**

A table shows the versions, user, and date information for the rules in the snapshots. The table also shows information about added and updated rules.

For example, the comparison between NewPolicy and InitialPolicy shows that one rule was added, and two rules were updated.

	Initial Policy	New Policy
Created by	Paul	Paul
Created on	May 3, 2017	May 3, 2017
<b>checkAge</b>	v1.0 Created by rtsAdmin Apr 21, 2017	v1.1 Last updated by Bea May 3, 2017
<b>salary2score</b>	v1.0 Created by rtsAdmin Apr 21, 2017	v1.1 Last updated by Bea May 3, 2017

- \_\_\_ 2. Take a couple of minutes to explore the snapshot comparison.
- \_\_\_ 3. When you are finished exploring the comparison between the two snapshots, click the **main** breadcrumb.

### 3.3. Viewing the timeline

The Business console timeline shows the activities, and includes the type of change, the person who made the change, and the comments that are added.

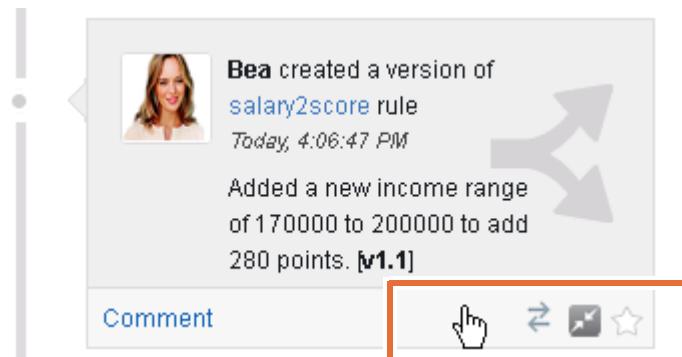
In this part of the exercise, you use the timeline to review the activities that took place in my-validation-rules-service.

- \_\_\_ 1. On the **main** toolbar, click **Timeline**.



All the changes that Bea and Paul made are displayed, with the most recent change listed first. Notice that the **Timeline** icon changes to the **Exit Timeline** icon.

- 2. Hover the mouse pointer over the bottom of one of the timeline activities to access the following features.



- Use **Comment** to post a comment on the activity.
  - Use the **Compare** icon to compare the version of the element with its previous version.
  - Click the star to follow or unfollow the element.
- 3. Take a couple of minutes to explore the features of the timeline, and when you are finished, click **Exit Timeline**.



- 4. Log out of the Business console and close the web browser window.

## End of exercise

## Exercise review and wrap-up

In this exercise, you learned how to work with the Business console rule authoring and management features. In the first part of the exercise, you logged in to the Business console as Paul, who is a manager. As Paul, you review the rule project and the updated policy. Next, you assigned Bea to implement the changes to the rules. In the second part of the exercise, you logged in to the Business console as Bea, who is a rule author. As Bea, you changed the rules according to the updated policy. In the last part of the exercise, you logged back in to the Business console as Paul to create and compare snapshots, and to view the timeline of project activities.

# Exercise 7. Understanding the case study

## Estimated time

00:30

## Overview

This exercise and the next several exercises focus on rule authoring. This exercise provides an overview of the business domain and object model that you work with during the rule authoring exercises.

## Objectives

After completing this exercise, you should be able to:

- Describe the business domain
- Describe the business policies and vocabulary requirements for this domain

## Introduction

This exercise involves reading through the *Understanding the Case Study* book that is provided with your course materials.

In this exercise, you look through the case study materials to learn more details about the car insurance scenario and its supporting documents.

The case study is the basis for many of the exercises that you work with during this course. For example, you worked with simplified materials from the case study in the rule modeling and implementing exercises, and the next set of rule authoring exercises are also based on the case study.

By becoming familiar with the case study documentation, you can better understand how to write the rules that implement these policies. The same principles apply to any rule project: to write rules effectively, you must be thoroughly familiar with the object model and vocabulary.

## Requirements

This exercise requires you to read the *Understanding the Case Study* book, and to look at the supporting case file documents on the computer lab environment.

A PDF-format copy of the case study, `CaseStudy.pdf`, is stored in the `<labfilesDir>\CaseStudyExercises` folder.



## Reminder

In the computer lab environment that is prepared for this course, `<labfilesDir>` is: `C:\labfiles`

---

The `CaseStudy.pdf` is also provided along with the Student Notebook and Student Exercises books for this course.

## Section 1. Reading the case study

Some of the exercises in this course are based on the car insurance domain. The business policies that you use to capture the rules are in the *Understanding the Case Study* book.

The car insurance case study is the basis for the modeling and rule authoring exercises in this course. You were introduced to some of the case study concepts in the rule modeling and implementing exercises. This exercise provides an opportunity to learn more details about the case study and its supporting materials.

Take some time to get familiar with the files that are used to support the exercises.

- \_\_\_ 1. In the computer lab environment, go to the `<labfilesDir>\CaseStudyExercises` directory and note the contents.

You use these files in the rule discovery and analysis exercises.

- \_\_\_ a. The `<labfilesDir>\CaseStudyExercises` directory has the following documents:
  - CaseStudy.pdf
  - start-decision-points.rtf
  - start-rule-analysis-atomic.rtf
  - start-rule analysis-dependency.rtf
  - start-rule-analysis-overlap.rtf
  - start-rules-template.rtf
  - start-use-case.rtf
- \_\_\_ b. The `<labfilesDir>\CaseStudyExercises` directory also includes the following folders and files:
  - **solutions:** This folder contains answer files for the rule discovery and analysis exercises.
    - answer-decision-points.rtf
    - answer-rule-analysis.rtf
    - answer-rules.rtf
    - answer-use-case.rtf
  - **usecases:** This folder contains use case information for the rule discovery and analysis exercises.
    - UseCase1-Main.rtf
    - UseCase2-Validate.rtf
    - UseCase3-Underwrite.rtf

- \_\_\_ 2. Open the *Understanding the Case Study* book.

You can use the copy that is included in the computer lab environment, or the copy that you received with the student exercises and student notebook.

- \_\_\_ 3. Note the table of contents in the *Understanding the Case Study* book.

- \_\_\_ 4. Read the introductory sections in *Understanding the Case Study*.
- \_\_\_ 5. In *Understanding the Case Study*, read "[Discovering vocabulary](#)" on page 1-8, and take some time to consider these questions.
  - What types of objects are included for this business domain?

---

---

---

---

---

- Which terms and phrases do you expect to be part of the rule vocabulary?  

---

---

---

---

---
- What is the relationship between the request object and other objects in the class diagram?  

---

---

---

---

---

- \_\_\_ 6. In the case study, read "[Eligibility rules](#)" on page 1-12.

The business policy descriptions in this case study are the basis for the exercises that you work through next.

## End of exercise

## Exercise review and wrap-up

In this exercise, you reviewed the car insurance case study and the supporting case study files.

# Exercise 8. Discovering rules

## Estimated time

01:30

## Overview

This exercise continues the focus on rule authoring. You learn suggested practices for discovering rules.

## Objectives

After completing this exercise, you should be able to:

- Elicit decisions from use cases
- Transform business policy into formal rule statements

## Introduction

The premise of this exercise is to walk through the steps of rule capture that you can expect to do in your own organization.

Discovery starts with identifying decisions in use cases, mapping those decisions to business policies, and then formalizing those decisions as actual rule statements that can be implemented in the tool.

This exercise is based on the car insurance scenario that you reviewed in [Exercise 7, "Understanding the case study"](#), and involves these tasks:

- [Section 1, "Identifying decisions in use cases"](#)
- [Section 2, "Creating a decision-points table"](#)
- [Section 3, "Formalizing raw decisions as rule statements"](#)

## Requirements

This exercise requires you to use files that are stored on the computer lab environment.

## Before starting the exercise

The exercises in this unit are based on the car insurance domain. The business policies that you use to capture the rules are in the *Understanding the Case Study* book.

Before you start this exercise, take a moment to review the files that are used to support the rule discovery and analysis exercises.

- 1. Open the `<labfilesDir>\CaseStudyExercises` folder to see the supporting files that you use during the exercises, including:

- `solutions` (folder)
  - `usecases` (folder)
  - `CaseStudy.pdf`
  - `start-decision-points.rtf`
  - `start-rule-analysis-atomic.rtf`
  - `start-rule-analysis-dependency.rtf`
  - `start-rule-analysis-overlap.rtf`
  - `start-rules-template.rtf`
  - `start-use-case.rtf`
- 



### Reminder

In the computer lab environment that is prepared for this course, `<labfilesDir>` is: `C:\labfiles`

---

The exercise instructions here tell you which files to use as a starting point.

The answers are provided in the `<labfilesDir>\CaseStudyExercises\solutions` folder.

---



### Hint

During the exercises, you can copy text from the `CaseStudy.pdf` file into the exercise files.

---

## Section 1. Identifying decisions in use cases

The goal of this exercise is to demonstrate how decisions are extracted from use cases.

---



### Information

For this exercise, the business process model and the use cases are provided.

---

- 1. If needed, review the introductory sections in *Understanding the Case Study*.
  - 2. Read the use cases that are provided in the `<labfilesDir>\CaseStudyExercises\usecases` folder:
    - `UseCase1-Main.rtf`
    - `UseCase2-Validate.rtf`
    - `UseCase3-Underwrite.rtf`

These use cases can help you understand how to complete the use case template during this exercise.
  - 3. Open the `start-use-case.rtf` file in the `<labfilesDir>\CaseStudyExercises` folder.  
For this exercise, you complete the template in the `start-use-case.rtf` file for “Use Case 4: Check eligibility.”
  - 4. To complete the **Decision** column of the use case, look for verb cues that indicate decisions.
- 



### Hint

For examples of how to complete the **Decision** column, see the use cases that are included in the `<labfilesDir>\CaseStudyExercises` folder.

---

- 5. Compare your answers with the answers that are provided in the `answer-use-case.rtf` solution file in the `<labfilesDir>\CaseStudyExercises\solutions` folder.

## Section 2. Creating a decision-points table

The purpose of this exercise is to demonstrate how to identify the raw data that is used in a decision and how to turn that data into formal rules.

In this exercise, you create a decision points table to map the decisions that are identified in the use case to relevant “raw” business policies.



### Hint

For help to understand insurance vocabulary and definitions, see [Appendix A, "Terminology"](#) in the case study.

- 1. In the case study, read "[Eligibility rules](#)" on page 1-12.
- 2. Open these files in the <labfilesDir>\CaseStudyExercises folder:
  - start-decision-points.rtf
  - CaseStudy.pdf

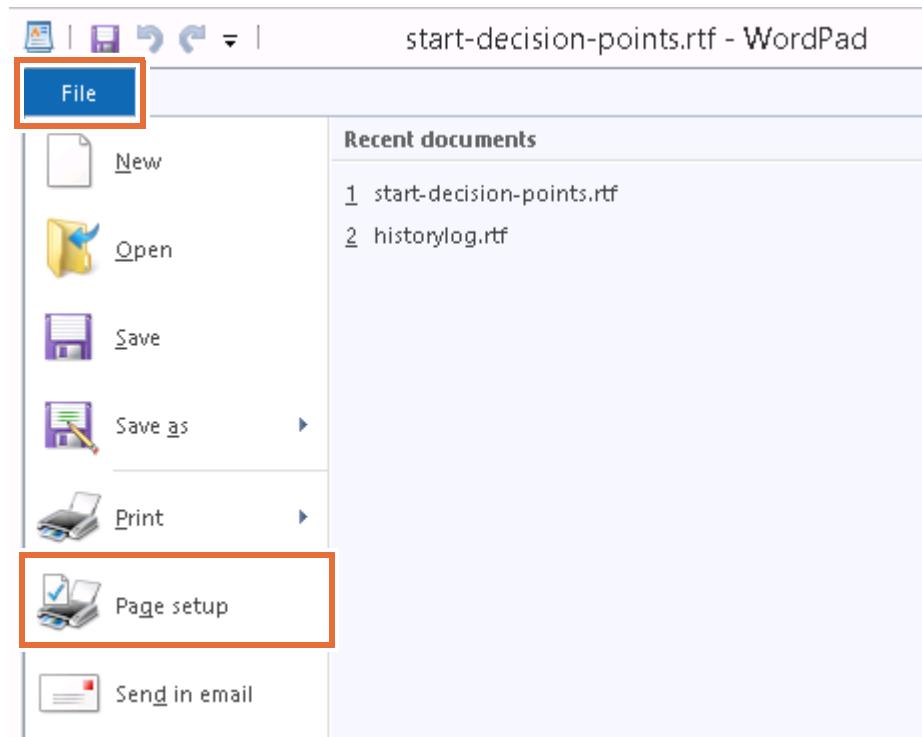


### Information

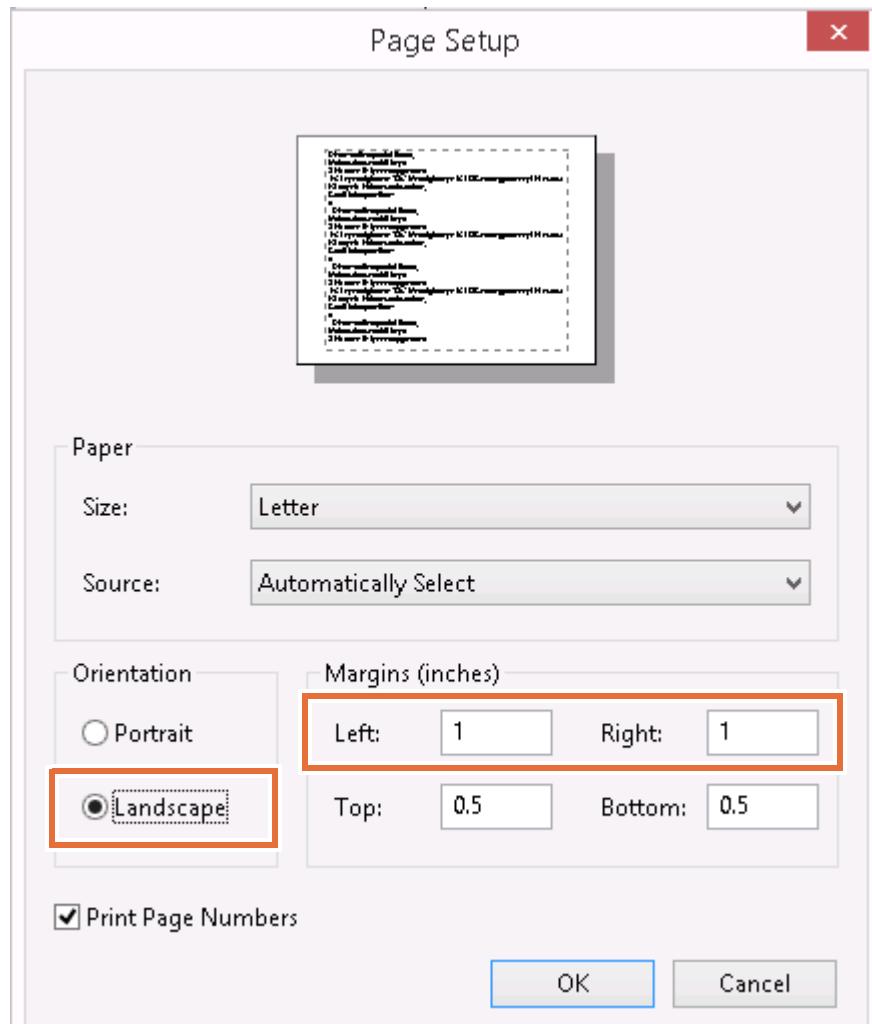
#### Changing the page orientation in WordPad

The `start-decision-points.rtf` file should open in WordPad. If you do not see all the content, change the page orientation to landscape, and adjust the size of the right and left margins.

- Click **File**, and then click **Page setup**.



- In the Orientation section, select **Landscape**.
- In the Margins section, change the value in the **Left** and **Right** fields to: 0.5



- Click **OK**. The page content should display correctly.
- 
3. Look at the examples of “raw” business rules in the **Raw business rules** column of the file. For example, look at the `Check policy type` decision. The raw text for this decision is taken directly from the policies that are described in the case study. For this part of the exercise, you are looking for the business policy only. In the next exercise, you transform the raw text into formal rule statements.
4. In the `CaseStudy.pdf` file, find the relevant text fragments to complete the **Raw business rules** column for the decision points listed in the `start-decision-points.rtf` file.



### Important

As you go through the use case, try to understand the circumstances or context of the decisions. Context can help you determine the correct classification of the rules.

- 
- 5. Compare your answers with the answers that are provided in the `answer-decision-points.rtf` solution file.
- 



### Information

If necessary, change the page orientation of the `answer-decision-points.rtf` file to landscape to display all of the content.

For more information about changing the page setup properties in WordPad, see "["Changing the page orientation in WordPad"](#) on page 8-4.

---

## Section 3. Formalizing raw decisions as rule statements

In this exercise, you transform the raw decisions that were identified in the `answer-decision-points.rtf` solution file into formal rules.

- 1. Open the following files.
  - In the `<labfilesDir>\CaseStudyExercises\solutions` folder:  
`answer-decision-points.rtf`
  - In the `<labfilesDir>\CaseStudyExercises` folder: `start-rules-template.rtf`
- 2. In the `start-rules-template.rtf` file, follow the instructions that are provided in the file.
- 3. Use the `Check policy type` rule as an example to help you complete the rule templates for the decisions that are listed in the `answer-decision-points.rtf` file.
- 4. Compare your answers with the `answer-rules.rtf` solution file in the `<labfilesDir>\CaseStudyExercises\solutions` folder.

### End of exercise

## Exercise review and wrap-up

In this exercise, you worked on eliciting decisions from the case study materials. You also worked on transforming business decisions into rule statements.

# Exercise 9. Analyzing rules

## Estimated time

01:30

## Overview

This exercise continues the focus on rule authoring. You learn suggested practices for analyzing rules that are based on the case study materials.

## Objectives

After completing this exercise, you should be able to:

- Make rules atomic
- Eliminate overlap, redundancy, and inconsistency
- Identify dependencies between individual rules and rule families

## Introduction

After discovering the rules, the next goal is to make sure that the rules form a complete and coherent set of rules. Each rule must be analyzed both individually and as part of the set of rules.

As you work through the steps in this exercise, consider how you can apply these rule analysis practices to the rulesets and domains in your own organization.



### Note

Rule analysis does not involve validation, optimization, or implementation.

This exercise is based on the car insurance scenario case study, and involves these tasks:

- [Section 1, "Making the rules atomic"](#)
- [Section 2, "Removing redundancies, overlaps, and inconsistencies"](#)
- [Section 3, "Identifying dependencies"](#)

## Requirements

This exercise uses files that are stored on your computer lab environment.

## Section 1. Making the rules atomic

During this exercise, you start the analysis of the rules that you identified in [Exercise 8, "Discovering rules,"](#) on page 8-1.

- 
- 1. Open the following files.
    - In the `<labfilesDir>\CaseStudyExercises\solutions` folder: `answer-rules.rtf`
    - In the `<labfilesDir>\CaseStudyExercises` folder: `start-rule-analysis-atomic.rtf`
- 



### Reminder

In the computer lab environment that is prepared for this course, `<labfilesDir>` is: `C:\labfiles`

---

- 2. In the `start-rule-analysis-atomic.rtf` file, follow the instructions.
- 



### Hint

Only two of the highlighted rules in step 6 of the `start-rule-analysis.rtf` file are *not* atomic.

---

- 3. Compare your answers with the `answer-rule-analysis.rtf` solution file in the `<labfilesDir>\CaseStudyExercises\solutions` folder.

## Section 2. Removing redundancies, overlaps, and inconsistencies

During this exercise, you analyze the subtasks of the eligibility rules. You also verify that there are no overlaps or redundancies between these tasks. To avoid ambiguity or gaps, some rules might need to be clarified, redefined, or even added.

As explained in the rule document `answer-rules.rtf` in the `<labfilesDir>\CaseStudyExercises\solutions` directory, the rules are presented according to use case, and are grouped into these rule families:

- Policy eligibility
- Driver eligibility
- Vehicle eligibility
- Primary Driver eligibility
- Options eligibility

For example, you might expect that the policy eligibility rules apply only to the policy eligibility family, but some policy rules apply also to the driver or the vehicle. Therefore, you must check for rules that refer to objects outside the rule family that might potentially cause problems.

- \_\_\_ 1. Open the following files.
  - In the `<labfilesDir>\CaseStudyExercises\solutions` folder: `answer-rules.rtf`
  - In the `<labfilesDir>\CaseStudyExercises` folder: `start-rule-analysis-overlap.rtf`
- \_\_\_ 2. In the `start-rule-analysis-overlap.rtf` file, follow the instructions.
- \_\_\_ 3. Compare your answers with the `answer-rule-analysis.rtf` solution file in the `<labfilesDir>\CaseStudyExercises\solutions` folder.

## Section 3. Identifying dependencies

- 1. Open the following files.
  - In the `<labfilesDir>\CaseStudyExercises\solutions` folder: `answer-rules.rtf`
  - In the `<labfilesDir>\CaseStudyExercises` folder:  
`start-rule-analysis-dependency.rtf`
- 2. In the `start-rule-analysis-dependency.rtf` file, follow the instructions.
- 3. Compare your answers with the `answer-rule-analysis.rtf` solution file in the `<labfilesDir>\CaseStudyExercises\solutions` folder.

**End of exercise**

## Exercise review and wrap-up

The exercise looked at how to analyze rules to eliminate problems before implementation.

# Exercise 10. Working with conditions in rules

## Estimated time

01:30

## Overview

This exercise continues the focus on rule authoring. You learn how to identify the appropriate BAL constructs to use in your condition statements when you write rules.

## Objectives

After completing this exercise, you should be able to:

- Identify BAL constructs that apply to specific types of conditions
- Use the BAL reference documentation to look up constructs and syntax
- Use the correct BAL constructs to define condition statements in rules

## Introduction

The activities in this exercise are designed to help you recognize which BAL constructs can be useful in building rules. You also learn how to find help on BAL in the product documentation.

In this exercise, you write condition statements, and identify the vocabulary building blocks to best implement the business policies. Hints are provided to help you see how the underlying business model and vocabulary are set up to support rule authoring.

This exercise is based on the car insurance scenario.

The exercise includes these sections:

- [Section 1, "Before starting the exercise"](#)
- [Section 2, "BAL condition constructs"](#)
- [Section 3, "Identifying conditions"](#)
- [Section 4, "Writing comparison conditions"](#)
- [Section 5, "Creating a compound condition"](#)
- [Section 6, "Creating rules with constants and ruleset variables"](#)
- [Section 7, "Updating ruleset variables"](#)
- [Section 8, "Creating a count condition"](#)
- [Section 9, "Creating a count condition with a "where" clause"](#)
- [Section 10, "Advanced practice"](#)
- [Section 11, "Solutions"](#)

The tasks begin with step-by-step instructions to help you get started. The ["Advanced practice"](#) on page 10-44 provides only minimal guidance so that you can test your skills.

---

**Scenario:**

There are a number of circumstances under which the insurance company does not want to provide coverage. Each constraint is implemented as an individual rule. Your tasks during this exercise involve writing the rules that implement business policies to describe when insurance coverage is rejected.

---

## Requirements

This exercise includes some activities on paper. It also requires that you work in the Business console and the Enterprise console on the computer lab environment. The sample server must be running.

You use a training-specific Decision Center database (`rtsdb.h2-training.zip`) that includes the rule projects for this exercise. This database is installed on the computer lab environment for this course.

## Section 1. Before starting the exercise

### Starting the sample server

Before proceeding, make sure that the sample server is running.

- 1. Click **Start**, and then click the **Start Sample Server** shortcut.
- 2. Wait until the server is started.

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



#### Hint

If you can open the Business console in a web browser, the sample server is running. If the browser cannot connect, then the sample server is not running.

### Working with car insurance decision service rule projects

During the rule authoring exercises, you work in the Business console with a decision service rule project that is based on the car insurance case study.

For your convenience, a series of decision services are provided. The following list of decision services are available in your Decision Center Repository:

- carinsurance-conditions-start
- carinsurance-definitions-start
- carinsurance-actions-start
- carinsurance-dt-start

In this exercise, you start with the first decision service: `carinsurance-conditions-start`



#### Information

You can continue to use this decision service throughout the remaining rule authoring exercises. Or, you can change decision services at the beginning of each exercise.

At the start of each exercise, the instructions tell you which decision service to use as the starting point for that exercise.

Before you begin actively working with the car insurance rules, make sure that you complete [Exercise 7, "Understanding the case study,"](#) on page 7-1.

A good understanding of the underlying business object model (BOM) can help you work with the rules in the rule editors.



### Important

Save your rule authoring work frequently while you work in the Business console rule editor. Saving your rule while it is in progress helps you to avoid losing your work if your Business console session times out.

You can save your work in the Compose editor by clicking **Save** and then **Create New Version**. Return to rule editor by clicking **Edit**.

## Section 2. BAL condition constructs

This section contains some examples of Business Action Language (BAL) constructs that are used in condition statements. You can complete the Example column with your own examples of the constructs. Use these tables as a reference for the condition statement exercises.

*Table 1. Comparison condition statements*

Building blocks for comparisons	Description	Example
contains does not contain	Tests that a string contains or does not contain another string	<i>if the name of the customer contains "Smith"</i>
ends with does not end with	Tests that a string ends with or does not end with another string	
equals does not equal	Tests that a number is equal or is not equal to another number	
is is not	Tests that two values or objects are or are not equivalent	
is between <number> and <number>	Tests that a number is within the specified range in which both values are included	
is empty is not empty	Tests that a string does not contain or does contain characters. An empty string contains no characters and is equivalent to ""	
is more than is at least	Value that precedes the operator is greater than, or is greater than or equal to, the value that follows it	

*Table 1. Comparison condition statements*

<b>Building blocks for comparisons</b>	<b>Description</b>	<b>Example</b>
is less than	What precedes the operator is less than, or is less than or equal to, what follows it	
is at most		
starts with		
does not start with	String starts with or does not start with another string	

*Table 2. Count condition statements*

<b>Building blocks for count tests</b>	<b>Description</b>	<b>Example</b>
the number of	Tests that there is a specified number of something	
there are	Tests that the number of objects of a specific type is equal to a specific value	
there are at least	Tests that the number of objects of a specific type is greater than or equal to a specific value	
there are at most	Tests that the number of objects of a specific type is lower than or equal to a specific value	
there is one	Tests that there is only one object of a specific type	

*Table 3. Test for existence condition statements*

<b>Building blocks that test for existence</b>	<b>Description</b>	<b>Example</b>
there is at least one	Tests that at least one occurrence of something exists	
there is no	Tests that there is not even a single occurrence of something	

*Table 4. Evaluate set membership condition statements*

<b>Building blocks to evaluate set membership</b>	<b>Description</b>	<b>Example</b>
is one of	Tests whether something is part of a set	
is not one of	Tests whether something is not part of a set	
contains	Tests whether an object is included in a collection of objects	
does not contain	Tests whether an object is not included in a collection of objects	

## 2.1. Looking up BAL constructs in the product documentation

In this section of the exercise, you learn how to find BAL reference information in the product documentation. You can access the documentation for ODM either online or locally on your computer system.

- 1. If you have internet access, enter the following web address in a browser to open the BAL reference in the product documentation:

[http://www.ibm.com/support/knowledgecenter/SSQP76\\_8.9.0/com.ibm.odm.dserver.rules.ref.designer/lang\\_bal\\_ref\\_topics/tpc\\_bal\\_intro.html](http://www.ibm.com/support/knowledgecenter/SSQP76_8.9.0/com.ibm.odm.dserver.rules.ref.designer/lang_bal_ref_topics/tpc_bal_intro.html)

Continue to step 3.

- 2. If you do not have internet access, you can use the local documentation, which is installed in the course lab environment.
  - a. Go to the Windows Apps page (in the taskbar, click the **Start** icon, then click the down arrow at the bottom of the page).
  - b. In the IBM Operational Decision Manager V8.9 section, click **Start Knowledge Center (local)**.

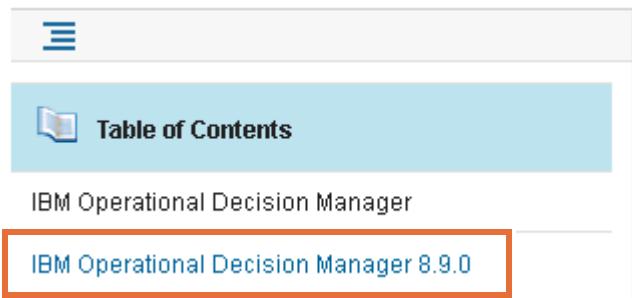


### Information

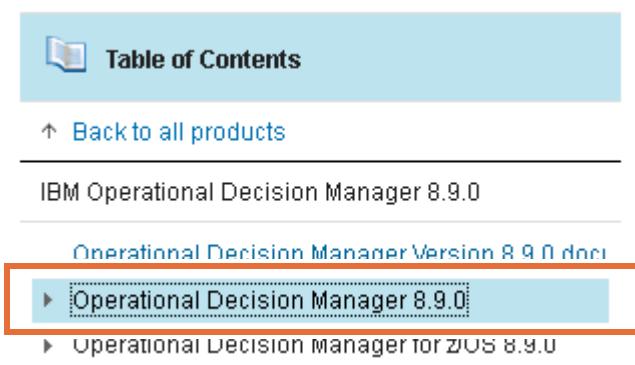
This exercise refers to the locally installed documentation.

If you access documentation on the internet, you might see a slightly different interface. However, the contents are the same.

- c. In the Table of Contents, click **Operational Decision Manager 8.9.0**.



- d. Expand **Operational Decision Manager 8.9.0** to see the list of documentation topics.



- e. Expand **Decision Server Rules > Reference > Rule Designer reference > Rule languages > Business Action Language (BAL)**.

Table of Contents

- ▼ **Decision Server Rules**
  - [Overview: Decision Server Rules](#)
  - ▶ Tutorials
  - ▶ Samples
  - ▶ Developing rulesets in Rule Designer
  - ▶ Designing projects for rule authoring
  - ▶ Testing and simulating rulesets
  - ▶ Building and running rules
  - ▶ Deploying rules from Rule Designer
  - ▶ Developing client applications
  - ▶ Managing rule execution in Rule Execution Server
  - ▶ Rule Execution Server console online help
- ▼ **Reference**
  - ▼ **Rule Designer reference**
    - ▶ Rule Designer API
    - [Rule Designer logging](#)
    - ▶ File types
- ▼ **Rule languages**
  - ▼ **Business Action Language (BAL)**
    - ▶ BAL constructs
    - ▶ BAL operators
    - ▶ BAL literals

- 3. Click **BAL constructs** to see how the information is organized. You see the list of constructs that you can use in your rules, along with the description of each construct.

- 4. Click one of the constructs to see a usage example.

**there are more than (number) (objects)**

This construct tests whether there are more than a specified number of objects of a given type.

**there is at least one (object)**

This construct tests whether there is at least one object of a given type in the current data set.

**there is at most one (object)**

This construct tests whether there is at most one object of a given type in the current data set.

**there is no (object)**

This construct tests whether a data set contains no objects of the given type.

**there is one (object)**

This construct tests whether the current data set contains one and only one object of a given type.

## Section 3. Identifying conditions

- 1. Think about some rules that relate to your business and how they would be phrased in day-to-day business language.
  - a. See some of the rules you noted earlier, or write down one or two rules here:

---

---

---

---

---

- b. Identify the condition part of those rules. Circle or underline the phrase that includes the condition.
- 2. Consider when each of the types of conditions in the following table might be used in a rule at your business, and write down some condition statements of each type.

For a reminder of key phrases, see the tables in [Section 2, "BAL condition constructs"](#) that show the BAL building blocks for each type of condition.

Condition Type	Example in your business context
----------------	----------------------------------

Comparison condition

Count condition

Condition Type	Example in your business context
Existence condition	
Evaluate set membership condition	

## Section 4. Writing comparison conditions

---



### Important

Save your rule authoring work periodically while you work in the Business console rule editor. Saving your rule while it is in progress helps you avoid losing your work if your session times out.

---

Read the business policy description and determine how to formulate this policy as a rule statement in the rule editor.

---

#### Business policy

The insurance company does not provide coverage for classic or antique cars. If a car is over 40 years old, the insurance company considers the car ineligible.

This policy is a mandatory constraint, so the rule action must reject cars that do not meet this eligibility criteria.

---

#### Task

Create the rule that tests the age of the vehicle by following the instructions that are provided here.

---

### 4.1. Creating the “Car age” rule

In this section, you write a rule that implements a rule about the age of a vehicle. The rule is based on the business policy description.

- 1. Open the Business console by the **Decision Center Business console** shortcut from the Start menu, or by entering the following URL in a browser:

`http://localhost:9090/decisioncenter`

---



### Note

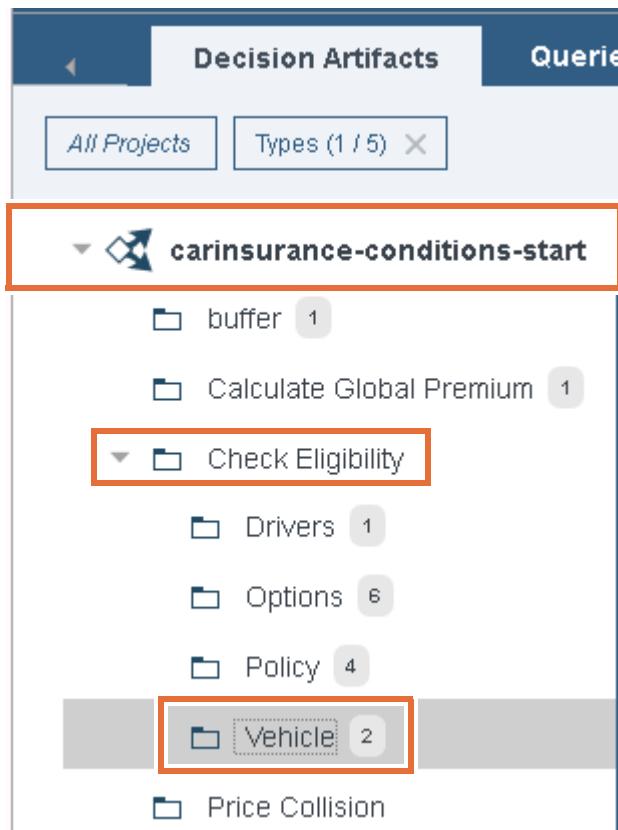
Make sure that you use the correct URL and port for your environment.

- 2. If you are currently signed in to the Business console with either the `rtsAdmin` or the `rtsConfig` user, sign out by clicking the link at the top of the screen.
- 3. Sign in with `rtsUser1` as the user name and password.
- 4. Open the main branch of the `carinsurance-conditions-start` decision service.
  - a. Make sure that you are on the **Library** tab.

- b. In the list of decision services, click the blank space of the **carinsurance-conditions-start** section, and then click **main**.



- 5. Make sure that you are on the **Decision Artifacts** tab.  
 — 6. In the left explorer pane, expand the **Check Eligibility** folder and click the **Vehicle** subfolder.



### Hint

A folder is also called a *rule package*. In Rule Designer, rules are organized in rule packages. When the project is published to Decision Center, rule packages are displayed as folders.

- \_\_\_ 7. In the center pane, you see the contents of the **Vehicle** folder, which includes two rules:
- Total insured value
  - Usage

The screenshot shows the Rational Rules interface with the title bar "Vehicle". Below the title bar are standard file and navigation icons. A "Filter:" input field is present. The main area displays a table of rules:

Name	Last Changed By	Last Changed On
Total insured value	rtsAdmin	April 21, 20...
Usage	rtsAdmin	April 21, 20...

- \_\_\_ 8. Create a rule called **Age**.
- \_\_\_ a. Click the **create an artifact** icon (the plus sign) and click **New Rule**.

The screenshot shows the Rational Rules interface with the title bar "Vehicle". The toolbar includes a "New Rule" icon, which is highlighted with a red box. Other icons include "New Decision Table", "Filter:", and search.

- \_\_\_ b. In the "Create new Rule" window, enter **Age** for the name.
- \_\_\_ c. Click **Create**.

The screenshot shows the "Create new Rule" dialog box. The title bar says "Create new Rule". The main area has a text input field labeled "Specify a name for the new rule." containing the text "Age", which is highlighted with a red box. Below the input field is a message: "Select **Create** to start authoring your rule." At the bottom right is a "Create" button.

The rule editor opens.

## 4.2. Writing the condition statement

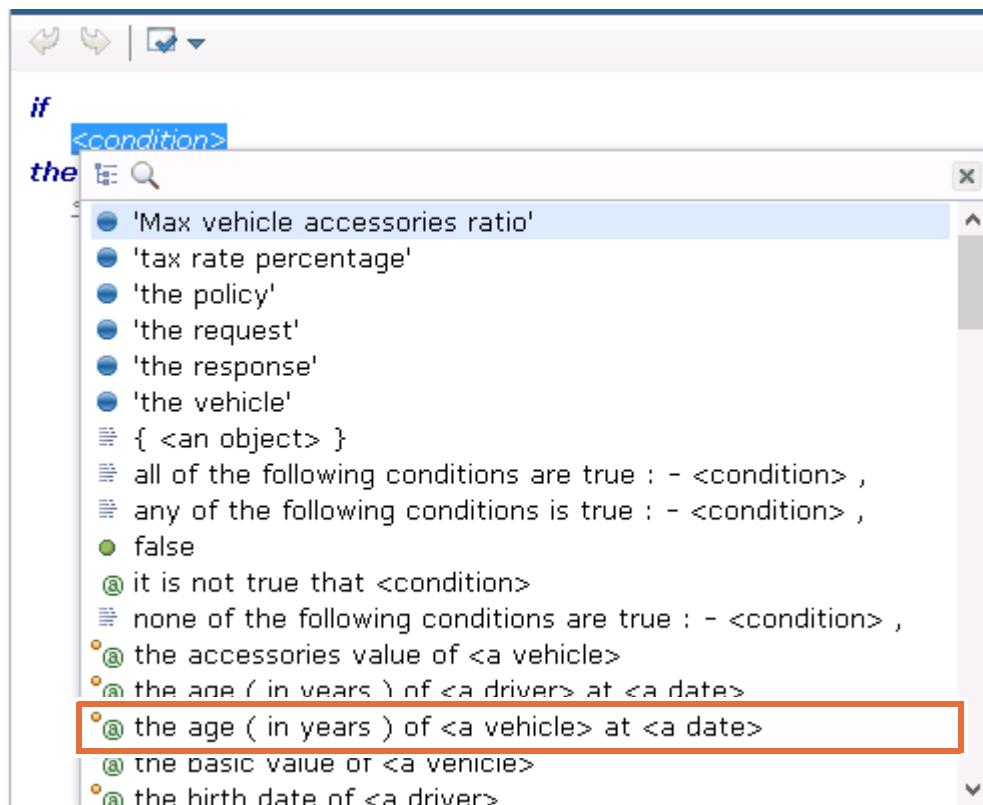


### Information

In the Business console rule editor, you can either click placeholders and select vocabulary from the completion menu, or you can type directly into the editor.

In this exercise, you use placeholders and the completion menu to build the condition and action statements for the rules.

- 1. In the ***if*** part of the rule, click **<condition>**, and click **the age (in years) of <a vehicle> at <a date>**.



- 2. In the selection menu that opens on the phrase **<a vehicle>**, select **'the vehicle'**.



### Note

If the selection menu does not open, you can click **<a vehicle>** to open it.



## Questions

When you change **<a vehicle>**, the rule editor suggests:

- the vehicle
- the vehicle of <a request>

What is the difference between these two vocabulary items?

---



---

## Answer

To answer the question, first look at the BOM in the case study. See "[Discovering vocabulary](#)" on page 1-8 in the case study.

---



---



## Questions

What is the relationship between `Vehicle` and `Request`? Do you see the `Request.vehicle` attribute?

---



---

## Answer

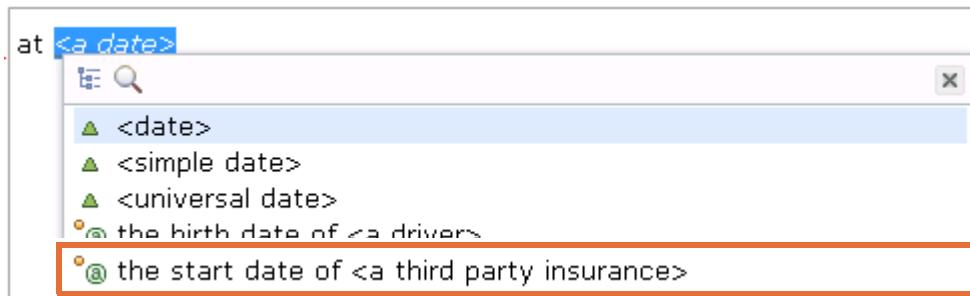
All vehicle information is passed to the rule engine through the `request` input ruleset parameter. The verbalization for `request.vehicle` is: **the vehicle of the request**

This wording can be awkward in a long rule, so the developers wanted to simplify rule authoring for you, and prepared a ruleset variable that is called `vehicle` that maps to the `request.vehicle` attribute. The verbalization for ruleset variable `vehicle` is verbalized as: **the vehicle**

When you see **the vehicle** in the vocabulary list, you know that it is an equivalent, easy-to-use term for: **the vehicle of the request**

---

- \_\_\_ 3. In the **if** statement, click **<a date>** and click **the start date of <a third party insurance>**.



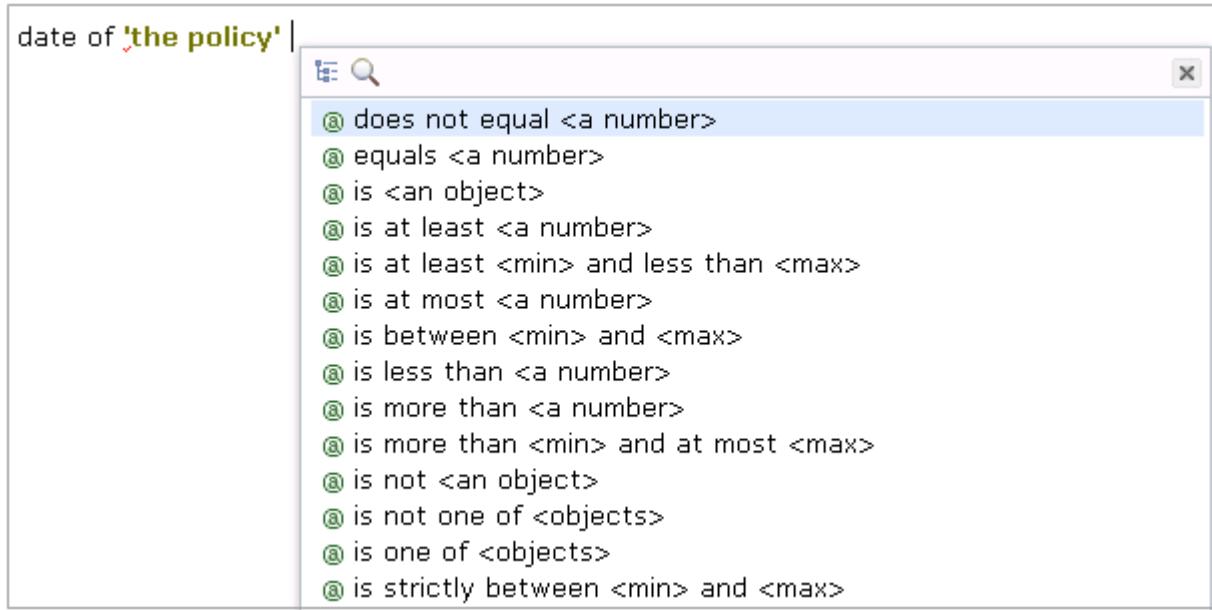
- \_\_\_ 4. In the completion menu that opens on the phrase **<a third party insurance>**, click 'the policy'.



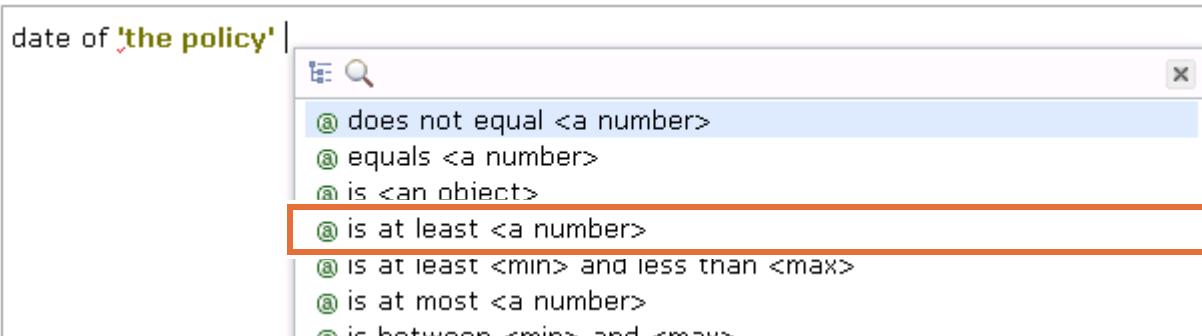
- \_\_\_ 5. Complete the condition to state:

the age ( in years ) of the vehicle at the start date of the policy is at least 40

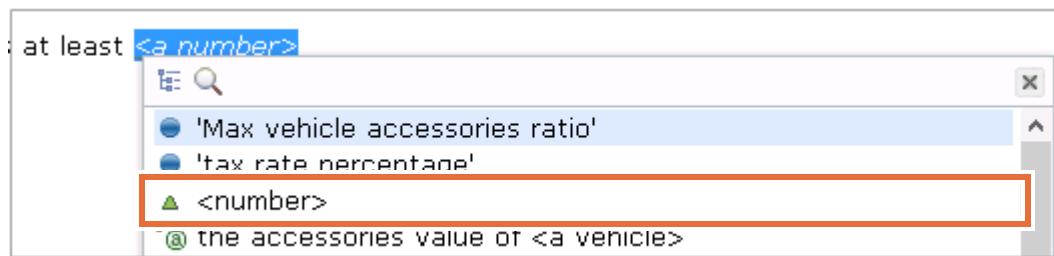
- \_\_\_ a. In the rule editor, the cursor is now directly after the phrase **'the policy'**. Press the space bar and wait for the completion menu.



- \_\_ b. In the menu, click **is at least <a number>**.



- \_\_ c. In the menu that opens on <a number>, select <number>.



- \_\_ d. After you select <number>, a 0 is in the rule editor.  
\_\_ e. Change the 0 to: 40

the start date of 'the policy' is at least 40

The condition statement is now complete.



## Questions

Why does the vocabulary list propose **the policy** and **the policy of <a request>** for **a third party insurance**?

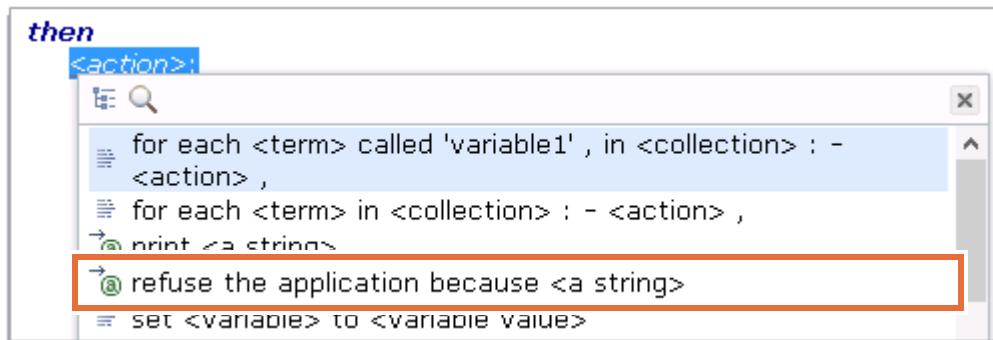
## Answer

Again, look at the BOM and notice the relationship between types of insurance and the Request object. As with the vocabulary for **the vehicle**, the developers prepared a ruleset variable that is called `policy` to simplify wording in rule statements. The ruleset variable is verbalized as: **the policy**

Because the minimum insurance that you can request is to cover third-party liability, the vocabulary uses “a third party insurance” as another term for a basic “insurance policy.”

### 4.3. Completing the rule with the action statement

- 1. In the rule editor, click <action> and click **refuse the application because <a string>**.



#### Hint

You use this same vocabulary item for all the action statements in this exercise.

- 2. In the menu that opens on <a string>, click <string>.  
 — 3. A pair of quotation marks ("") is now in the editor, which indicates that you can enter a text string. Make sure that the cursor is between the quotation marks, and type:

This insurance policy does not cover vehicles that are over 40 years

- 4. Make sure that a semicolon (;) is at the end of the action statement, which now reads:

then

refuse the application because "This insurance policy does not cover vehicles that are over 40 years";

- 5. Click **Save**, and in the Create New Version window, click **Create New Version**.

The Business console opens on the Rule view for the Age rule, which is at version 1.0.



#### Questions

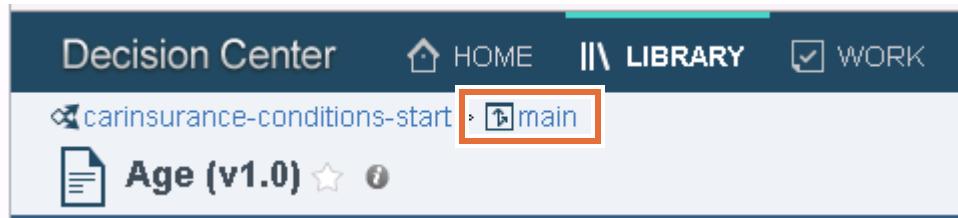
What type of condition did you create?

---

#### Answer

Comparison ("is at least").

- 6. Click the **main** breadcrumb to return to the main page for the carinsurance-conditions-start decision service.



## Section 5. Creating a compound condition

Based on the business policy description that is provided, use the appropriate BAL constructs to build the condition statements that are required in this rule.

---

### Business policy

---

The insurance company refuses to provide third-party coverage only on vehicles that are not owned outright. If the car is financed or leased, the insurance company insists that the customer purchase collision insurance.

---

### Task

---

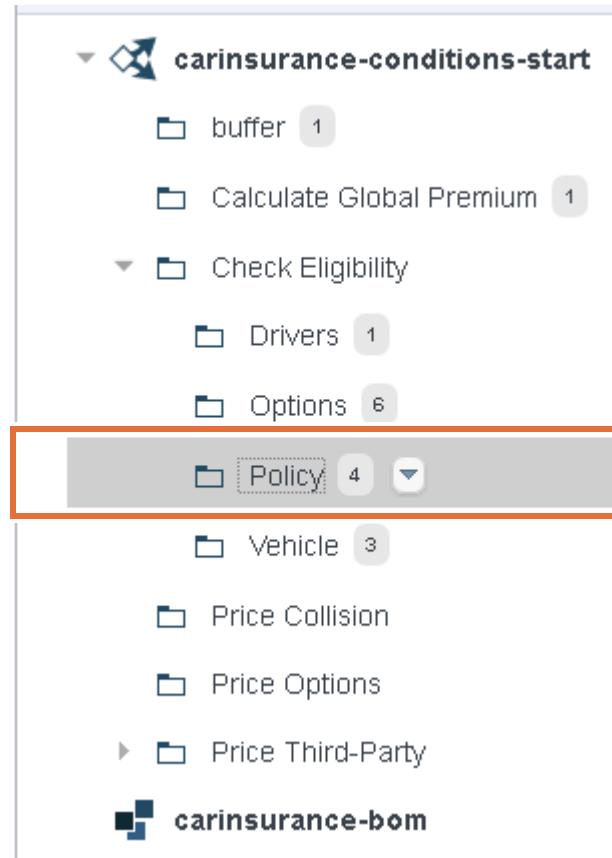
Create a rule that tests whether the vehicle is owned, financed, or leased, and that tests the type of insurance policy requested. The action statement rejects coverage for vehicles that are not owned, and provides an explanation.

---

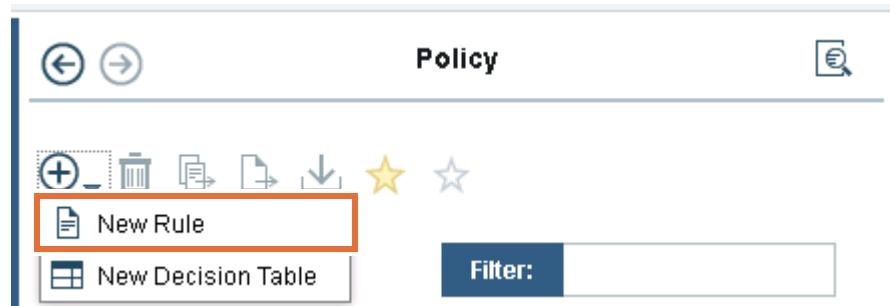
### 5.1. Creating the “Policy type” rule

- 1. In the main branch page for the `carinsurance-conditions-start` decision service, make sure that you are on the **Decision Artifacts** tab.
- 2. In the left explorer pane, make sure that the **Check Eligibility** folder is expanded.

- \_\_\_ 3. Under the **Check Eligibility** folder, click the **Policy** subfolder.



- \_\_\_ 4. Create a rule that is called `Check ownership`.
- \_\_\_ a. In the Center pane, click the **create an artifact** icon (the plus sign) and click **New Rule**.



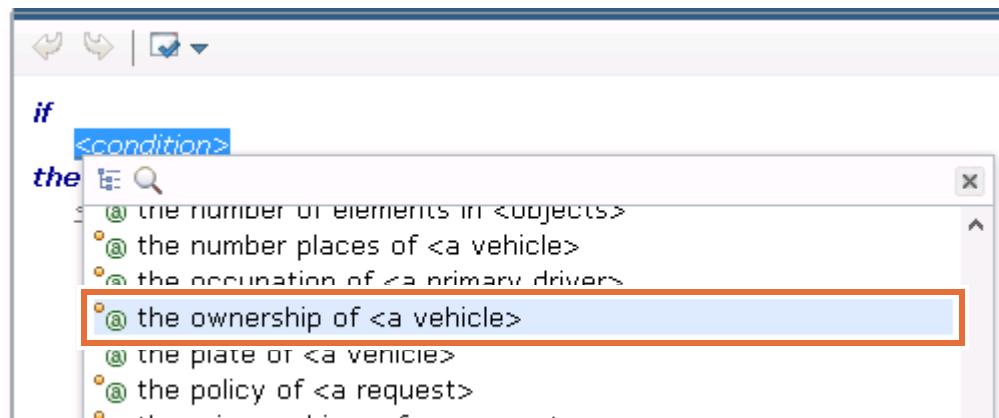
- \_\_\_ b. In the "Create new Rule" window, enter `check ownership` for the rule name.
- \_\_\_ c. Click **Create**.

## 5.2. Writing the condition statement that tests whether the car is financed or leased

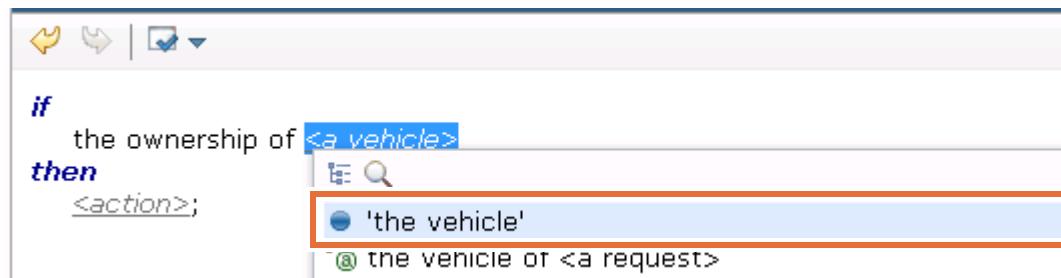
In this section, you author the following condition statement:

```
if
    the ownership of 'the vehicle' is one of { Financed by credit, Leased }
    and the type of the policy is ThirdParty
```

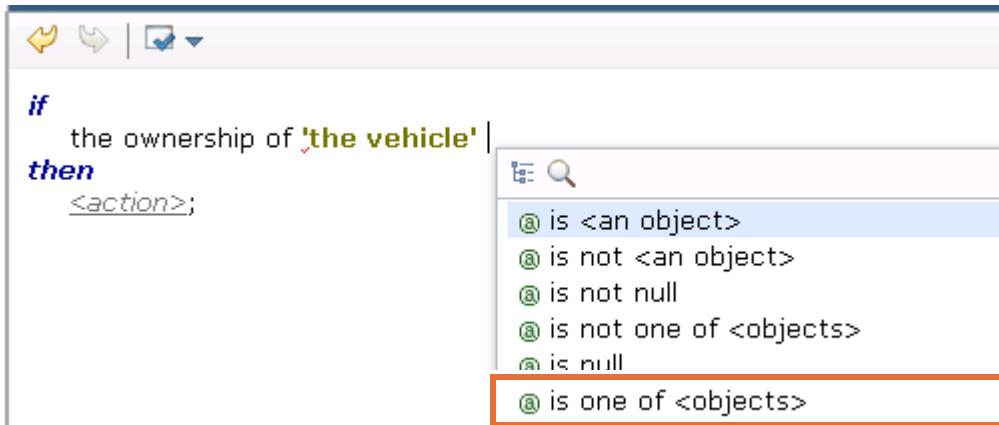
- 1. In the **if** part of the rule, click **<condition>**, scroll down the list, and click **the ownership of <a vehicle>**.



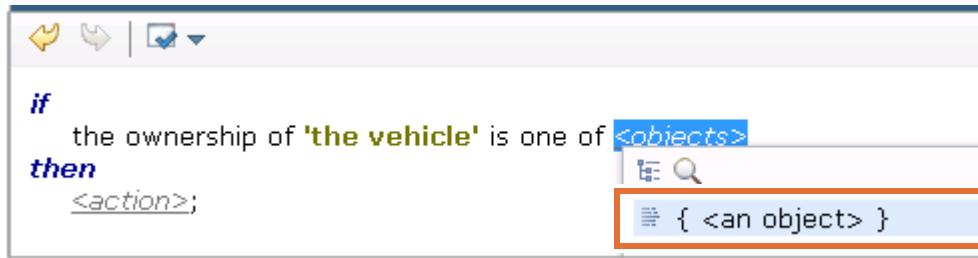
- 2. In the menu that opens on **<a vehicle>**, select '**the vehicle**'.



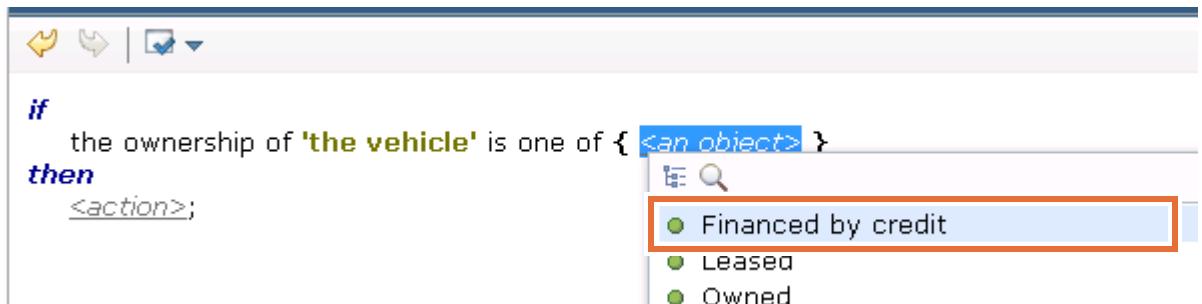
- 3. The cursor is now directly after the phrase '**the vehicle**'. Press the Space bar and wait for the completion menu.
- 4. In the menu, click **is one of <objects>**.



- \_\_\_ 5. In the menu that opens on <objects>, select { <an object> }.



- \_\_\_ 6. In the menu that opens on { <an object> }, select **Financed by credit**.

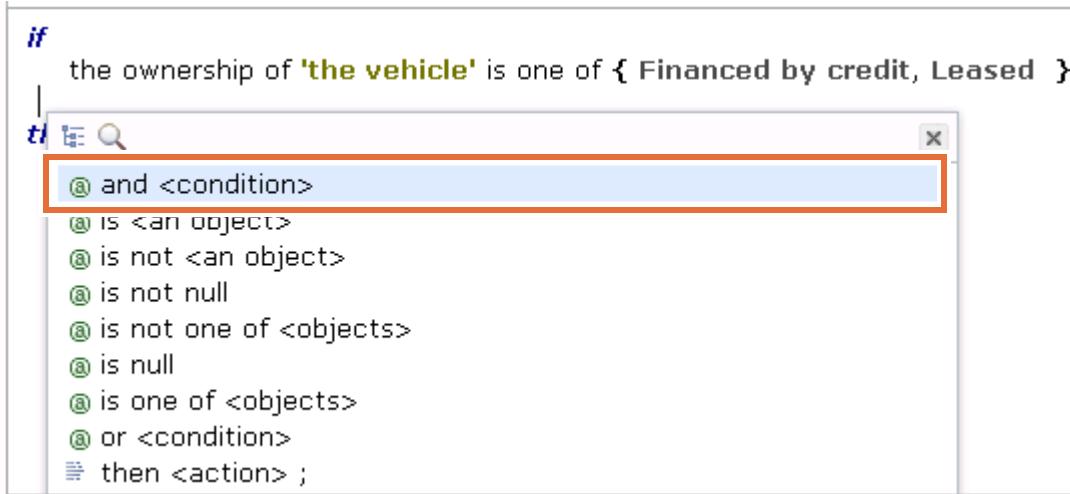


- \_\_\_ 7. The cursor is directly after the word **credit**. Enter a comma (,), press the Space bar, and wait for the completion menu.  
 \_\_\_ 8. In the completion menu, select **Leased**.  
 \_\_\_ 9. Close the completion menu that opens after you select **Leased** by clicking the X in the upper-right corner.

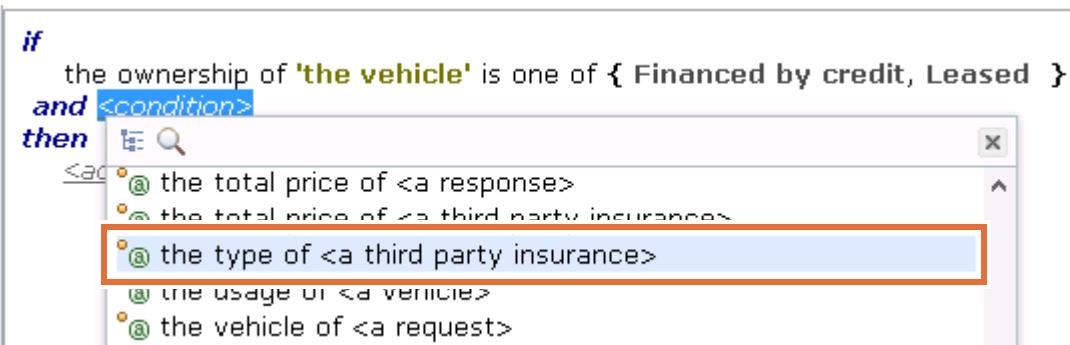


- \_\_\_ 10. Add a condition that tests the type of insurance requested.  
 \_\_\_ a. Place the cursor after the closing brace () of the phrase **{ Financed by credit, Leased }** and press Enter to create a line.  
 \_\_\_ b. Press the Space bar and wait for the completion menu.

- \_\_ c. In the completion menu, click **and <condition>**.



- \_\_ d. In the menu that opens on <condition>, click **the type of <a third party insurance>**.



- \_\_ e. In the menu that opens on <a third party insurance>, click '**the policy**'.
- \_\_ f. The cursor is now directly after the phrase '**the policy**'. Press the Space bar and wait for the completion menu.
- \_\_ g. Click the placeholders and select the appropriate vocabulary to create the following condition statement:

the type of the policy is ThirdParty

### 5.3. Completing the rule with the action statement

- \_\_ 1. In the **then** statement, click **<action>**; and click **refuse the application because <a string>**.
- \_\_ 2. In the menu that opens on <a string>, select **<string>**.

- \_\_\_ 3. Make sure that the cursor is between the two double quotation marks ("") and enter:

Collision insurance is compulsory for vehicles that are financed or leased but not owned

**if**

the ownership of '**the vehicle**' is one of { Financed by credit, Leased }  
**and** the type of '**the policy**' is ThirdParty

**then**

refuse the application because "**Collision insurance is compulsory for vehicles that are financed or leased but not owned**";

- \_\_\_ 4. Click **Save**, and in the Create New Version window, click **Create New Version**.



## Questions

What type of conditions did you create and which BAL constructs did you use to define them?

- \_\_\_ 5. For solutions to the rule and the questions, check your work against [Section 11, "Solutions", "Solution to Section 5, "Creating a compound condition""](#) on page 10-46.
- \_\_\_ 6. Click the **main** breadcrumb at the top of the page to return to the main page for the decision service.

## Section 6. Creating rules with constants and ruleset variables

---

### Business policy

From an insurance perspective, certain accessories can increase the value of a vehicle. However, if the value of the accessories is excessive in relation to the basic value of the car, coverage is refused.

The acceptable maximum ratio for the accessories value to car value is 0.8.

### Task

- Write a rule that specifies a constant (fixed number) to constrain the ratio of the value of accessories to the basic value of the car to 0.8 or less. The purpose of the rule is to flag violations of the constraint, in other words, when the accessories value is divided by the car value, the result must not be more than 0.8.  
In the next part of this exercise, you rework this rule to use a ruleset variable instead of a constant.
  - Add the rule to the rule package: Check\_Eligibility/Vehicle
- 

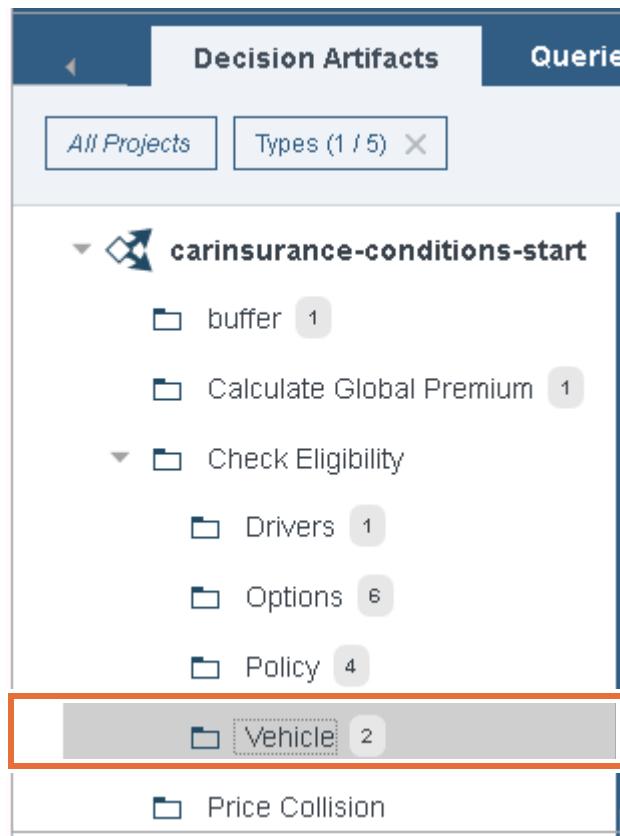
### 6.1. Creating the “Accessories value” rule with a constant

In this section, based on the business policy description, you write the rule that implements the policy. The condition statement that you author is:

```
if
    the accessories value of the vehicle / the basic value of the vehicle is
    more than 0.8
```

- 1. Make sure that you are on the **Decision Artifacts** tab.

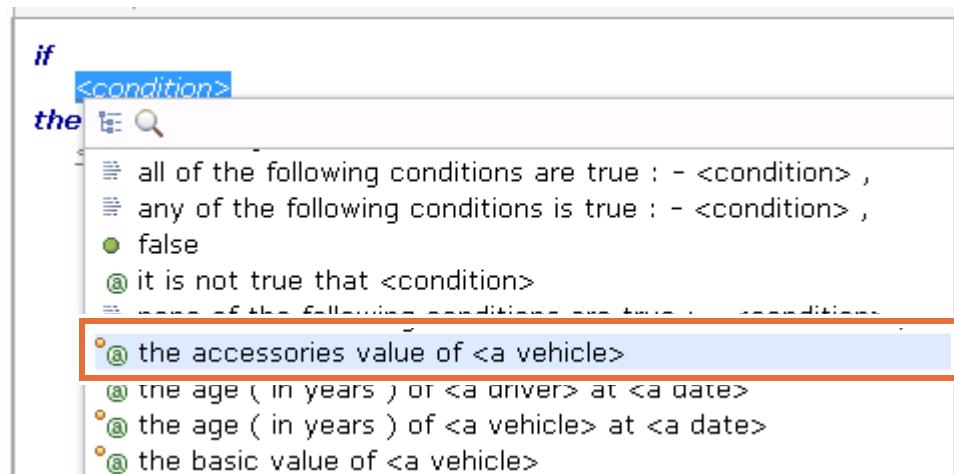
- 2. In the left pane, make sure that the **Check Eligibility** folder is expanded, and click the **Vehicle** subfolder.



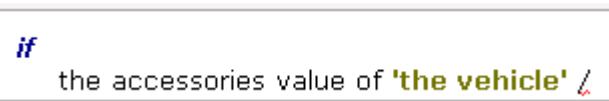
- 3. In the center pane, click the **create an artifact** icon (the plus sign) and click **New Rule**.  
 — 4. Enter **Accessories** value for the rule name, and click **Create**.

## 6.2. Defining the rule

- 1. In the rule editor, define the condition statement for the rule.  
 — a. In the **if** part of the rule, click **<condition>** and click **the accessories value of <a vehicle>**.



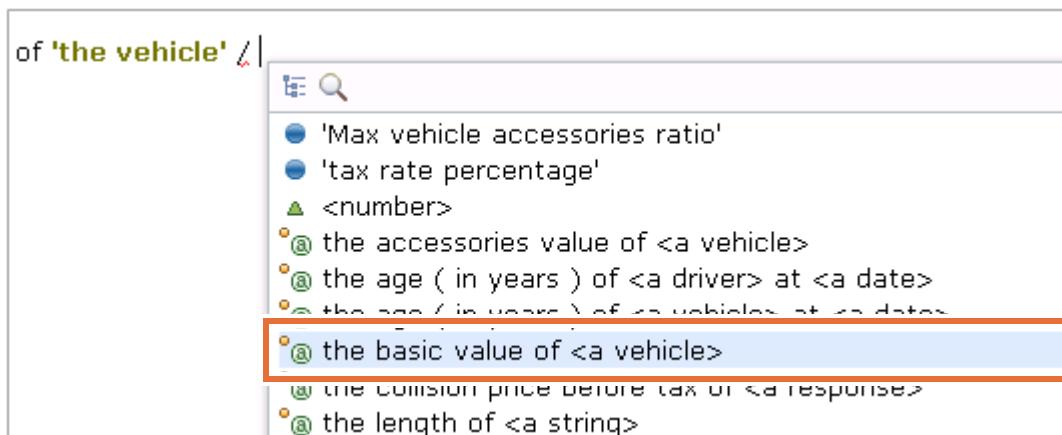
- \_\_ b. In the menu that opens on the phrase <a vehicle>, click ‘the vehicle’.
- \_\_ c. Press the Space bar, and type a forward slash (/).



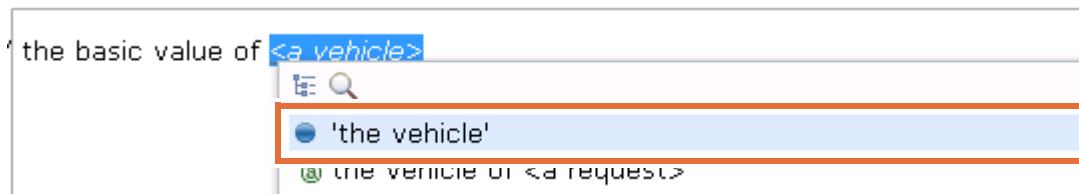
### Note

The forward slash (/) is the division operator.

- \_\_ d. Press the Space bar again and wait for the completion menu.
- \_\_ e. In the completion menu, select the basic value of <a vehicle>.

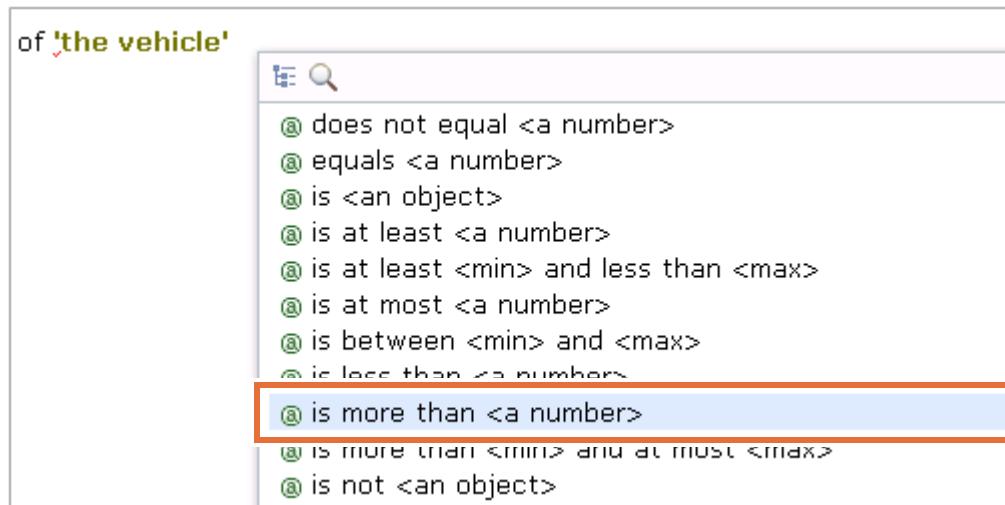


- \_\_ f. In the menu that opens on <a vehicle>, select ‘the vehicle’.

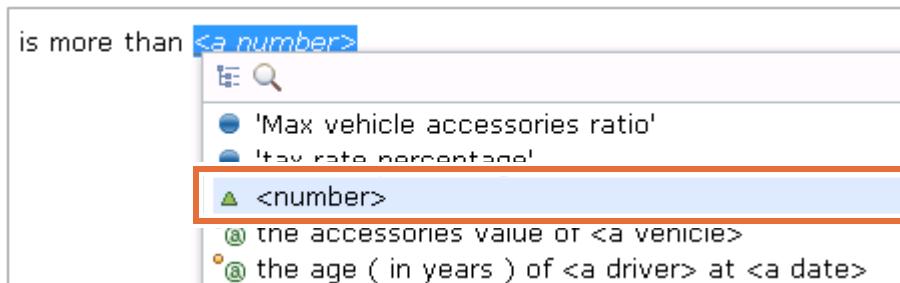


- \_\_ g. Press the Space bar and wait for the completion menu.

- \_\_\_ h. In the menu, click **is more than <a number>**.



- \_\_\_ i. In the menu that opens on <a number>, click <number>.



- \_\_\_ j. Change 0 to: 0.8
- \_\_\_ 2. Create an action statement that refuses the application with an explanation that the ratio is too high.
- In the **then** part of the rule, click **<action>**.
  - In the menu, select **refuse the application because <a string>**.
  - In the menu that opens on <a string>, select **<string>**.
  - Enter the following text between the double quotation marks (""):  
Vehicle accessories to base value ratio too high
- \_\_\_ 3. Save your work.
- \_\_\_ 4. Check your work against the answers in [Section , "Solution to Section 6, "Creating rules with constants and ruleset variables","](#) on page 10-46.
- \_\_\_ 5. Click the **main** breadcrumb to return to the **Decision Artifacts** tab for the decision service.

## 6.3. Reworking the “Accessories value” rule with a ruleset variable

### Business policy

The maximum ratio of the accessories value to the basic car value is 0.8, but business policy updates might result in changes to that ratio.

To better maintain these types of policy changes, you can define the ratio as a ruleset variable that is maintained independently of the rules.

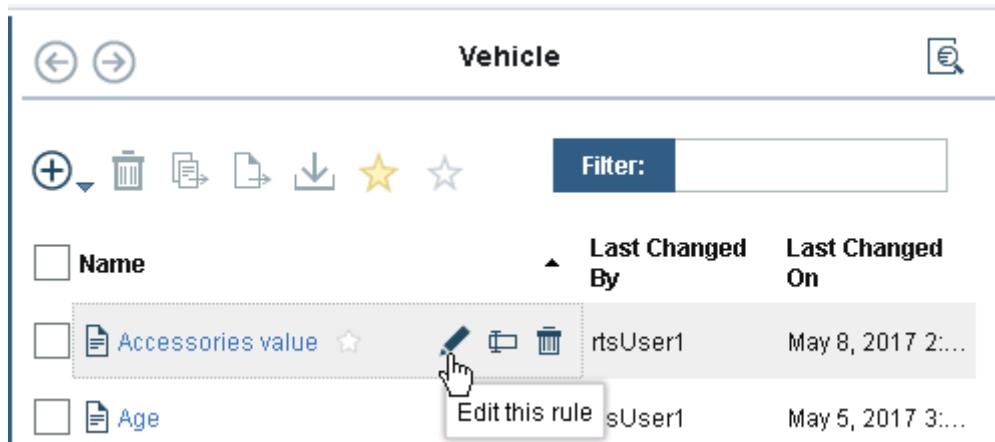
A variable that is called `maxRatio` is already defined for this decision service. The variable is verbalized as: `Max vehicle accessories ratio`

The variable value is set to 0.4, but you change it in the next section of this exercise by using the Decision Center Business console.

### Task

Change the rule that you wrote in [“Creating the “Accessories value” rule with a constant”](#) to use the predefined ruleset variable: **Max vehicle accessories ratio**

- 1. In the **Decision Artifacts** tab explorer pane, make sure that you are in the **Check Eligibility > Vehicle** folder.
- 2. Open the rule in the rule editor by hovering over the **Accessories value** rule name in the **Vehicle** folder pane, and then clicking the **Edit this rule** icon (the pencil).



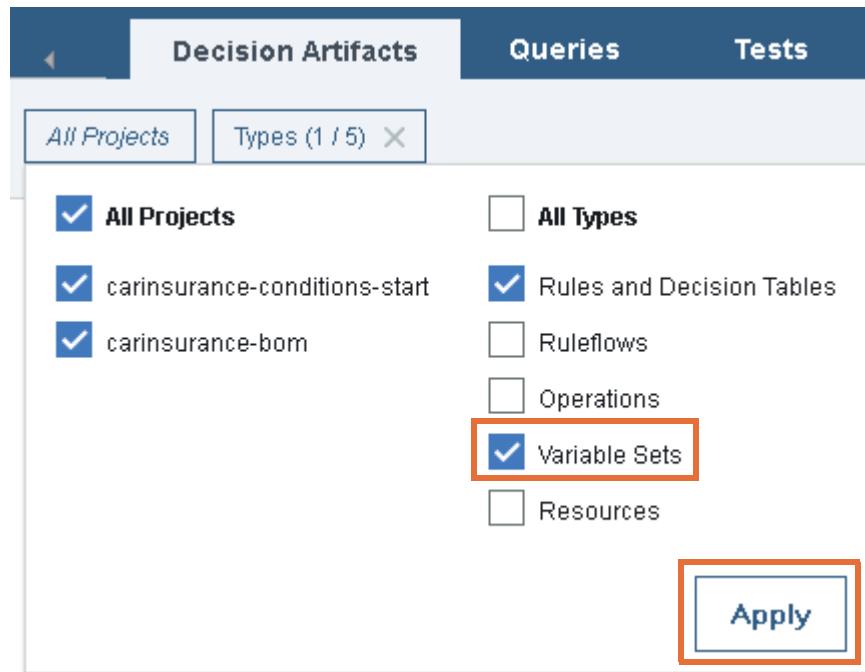
- 3. Double-click **0.8** to open the completion menu.
- 4. From the menu, select ‘**Max vehicle accessories ratio**’.
- 5. Save your work.
- 6. Check your work against [“Solution to Section 6, “Creating rules with constants and ruleset variables””](#) on page 10-46.
- 7. When you are finished checking your work, click the **main** breadcrumb to return to the **Decision Artifacts** tab.

## Section 7. Updating ruleset variables

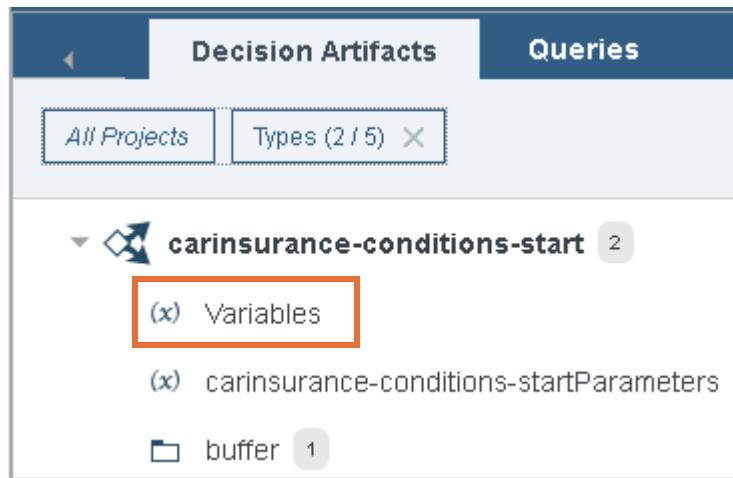
In this part of the exercise, you work in the Business console to edit the ruleset variable that is defined for this rule project, which is the `maxRatio` variable. In this section, you see how straightforward it is to access and update the variables in the variable set.

### 7.1. Viewing variable sets in the Decision Artifacts tab

- 1. Make sure that you are on the **Decision Artifacts** tab.
- 2. Enable variable sets as a viewable decision artifact type.
  - a. On the **Decision Artifacts** tab, click **Types**.
  - b. In the Types menu, select **Variable Sets**, and then click **Apply**.



- c. The Decision Artifacts explorer pane reloads, and shows the **Variables** menu item.



## 7.2. Editing the maxRatio variable

In this section, you edit the `maxRatio` variable to match the value that you originally used for the `Accessories` value rule. The `maxRatio` variable is defined as 0.4, and it must be updated to 0.8.

- 1. In the left explorer pane, click **Variables** to see the list of ruleset variables.

Name	Type	Verbalization	Initial Value
maxRatio	java.lang.Double	Max vehicle accessories ratio	0.4
policy	third party	the policy	null
taxRate	java.lang.Double	tax rate per vehicle	21.0
vehicle	vehicle	the vehicle	null

- 2. In the Variables pane toolbar, click **Edit this variable set** (the pencil icon).
- 3. In the `maxRatio` variable row, in the **Initial Value** column, click `0.4` to activate the text field.

Variables:			
<b>+ <input type="button" value=""/></b>	<b><input type="button" value=""/></b>		
<input type="checkbox"/>	<b>Name</b>	<b>Type</b>	<b>Verbalization</b>
<input type="checkbox"/>	maxRatio	java.lang.Double	Max vehicle accessories ratio
			<b>Initial Value</b>
			0.4

- 4. Delete 0.4 and type: 0.8

**Initial Value**  
  
0.8

- 5. Click **Save**, and then click **Create New Version** to save your work.
- 6. Click the **main** breadcrumb to return to the **Decision Artifacts** tab.

## Section 8. Creating a count condition

---

### Business policy

Insurance policies cover a maximum of six drivers, regardless of the type of coverage requested. If the car owner wants to include more than six drivers on the policy, the request is rejected.

---

### Task

- Write a rule that constrains the number of drivers that can be covered by the policy. The rule limits the number of drivers to six. The rule should flag requests that fail this constraint as *ineligible*.
  - Start from your Insurance Rejection template to create a rule that is named: Number of drivers
  - Add the rule to the rule folder: Check Eligibility/Drivers
- 

### 8.1. Creating the “Number of drivers” rule

- 1. Make sure that you are on the **Decision Artifacts** tab.
- 2. In the left explorer pane, click **Check Eligibility > Drivers**.
- 3. In the center pane, click the **create an artifact** icon (the plus sign) and click **New Rule**.
- 4. Enter **Number of drivers** for the rule name, and click **Create**.

### 8.2. Defining the conditions

- 1. Follow these steps to write the first condition statement, with this meaning:  
*More than six drivers are included in the insurance request.*
  - a. In the **if** part of the rule, click **<a condition>** to open the completion menu, and scroll through the list to find the appropriate phrase.
- 



### Questions

Considering the statement that you want to build, which tests the number of drivers, should you look for vocabulary phrases that are related to “drivers” or to “numbers”?

---



---



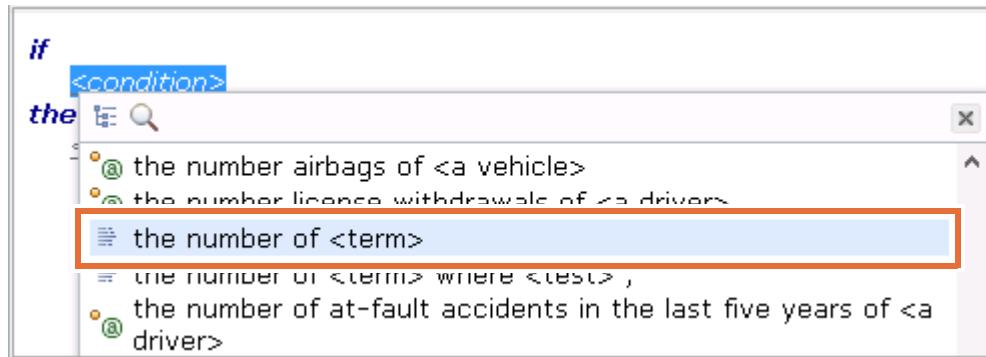
---

### Answer

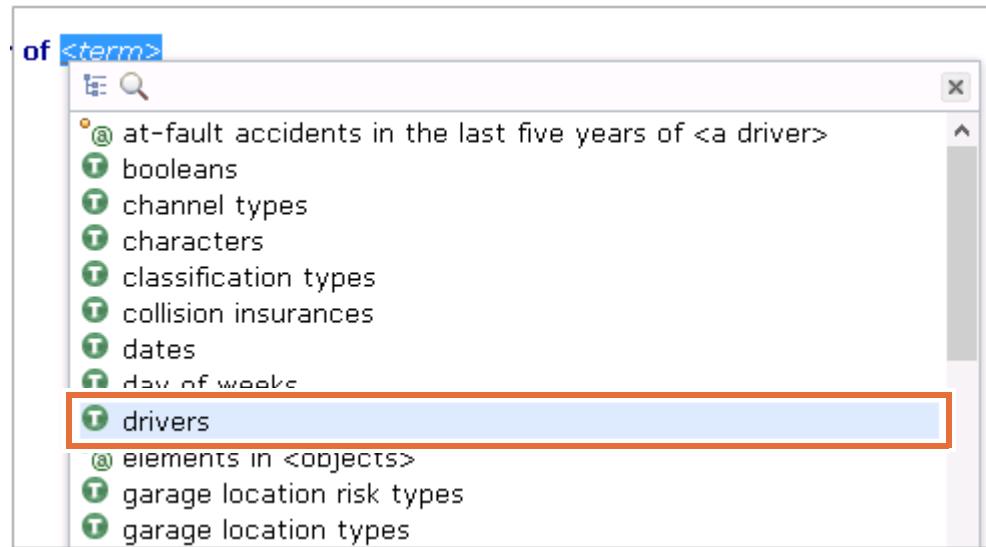
In this case, look for the phrase “the number of <term>.” You can then specify “drivers” as the object of this phrase in the condition statement.

---

- \_\_ b. In the menu, select **the number of <term>**.

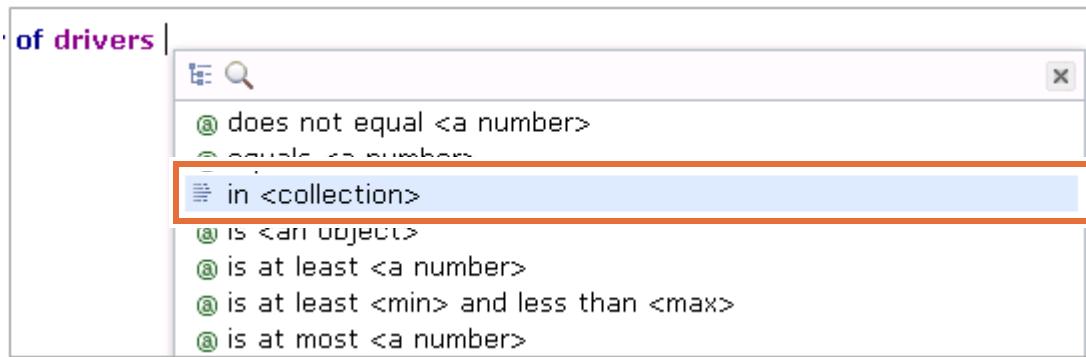


- \_\_ c. In the menu that opens on <term>, select **drivers**.

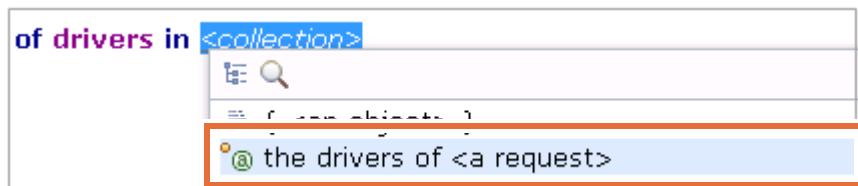


- \_\_ d. Press the Space bar and wait for the completion menu.

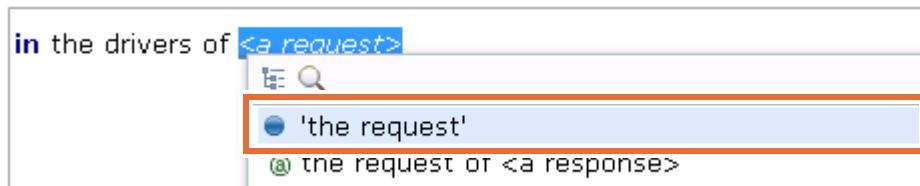
- \_\_ e. In the menu, select **in <collection>**.



- \_\_ f. In the menu that opens on <collection>, select **the drivers of <a request>**.



- \_\_ g. In the menu that opens on <a request>, select ‘the request’.



### Information

The condition that you create uses the **in** construct. By using the **in** construct, you can retrieve objects or collections of objects that are related to other objects.

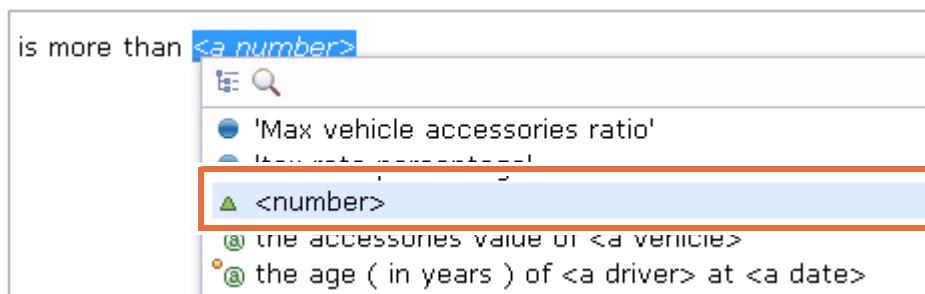
The **in** construct recognizes the collection of drivers from the insurance request, and provides access to individual drivers in that collection. By being able to access individual objects in a collection, you can get information such as how many objects are in the list or details specific to those objects.

You learn more about the **in** construct and the **from** construct, which is also used with collections, in a later unit.

- \_\_ 2. Complete the remaining placeholders so that the phrase specifies:

if the number of drivers in the drivers of the request is more than 6

- \_\_ a. The cursor is now directly after the phrase ‘**the request**’. Press the Space bar and wait for the completion menu.
- \_\_ b. In the menu, click **is more than <a number>**.
- \_\_ c. In the menu that opens on <a number>, select <number>.



- \_\_ d. Delete the 0 and type: 6

## 8.3. Completing the action statement

- \_\_ 1. In the **then** part of the rule, click <**action**>; and select **Refuse the application because <a string>**.
- \_\_ 2. On the menu that opens on <a string>, select **string**.
- \_\_ 3. Make sure that the cursor is between the double quotation marks ("").

- \_\_\_ 4. Type a sentence that explains why the request is ineligible.
  - \_\_\_ 5. Click **Save** and then click **Create New Version**.
- 



## Questions

Which BAL constructs were required to define the count condition in this rule?

---

- \_\_\_ 6. To see the solutions to the rule and the questions, check your work against "["Solution to Section 8, "Creating a count condition""](#) on page 10-47.
- \_\_\_ 7. Click the main breadcrumb to return to the **Decision Artifacts** tab.

## Section 9. Creating a count condition with a “where” clause

---

### Business policy

The driving history is a determining factor when evaluating eligibility and risk that are associated with the policy request. The driving history of all drivers included in the policy is considered. If any of the drivers were responsible for more than two accidents in the past five years, the request is rejected.

**Note:** The rule specification also clarifies that, at most, one message should be generated. Do not generate one message per ineligible driver.

---

### Task

- Write a rule that tests the driving history of all the drivers to see whether any driver was responsible for more than two accidents in the past five years.
  - The rule rejects the request as **ineligible** if any one of the drivers fails this constraint.
  - Name this rule: Number of at-fault accidents
  - Add the rule to the rule package: Check Eligibility/Drivers
- 

### 9.1. Creating the “Number of at-fault accidents” rule

In this section, you create a condition that addresses the following business policy:

*At least one of the drivers included in the insurance request had too many accidents.*

The condition statement that you author is:

```
if
    there is at least one driver in the drivers of 'the request'
    where the number of at-fault accidents in the last five years of this driver
    is more than 2 ,
```

---



#### Note

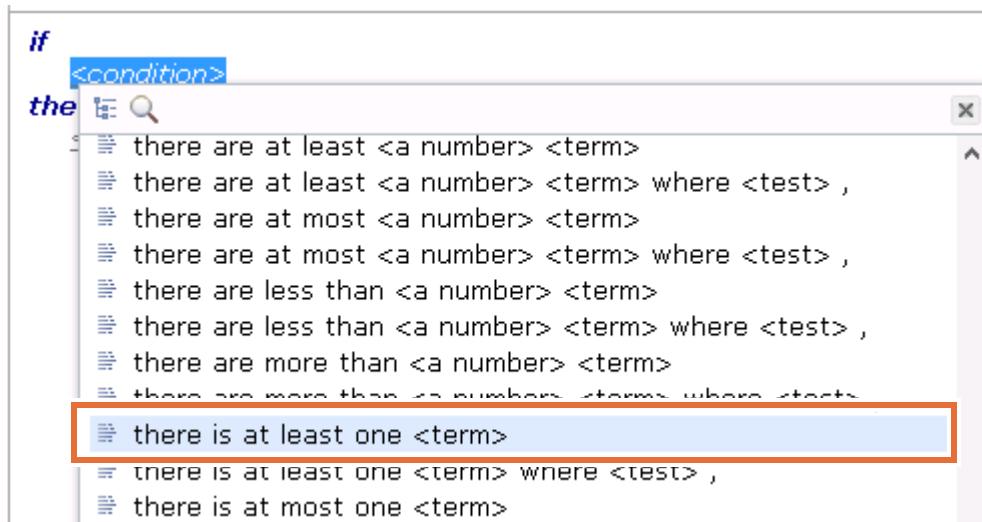
The following vocabulary is available in the selection lists to help build the condition:

- **there is at least one**
- **number of accidents of <a driver>**

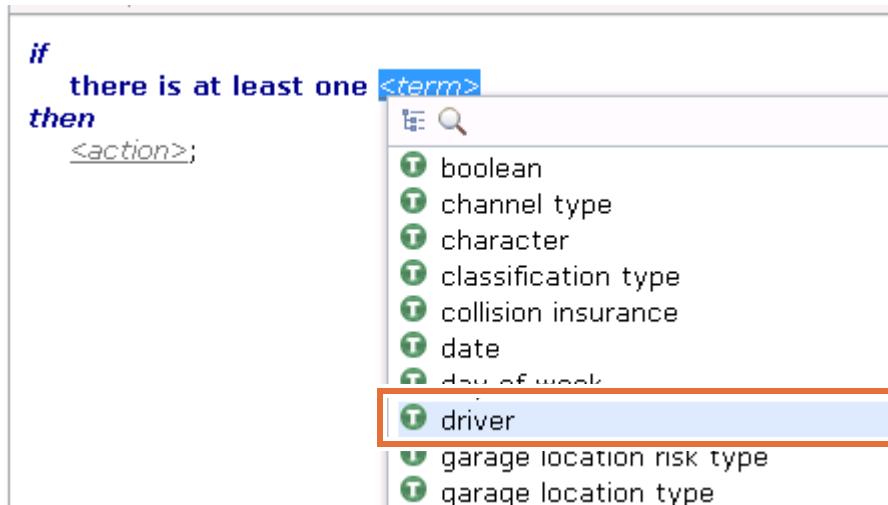
Because a request can include multiple drivers, you must test each driver in the request to be sure that no driver fails this condition. You use the “at least one” BAL construct to test the set of drivers.

- 
- 1. Make sure that you are on the **Decision Artifacts** tab, and that the **Check Eligibility > Drivers** folder is selected.

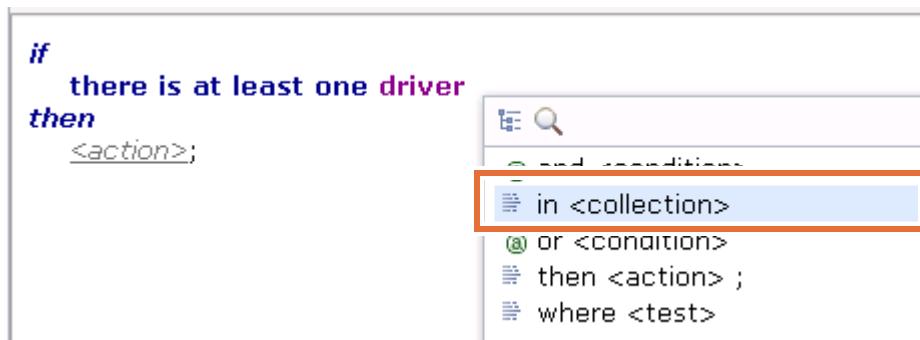
- \_\_ 2. Create a rule that is named: Number of at-fault accidents
  - \_\_ a. In the **Drivers** pane, click the **create an artifact** icon (the plus sign).
  - \_\_ b. Enter Number of at-fault accidents for the rule name, and click **Create**.
- \_\_ 3. Create the count condition.
  - \_\_ a. Click **<condition>** and select **there is at least one <term>**.



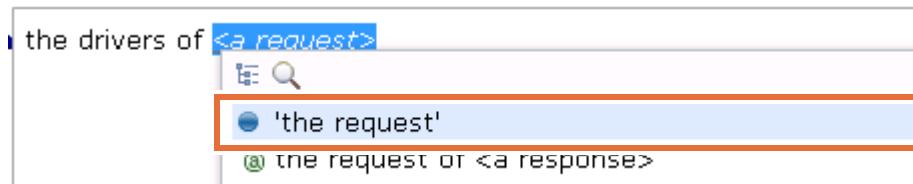
- \_\_ b. In the menu that opens on **<term>**, select **driver**.



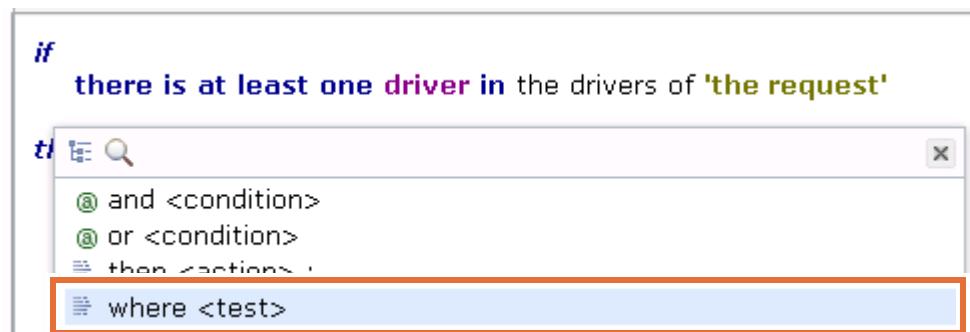
- \_\_ c. Press the Space bar and in the completion menu, select **in <collection>**.



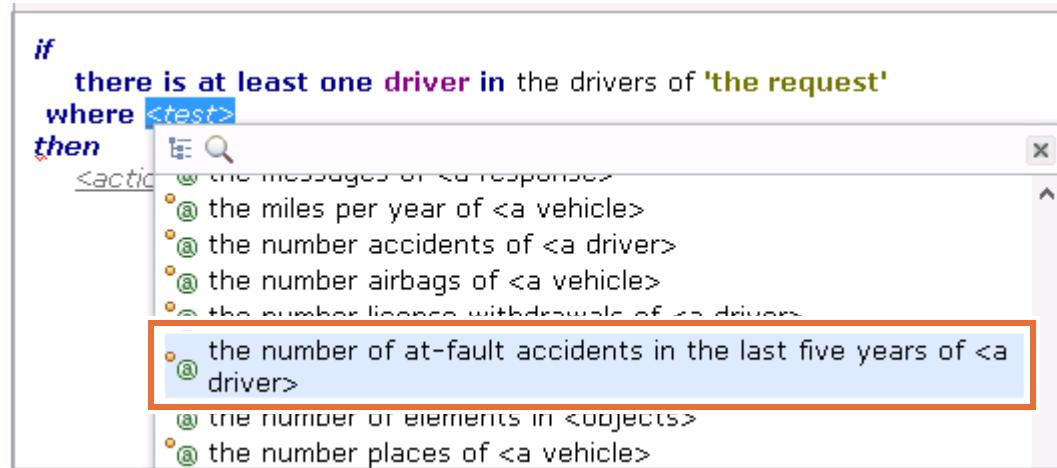
- \_\_ d. In the menu that opens on <collection>, select **the drivers of <a request>**.
- \_\_ e. In the menu that opens on <a request>, select '**the request**'.



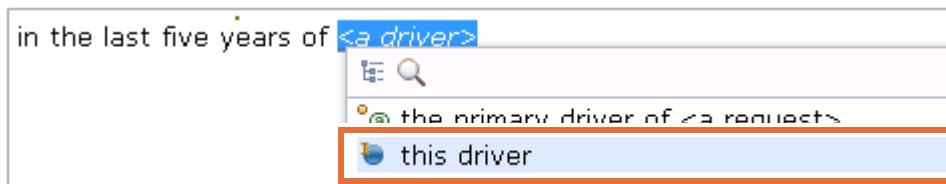
- \_\_ 4. Create the **where** clause.
- \_\_ a. In the editor, press Enter to go to a new line.
- \_\_ b. Press the Space bar, and in the completion menu, select **where <test>**.



- \_\_ c. In the menu that opens on <test>, select **the number of at-fault accidents in the last five years of <a driver>**.

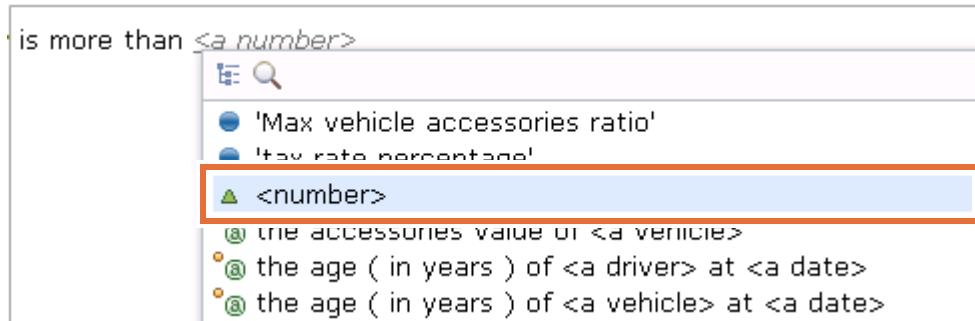


- \_\_ d. In the menu that opens on <a driver>, select **this driver**.



- \_\_ e. Press the Space bar, and in the completion menu, select **is more than <a number>**.

- \_\_ f. In the menu that opens on <a number>, select <number>.



- \_\_ g. In the editor, change the 0 to: 2
- \_\_ h. Enter a comma (,) to complete the test condition. You can either:
- Type a comma directly after the 2
  - or
  - Press the Space bar and in the menu, click the comma. Close the completion menu window by clicking the X in the upper-right corner.



### Important

The comma indicates that the test condition for the **where** clause is complete, and it is required. If you do not include a final comma for the **where** clause, the Business console rule editor shows an error.

Be careful where you place the comma. Any condition statements that are placed after the comma are not considered to be part of the **where** clause.

Thus, if your rule has a **where** clause with multiple test conditions, the comma should go at the end of the last test condition for that **where** clause. The **where** clause for this rule has only one test condition, so the comma must go at the end of the test condition.



### Information

When you use the BAL construct **there is at least one**, it often includes the **where** clause, which is how you specify the tests for the set of objects. For this rule, the **where** clause tests the number of accidents for each driver in the past five years.

The rule engine tests each driver against the **where** clause. As soon as the rule engine finds an object that meets the condition, the rule engine stops testing conditions against the objects and jumps to the action part of the rule. In this case, as soon as it finds a driver in the set of drivers that had more than two accidents in the past five years, the rule engine stops evaluating the insurance requests. It also sends a rejection message.

- \_\_\_ 5. Create an action statement that refuses the application and includes a message that explains why the request is refused.
  - \_\_\_ a. In the **then** part of the rule, click <action>; and click **Refuse the application because <a string>**.
  - \_\_\_ b. In the menu that opens on <a string>, click <string>.
  - \_\_\_ c. Make sure that the cursor is between the double quotation marks ("") and enter a reason why the request is ineligible.
- \_\_\_ 6. Save your work by clicking **Save**, and then clicking **Create New Version**.
- \_\_\_ 7. Check your work against "[Solution to Section 9, "Creating a count condition with a "where" clause""](#) on page 10-48.
- \_\_\_ 8. Click the **main** breadcrumb to return to the **Decision Artifacts** tab.

## Section 10. Advanced practice

In the previous parts of the exercise, you followed step-by-step instructions to build condition statements in rules. In this part of the exercise, you apply your skills to write a new rule.

### Business policy

Insurance policies can cover up to six drivers for the vehicle. Secondary drivers over the age of 65 must not be responsible for more than three accidents in the past five years. If any of the secondary drivers fail these conditions, the policy cannot be issued.

The raw business rule says:

*If one of the secondary drivers is over age 65 and has had more than 3 accidents, reject this request as a high risk.*

During the discovery phase, the meaning of this rule was verified to determine whether to apply “strictly greater than” or “equal to or greater than” to the age constraint. The policy manager validated the “is over age 65” constraint to mean “equal to or greater than” and the age rule was formalized in the Business Rule Document (BRD). The rule template is provided here:

#### 4.4.5. Rule Check senior secondary driver risk

<b>Rule name</b>	cons_senior_secondary_drivers	<b>Rule ID</b>	R_4_4_5
<b>Classification</b>	Constraint	<b>Creation date</b>	<date>
<b>References</b>	Check Eligibility (UC 4), Driver Eligibility (Step 4)		
<b>Decisions</b>			
<b>Business rule</b>	For each secondary driver: IF age of the secondary driver IS AT LEAST 65 THEN the number of accidents (in the last five years) of this secondary driver <b>MUST BE AT MOST 3</b>		
<b>Policy implemented</b>			
<b>Business motivation</b>			
<b>Objects affected</b>	Driver		
<b>Changes</b>			
<b>Who can change it</b>			
<b>How</b>			
<b>When</b>			



### Note

The “References” section of the rule template includes the name and number of the use case (UC) and the specific use case step from which this business rule was derived. In your organization, you might want access to this type of documentation to help provide context for the rules that you are implementing.

---

## Task

- Implement the formal rule to test both the age and the driving history of secondary drivers according to the business policy. Include an explanation in the action statement that coverage is refused because of the high risk.
  - Add the rule to the rule package: Check\_Eligibility/Drivers
  - After you finish, check your work against "[Solution to Section 10, "Advanced practice""](#) on page 10-48.
- 



### Questions

Consider:

- How can you test that the driver is not the primary driver?
  - Does this rule need a **where** clause?
- 



### Hint

#### Vocabulary and the BOM

The following selections are available in the vocabulary lists:

- **there is at least one**
- **if it is not true**
- **primary driver**
- **number of accidents**

If you need help to decide what vocabulary to select to build the statements, look at the BOM.

---

## End of exercise

## Section 11. Solutions

This section provides solutions to the previous exercises.



### Important

The syntax for the rules in the solutions matches the rule syntax when you open the Details view for a rule in the Decision Center Enterprise console. This syntax is slightly different from the syntax that you see in the Guided editor. The Guided editor does not require that you use quotation marks, commas, or semicolons. However, that syntax is inserted when you view completed rules in the Preview or Details view.

## Solution to [Section 5, "Creating a compound condition"](#)

### 11.1. Solution for ["Creating the “Policy type” rule"](#)

```

if
    the ownership of the vehicle is one of { Financed by credit , Leased }
        and the type of the policy is ThirdParty

then
    refuse the application because "Collision insurance is compulsory if the
    vehicle is not owned" ;

```



### Questions

**Answer to ["What type of conditions did you create and which BAL constructs did you use to define them?"](#)** on page 10-27

- Evaluate set membership ("is one of")
- Comparison ("if the policy type is third party")
- Compound AND

## Solution to [Section 6, "Creating rules with constants and ruleset variables"](#)

### 11.2. Solution for ["Creating the “Accessories value” rule with a constant"](#)

```

if
    the accessories value of the vehicle / the basic value of 'the vehicle' is
        more than 0.8

then
    refuse the application because "Vehicle accessories to base value ratio too
    high" ;

```

## Solution for "Reworking the “Accessories value” rule with a ruleset variable"

```

if
    the accessories value of the vehicle / the basic value of 'the vehicle' is
    more than 'Max vehicle accessories ratio'

then
    refuse the application because "Vehicle accessories to base value ratio too
    high" ;

```

## Solution to Section 8, "Creating a count condition"

### Solution for "Creating the “Number of drivers” rule"

```

if
    the number of drivers in the drivers of the request is more than 6

then
    refuse the application because "This insurance policy does not cover more
    than 6 drivers" ;

```

---



#### Questions

**Answer to "Considering the statement that you want to build, which tests the number of drivers, should you look for vocabulary phrases that are related to “drivers” or to “numbers”?"** on page 10-35

- The vocabulary term starts with the attribute first: “numbers”
- 



#### Questions

**Answer to "Which BAL constructs were required to define the count condition in this rule?"** on page 10-38

- in <collection>
  - is more than
-

## Solution to [Section 9, "Creating a count condition with a “where” clause"](#)

### 11.3. Solution for ["Creating the “Number of at-fault accidents” rule"](#)

```

if
    there is at least one driver in the drivers of the request
        where the number of at-fault accidents in the last five years of this driver
            is more than 2,
then
    refuse the application because "This request includes a high risk driver" ;

```

## Solution to [Section 10, "Advanced practice"](#)

### 11.4. Solution for [“Senior secondary drivers” rule](#)

```

if
    there is at least one driver in the drivers of the request
        where the age (in years) of this driver at the start date of the policy is
            at least 65
            and it is not true that this driver is the primary driver of the
                request
            and the number accidents of this driver is more than 3 ,
then
    refuse the application because "Risk too high" ;

```

### 11.5. Steps for creating the [“Senior secondary drivers” rule](#)



#### Important

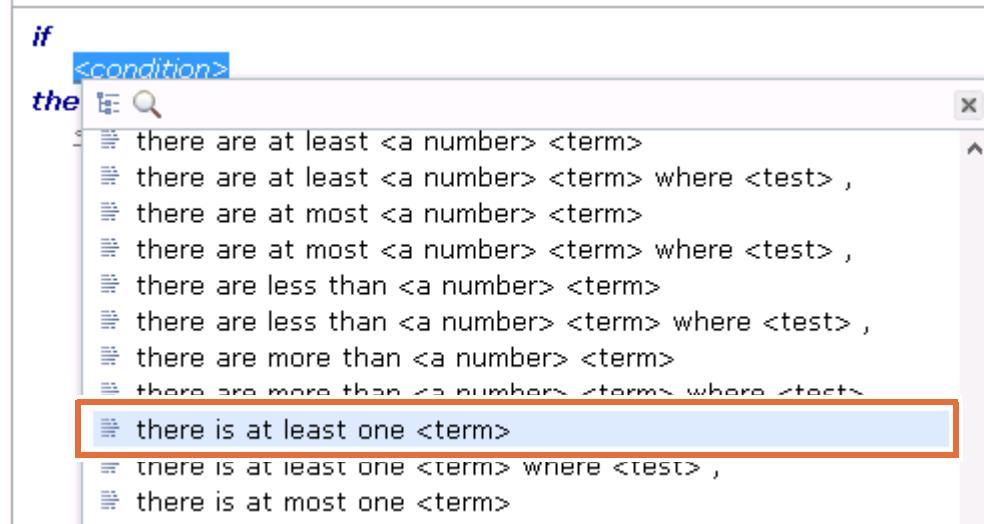
##### **Building the condition statement**

Be careful where you place the comma (,) in your set of test conditions for the **where** clause.  
The comma should go at the end of the last test condition.

- 1. Make sure that you are on the **Decision Artifacts** tab, and that the **Check Eligibility > Drivers** folder is selected.
- 2. Create a rule that is named: Senior secondary drivers
- 3. Create a condition that addresses the following business policy:

If one of the secondary drivers is at least 65 and has had more than 3 accidents, reject this request as a high risk.

- \_\_\_ a. Click <condition> and select: **there is at least one <term>**.



- \_\_\_ b. In the menu that opens on <term>, click **driver**.
- \_\_\_ c. Press the Space bar, and in the completion menu, click **in <collection>**.

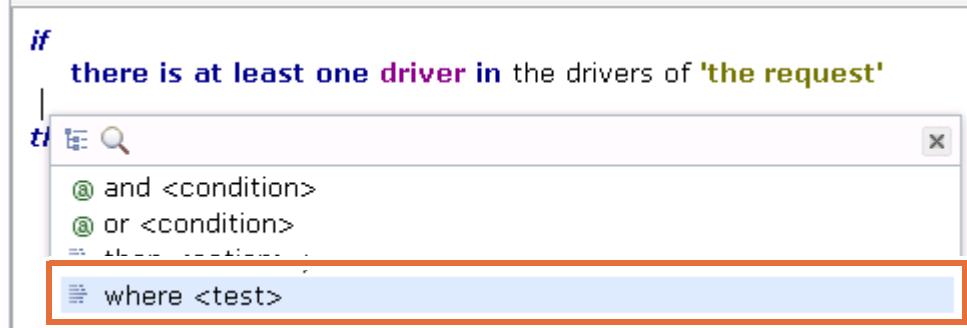


- \_\_\_ d. In the menu that opens on <collection>, click **the drivers of <a request>**.

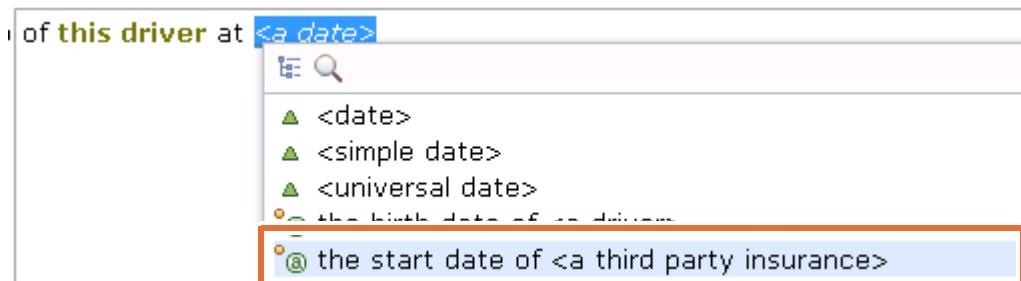
- \_\_ e. In the menu that opens on <a request>, click ‘the request’.



- \_\_ a. Press Enter to go to a new line, press the Space bar to open the completion menu, and select where <test>.

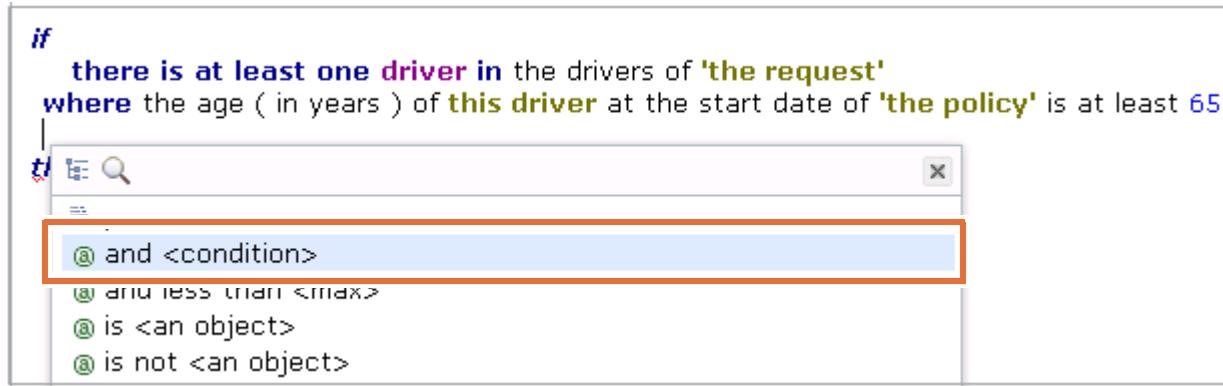


- \_\_ b. In the menu that opens on <test>, click the age (in years) of <a driver> at <a date>.  
 \_\_ c. In the menu that opens on <a driver>, click this driver.  
 \_\_ d. Click <a date> and click the start date of <a third party insurance>.

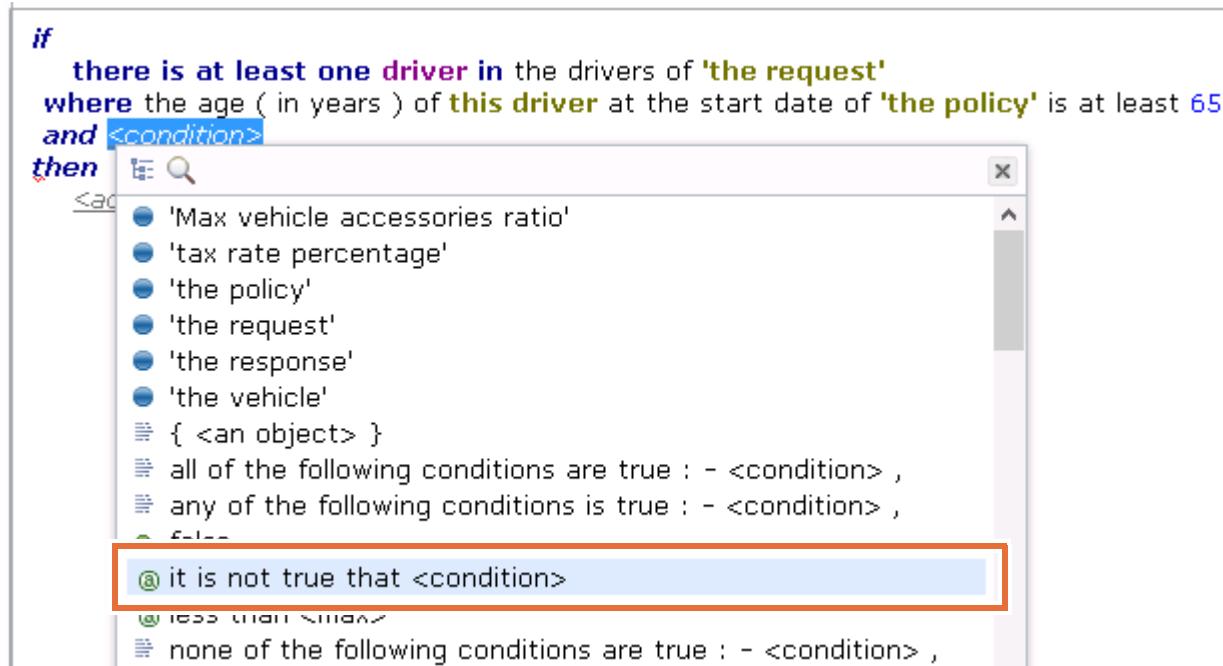


- \_\_ e. In the menu that opens on <a third party insurance>, click ‘the policy’.  
 \_\_ f. Press the Space bar, and in the completion menu, click is at least <a number>.  
 \_\_ g. In the menu that opens on <a number>, click <number>, change the 0 to: 65

- 4. Add another condition to test whether the driver is a secondary driver.
- a. Press Enter to go to a new line, press the Space bar to open the completion menu, and click **and <condition>**.



- b. In the menu that opens on **<condition>**, click **it is not true that <condition>**.



- \_\_ c. In the menu that opens on <condition>, scroll down in the list and click **this driver**.

**if**

there is at least one driver in the drivers of 'the request'  
where the age ( in years ) of **this driver** at the start date of 'the policy' is at least 65  
and it is not true that <condition>

**then**

<action>;

The completion menu lists several items related to drivers and requests. The item 'this driver' is highlighted with a red rectangle.

- ① the status of <a response>
- ② the third party max liability amount desired of <a third party insurance>
- ③ the third party max liability amount proposed of <a third party insurance>
- ④ the third party price before tax of <a response>
- ⑤ the third party price before tax of <a third party insurance>
- ⑥ the total price before tax of <a response>
- ⑦ the total price before tax of <a third party insurance>
- ⑧ the total price of <a response>
- ⑨ the total price of <a third party insurance>
- ⑩ the type of <a third party insurance>
- ⑪ the usage of <a vehicle>
- ⑫ the vehicle of <a request>
- ⑬ the year of <a vehicle>
- ⑭ this driver

true

- \_\_ d. Press the Space bar, and in the completion menu, click **is <an object>**.

- \_\_ e. In the menu that opens on <an object>, click **the primary driver of <a request>**.

**if**

there is at least one driver in the drivers of 'the request'  
where the age ( in years ) of **this driver** at the start date of 'the policy' is at least 65  
and it is not true that **this driver** is <an object>

**then**

<action>;

The completion menu lists several items related to objects. The item 'the primary driver of <a request>' is highlighted with a red rectangle.

- ① not <an object>
- ② not null
- ③ not one of <objects>
- ④ null
- ⑤ one of <objects>
- ⑥ this driver
- ⑦ the primary driver of <a request>

- \_\_ f. In the menu that opens on <a request>, click **'the request'**.

- \_\_ 5. Add another condition to test the number of accidents for this driver.

- \_\_ a. Press Enter to go to a new line, press the Space bar, and in the menu, click **and <condition>**.

- \_\_ b. In the menu that opens on <condition>, click **the number accidents of <a driver>**.

- \_\_ c. In the menu that opens on <a driver>, click **this driver**.

**if**

there is at least one driver in the drivers of 'the request'  
**where** the age ( in years ) of **this driver** at the start date of 'the policy' is at least 65  
**and it is not true that** **this driver** is the primary driver of 'the request'  
**and** the number accidents of <a driver>

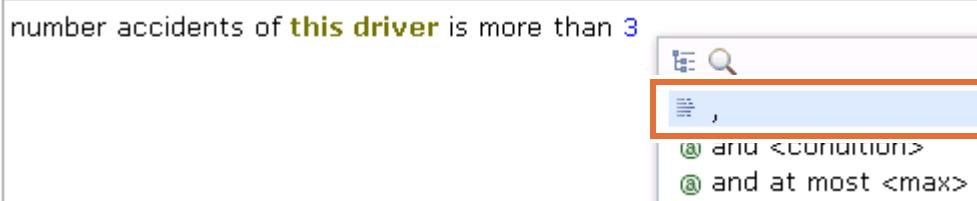
**then**

<action>;



The completion menu shows the text "the primary driver of the request" above the option "this driver", which is highlighted with a red border.

- \_\_ d. Press the Space bar, and in the completion menu, click **is more than <a number>**.  
 \_\_ e. In the menu that opens on <a number>, click <number> and change the 0 to: 3  
 \_\_ f. Enter a comma to complete the **where** clause by either typing a comma (,) or by using the completion menu by pressing the Space bar and clicking the comma (,).



The completion menu shows the text "number accidents of this driver is more than 3" above the option ",", which is highlighted with a red border.

Completion options below the menu:  
 @ and <condition>  
 @ and at most <max>



### Note

If you use the completion menu to insert a comma, close the menu by clicking the X in the upper-right corner.



The completion menu shows the text "@ and <condition>" above the option "then <action> ;", which is highlighted with a red border. The close button (X) in the top right corner is also highlighted with a red border.

- \_\_ 6. Complete the action statement so that it refuses the application and provides an explanation for refusing this request.
- \_\_ a. In the **then** part of the rule, click <action>; and click **Refuse the application because <a string>**.
  - \_\_ b. In the menu that opens on <a string>, click <string>.
  - \_\_ c. Make sure that the cursor is between the double quotation marks ("") and enter a message that explains why the request is refused.
- \_\_ 7. Click **Save**, and click **Create New Version** to save your work.

## Exercise review and wrap-up

The first part of the exercise looked at BAL constructs and how to find them in the product documentation reference. Next, you learned how to write various types of condition statements in the Business console, and used the Business console to edit a variable. Finally, you tested your skills by creating a rule without step-by-step instructions.

# Exercise 11. Working with definitions in rules

## Estimated time

01:00

## Overview

This exercise continues the focus on rule authoring. You learn how to define and use variables.

## Objectives

After completing this exercise, you should be able to:

- Identify and use common BAL constructs to write definition statements, including *in*, *from*, and *where*
- Use variables to make a rule easier to read
- Write definition statements that use objects and collections of objects

## Introduction

The exercise includes these sections:

- [Section 1, "Identifying variables for definitions"](#)
- [Section 2, "Using the "from <object>" construct"](#)
- [Section 3, "Using variables to improve readability"](#)
- [Section 4, "Accessing objects in a collection"](#)
- [Section 5, "Using a precondition"](#)
- [Section 6, "Defining a collection variable"](#)
- [Section 7, "Solutions"](#)

At the end of this exercise, you review and compare the various solutions with your instructor.

## Requirements

This exercise requires that you work in the Business console on the computer lab environment that is provided for this course. This exercise also includes some activities on paper.

For the computer lab portion of this exercise, you can continue to work in the decision service that you used for the previous exercise, or you can work in the `carinsurance-definitions-start` decision service.

To do this exercise, the following exercise must be completed:

- [Exercise 10, "Working with conditions in rules,"](#) on page 10-1

## Section 1. Identifying variables for definitions

This part of the exercise helps you recall how variables can be defined, and where such definitions might fit into the rules that you work with.

- 1. Think again about some rules that relate to your business.

Write some of your rules here.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

- 2. Identify where variable definitions might be useful in those rules. Circle or underline the phrase that might be defined as a variable.
- 3. Are these variables likely to be global (available for all rules), or local (used only in a single rule)?
- If a single rule uses the variables, you can define them in the definitions part of the rule. They can help restrict the objects that the rule should evaluate (acting as a precondition), and they can help make the rule easier to read.
  - If the variables fall under the following conditions, developers are likely to define them so that you can use when you author rules.
  - Variables should be available for all rules; for example, a constant such as value-added tax (VAT).
  - The values of the variables change as a result of calculations across rules.

- 4. Consider when each of the following types of variables might be used in a rule at your business.

In this table, write some definition statements for each type of variable.

Variable definition type	Example in your business context
--------------------------	----------------------------------

A constant

---

An expression (formula)

---

An object

- When you must refer to more than one object of the same type
  - When you want to clarify or simplify how you refer to something
- 

A collection of objects

---

A precondition

- When you want to refer to an object that meets a particular criterion
-

## Section 2. Using the “from <object>” construct

---

### Business policy:

In addition to basic third-party liability insurance, the insurance company offers collision insurance for amounts in the range of 2500 to 125 000. Requests that fall into this range are accepted and priced accordingly.

---

### Task:

- Write a business rule that tests the requested amount of collision insurance to determine whether that amount is within the accepted insurable range.
  - If the requested amount is acceptable, the rule sets the proposed collision insurance amount to the requested amount. Later calculations compute the premium, which is based on the collision amount that is proposed.
  - Test the amount of collision insurance that was requested. You use the **from <object>** construct to access the collision object, and all its attributes, from the CollisionInsurance subclass of the basic ThirdPartyInsurance.
  - Name this rule: Collision cover in [min, max]
  - Add this rule to the Check Eligibility/Policy rule package.
  - After you finish, check your work against "[Section 2, "Using the “from <object>” construct"](#)" on page 11-26.
- 



### Important

Save your rule authoring work frequently while you work in the Business console rule editor. Saving your rule while it is in progress helps you to avoid losing your work if your Business console session times out.

You can save your work in the Business console rule editor by clicking **Save** and then **Create New Version**. Return to rule editor by clicking **Edit**.

---

### 2.1. Creating the “Collision cover in [min, max]” rule

Based on the business policy description, write the rule to implement this policy.

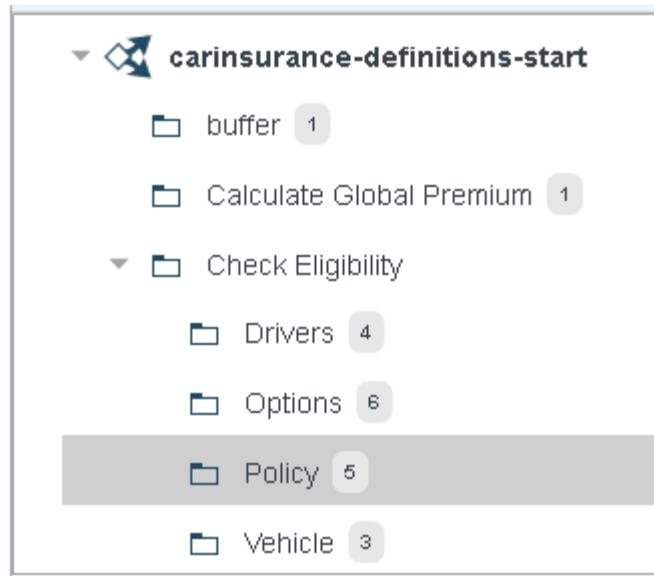
- 1. Make sure that the sample server is run-in.
- 2. If you are not logged in to the Business console, open the Business console and sign in as rtsUser1.
  - a. Double-click the Decision Center Business console shortcut, or enter the following URL in a web browser window:  
`http://localhost:9090/decisioncenter`

If your computer lab environment uses a different port, use it here.

- \_\_\_ b. Enter `rtsUser1` in the **Username** and **Password** fields, and click **Sign In**.
- \_\_\_ 3. On the **Library** tab, you can continue to work in `carinsurance-conditions-start`, or you can work in the `carinsurance-definitions-start` decision service.  
To switch to the `carinsurance-definitions-start` decision service:
  - \_\_\_ a. In the **Library** tab, in the list of decision services, find the `carinsurance-definitions-start` decision service.
  - \_\_\_ b. Click the white space in the **carinsurance-definitions-start** section, and click the **main** link to open the main branch of the decision service.

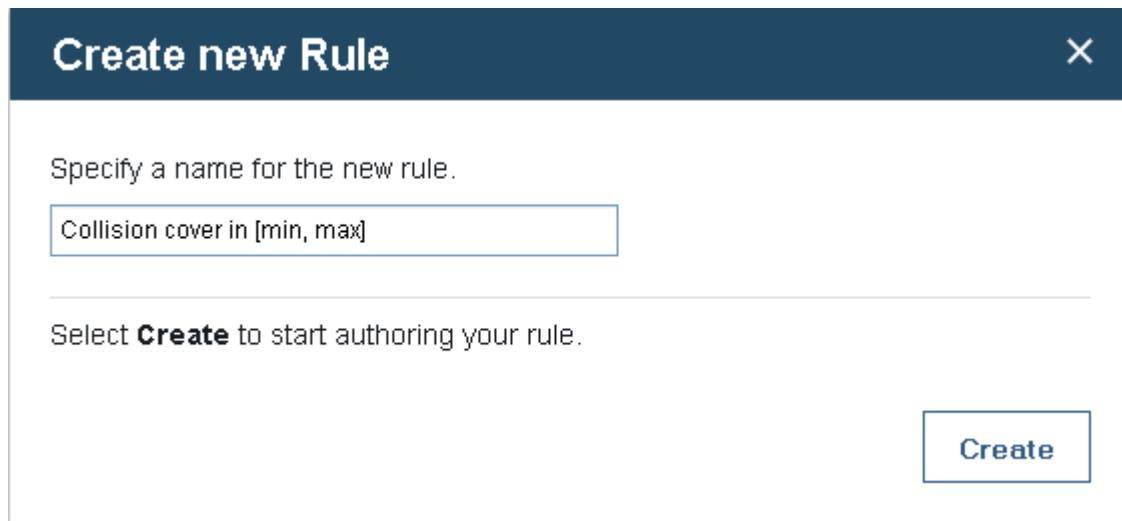


- \_\_\_ 4. Make sure that you are on the **Decision Artifacts** tab.
- \_\_\_ 5. In the left explorer pane, expand the **carinsurance-definitions > Check Eligibility** folder, and click the **Policy** subfolder.



- \_\_\_ 6. Create a rule that is called Collision cover in [min, max].
  - \_\_\_ a. In the Policy pane, create a rule by clicking the **create an artifact** icon (the plus sign) and clicking **New Rule**.

- \_\_\_ b. In the “Create new Rule” window, enter Collision cover in [min, max] for the rule name, and click **Create**.



## 2.2. Defining a variable

- \_\_\_ 1. Create a **definitions** section for the rule.
- \_\_\_ a. In the rule editor, place the cursor before the **if** heading, and press Enter to move the **if** part of the rule to a new line.
- \_\_\_ b. Place the cursor in the empty line at the top of the rule editor and type: **definitions**
- \_\_\_ c. Press Enter to go to a new line.

```

carinsurance-definitions-start > main
Collision cover in [min, max]
definitions
if
<condition>
then
<action>;

```



### Note

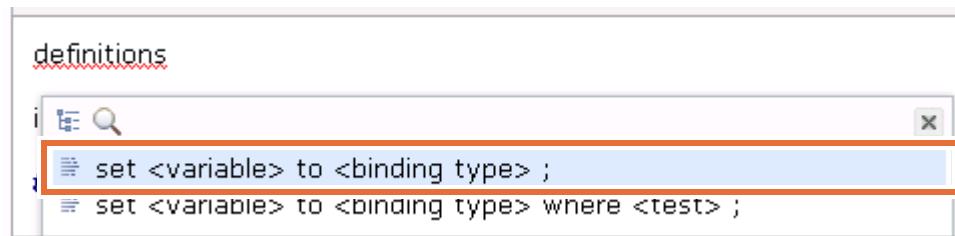
You can ignore the warning indicator on the word **definitions** and the warning message at the bottom of the editor that states:

The word 'definitions' is not required.

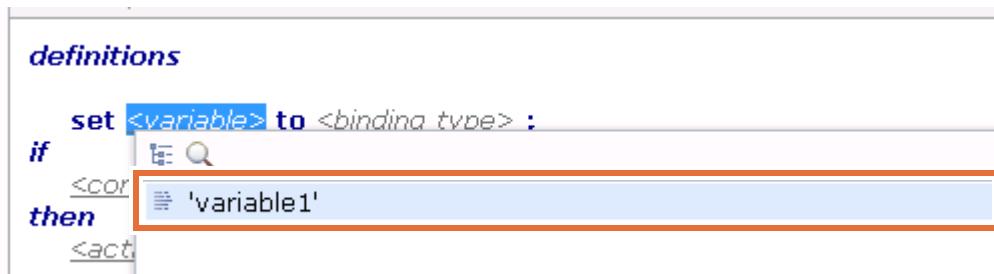
The warning goes away after you define the local variable.

- \_\_ d. Press the Space bar and in the completion menu, click:

```
set <variable> to <binding type> ;
```



- \_\_ e. In the menu that opens on <variable>, click 'variable1'.



- \_\_ 2. Define a variable that is called collision policy.

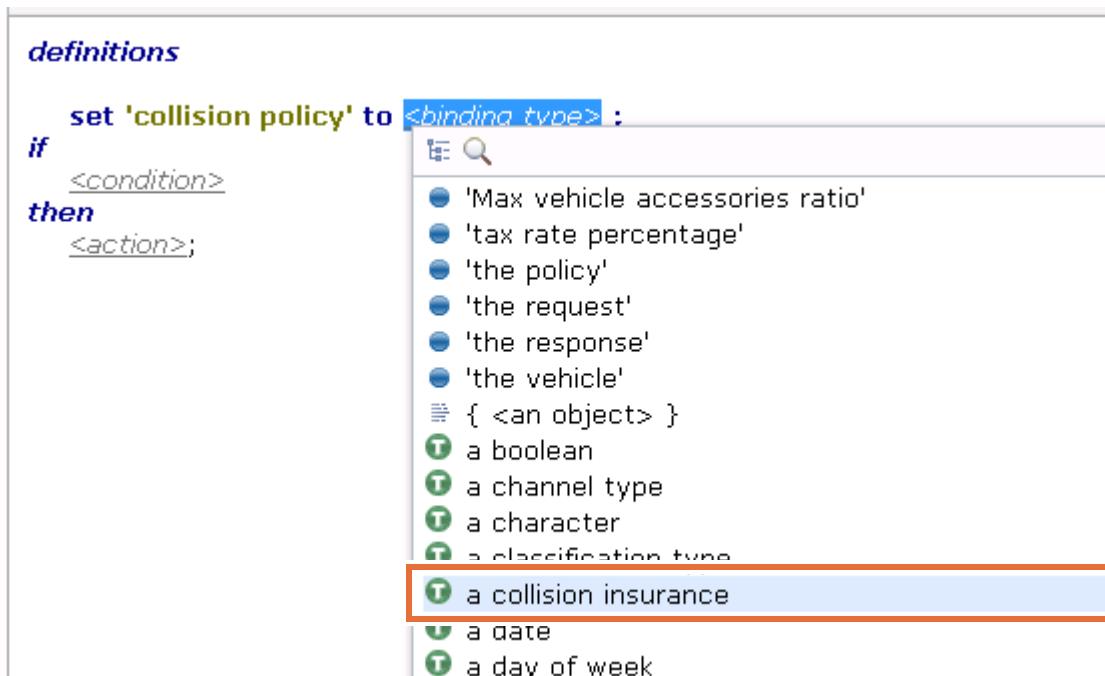
- \_\_ a. In the editor, change **variable1** to: collision policy



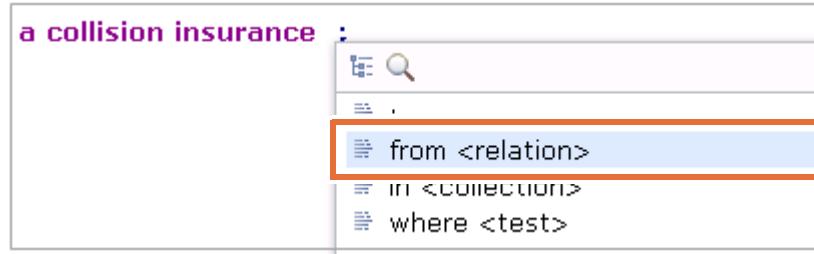
### Important

Make sure that the term `collision policy` is between the single quotation marks (' ').

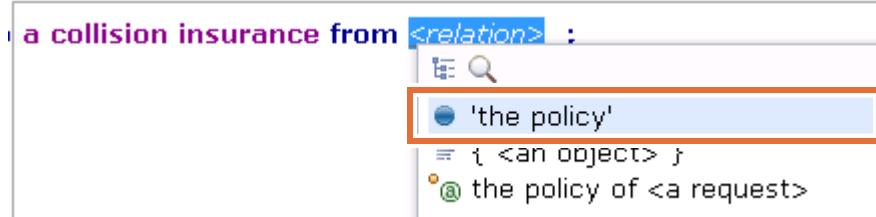
- \_\_ b. In the editor, click <binding type> and in the menu, click a **collision insurance**.



- \_\_ c. Press the Space bar, and in the completion menu, click **from <relation>**.



- \_\_ d. In the menu that opens on <relation>, click 'the policy'.



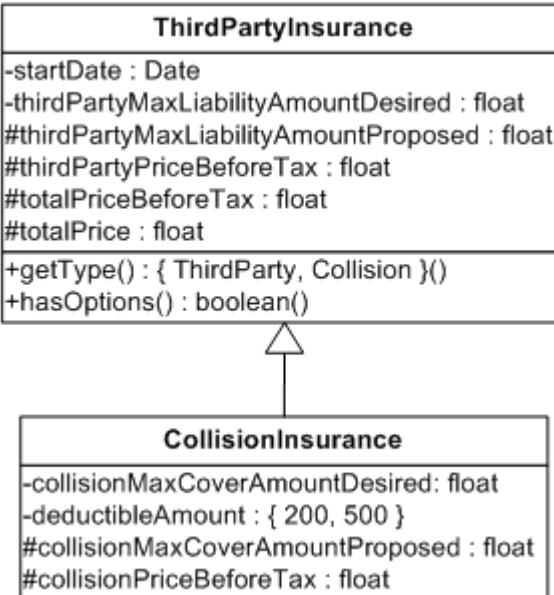
The definitions statement now reads:

```
definitions
  set 'collision policy' to a collision insurance from 'the policy' ;
```

**definitions**

**set 'collision policy' to a collision insurance from 'the policy' ;**

- \_\_ 3. Recall the BOM structure.





## Questions

Why is the **from** construct used to define the variable?

---



---

## Answer

Remember in the BOM, that CollisionInsurance is a subclass of the basic ThirdPartyInsurance. You can use the **from** construct to get a specialized class from the parent class.

After your variable is set to the `CollisionInsurance` class, you can then access the attributes of that class, including the attribute:

`CollisionInsurance.collisionMaxCoverAmountDesired`

## 2.3. Defining the condition and action statements

- 1. In the **if** part of the rule, test your `collision policy` variable to check whether the amount that is requested for collision coverage is within the acceptable range, which is 2500 - 125000.



### Hint

To test whether a value is within a range, you can click **is** and choose **is between <min> and <max>**.

The screenshot shows a dropdown menu with the following options:

- ⑧ does not equal <a number>
- ⑧ equals <a number>
- ⑧ is <an object>
- ⑧ is at least <a number>
- ⑧ is at least <min> and less than <max>
- ⑧ is at most <a number>
- ⑧ is between <min> and <max>** (highlighted with a red border)
- ⑧ is less than <a number>
- ⑧ is more than <a number>
- ⑧ is more than <min> and at most <max>
- ⑧ is not <an object>
- ⑧ is not one of <objects>
- ⑧ is one of <objects>
- ⑧ is strictly between <min> and <max>

This construct automatically adds placeholders to enter the minimum and maximum values.

- \_\_\_ 2. Define the action statement to set the maximum amount of collision coverage that is proposed in the policy to match the amount that is intended.
- \_\_\_ 3. For an example solution, see ["Solution for "Creating the “Collision cover in \[min, max\]” rule""](#) on page 11-26.

## Section 3. Using variables to improve readability

Because a rule variable is applicable only within the rule, you can create variables specific to your rule with short names for long, awkward vocabulary terms.

For example, in the previous rule, although the ***if*** and ***then*** parts of the rule are simple, they might seem to be complex because of the wording in the vocabulary. However, you might not want to change the verbalization because the names are meaningful. So to improve readability, you can define variables.

### 3.1. Rewriting the “Collision cover in [min, max]” rule to improve readability

- 1. Reread the rule statement.

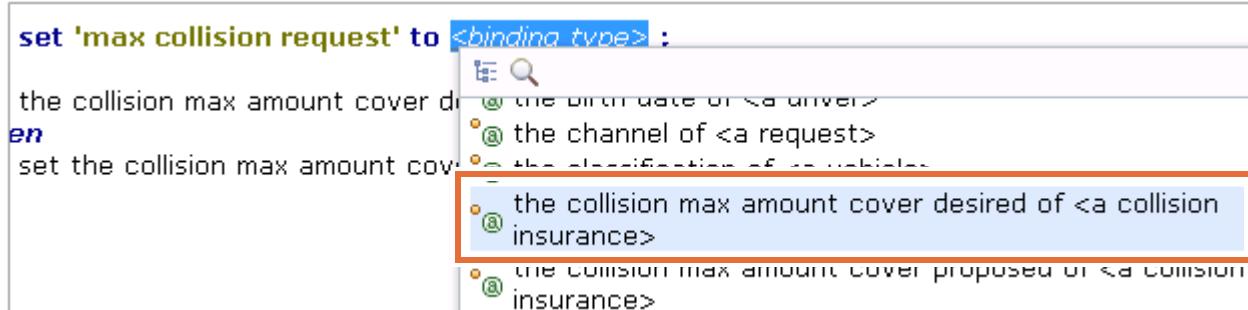
```
definitions
    set collision policy to a collision insurance from 'the policy' ;
    if
        the collision max amount cover desired of collision policy is between 2500
        and 125000
    then
        set the collision max amount cover proposed of 'collision policy' to the
        collision max amount cover desired of 'collision policy' ;
```

- 2. Consider how to improve readability for the term:

the collision max amount cover desired of 'collision policy'

- 3. In the rule editor, add another variable: max collision request

- a. Click **Edit** to open the rule in the rule editor.
- b. Place the cursor at the end of the first variable definition for 'collision policy' (after the semicolon [:]) and press Enter to go to a new line.
- c. Press the Space bar and in the menu, click: **set <variable> to <binding type>** ;
- d. In the menu that opens on **<variable>**, click '**variable1**'.
- e. Change the text '**variable1**' to '**max collision request**'.
- f. Click **<binding type>** and in the menu, click **the collision max amount cover desired of <a collision insurance>**.



- g. In the menu that opens on **<a collision insurance>**, click '**collision policy**'.

The new variable statement reads:

```
set 'max collision request' to the collision max amount cover desired  
of 'collision policy' ;
```

### **definitions**

**set 'collision policy' to a collision insurance from 'the policy' ;**

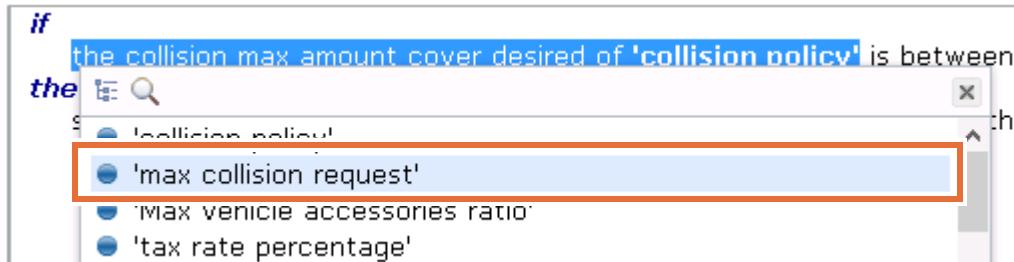
**set 'max collision request' to the collision max amount cover desired of 'collision policy'**

- 4. Change the condition and action statements to use your new variable.



### **Hint**

To replace the original phrase the collision max amount cover desired of 'collision policy' with the new variable, double-click the first word in the phrase (the) and in the menu, click 'max collision request'.



Notice that your new variable is near the top of the vocabulary list for easy access.

The updated rule reads:

```
if  
  'max collision request' is between 2500 and 125000  
then  
  set the collision max amount cover proposed of 'collision policy' to  
  'max collision request' ;
```

```
if  
  'max collision request' is between 2500 and 125000  
then  
  set the collision max amount cover proposed of 'collision policy' to 'max collision request';
```

- 5. Save your rule.

This rule is now shorter and easier to read. This rule is a simple example of how you can use rule variables to enhance readability in your rules.

- 6. Click the **main** breadcrumb to return to the **Decision Artifacts** tab.

## Section 4. Accessing objects in a collection

In this first part of the exercise, you define a variable that accesses each driver in an insurance request. Consider the business policy that is described here.

---

### **Business policy:**

To assess the eligibility of a car insurance request, the insurance company looks not only at the profile of the primary driver, but also that of secondary drivers.

If any of the drivers included on the policy fail the eligibility requirements, the request is denied.

One requirement is that all drivers on the policy must be between the ages of 18 and 80.

---

### **Task:**

- Create a rule that rejects insurance requests for drivers that do not meet the age constraints.
  - Name the rule: Driver age
  - Add this rule to the rule folder: Check Eligibility/Drivers
- 



### Questions

Consider:

- Which BAL construct can you use to define a variable that accesses each driver?
  - Is there more than one way to access the profile for each driver? Should you define a collection variable or a single object variable?
  - Should you test the age of the driver as a condition or use a precondition?
- 

### 4.1. Creating the “Driver age” rule without preconditions

Based on the business policy description, write the rule to implement this policy by using a single object variable and no precondition.

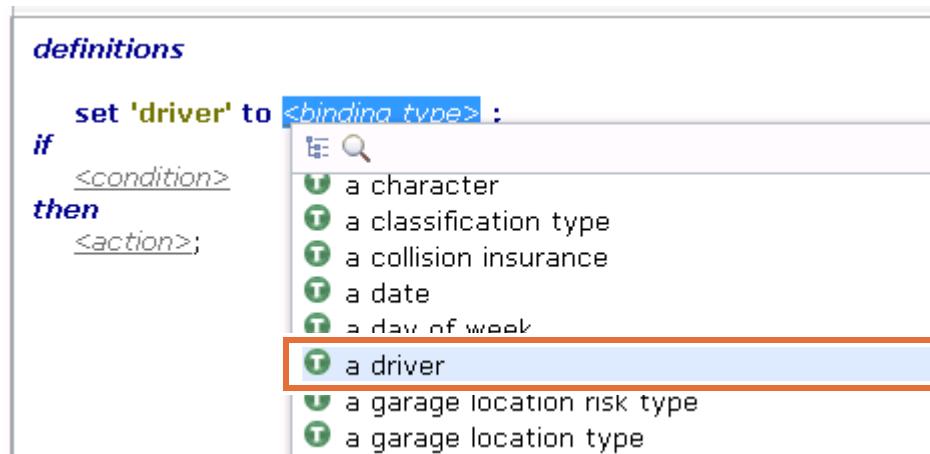
- 1. Make sure that you are on the **Decision Artifacts** tab.
- 2. In the left pane, click the **Check Eligibility > Drivers** folder.
- 3. Create a rule that is called **Driver age**.
  - a. In the center pane, click the **create an artifact** icon (the plus sign) and click **New Rule**.
  - b. In the “Create new Rule” window, enter **Driver age** for the rule name, and click **Create**.

## Defining a variable

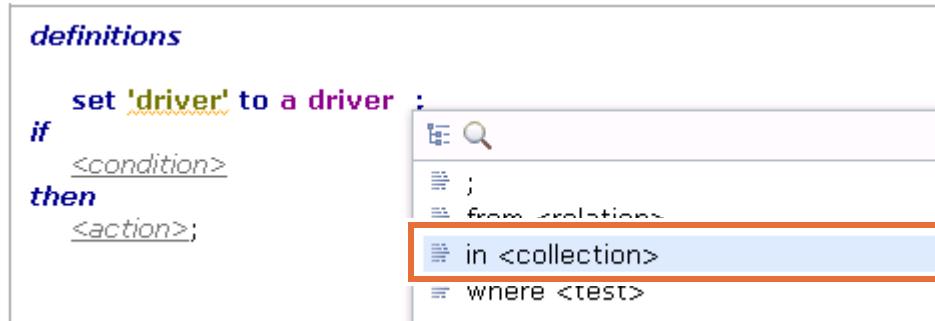
- \_\_\_ 1. On the Content page, define a variable that is called `driver`.
  - \_\_\_ a. In the rule editor, place the cursor before the `if` heading and press Enter to move it to the next line.
  - \_\_\_ b. Move the cursor to the new line at the top of the editor, type `definitions`, and press Enter to go to a new line.
  - \_\_\_ c. Press the Space bar, and in the menu, click: `set <variable> to <binding type>` ;
  - \_\_\_ d. In the menu that opens on `<variable>`, click '`variable1`'.
  - \_\_\_ e. Change the text '`variable1`' to: '`driver`'



- \_\_\_ 2. Complete the variable definition.
  - \_\_\_ a. Click `<binding type>` and in the menu, click `a driver`.



- \_\_\_ b. Press the Space bar, and in the menu, click `in <collection>`.



- \_\_\_ c. In the menu that opens on `<collection>`, click `the drivers of <a request>`.
- \_\_\_ d. In the menu that opens on `<a request>`, click `the request`.

- \_\_\_ 3. The completed definition statement is:

```
definitions
    set 'driver' to a driver in the drivers of 'the request' ;
```

## Writing the condition

- \_\_\_ 1. Complete the condition to say:

```
if it is not true that the age ( in years ) of driver at the start date of
'the policy' is between 18 and 80
```



### Questions

Where is the “it is not true” construct in the vocabulary list?

Which vocabulary item in the vocabulary list tests the age of the driver?

- \_\_\_ 2. Define the condition statement.

- \_\_\_ a. In the **if** section, click <condition> and click **if it is not true that <condition>**.

**definitions**

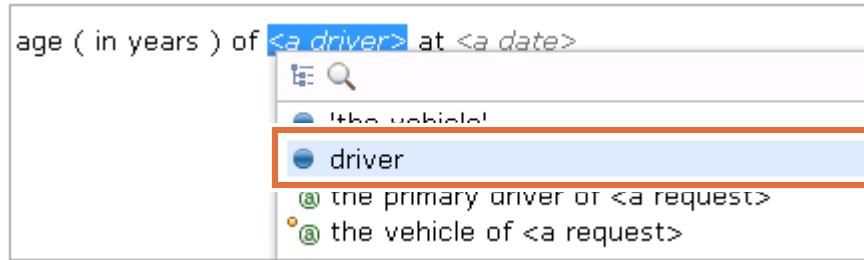
```
set 'driver' to a driver in the drivers of 'the request' ;
if
<condition>
```

**the**

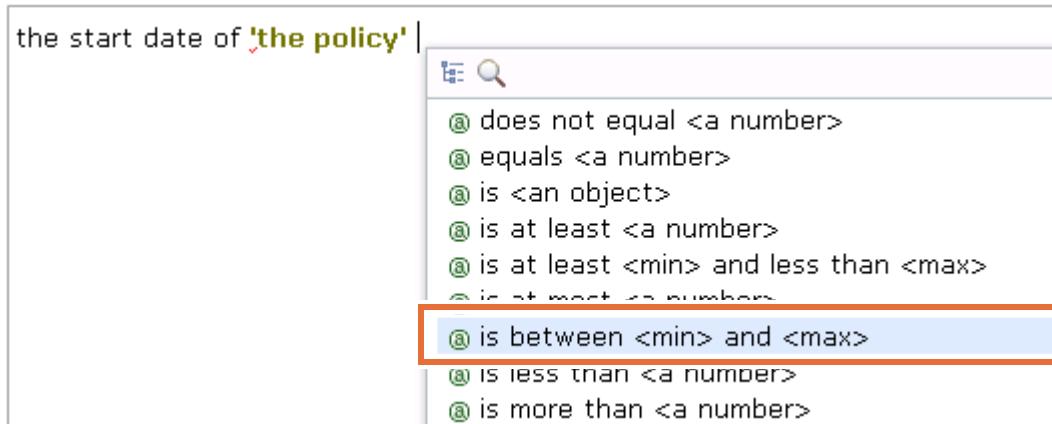
- 'Max vehicle accessories ratio'
- 'tax rate percentage'
- 'the policy'
- 'the request'
- 'the response'
- 'the vehicle'
- { <an object> }
- all of the following conditions are true : - <condition> ,
- any of the following conditions is true : - <condition> ,
- driver
- &lt;&gt;
- @ it is not true that <condition> **(highlighted)**
- = none or the following conditions are true : - <condition> ,
- @ the accessories value of <a vehicle>

- \_\_\_ b. In the menu that opens on <condition>, click **the age (in years) of <a driver> at <a date>**.

- \_\_ c. In the menu that opens on <a driver>, click the **driver** variable.



- \_\_ d. Click <a date> and in the menu, click **the start date of <a third party insurance>**.  
 \_\_ e. In the menu that opens on <a third party insurance>, click 'the policy'.  
 \_\_ f. Press the Space bar and in the completion menu, click **is between <min> and <max>**.



- \_\_ g. In the menu that opens on <min>, click <number>, and change **0** to: 18  
 \_\_ h. Click <max>, click <number>, and change **0** to: 80

The condition statement now looks like the following example:

```
definitions
  set 'driver' to a driver in the drivers of 'the request' ;
  if
    it is not true that the age (in years) of driver at the start date of
    'the policy' is between 18 and 80
```

## Completing the action statement

- \_\_ 1. In the **then** part of the rule, click <action>; and in the menu, click **refuse the application because <a string>**.
- \_\_ 2. In the menu that opens on <a string>, click <string>.
- \_\_ 3. Provide an explanation for refusing insurance coverage because of age constraints.
  - \_\_ a. Make sure that the cursor is between the double quotation marks ("").
  - \_\_ b. Type the following explanation:  
  
At least one driver does not meet the age constraints
- \_\_ 4. Save your rule.

Your rule now looks like the following example.

```

definitions
    set 'driver' to a driver in the drivers of 'the request';
if
    it is not true that the age (in years) of driver at the start date of 'the
    policy' is between 18 and 80
then
    refuse the application because "At least one driver does not meet the age
    constraints" ;

```

**definitions**

set 'driver' to a driver in the drivers of 'the request';

**if**

*it is not true that* the age (in years) of **driver** at the start date of **'the policy'** is  
more than **18** and at most **80**

**then**

refuse the application because  
**"At least one driver does not meet the age constraints"**;

- 
- 5. For an example solution, see "["Solution for "Creating the “Driver age” rule without preconditions""](#)" on page 11-26.
- 



## Questions

Consider these questions:

- Can you write this rule differently?
  - Can the condition test be included in the definition statement?
- 

- 6. Click the **main** breadcrumb to return to the **Decision Artifacts** tab.

## Section 5. Using a precondition

In this part of the exercise, you rework the previous rule by using the variable along with a precondition to access only drivers that match the precondition test.

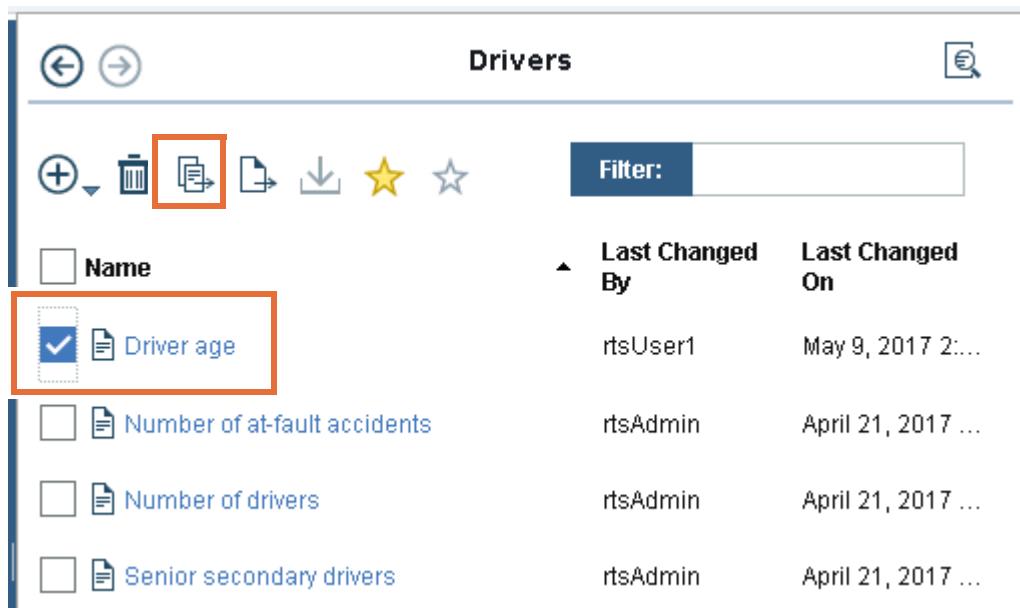
### Task:

- Rework the `Driver age` rule so that it uses a precondition
- Name the rule: `Driver age - precondition`
- Add this rule to the rule folder: `Check Eligibility/Drivers`

### 5.1. Rewriting the “Driver age” rule with a precondition

Change the rule to now include a `where` clause in the variable statement that defines the precondition.

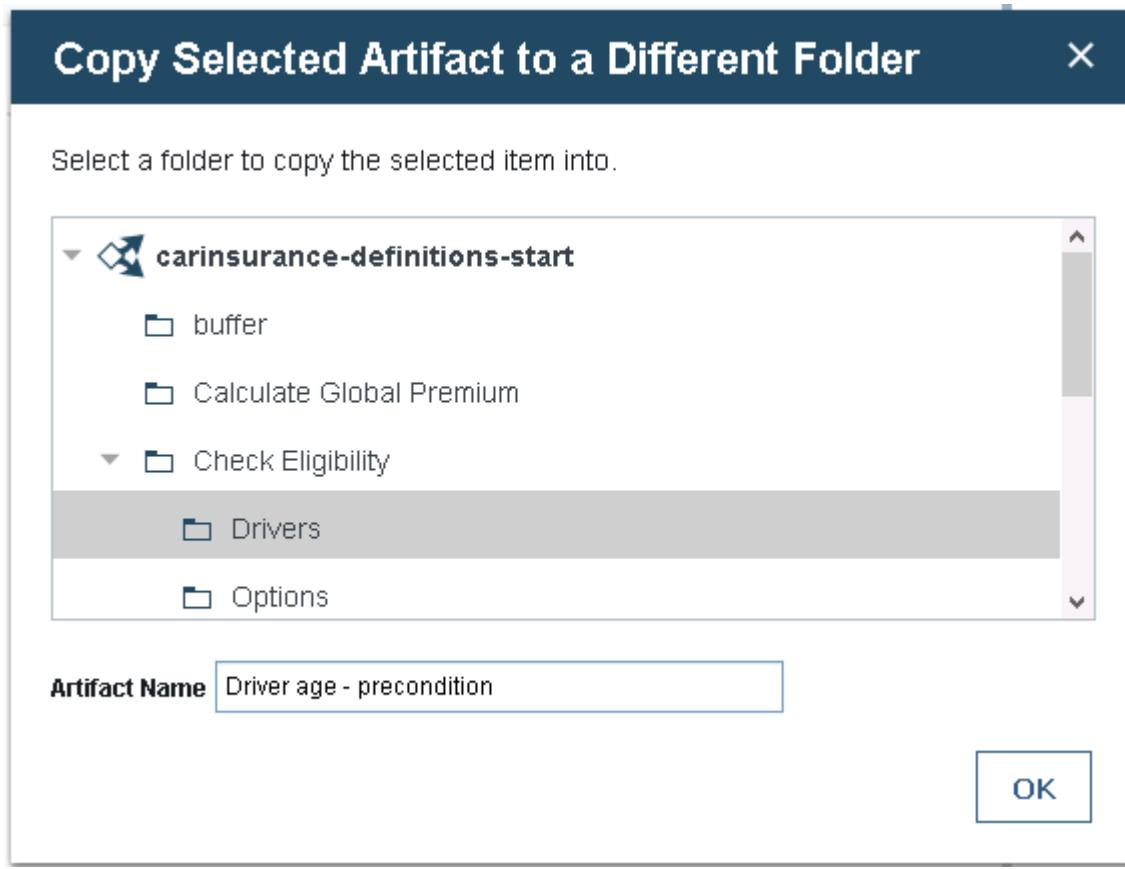
- 1. Make sure that you are on the **Decision Artifacts** tab, and that the **Check Eligibility > Drivers** folder is selected.
- 2. Copy the `Driver age` rule.
  - a. In the center pane, select the `Driver age` rule and click the **copy selected items** (documents and arrow) icon.



Name	Last Changed By	Last Changed On
<input checked="" type="checkbox"/> Driver age	rtsUser1	May 9, 2017 2:...
<input type="checkbox"/> Number of at-fault accidents	rtsAdmin	April 21, 2017 ...
<input type="checkbox"/> Number of drivers	rtsAdmin	April 21, 2017 ...
<input type="checkbox"/> Senior secondary drivers	rtsAdmin	April 21, 2017 ...

- b. In the “Copy Selected Artifact to a Different Folder” window:
  - Expand **carinsurance-definitions-start > Check Eligibility**, and click **Drivers**.
  - In the **Artifact Name** field, type: `Driver age - precondition`

- Click OK.



- 3. You now see the Driver age - precondition rule in the list of rule artifacts for **Check Eligibility > Drivers**.

Drivers		
<span style="float: left;">+ <input type="button" value="New"/></span> <span style="float: right;"><input type="button" value="Filter:"/></span>		
<input type="checkbox"/> Name	<input type="checkbox"/> Last Changed By	<input type="checkbox"/> Last Changed On
<input type="checkbox"/> Driver age	rtsUser1	May 9, 2017 2:...
<input checked="" type="checkbox"/> Driver age - precondition	rtsUser1	May 9, 2017 2:...
<input type="checkbox"/> Number of at-fault accidents	rtsAdmin	April 21, 2017 ...
<input type="checkbox"/> Number of drivers	rtsAdmin	April 21, 2017 ...

## Updating the definition statement

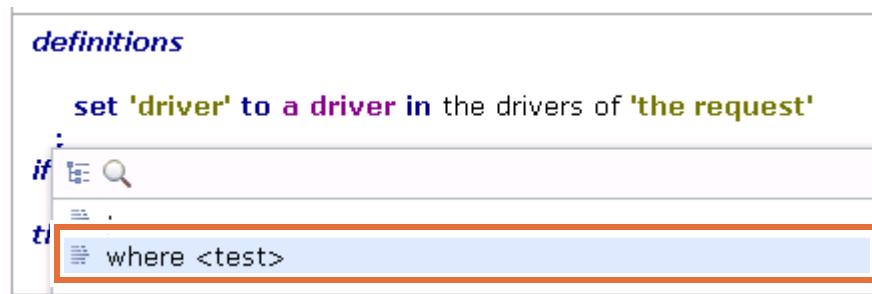
- 1. Open the Driver age - precondition rule in the rule editor.
- a. In the Drivers pane, hover the mouse pointer next to Driver age - precondition and click the **Edit this rule** icon (the pencil).



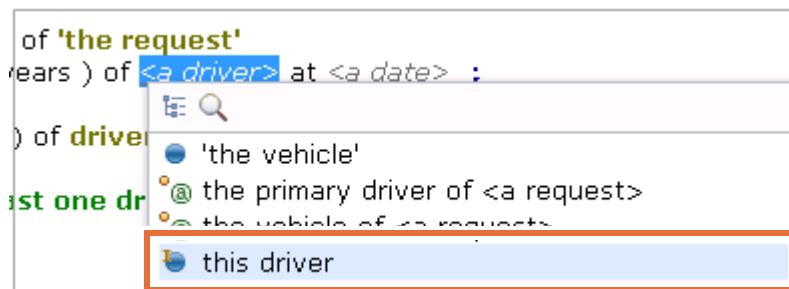
Alternatively, you can click Driver age - precondition, and in the Drivers pane toolbar, click the **Edit this rule** icon (the pencil).



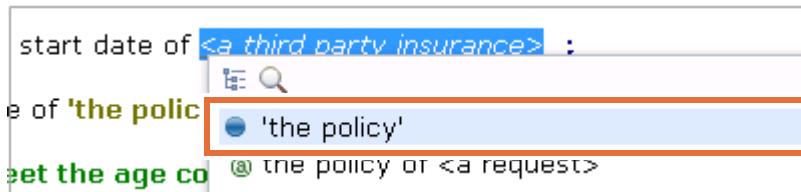
- 2. Add a **where** clause to the definitions statement that re-creates the condition statement from the **if** part of the rule.
- a. Place the cursor before the closing semicolon (;) at the end of the definitions statement:  
set 'driver' to a driver in the drivers of 'the request' ;
- b. Press Enter to go to a new line and then press the Space bar to open the completion menu.
- c. In the menu, click **where <test>**.



- d. In the menu that opens on <test>, click **it is not true that <condition>**.
- e. In the menu that opens on <condition>, click **the age (in years) of <a driver> at <a date>**, and click **this driver**.



- \_\_\_ f. Click <a date>, click **the start date of <a third party insurance>**, and click 'the policy'.



- \_\_\_ 3. Complete the where clause by including the 18-80 age range.
- \_\_\_ a. Press the Space bar and in the menu, click **is between <min> and <max>**.
  - \_\_\_ b. In the menu that opens on <min>, click <number>, and change **0** to: 18
  - \_\_\_ c. Click <max>, click <number>, and change **0** to: 80



## Information

### **Working with the “where” clause**

The **where** clause automatically includes an implicit variable for the object that is being tested. As this definition is tested against the list of drivers in the request, **this driver** takes each driver in that list as a value.

As soon as **this driver** does not meet the age constraint, your **driver** variable is assigned the value of **this driver**, and the rest of the rule statement is then evaluated.

- \_\_\_ 4. Delete the entire **if** part of the rule.
- \_\_\_ 5. Save the rule.



## Note

You can ignore the warning message about the **driver** variable.

- \_\_\_ 6. For an example solution, see "[Solution for "Rewriting the “Driver age” rule with a precondition"](#)" on page 11-27.
- \_\_\_ 7. Click the **main** breadcrumb to return to the **Decision Artifacts** tab.

## Section 6. Defining a collection variable

In this part of the rule, your variable defines a subset of all drivers that match the precondition.

### Task:

- Rework the `Driver age` rule so that it defines a collection of drivers who do not meet the age requirements. The rule action executes for each object in the collection to indicate the number of ineligible drivers.
- Name the rule: `Driver age - collection`
- Add this rule to the rule package: `Check Eligibility/Drivers`

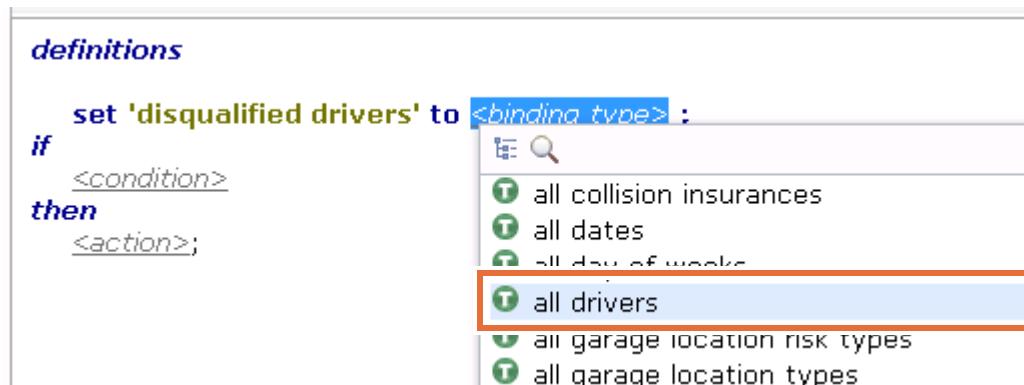
### 6.1. Rewriting the “Driver age” rule with a collection variable

This rule uses a collection variable to iterate through all the drivers included in the request. The action statement also iterates through all the drivers “captured” in the collection variable.

- 1. Make sure that you are on the **Decision Artifacts** tab, and that the **Check Eligibility > Drivers** folder is selected.
- 2. If needed, return to the list of decision artifacts by clicking **Drivers** in the left explorer pane.
- 3. Create a rule and specify a name for the rule.

#### Defining a collection variable

- 1. In the rule editor, create a **definitions** section.
- 2. Define the variable.
- a. Define **variable1** as `disqualified drivers` and set it to `all drivers`.



- b. Press the Space bar and in the menu, click **in <collection>**.
- c. Click **the drivers of <a request>** and then click **'the request'**.
- d. Making sure that the cursor is before the final semicolon (;), press Enter to go to a new line.
- 3. Add a **where** clause that defines the same test that you used in ["Rewriting the “Driver age” rule with a precondition"](#) on page 11-18.



## Important

Notice that because you used a collection variable, the **where** test automatically defines an implicit variable called “each driver” (instead of “this driver”).

**definitions**

```
set 'disqualified_drivers' to all drivers in the drivers of 'the request'
where it is not true that the age ( in years ) of <a driver> at <a date> :
if
  <condition>
then
  <action>;
```

The screenshot shows a context menu with several options: 'the vehicle', 'each driver' (which is highlighted with a red box), 'the primary driver of <a request>', and 'the vehicle of <a request>'.

## Defining the action statement

- 1. Delete the **if** part of the rule, including the **<condition>** placeholder.
- 2. In the **then** part of the rule, click: **<action>**;
- 3. Notice that the menu includes two different options that include a **for each** statement.

**then**

**<action>:**

The screenshot shows a context menu with several options: 'for each <term> called 'variable1'', 'in <collection> : - <action> ,', 'for each <term> in <collection> : - <action> ,', '@ print <a string>', and '@ refuse the application because <a string>'.

Note how the **for each** constructs are formulated. They both have a similar structure, though the first **for each** construct references a variable definition for the term.

The basic for each construct comprises the following structure:

```
then
  for each <term> in <collection> :
    - <action> ,
```

In this construct, you first define which objects and which list to evaluate. The hyphen (-) is like a bullet in a bulleted list. You can define one or several actions to take for each object in the list.



## Questions

In this rule, you want to perform the action for each driver. Which construct do you choose?

---

## Answer

You must use the `for each <term>` in `<collection>` construct. In this rule, you define the term as each driver, and the variable `disqualified drivers` is a collection.

Because your action is to refuse ineligible drivers, you iterate only through the list of drivers that is captured in the **disqualified drivers** collection variable.

- 
- 4. Complete the action statement to read:

```
then
  for each driver in 'disqualified drivers' :
    - refuse the application because At least one driver does not meet the age
constraints, ;
```



### Important

The action statement must include both a comma (,) and a semicolon (;):

- The purpose of the comma is to separate nested “for each” statements in the situations where they are used.
  - While this rule does not have multiple “for each” statements, the comma is still included in the statement by the rule editor, and is not a bug.
- The semicolon is required because it indicates the end of the **then** part of the rule.

```
then
  for each driver in 'disqualified drivers' :
    - refuse the application because
"At least one driver does not meet the age constraints" ;
```

- 
- 5. Save your rule.

## 6.2. Reviewing the “Driver age” rules

See Unit 9, "Working with definitions in rules", [Figure 9-56, "Exercise review: Solution 1,"](#) on page 9-66 for a review and discussion of this exercise.

In this exercise, you wrote the “Driver age” rule three different ways:

- With an **if** statement
- With a precondition
- With a collection variable



## Questions

Consider the following questions:

- Which approach do you prefer?
- Why do you prefer this approach?
- In what situations might you want to take each approach?

---

— 6. For an example solution, see ["Solution for "Rewriting the “Driver age” rule with a collection variable""](#) on page 11-27.

**End of exercise**

## Section 7. Solutions

This section shows some example solution rules for the previous exercises.

### Section 2, "Using the “from <object>” construct"

#### 7.1. Solution for "Creating the “Collision cover in [min, max]” rule"

```
definitions
  set 'collision policy' to a collision insurance from 'the policy'
if
  the collision max amount cover desired of 'collision policy' is between 2500
  and 125000
then
  set the collision max amount cover proposed of 'collision policy' to the
  collision max amount cover desired of 'collision policy'
```

### Section 4, "Accessing objects in a collection"

#### 7.2. Solution for "Creating the “Driver age” rule without preconditions"

```
definitions
  set 'driver' to a driver in the drivers of 'the request';
if
  it is not true that the age (in years) of driver at the start date of 'the
  policy' is between 18 and 80
then
  refuse the application because "At least one driver does not meet the age
  constraints" ;
```

```
definitions
  set 'driver' to a driver in the drivers of 'the request';
if
  it is not true that the age (in years) of driver at the start date of 'the
policy' is more than 18 and at most 80
then
  refuse the application because
"At least one driver does not meet the age constraints";
```

### Section 5, "Using a precondition"

### 7.3. Solution for "Rewriting the “Driver age” rule with a precondition"

```

definitions
  set 'driver' to a driver in the drivers of 'the request'
    where it is not true that the age (in years) of this driver at the
    start date of 'the policy' is between 18 and 80;

then
  refuse the application because "At least one driver does not meet the age
constraints" ;

```

```

definitions
set 'driver' to a driver in the drivers of 'the request'
  where it is not true that the age (in years) of this driver at the start date of 'the
policy' is between 18 and 80 ;
then
  refuse the application because
"At least one driver does not meet the age constraints" ;

```

## Section 6, "Defining a collection variable"

### 7.4. Solution for "Rewriting the “Driver age” rule with a collection variable"

```

definitions
  set 'disqualified drivers' to all drivers in the drivers of 'the request'
    where it is not true that the age ( in years ) of each driver at the
    start date of 'the policy' is between 18 and 80 ;
then
  for each driver in 'disqualified drivers' :
  - refuse the application because "At least one driver does not meet the age
constraints" , ;

```

```

definitions
set 'disqualified drivers' to all drivers in the drivers of 'the request'
  where it is not true that the age (in years) of each driver at the
start date of 'the policy' is between 18 and 80 ;
then
  for each driver in 'disqualified drivers':
  - refuse the application because
"At least one driver does not meet the age constraints" , ;

```

## Exercise review and wrap-up

The first part of the exercise covered how to use various BAL constructs to write rule definitions.  
The last part of the exercise looked at possible approaches to writing the “Driver age” rule.

# Exercise 12. Writing complete rules

## Estimated time

01:45

## Overview

This exercise continues the focus on rule authoring. It provides practice with various types of action statements and writing complete rules.

## Objectives

After completing this exercise, you should be able to:

- Write clear and unambiguous action statements in rules
- Apply the concepts of building definition, condition, and action statements to write complete rules

## Introduction

The tasks in this exercise help you to further apply the skills you learned to build definition, condition, and action statements.

In the previous exercises, you followed step-by-step instructions. During this exercise, you are given the business policy and some hints, but you apply the skills to formulate the statements yourself.

This exercise is based on the car insurance scenario.

The exercise includes these sections:

- [Section 1, "Writing actions that use expressions"](#)
- [Section 2, "Writing a computation rule that always executes"](#)
- [Section 3, "Writing rules that perform actions on each object in a collection"](#)
- [Section 4, "Solutions"](#)

## Requirements

This exercise requires that you work in the Decision Center Business console in the computer lab environment.

To do this exercise, the following exercise must be completed:

- [Exercise 10, "Working with conditions in rules,"](#) on page 10-1
- [Exercise 11, "Working with definitions in rules,"](#) on page 11-1

# Before starting the exercise

## Starting the sample server

Before proceeding, make sure that the sample server is running.

## Working with car insurance rule projects

For this exercise, you can continue working in the decision service that you used for the previous exercise, or you can switch projects and start with: `carinsurance-actions-start`

To switch projects:

1. On the **Library** tab, find the `carinsurance-actions-start` decision service.
2. Click the white space of the `carinsurance-actions-start` section, and then click **main**.
3. Make sure that you are on the **Decision Artifacts** tab.
4. To access the rules in this decision service, expand **carinsurance-actions-start** to see its subfolders.



### Important

Save your rule authoring work frequently while you work in the Business console rule editor. Saving your rule while it is in progress helps you to avoid losing your work if your Business console session times out.

You can save your work in the rule editor by clicking **Save** and then **Create New Rule**. You can return to the rule editor by clicking **Edit**.

## Section 1. Writing actions that use expressions

In the previous exercises, you defined actions that executed as soon as the condition or precondition was met. You also defined actions for each object that met the condition or precondition.

In this part of the exercise, you use an expression in the action to provide the name of the driver that matches the ineligibility condition.

### 1.1. Creating the “Total number of accidents” rule

Based on the business policy, write a rule to identify which drivers fail eligibility constraints.

---

#### **Business policy:**

Insurance coverage is refused if any driver named on the policy had more than four accidents.

---

#### **Task:**

- Write a rule that checks each driver included in the insurance request and tests how many accidents they had.
- The rule rejects the request when any of the drivers had more than four accidents, and includes the name of the problem driver in the explanation.

The refusal statement should include the following information, in this format:

```
"Driver " + <driver.firstName> + "<space>" + <driver.lastName> +
" <explanation>"
```

- Name this rule: Total number of accidents
- Add this rule to the rule folder: Check\_Eligibility/Drivers
- Create a collection variable: bad\_driver
- You do not need a precondition.
- Compare your rule statement to "[Solution for "Creating the “Total number of accidents” rule"](#)" on page 12-13.



#### Note

You can start in the carinsurance-actions-start project. See "[Before starting the exercise](#)" on page 12-2.



### Hint

To build the refusal statement, keep in mind the following items:

- Putting text between double quotation marks ("") inserts the text as a string into the action statement.
  - Putting a space between double quotation marks inserts the space as a blank character into the action statement.
  - You can concatenate strings and vocabulary by typing a plus sign (+) to connect them.
- 
- 1. To start building the refusal statement, click **<action>**; and click **refuse the application because <a string>**.
  - 2. In the menu, click **<string>** and type the following text for the initial string: "Driver "
- 



### Important

The text string should include a space after the word **Driver**.

---

- 3. To concatenate a vocabulary item for the driver first name, place the cursor after the closing quotation mark ("") of the first string, type a space, and type a plus sign (+).
  - 4. Press the Space bar to open the completion menu.
  - 5. Select **the first name of <a driver>** from the menu and complete the **<a driver>** placeholder by clicking **'bad driver'**.
  - 6. Continue building the refusal statement by typing the plus sign (+) and by using the completion menu to add the remaining strings and vocabulary phrases.
-

## 1.2. Creating the “Number of license withdrawals” rule

---

### Business policy:

Insurance coverage is refused when any driver or drivers that are named on the policy lost their driver's license more than once.

---

### Task:

- Write a rule that checks each driver included in the insurance request to test whether they ever had their driver's license revoked (or withdrawn).
  - The rule rejects the request if any of the drivers lost their license more than once, and includes the name of the problem driver in the explanation.
  - Name this rule: Number of license withdrawals
  - Add this rule to the rule folder: Check Eligibility/Drivers
  - Define a variable and use a precondition.
  - Compare your rule statement to ["Solution for "Creating the “Number of license withdrawals” rule""](#) on page 12-13.
-

## Section 2. Writing a computation rule that always executes

Computation rules often do not require a condition statement. You can calculate a value in a computation rule, and reuse that calculated value in other rules.

---

### Business policy:

A total premium is calculated for all insurance requests that meet the eligibility requirements. All quotes for insurance policies that are proposed to customers must include tax.

The equation for calculating the price is:

`Total premium = (Tax rate % + 100) * Total price before tax / 100`

---

### Task:

- Write a rule that computes the global premium plus tax for all insurance quotations.
- Name this rule: Calculate collision global price tax included
- No variables or conditions are required.
- Add this rule to the rule folder: Calculate Global Premium
  - In the left explorer pane, click the **Calculate Global Premium** folder to make it active.
- Compare your rule statement with "[Solution for "Creating the “Calculate collision global price tax included” rule"](#)" on page 12-13.

---

## 2.1. Creating the “Calculate collision global price tax included” rule

To define the equation to calculate the insurance premium, you must create an expression in the action statement that represents:

`(amount_1 + amount_2) * amount_3 / amount_2`



### Questions

Why does this rule *not* require a definition or condition statement?

---



---



### Hint

In the **then** part of the rule, look for the statement **set the total price of <a third party insurance>** to start the action statement.

---



## Information

### Expressions and operators

- To build an expression that requires parentheses [( )], type an opening parenthesis at the beginning of the expression and a closed parenthesis at the end of the expression.
- To use mathematical operators in an expression, type the operator that you need.
- Use a combination of typing and using the completion menu to build the action statement.

## Section 3. Writing rules that perform actions on each object in a collection

In the previous exercises, your actions executed once for each rule. Next, you write actions that are repeated for each object that matches the rule conditions.

---

### **Business policy:**

In addition to basic third-party liability and collision insurance, customers can select optional coverage, such as theft, vandalism, or fire. Each option is priced separately. The total premium is calculated according to the coverage amounts requested and the options chosen.

The formula for calculating the premium with options before tax:

Total policy price before tax = (basic third-party premium + collision package premium) + (for each option selected: the option price)

---

### **Task:**

- Write a computation rule that calculates the total price (before tax) of collision insurance plus options. The rule includes the amount for each additional option.
- Name this rule: Calculate collision global premium before tax
- Add this rule to the rule folder: Calculate global premium
- Define a variable that matches the collision insurance type (by using the “from” construct)
- Define a collection variable to capture all options that are selected in the request.
- No conditions are required.
- Compare your rule statement to ["Solution for "Creating the “Calculate collision global premium before tax” rule""](#) on page 12-14

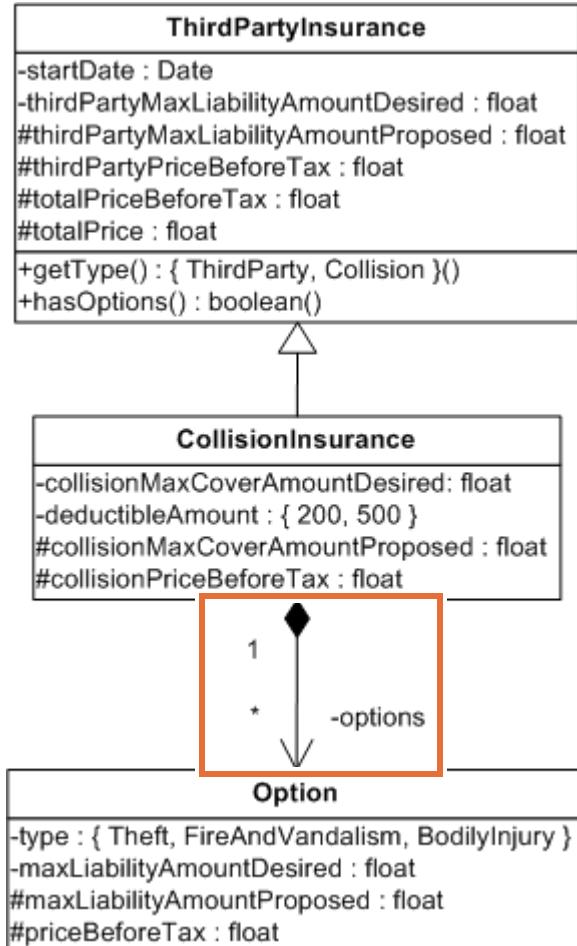
---

### 3.1. Creating the “Calculate collision global premium before tax” rule

As you learned earlier, long verbalization names are meaningful, but they can become awkward when they are used repeatedly in a rule statement. To clarify this computation rule, you can define shorter names to use within the scope of the rule so that the calculation is easier to read and understand.

## Strategy

To determine which options were selected in the request, recall the BOM structure.



Options are only available from the **CollisionInsurance** class. Therefore, only requests for collision insurance include options.

To identify each option that is included in the request, you create a collection variable, and during the computation, you iterate over that variable to access the additional price per option.



### Hint

#### Building the rule

- Remember to use parentheses and mathematical operators as needed to create or perform calculations.
- Click placeholders and use the completion menu to select vocabulary items to build the rule.
- As depicted in the class diagram, you must define the collision insurance variable before you define a variable for the options.

## 3.2. Creating the “Risky co-drivers” rule

### **Business policy:**

The insurance premium is based on the driving history of the primary driver and any secondary drivers that are to be included.

Secondary drivers who were at fault for one or more accidents in the past five years, or lost their license at least once, are considered high risks. Insurance coverage can still be offered if these high-risk drivers are not the primary driver. However, the premiums are higher.

The formula for calculating the premium before tax with a surcharge for each additional high-risk driver:

Total before tax = basic premium + (20 for each high risk driver)

---

### **Task:**

- Write a business rule that checks the driving history of secondary drivers for the number of at-fault accidents or license withdrawals. The rule action increases the premium by 20 for each high-risk secondary driver.
  - To evaluate the driving history for each secondary driver, define a collection variable that captures only secondary drivers who meet the risk criteria: `risky codrivers`
  - Name this rule: `Risky co-drivers`
  - Add this rule to the rule folder: `Calculate Global Premium`
  - Compare your rule statement to "[Solution for "Creating the “Risky co-drivers” rule"](#)" on page 12-14
- 



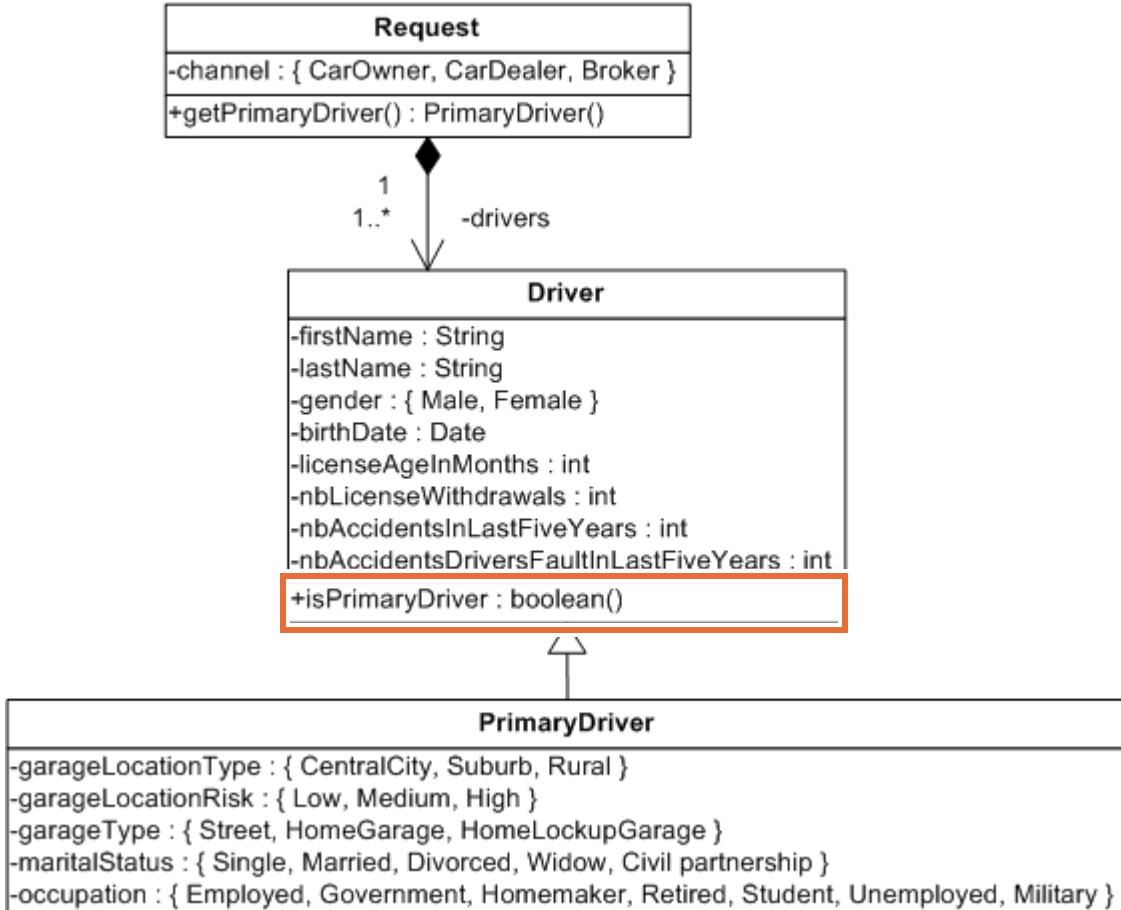
### **Questions**

Consider:

- Which BAL construct can you use to define a variable that accesses each driver?
  - Look at the `Driver` class in the BOM. Which BOM member can you use to test to determine whether the variable is set to the primary driver?
-

## Strategy

To determine whether a driver is the primary driver, recall the BOM structure.



The **Driver** class includes the `isPrimaryDriver` method, which returns a value of true or false. You can test that method for each driver to exclude primary drivers from your collection variable.



### Hint

Because you are using a collection variable, the vocabulary menu automatically includes these BAL constructs so that you can iterate over all the objects that are collected in that variable:

- **each** construct in the **where** clause
- **for each** construct in the **then** part of the rule



### Reminder

#### Building the rule

- Use parentheses and mathematical operators as needed to create expressions and perform calculations.
- Click placeholders and use the completion menu to select vocabulary items.

## End of exercise

## Section 4. Solutions

This section provides solution rule statements to the previous exercises.

### Section 1, "Writing actions that use expressions"

#### 4.1. Solution for "Creating the “Total number of accidents” rule"

```

definitions
  set 'bad driver' to a driver in the drivers of 'the request' ;
  if
    the number accidents of 'bad driver' is more than 4
  then
    refuse the application because "Driver " + the first name of 'bad driver' +
    " + the last name of 'bad driver' + " has had too many accidents" ;

```

#### 4.2. Solution for "Creating the “Number of license withdrawals” rule"

```

definitions
  set 'bad driver' to a driver in the drivers of 'the request'
  where the number license withdrawals of this driver is more than 1;
  then
    refuse the application because "Driver " + the first name of 'bad driver' +
    " " + the last name of 'bad driver' + " has had too many license withdrawals"
    ;

```

### Section 2, "Writing a computation rule that always executes"

#### 4.3. Solution for "Creating the “Calculate collision global price tax included” rule"

```

then
  set the total price of 'the policy' to (100 + 'tax rate percentage') * the
  total price before tax of 'the policy' / 100 ;

```

### Answer

This rule does not require a definition or condition statement because it always executes.

## Section 3, "Writing rules that perform actions on each object in a collection"

### 4.4. Solution for "Creating the “Calculate collision global premium before tax” rule"

definitions

```
set 'collision policy' to a collision insurance from 'the policy' ;
set 'collision options' to the options of 'collision policy' ;
then
  set the total price before tax of 'the policy'
  to ( the third party price before tax of 'the policy'
    + the collision price before tax of 'collision policy' );
for each option in 'collision options' :
  - set the total price before tax of 'the policy'
    to the total price before tax of 'the policy'
    + the price before tax of this option, ;
```



#### Information

In the “Calculate collision global premium” rule, a comma (,) appears before the closing semicolon (;) of the “for each” collection variable statement in the action statement.

**then**

set the total price before tax of '**the policy**' to ( the third party price before tax of '**the policy**' + the collision price before tax of '**collision policy**' );

**for each option in 'collision options'**:

- set the total price before tax of '**the policy**' to the total price before tax of '**the policy**' + the price before tax of **this option**, ;

The purpose of this comma is to separate nested “for each” statements in the situations where they are used.

While this rule does not have multiple “for each” statements, the comma is still included in the statement by the rule editor, and is not a bug.

### 4.5. Solution for "Creating the “Risky co-drivers” rule"

definitions

```
set 'risky codrivers' to all drivers in the drivers of 'the request'
  where it is not true that each driver is the primary driver of 'the
  request'
```

```
    and ( the number of at-fault accidents in the last five years of each
driver is at least 1
        or the number license withdrawals of each driver is at least 1 );
then
    for each driver in 'risky codrivers' :
        - set the total price before tax of 'the policy' to the third party price
before tax of 'the policy' + 20, ;
```

---



### Information

As with the previous rule, a comma (,) appears before the closing semicolon (;) of the “for each” collection variable statement.

However, this comma is not a bug. Its purpose is to separate nested “for each” statements in the situations where they are used. While this rule does not have multiple “for each” statements, the comma is still included in the statement by the rule editor.

---

## Exercise review and wrap-up

The first part of the exercise looked at how to write clear and unambiguous action statements in rules. In the last part of the exercise, you applied the concepts of building definition, condition, and action statements to write complete rules.

# Exercise 13. Authoring decision tables

## Estimated time

01:00

## Overview

This exercise continues the focus on rule authoring. You learn how to create decision tables.

## Objectives

After completing this exercise, you should be able to:

- Use the decision table editor to create a decision table
- Change rule conditions in a condition cell

## Introduction

This exercise includes these sections:

- [Section 1, "Creating a decision table"](#)
- [Section 2, "Editing conditions in a decision table"](#)
- [Section 3, "Solutions"](#)

---

### Scenario:

After analyzing the rules captured during the discovery phase, several rule patterns were identified as good candidates to implement as decision tables. Some of these rules involve eligibility, while others are computation rules that calculate premiums according to certain criteria. Your task is to identify symmetrical rule patterns that you can implement as decision tables.

---

## Requirements

This exercise requires that you work in both the Decision Center Business console and the Decision Center Enterprise console in the computer lab environment.

The first part of the exercise first outlines the strategy of transforming business policy into a decision table. Then, you walk through the steps of implementing a decision table in the Business console.

The next part of the exercise provides another business policy from a typical Business Rule Document (BRD). This section gives you the opportunity to build a decision table without detailed step-by-step instructions. A solution for this decision table is provided at the end of the exercise.

Finally, you work with the Decision Tree editor in the Enterprise console to see how to build a decision tree.

To do this exercise, the following exercise must be completed:

- [Exercise 10, "Working with conditions in rules,"](#) on page 10-1
- [Exercise 11, "Working with definitions in rules,"](#) on page 11-1
- [Exercise 12, "Writing complete rules,"](#) on page 12-1

## Before starting the exercise

### Starting the sample server

Before proceeding, make sure that the sample server is running.

### Working with car insurance rule projects

For this exercise, you can continue working in the same project that you used during the previous exercise, or you can switch projects and start with: `carinsurance-dt-start`



#### Important

Save frequently to avoid losing work on decision tables if your Business console session times out. Click **Save** and then **Create New Version** to save your changes. You can reopen the decision table editor by clicking **Edit**.

## Section 1. Creating a decision table

In this exercise, you create a decision table that calculates insurance premiums.

### Business policy

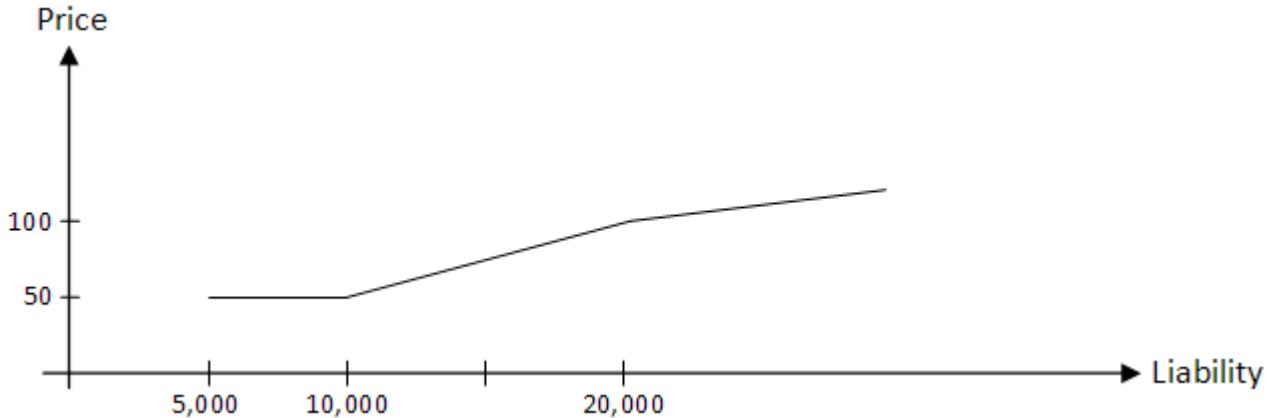
Insurance premiums are calculated according to the type and amount of coverage. For minimum third-party liability, the premium is composed of a base price plus a percentage of the liability amount. Both the base price and the rate of percentage vary according to the amount of liability.

The formula to calculate the base premium:

$$\text{Premium} = \text{Base price} + \text{Variable rate} * \text{Third-party liability amount}$$

### Strategy

Calculation of the basic premium is a piecewise linear function that depends on the third-party liability amount.



Implementing this formula in a decision table can be tricky if you try to include the slope value in an action statement. For example, one implementation might include redundant columns if you use this formula:

```
if liability is between x and x
then apply slope starting from x and ending at x
```

This table defines the values that are used in the piecewise linear function:

Third-party liability		Base amount	Variable rate
[ min	max ]		
5 000	10 000	50	
10 000	20 000		0.0005
>= 20 000		80	0.0001

Note the empty cells in the **Base amount** column and the **Variable rate** column. When you implement this formula as a decision table, you can set the value to 0.

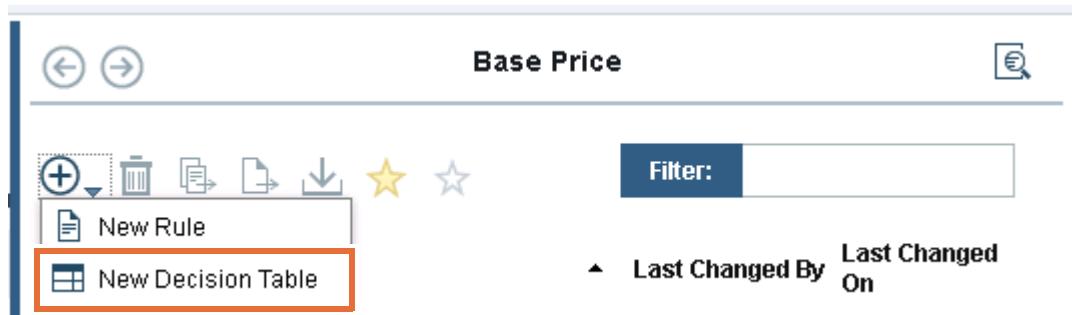
## Task

- Create a decision table that calculates the base price for third-party liability insurance according to liability amounts.
- Make sure that you are in the rule folder: Price Third-Party/Base Price
- Create a decision table artifact
- Name this decision table: Compute base price

### 1.1. Creating the “Basic coverage premium” decision table

To become familiar with the Decision Table editor, follow the provided steps to create a simple decision table.

- 1. Make sure that you are logged in to the Business console as `rtsUser1`.
- 2. You can continue to work in the same decision service as before, or you can select the `carinsurance-dt-start` decision service.
  - a. To open the `carinsurance-dt-start` decision service, go to the **Library** tab.
  - b. Click the white space in the `carinsurance-dt-start` section, and click **main**.
  - c. Make sure that you are on the **Decision Artifacts** tab.
- 3. Select the folder for the decision table.
  - a. In the left explorer pane, expand **carinsurance-dt-start**.
  - b. Expand the **Price Third-Party** folder, and click the **Base Price** subfolder.
- 4. Create a decision table that is called `Compute base price table`.
  - a. In the center pane, click the **create an artifact** icon (the plus sign) and click **New Decision Table**.



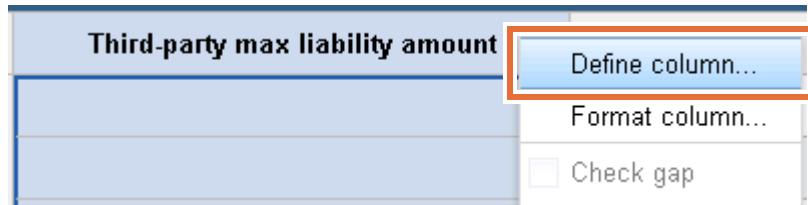
- b. Enter `Compute base price table` for the name, and click **Create**.

A new decision table opens in the decision table editor. Gray shading indicates a condition column, and the columns are labeled **C0**, **C1**, and **C2** to indicate that they are condition columns. Blue shading indicates an action column, and the column is labeled **A0** to indicate that it is an action column.

	C0	C1	C2	A0
1				
2				
3				
4				

## 1.2. Defining the condition column

- 1. Double-click the column heading **C0** and overwrite it with a new heading: **Third-party max liability amount**
- 2. Add the condition expression for the **Third-party max liability amount column**.
  - a. Right-click the column header and in the menu, click **Define column...**.

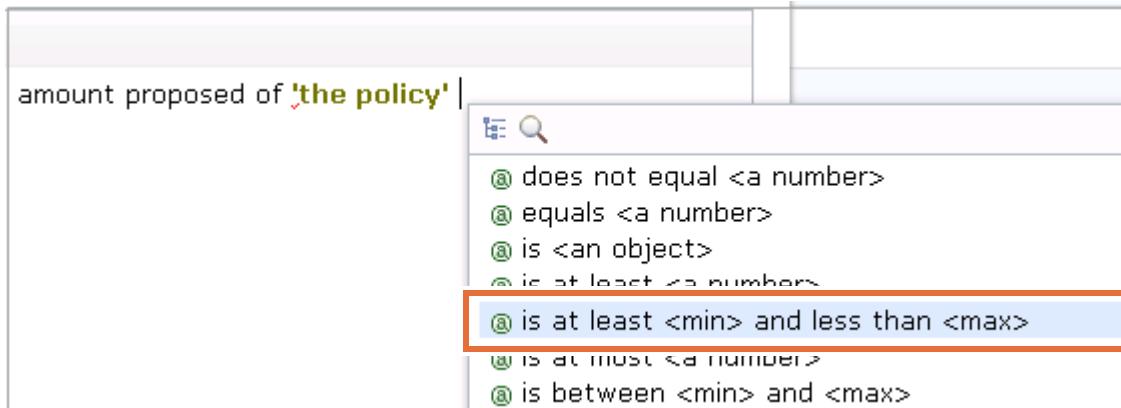


- b. In the Define Condition Column window, press the Space bar, and in the menu, click the **third party max liability amount propose of <a third party insurance>**.

The dialog box has a title bar 'Define Condition Column'. Below it is a instruction 'Use this editor to write a condition for the column.' At the top left are navigation icons. A search bar is present above a list of conditions. The list includes several items starting with '@', followed by a brief description. One item, 'the third party max liability amount proposed of <a third party insurance>', is highlighted with a red box.

- c. Click **<a third party insurance>** and click 'the policy'.

3. Press the Space bar and in the menu, click **is at least <min> and less than <max>**.



Do *not* set the **min** and **max** placeholders. These placeholders are filled through the values that you enter in the table cells.

4. To finish editing the condition column, click **Define**.

The screenshot shows the "Define Condition Column" dialog box. The title bar says "Define Condition Column" and has a close button. The main area contains the text "the third party max liability amount proposed of 'the policy' is at least <min> and less than <max>". Below this is a table with three columns: "Severity", "Line", and "Message". At the bottom right of the dialog is a "Define" button, which is highlighted with a red border.

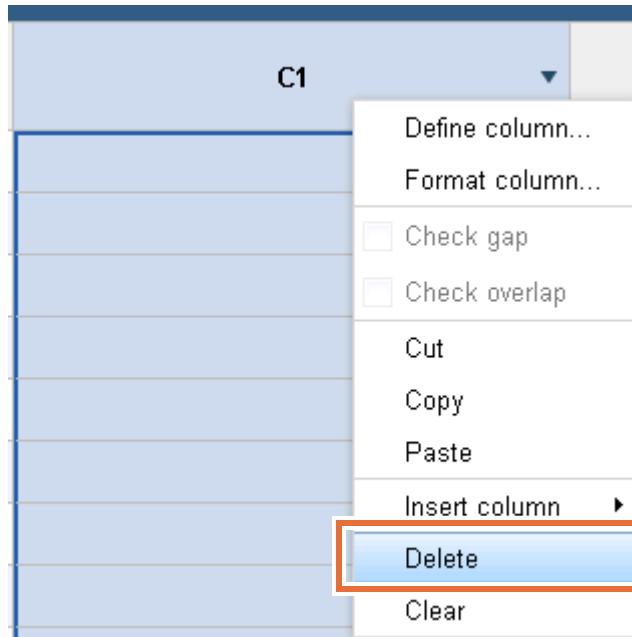


## Information

You can see the expressions that are defined for columns by hovering the mouse pointer in the column header.

	Third-party max liability amount		C1
	min	max	
1	the third party max liability amount proposed of ' <b>the policy</b> ' is at least <min> and less than <max>		
2			

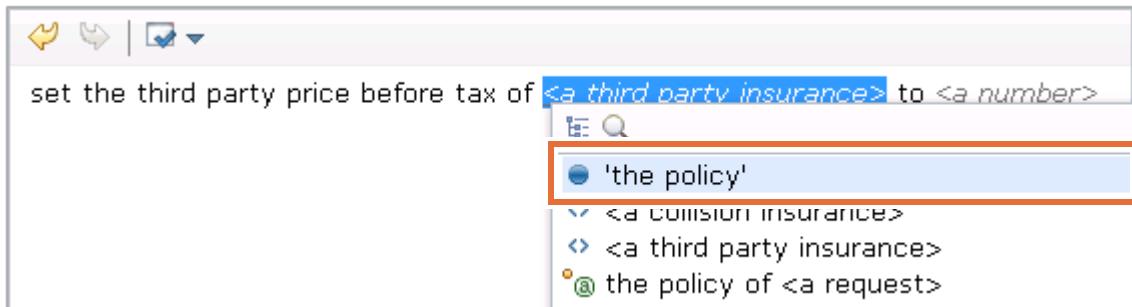
- 5. Because the table has no other conditions, remove the remaining condition columns by right-clicking the column headers and in the menu, clicking **Delete**.



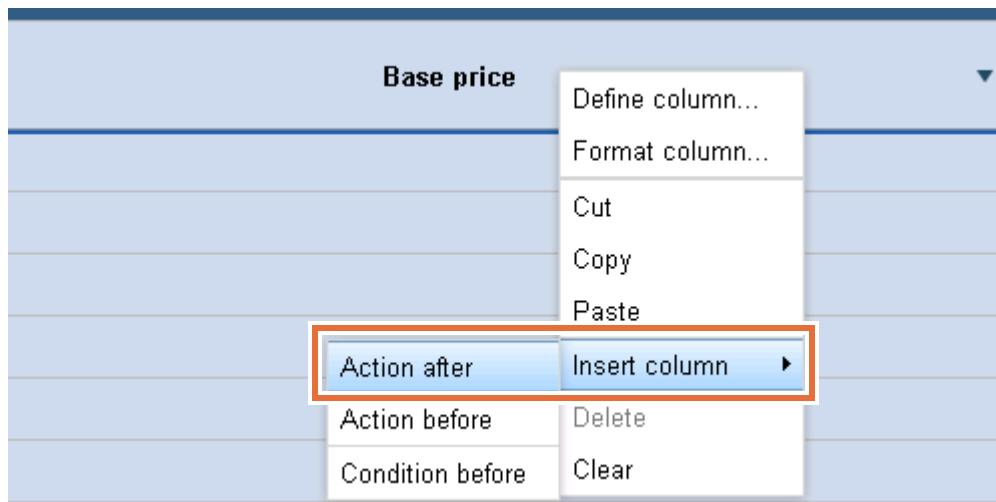
### 1.3. Defining the action columns

- 1. Define the first action column, which is called **Base price**.
  - a. Double-click the action column heading **A0** and overwrite it with a new heading: **Base price**
  - b. Right-click the **Base price** column header and in the menu, click **Define column**.
  - c. In the Define Action Column window, press the Space bar and in the menu, click **set the third party price before tax of <a third party insurance> to <a number>**.

- \_\_\_ d. Click the <a third party insurance> placeholder and click 'the policy', but do not set the number placeholder.



- \_\_\_ e. Save the action in this column by clicking **Define**.
- \_\_\_ 2. Add another action column after (to the right of) the **Base price** column.
- \_\_\_ a. Right-click the **Base price** column header, and in the menu, click **Insert column > Action after**.



A new action column that is labeled **A2** is added to the table.

- \_\_\_ 3. Change the column heading from **A2** to: **Variable rate**
- \_\_\_ 4. Define the action for the **Variable rate** column.
- \_\_\_ a. Set this column action to this statement:

```
set the third party price before tax of the policy to the third party
price before tax of the policy + <a number> * the third party max
liability amount proposed of the policy
```



### Reminder

You can type the mathematical operators and use the completion menu to complete the action statement.

- \_\_ b. Complete all the placeholders except the <a number> placeholder.

Use this editor to write an action for the column.

set the third party price before tax of 'the policy' to the third party price before tax of 'the policy' · **<a number>**

- \_\_ c. Complete this column by clicking **Define**.



### Reminder

Remember to save your work frequently by clicking **Save** and then **Create New Version**. You can reopen the decision table in the editor by clicking **Edit**.

## 1.4. Completing the rows of the table

- \_\_ 1. Starting with the first row, type the values directly into each cell to match the values from the business policy in "[Strategy](#)" on page 13-4.

Third-party liability		Base price	Variable rate
[ min	max [		
5 000	10 000	50	
10 000	20 000		0.005
>= 20 000		80	0.001



## Information

After you complete conditions and actions for each row, you can see the complete rule for each row by hovering the mouse pointer over the row number.

For row 1, the complete rule is:

```
if
  all of the following conditions are true:
    - (the third party max liability amount proposed of 'the policy' is at
      least 5000 and less than 10000),
  then
    set the third party price before tax of 'the policy' to 50;
```

	Third-party max liability amount		Base price
	min	max	
1	5,000	10,000	50
	10,000	20,000	

**if**  
**all of the following conditions are true :**  
 - (the third party max liability amount proposed of 'the policy' is at least 5000 and less than 10000 ),  
**then**  
 set the third party price before tax of 'the policy' to 50 ;

2. Notice that in the **Third-party max liability amount** column, row 3 shows an error because that column has only one numeric value: 20,000

	Third-party max liability amount	
	min	max
1	5,000	10,000
2	10,000	20,000
3	20,000	
4	Error Expression is incomplete.	
5		



## Questions

How do you resolve the error and merge the condition column cells row 3 so that the value changes to: **>=20000** ?

---

**Answer**

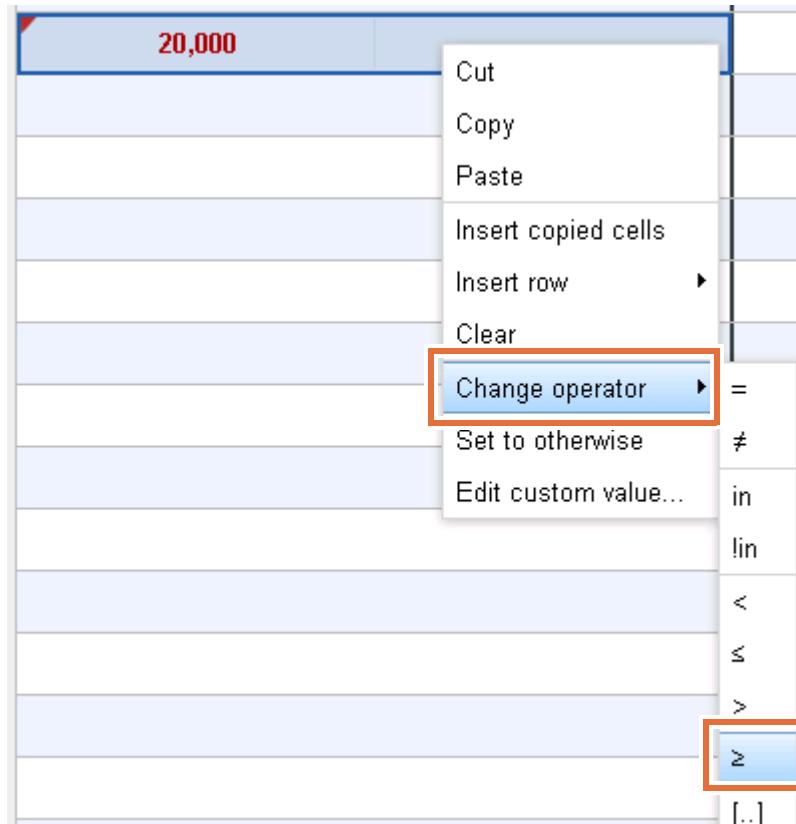
You must change the expression for row 3.

---

- \_\_\_ 3. Modify the BAL operator for the last row so that the condition becomes:  
the third party max liability amount proposed of the policy is at least 20000
- \_\_\_ a. In the **Third-party max liability amount** column, right-click the cells in row 3 to open the menu.



- b. In the menu, click **Change operator** and then click  $\geq$  (greater than or equal to).



The row 3 shows the new operator, and the error message no longer shows.

3	$\geq 20,000$
---	---------------

- c. Click **Save** and then **Create New Version** to save your changes.  
 d. Click the **Edit** icon to reopen the decision table in the editor.

## 1.5. Handling empty cells

Two of the cells in this table have no values.

When you have empty cells in condition columns, the rule is not affected. Condition statements are optional in rules, so an empty condition cell indicates that the condition does not apply in that rule.

However, you cannot have empty cells in action columns. Actions are required in rules, so an empty action cell causes an error.

## Entering zero (0) in cells



### Questions

If you set the empty cells to 0, what happens to the rule?

## Answer

For this rule, you can set the cell values to 0 without affecting the rule outcome. The calculations for those action statements produce a result of 0 and do not change the final computed price.

- 1. For the first row, change the cell value of the **Variable rate column** to: 0
- 2. Hover the mouse pointer over the row number (1) to see the full rule statement:

if

all of the following conditions are true:

- (the third party max liability amount proposed of 'the policy' is at least 5000 and less than 10000),

then

set the third party price before tax of 'the policy' to 50;

set the third party price before tax of 'the policy' to the third party price before tax of 'the policy' + 0 \* the third party max liability amount proposed of 'the policy';

	Third-party max liability		Base price	Variable rate
	min	max		
1	5,000	10,000	50	0
	10,000	20,000		0.005

**if**  
**all of the following conditions are true :**  
- (the third party max liability amount proposed of 'the policy' is at least 5000 and less than 10000 ),  
**then**  
set the third party price before tax of 'the policy' to 50 ;  
set the third party price before tax of 'the policy' to the third party price before tax of 'the policy' + 0 \* the third party max liability amount proposed of 'the policy' ;

The total base price in row 1 becomes:

the base price + 0 \* liability amount

That is, the value is 0 in this case.

- 3. In row 2, enter 0 for the **Base price column**.

For the second row, the action calculation becomes:

0 + 0.005 \* liability amount

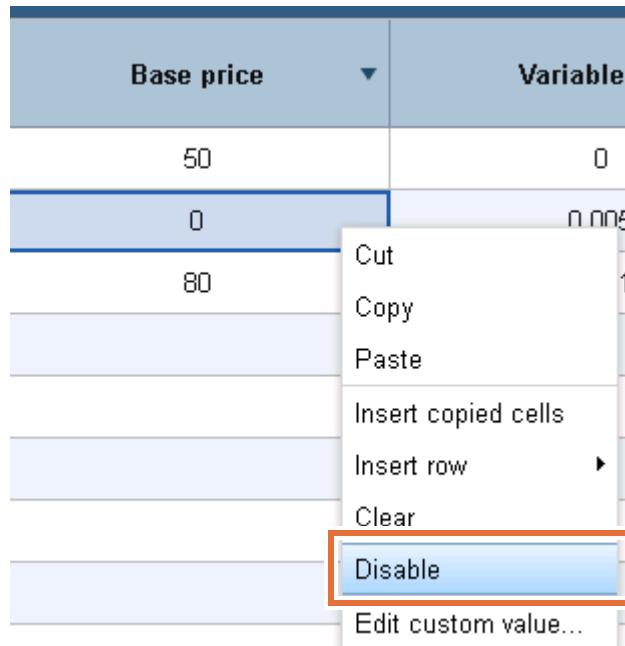
In both cases, the 0 causes the corresponding action statement to have no effect when it executes.

## Disabling cells

Another option for this rule is to disable the cell. When a rule includes only one action statement, disabling the action cell also disables that row (or rule).

Because this rule includes more than one action, the rule can still execute, but the action statement that corresponds to this cell does not execute.

- 1. Disable both of the cells with the value of 0 by right-clicking the cell, and in the menu, clicking **Disable**.



- 2. Notice that the cells display the **disabled** icon (a red circle with a diagonal line).

A screenshot of the same decision table after disabling the cells. The first two rows now have a red circle with a diagonal line icon in their "Base price" cells. The first row's "Variable" cell also contains this icon.

Base price	Variable rate
50	0
0	0.005
80	0.001

- 3. When you are finished, save the decision table.
- 4. Click the **main** breadcrumb to return to the **Decision Artifacts** tab.

## Section 2. Editing conditions in a decision table

In this section, you create a decision table that uses different operators in a condition statement.

### Business policy

The Business Rule Document (BRD) contains the description of each rule that was captured during the discovery phase.

The following excerpt comes from the BRD that describes the inference rule for senior primary driver risk.

#### 4.5.2. Rule Infer senior primary driver risk

<b>Rule name</b>	infer_senior_primary_driver_risk			<b>Rule ID</b>	R_4_5_2																	
<b>Classification</b>	Constraint			<b>Creation date</b>	<date>																	
<b>References</b>	Check Eligibility (UC 4), Primary Driver Eligibility (Step 5)																					
<b>Decisions</b>																						
<b>Business rule</b>	<table border="1"> <thead> <tr> <th>Driver age</th> <th>Number accidents in last 5 years</th> <th>Number at-fault accidents in last 5 years</th> <th>Result</th> </tr> </thead> <tbody> <tr> <td>[65, 70]</td> <td>&gt; 3</td> <td></td> <td>Refused</td> </tr> <tr> <td rowspan="2">] 70, 80]</td> <td rowspan="2"><math>\leq 3</math></td> <td><math>\leq 1</math></td> <td>P. Manually</td> </tr> <tr> <td>&gt; 1</td> <td>Refused</td> </tr> <tr> <td></td> <td>&gt; 3</td> <td></td> <td>Refused</td> </tr> </tbody> </table>				Driver age	Number accidents in last 5 years	Number at-fault accidents in last 5 years	Result	[65, 70]	> 3		Refused	] 70, 80]	$\leq 3$	$\leq 1$	P. Manually	> 1	Refused		> 3		Refused
Driver age	Number accidents in last 5 years	Number at-fault accidents in last 5 years	Result																			
[65, 70]	> 3		Refused																			
] 70, 80]	$\leq 3$	$\leq 1$	P. Manually																			
		> 1	Refused																			
	> 3		Refused																			
<b>Policy implemented</b>																						
<b>Business motivation</b>																						
<b>Objects affected</b>	Driver																					
<b>Changes</b>																						
<b>Who can change it</b>																						
<b>How</b>																						
<b>When</b>																						

**Note**

The term “P. Manually” means “process manually.” By default, the insurance request is accepted. However, note the following items:

- If the driver fails one of the constraints, the request is rejected.
- If the request is not rejected, and if at least one rule leads to the conclusion that further review is required, the request is passed to an underwriter to process manually.

## Tasks

- In the **Decision Artifacts** tab, in the **Check Eligibility** folder, create a subfolder that is called: **Primary Driver**
- Create a decision table in the new rule folder that is named: **Senior primary drivers**
- Check your work with the ["Solution for "Creating the “Senior primary drivers” decision table"](#).

## Creating the “Senior primary drivers” decision table

### 2.1. Creating the rule folder and starting the decision table

Before creating the table, first create a folder to contain rules that are specific to primary drivers.

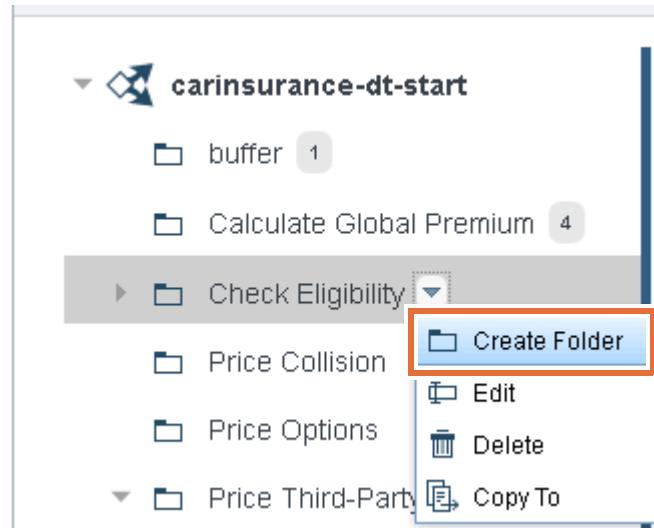
**Information**

You can use the carinsurance-dt-start decision service, or if you are not using this decision service, you can continue with the same decision service that you are already using.

- 1. Make sure that you are on the **Decision Artifacts** tab.
- 2. In the **Check Eligibility** folder, create a subfolder that is called **Primary Driver**.
  - a. In the left pane, hover the mouse pointer over the **Check Eligibility** folder, and click the down arrow to open the menu.



- \_\_ b. In the menu, click **Create Folder**.



- \_\_ c. In the “Create new Folder” window, type the following name for the folder: Primary Driver

**Create new Folder**

Specify a name for your new folder

Optionally, you can specify a group

Optionally, you can add a description

**Create**

- \_\_ d. Click **Create**.

Your new folder is now listed under **Check Eligibility**.

- \_\_ 3. In the new **Primary Driver** folder, create a decision table that is called: Senior primary drivers
- \_\_ a. Make sure that the **Check Eligibility > Primary Driver** folder is selected.

- \_\_\_ b. In the center pane, click the **create an artifact** icon (the plus sign) and click **New Decision Table**.
- \_\_\_ c. In the “Create new Decision Table” window, type senior primary drivers for the table name, and click **Create**.

## 2.2. Defining the columns

To create a decision table that implements the inference rule from the BRD, your table must test these conditions for the primary driver:

- The age of the driver
- The total number of accidents
- The total number of accidents in the past five years where the driver was at fault

Based on a combination of these factors, there is a single rule action to set the request status to the outcome of this rule evaluation.

In this case, the rule is only evaluated when the age of a primary driver is in the “senior” range and had an accident within the past five years. If the primary driver did not have an accident, this rule does not come into play, which means the rule has only two outcomes:

- See an underwriter for manual processing
- Reject



### Questions

Consider:

- How many condition columns and action columns do you need in your table?
- How can you label them?

### Answers

- The table has three condition columns, and one action column.
- The column labels should match the content of the rule statements that are represented in the column.

- \_\_\_ 1. Label your condition and action columns by overwriting the default column headings with the following names.
  - Condition column **C0**: Driver age
  - Condition column **C1**: Number of accidents
  - Condition column **C2**: Number of at-fault accidents

- Action column **A0**: Result

Before you worry about cell values, first think what the condition and action statements should be for each column.



## Questions

What should the rule look like for the first row?



## Reminder

To open the expression editor for a decision table column, right-click the column header and click **Define column**.



## Important

Save your work frequently when you work in the Business console editor. If you save frequently, you can avoid losing your work if your Business console session times out unexpectedly.

2. Define the condition columns.

- **Driver age:**

the age (in years) of the primary driver of 'the request' at the start date of 'the policy' is more than <min> and at most <max>

- **Number of accidents:**

the number accidents of the primary driver of 'the request' equals <a number>



## Note

Unlike the phrase "number of at-fault accidents in the last five years," the vocabulary does not include "in the last five years" as part of the vocabulary phrase "number of accidents."

- **Number of at-fault accidents:**

the number of at-fault accidents in the last five years of the primary driver of 'the request' equals <a number>

3. Define the action column:

set response status to <a status type> and add message <enter a string>

	Driver age		Number of accidents	Number of at-fault accidents	Result	
	min	max			a status type	a string
1					Accepted	
2					Process manually	
3					Refused	
4					Undefined	

This action statement creates two columns as placeholders. The first **Result** subcolumn uses a predefined domain of values.

You can change the headings for these subcolumns. For example, you can change the **a status type** subcolumn heading to **Status** and the **a string** subcolumn heading to **Message**.

4. Complete the cell values for the first row so that you can see the full rule.
  - a. Enter these cell values for the first row: 70, 80, 3, 1, Process manually, Borderline
  - b. Hover the mouse pointer over the row 1 header to see the complete rule, and compare the current rule to the business policy.

The current rule reads:

```

if
  all of the following conditions are true:
    - ( the age (in years) of the primary driver of 'the request' at the
      start date of 'the policy' is more than 70 and at most 80 )
    - ( the number accidents of the primary driver of 'the request' equals
      3 )
    - ( the number of at-fault accidents in the last five years of the
      primary driver of 'the request' equals 1 )

then
  set the response status to Process manually and add message
  "Borderline";

```

	Driver age		Number of accident ▾	Number of at-fault ▾	Result	
	min	max			Status	Message
1	70	80	3	1	Process manually	Borderline

**if**  
all of the following conditions are true :  
 - (the age (in years) of the primary driver of "the request" at the start date of "the policy" is more than 70 and at most 80 )  
 - (the number accidents of the primary driver of "the request" equals 3 )  
 - (the number of at-fault accidents in the last five years of the primary driver of "the request" equals 1 ),  
**then**  
set response status to Process manually and add message "Borderline" ;

This initial look at the rule can help you identify how to modify the BAL constructs for each row.



## Questions

How must you change the BAL constructs for the number of accidents and at-fault accidents?

---



---

## Answer

Recall the values in the rule policy.

Driver age	Number accidents in last 5 years	Number at-fault accidents in last 5 years	Result
] 70, 80 ]	=< 3	=< 1	Process manually
		> 1	Refused
	> 3		Refused
[ 65, 70 ]	> 3		Refused

---

## 2.3. Completing the table

To complete your table, first enter the cell values for each row. You can then change the BAL number operators for each cell.

- 1. For row 1, you must change the operators in some of the conditions to match the rule policy.
  - a. Review the operators in the **Driver age** column.

The **Driver age** column operators are **is more than <min>** and **is at most <max>**, which are also represented through the symbols **]..]**. Because these operators match the operators in the rule policy, you do not need to change them.

- \_\_\_ b. In row 1, change the number operator in the **Number of accidents** column by right-clicking the cell, clicking **Change operator**, and clicking the **is at most** operator (**=<**).
- \_\_\_ c. In the row 1 cell for the **Number at-fault accidents** column, right-click the cell to open the menu, click **Change operator**, and click the **is at most** operator (**=<**).
- \_\_\_ 2. Enter the values for row 2, and update the operators to match the rule policy.
  - \_\_\_ a. Enter the following values in the cells for row 2:  
70, 80, 3, 1, Refused, Risk too high
  - \_\_\_ b. Use the cell menu to change the operator in the **Number of accidents** column to **is at most** (**=<**).
  - \_\_\_ c. Use the cell menu to change the operator in the **Number at-fault accidents** column to **is more than** (**>**).
- \_\_\_ 3. Enter the values for row 3, and update the operators.
  - \_\_\_ a. In row 3, enter the following values:  
70, 80, 3, **<no value>**, Refused, Risk too high
  - \_\_\_ b. Change the number operator in the **Number of accidents** column to **>** (is more than).
  - \_\_\_ c. Leave the **Number at-fault accidents** column empty. You do not need to change the condition statement.
- \_\_\_ 4. In the cells for row 4, enter the following values:  
65, 70, 3, **<no value>**, Refused, Risk too high



### Note

While the **Driver age** column for row 4 can be edited by changing the operator, as you did in the previous steps, you can also use the **Edit custom value** option to edit the condition statement.

The following steps walk you through the process of using the **Edit custom value** option to change a condition statement. This option is available in the cell menu for a column.

- 
- \_\_\_ 5. In the **Driver age** cell for row 4, use the **Edit custom value** option to update the condition.

**Reminder**

You can use the **Edit custom value** option to create a custom condition or action statement for a decision table column. This option displays the rule statement that was defined for the column in a rule editor window. You can either type directly in the rule editor or use the completion menu to build the custom rule statement.

- \_\_\_ a. Right-click the **Driver age** cell for row 4 and click **Edit custom value**.
- \_\_\_ b. In the Edit Custom Value window rule editor, double-click the **is more than** statement to open the rule editor.
- \_\_\_ c. In the rule editor, double-click the **is more than** phrase.

**Edit Custom Value**

Use this editor to write an expression for the cell.

the age (in years) of the primary driver of '**the request**' at the start date of '**the policy**' **is more than 65 and at most 70**

The completion menu opens.

- \_\_\_ d. In the completion menu, click **is between <min>**.

is more than 65 and at most 70

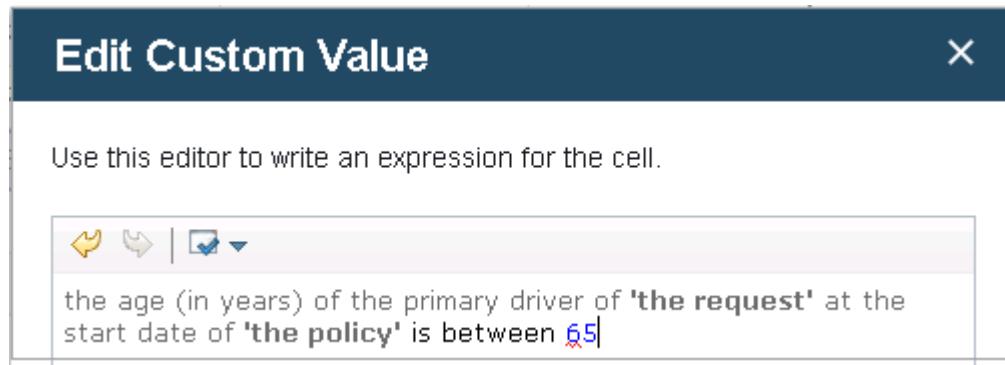
① does not equal <a number>
② equals <a number>
③ is <an object>
④ is at least <a number>
⑤ is at least <min>
⑥ is at most <a number>
⑦ is between <min>
⑧ is less than <a number>
⑨ is more than <a number>

- \_\_\_ e. In the rule editor, press the Space bar and in the menu, click **<number>**.

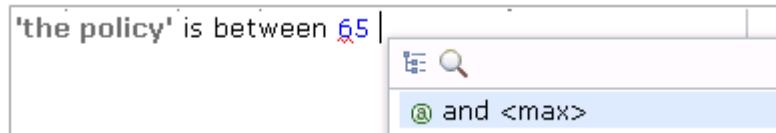
'**the policy**' is between |

① 'Max vehicle accessories ratio'
② 'tax rate percentage'
▲ <number>
③ @ the accessories value of <a vehicle>
④ @ the age ( in years ) of <a driver>

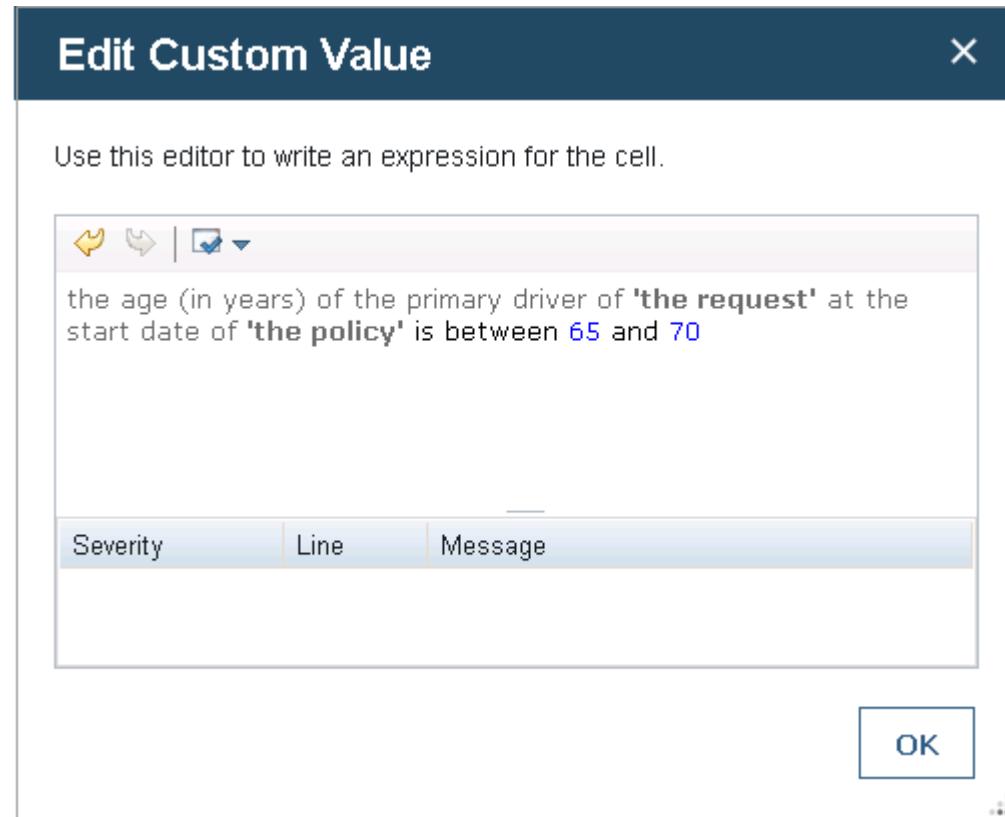
- \_\_ f. In the rule editor, replace the 0 with: 65



- \_\_ g. Press the Space bar, and in the menu, click **and <max>**.



- \_\_ h. In the editor, type 70 and click **OK** to save your changes.



- \_\_ i. In the table, you see that the operators for row 4 changed to **is between <min> and <max>**, which is represented by an open bracket and a closed bracket ([ ]).
- \_\_ 6. Complete the remaining condition columns for row 4.
- \_\_ a. By using the **Change operator** option, change the number operator of the **Number of accidents** column to **>** (is more than).

- \_\_ b. Leave the **Number at-fault accidents** column empty. You do not have to change the condition statement.

The rule for row 4 now reads:

if all of the following conditions are true:

- (the age (in years) of the primary driver of 'the request' at the start date of 'the policy' is between 65 and 70)
- (the number accidents of the primary driver of 'the request' is more than 3),

then

set response status to Refused and add message "Risk too high";

4	[65 70]	> 3		Refused	Risk too high
<b>if</b> <b>all of the following conditions are true :</b> - (the age (in years) of the primary driver of <b>'the request'</b> at the start date of <b>'the policy'</b> is between <b>65</b> and <b>70</b> ) - (the number accidents of the primary driver of <b>'the request'</b> is more than <b>3</b> ), <b>then</b> set response status to <b>Refused</b> and add message " <b>Risk too high</b> ";					

- \_\_ 7. Click **Save** and then **Create New Version**.

After you save the table, you might notice that the row order changed. Row 4 moved to row 1, so that the age range 65 - 70 is listed before the age range 70 - 80.

The Business console changed the row order because the decision table row ordering property is set to **Automatic**. This option automatically reorders the rows to optimize them.

	Driver age		Number of	Number of at-fa	Result	
	min	max			Status	Message
1	[65	70]	> 3		Refused	Risk too high
2	70	80	≤ 3	≤ 1	Process manually	Borderline
3	70	80	≤ 3	> 1	Refused	Risk too high
4	70	80	> 3		Refused	Risk too high



## Information

You can set decision table properties, such as row ordering and gap or overlap checking, in the Details window for the decision table.

To open the Details window, open the decision table in the editor and click the **Details** icon.



On the **Properties** tab, you can select or change various decision table properties and options.

### Senior primary drivers

- Properties**
- Stream**

Updated by rtsUser1  
May 11, 2017

Click to add a description.

Name	Senior primary drivers
Folder	/Check Eligibility/Primary Driver
Status	New
Categories	Any
Group	
Effective Date	m/d/yyyy
Expiration Date	m/d/yyyy

► Row Ordering

<input checked="" type="checkbox"/> Check gaps	<input checked="" type="checkbox"/> Auto-resize columns
<input checked="" type="checkbox"/> Check overlaps	<input type="checkbox"/> Use check boxes for Boolean values

## End of exercise

## Section 3. Solutions

### Section 2, "Editing conditions in a decision table"

#### 3.1. Solution for "Creating the “Senior primary drivers” decision table"

Each row is a rule, so the solution rule statements that are outlined below the table are for each row.

	Driver age		Number of accidents	Number of at-fault accidents	Result	
	min	max			Status	Message
1	[65	70]	> 3		Refused	Risk too high
2	70	80	$\leq 3$	$\leq 1$	Process manually	Borderline
3	70	80		$> 1$	Refused	
4	70	80	$> 3$		Refused	Risk too high

#### Row 1

```

if all of the following conditions are true :
  - ( the age (in years) of the primary driver of 'the request' at the start
    date of 'the policy' is between 65 and 70 )
  - ( the number accidents of the primary driver of 'the request' is more than
    3 ) ,
then
  set response status to Refused and add message "Risk too high" ;

```

#### Row 2

```

if all of the following conditions are true :
  - ( the age (in years) of the primary driver of 'the request' at the start
    date of 'the policy' is more than 70 and at most 80 )
  - ( the number accidents of the primary driver of 'the request' is at most 3
    )
  - ( the number of at-fault accidents in the last five years of the primary
    driver of 'the request' is at most 1 ) ,
then
  set response status to Process manually and add message "Borderline" ;

```

**Row 3**

```

if all of the following conditions are true :
  - ( the age (in years) of the primary driver of 'the request' at the start
    date of 'the policy' is more than 70 and at most 80 )
  - ( the number accidents of the primary driver of 'the request' is at most 3
)
  - ( the number of at-fault accidents in the last five years of the primary
    driver of 'the request' is more than 1 ) ,
then
  set response status to Refused and add message "Risk too high" ;

```

**Row 4**

```

if all of the following conditions are true :
  - ( the age (in years) of the primary driver of 'the request' at the start
    date of 'the policy' is more than 70 and at most 80 )
  - ( the number accidents of the primary driver of 'the request' is more than
    3 ) ,
then
  set response status to Refused and add message "Risk too high" ;

```

## Exercise review and wrap-up

This exercise showed you how to work with decision tables.

# Exercise 14. Authoring rules: Putting it all together

## Estimated time

01:45

## Overview

This exercise continues the focus on rule authoring. You practice the rule authoring skills that you learned in the previous units and exercises.

## Objectives

After completing this exercise, you should be able to:

- Use the rule editors to transform business policy into formal rule statements

## Introduction

### **Scenario:**

More business policies that were captured during the discovery phase must be implemented. Some of these rules involve vehicle eligibility, while other rules are for driver eligibility. Your task is to review the business policies from the BRD and implement them as rules.

This exercise is based on the car insurance scenario, and includes these sections:

- [Section 1, "Implementing the mandatory constraints about the maximum insurable value of vehicles"](#)
- [Section 2, "Implementing maximum collision liability amounts"](#)
- [Section 3, "Implementing a decision table to infer the risk for young primary drivers"](#)
- [Section 4, "Solutions"](#)

## Requirements

This exercise requires that you work in the Decision Center Business console in the computer lab environment.

This exercise provides the business policies and strategies for you to build the rules without step-by-step guidance. Solutions for the rules are provided at the end of this exercise.

To do this exercise, the following exercise must be completed:

- [Exercise 10, "Working with conditions in rules,"](#) on page 10-1
- [Exercise 11, "Working with definitions in rules,"](#) on page 11-1

- [Exercise 12, "Writing complete rules," on page 12-1](#)
- [Exercise 13, "Authoring decision tables," on page 13-1](#)

## Before starting the exercise

### Starting the sample server

Before proceeding, make sure that the sample server is running.

### Working with car insurance rule projects

For this exercise, you can continue working in the `carinsurance-dt-start` decision service.



#### Important

Save your rule authoring work frequently while you work in the Business console rule editor. Saving your rule while it is in progress helps you to avoid losing your work if your Business console session times out.

You can save your work in the Compose editor by clicking **Save** and then **Create New Version**. Return to the rule editor by clicking **Edit**.

## Section 1. Implementing the mandatory constraints about the maximum insurable value of vehicles

### Business policy

The Business Rule Document (BRD) includes a mandatory constraint that excludes luxury vehicles from coverage. See the original policy for "[Criteria for the vehicle](#)" on page 1-13 in the case study.

The original business policy was captured in a business rule template:

#### Rule 4.3.3: Check vehicle category and value

<b>Rule name</b>	cons_vehicle_category_and_value	<b>Rule ID</b>	R_4_3_3
<b>Classification</b>	constraint	<b>Creation date</b>	<date>
<b>References</b>	Check Eligibility (UC 4), Vehicle Eligibility (Step 3)		
<b>Decisions</b>			
<b>Business rule</b>	(The vehicle category <b>MUST NOT BE IN LIST</b> {High-end luxury, Ultra luxury}) AND (The vehicle base value <b>MUST NOT BE GREATER THAN</b> 120,000)		
<b>Policy implemented</b>			
<b>Business motivation</b>			
<b>Objects affected</b>	Vehicle		
<b>Changes</b>			
<b>Who can change it</b>			
<b>How</b>			
<b>When</b>			

### Strategy

Determine whether this policy should be implemented in one rule or in two separate rules by reviewing this statement:

Any vehicle that is valued at more than 120,000



## Questions

Which of the following items defines this value?

- Base value of the vehicle
- Total insured value of the vehicle
- Base value plus the value of vehicle accessories

## Answer

Based on discussion with the subject matter expert, you determine that the vehicle value for this context includes only the base value.

The decision is made to implement the luxury constraints as two separate business rules:

- A rule to specify the vehicle classification according to a predefined list of vehicle classifications: Luxury
- A rule to specify the maximum insurable value: Base value

The new rules are captured in these business rule templates.

### Rule 4.3.3a: Check vehicle category

<b>Rule name</b>	cons_vehicle_category	<b>Rule ID</b>	R_4_3_3a
<b>Classification</b>	constraint	<b>Creation date</b>	<date>
<b>References</b>	Check Eligibility (UC 4), Vehicle Eligibility (Step 3)		
<b>Decisions</b>			
<b>Business rule</b>	The vehicle category <b>MUST NOT BE IN LIST</b> {High-end luxury, Ultra luxury}		
<b>Policy implemented</b>			
<b>Business motivation</b>			
<b>Objects affected</b>	Vehicle		
<b>Changes</b>			
<b>Who can change it</b>			
<b>How</b>			
<b>When</b>			

### Rule 4.3.3b: Check vehicle value

<b>Rule name</b>	cons_vehicle_value	<b>Rule ID</b>	R_4_3_3b
<b>Classification</b>	constraint	<b>Creation date</b>	<date>
<b>References</b>	Check Eligibility (UC 4), Vehicle Eligibility (Step 3)		
<b>Decisions</b>			
<b>Business rule</b>	The vehicle base value <b>MUST NOT BE GREATER THAN</b> 120,000		
<b>Policy implemented</b>			
<b>Business motivation</b>			
<b>Objects affected</b>	Vehicle		
<b>Changes</b>			
<b>Who can change it</b>			
<b>How</b>			
<b>When</b>			

## Task

- \_\_\_ 1. Continue working in the same decision service that you used for the previous exercise.
- \_\_\_ 2. Create two rules to implement the business policies:
  - \_\_\_ a. Luxury
  - \_\_\_ b. Base value
- \_\_\_ 3. Add your rules to the rule folder: Check Eligibility/Vehicle
- \_\_\_ 4. Check your work against the solution to "[Section 1, "Implementing the mandatory constraints about the maximum insurable value of vehicles""](#) on page 14-15.



### Reminder

To include multiple items in a domain:

- \_\_\_ 1. In the completion menu, click the **is one of <objects>** construct.
- \_\_\_ 2. Click { **<an object>** }.
- \_\_\_ 3. Click the first item in the list.
- \_\_\_ 4. To add additional items to the list:
  - \_\_\_ a. Press the Space bar to open the completion menu.
  - \_\_\_ b. Click , **<an object>**.
  - \_\_\_ c. In the menu, click the item that you need.

## Section 2. Implementing maximum collision liability amounts

### Business policy

The Business Rule Document (BRD) includes an inference rule that determines the collision maximum amount of collision coverage that can be offered for a particular vehicle. See the original policy for "[Maximum liability](#)" on page 1-12 in the case study.

The business policy generates two inference rules: **Rule 4.2.3** and **Rule 4.2.4**.

#### Rule 4.2.3: Check collision policy liability amount

<b>Rule name</b>	infer_collision_proposed_amount_range	<b>Rule ID</b>	R_4_2_2
<b>Classification</b>	inference	<b>Creation date</b>	<date>
<b>References</b>	Check Eligibility (UC 4), Policy Eligibility (Step 2)		
<b>Decisions</b>	Check the collision policy liability amount and adjust if necessary		
<b>Business rule</b>	<b>IF</b> the collision policy liability amount requested IS IN RANGE [2,500 - 125,000] <b>THEN</b> the collision policy liability amount proposed = the collision policy liability amount requested <b>ELSE</b> the collision policy liability amount proposed = NEAREST-BOUND-TO([2,500 - 125,000], the collision policy liability amount requested)		
<b>Policy implemented</b>			
<b>Business motivation</b>			
<b>Objects affected</b>	Policy, Vehicle		
<b>Changes</b>			
<b>Who can change it</b>			
<b>How</b>			
<b>When</b>			



#### Note

The statement NEAREST-BOUND-TO (*range, value*) corresponds to the bound value in range that is closest to *value*. For example:

NEAREST-BOUND-TO ([4, 10], 18)

The result of this expression is 10 because 18 is closer to 10 than it is to 4.

---

#### Rule 4.2.4: Check collision policy liability amount and vehicle value

<b>Rule name</b>	infer_collision_proposed_amount_vehicle_total_value	<b>Rule ID</b>	R_4_2_2
<b>Classification</b>	inference	<b>Creation date</b>	<date>
<b>References</b>	Check Eligibility (UC 4), Policy Eligibility (Step 2)		
<b>Decisions</b>	Check the collision policy liability amount and adjust if necessary		
<b>Business rule</b>	<b>IF</b> the collision policy liability amount proposed IS MORE THAN the vehicle total insured value <b>THEN</b> the collision policy liability amount proposed = the vehicle total insured value		
<b>Policy implemented</b>			
<b>Business motivation</b>			
<b>Objects affected</b>	Policy, Vehicle		
<b>Changes</b>			
<b>Who can change it</b>			
<b>How</b>			
<b>When</b>			

## Strategy

- First, consider **Rule 4.2.4**. This rule is implemented as the rule: collision cover greater than total insured value
 

You can find this rule in the Check Eligibility/Policy rule package.
  - Next, consider the rule statement for **Rule 4.2.3** and how you might implement the **else** statement.
- 



### Questions

Rules with **else** are more difficult to maintain. How can you rewrite this rule to remove the **else** part?

---



---

### Answer

Consider writing the **else** statement as separate rules.

In this case, **Rule 4.2.3** can then be broken into three atomic rules.

---



## Questions

If **Rule 4.2.3** can be written as three atomic rules, what are they and how should they be formulated?

### Task 1

Write the formal rule statement for the new rules:

- **Rule 4.2.3a**

if

then

- **Rule 4.2.3b**

if

then

- **Rule 4.2.3c**

if

then

### Task 2

You already wrote one of these rules during [Exercise 11, "Working with definitions in rules"](#) (["Creating the "Collision cover in \[min, max\]" rule"](#) on page 11-4). This rule represents the main action statement.

You now implement the remaining two rules, which represent the rewritten **else** statement.

- 1. Continue working in the same decision service that you used for the previous exercise.
- 2. Create these rules:
  - a. Collision cover in min[
  - b. Collision cover in ]max
- 3. Add the new rules to the rule folder: Check Eligibility/Policy

- 4. Compare your rules to the solution to ["Section 2, "Implementing maximum collision liability amounts""](#) on page 14-15.
- 



### Hint

What definition can you create for this rule to assist in rule authoring?

---



### Reminder

To start a definitions section and to create a local variable:

- In the rule editor, put the cursor before the **if** statement and press Enter to create a new line.
  - Place the cursor in the new line, type **definitions**, and press Enter.
  - Press the Space bar, and select **set <variable1> to <a binding type>**.
  - Overwrite **variable1** with the variable name, and click **<a binding type>** to select an item from the completion menu.
  - If needed, continue to use the completion menu to complete the binding type.
-

## Section 3. Implementing a decision table to infer the risk for young primary drivers

### Business policy

Based on the BRD, create a decision table that evaluates the risk for primary drivers who are 25 years of age or younger.

This rule template outlines the guidelines and factors that are involved in determining the outcome for an insurance request from a young driver.

#### Rule 4.5.1: Infer young primary driver risk

<b>Rule name</b>	infer_young_primary_driver_risk	<b>Rule ID</b>	R_4_3_3b
<b>Classification</b>	constraint	<b>Creation date</b>	<date>
<b>References</b>	Check Eligibility (UC 4), Vehicle Eligibility (Step 3)		
<b>Decisions</b>			

<b>Business rule</b>	<b>Driver age</b>	<b>Car power</b>	<b>Number at-fault accidents in last five years</b>	<b>License age in years</b>	<b>Number license withdrawals</b>	<b>Result</b>
	=< 21	Class D	=< 1		= 0 => 1	P. Manually Refused
		Class E	=< 1			Refused
			> 1			Refused
	[21, 25 [	{ Class D, Class E }		=< 3	= 0 => 1	P. Manually Refused

<b>Policy implemented</b>	
<b>Business motivation</b>	
<b>Objects affected</b>	Driver
<b>Changes</b>	
<b>Who can change it</b>	
<b>How</b>	
<b>When</b>	

## Strategy

Look at the rule template and count the number of conditions and actions to determine the number of columns that you need in your table.

Count the number of rows of values in the table to know how many rows to include in your table. Some cells are empty, which indicates that the condition column does not apply to the rule for that row.

## Task

- \_\_\_ 1. Continue working in the same decision service that you used for the previous exercise.
  - \_\_\_ 2. Create a decision table named: Young primary drivers
  - \_\_\_ 3. Add the table to the rule folder: Check Eligibility/Primary driver
  - \_\_\_ 4. After you finish, check your work against the ["Solution for ‘Young Primary Driver’ table"](#) on page 14-16.
- 



### Hint

Remember these tips as you work with the table.

#### Working with condition and action columns

- To define the condition or action for a column, right-click the column and click **Define condition** or **Define action**.
- To insert a condition column:
  - Right-click a condition column
  - Click **Insert condition**
  - Click either **Condition before** or **Condition after**
- To edit the rule statement in a cell:
  - To change operators: Right-click a cell and click **Change operator**.
  - To create a custom value: Right-click a cell and click **Edit custom value**

#### Working with rule vocabulary

- For the **Driver age** column, use the **Edit custom value** option to create a rule statement that handles an age range.
- The vocabulary list includes these terms:
  - **the horse power of a vehicle**
  - **the license age of a driver in months**
- Use **equals** as the BAL operator to compare number values.
- Use the **is one of** construct to enter a list of values in a cell.

## Completing cells and rows

- When you enter the cell value for license age, remember to use the number of months and not years.
  - When you change conditions for cells, change the conditions for one row before moving on to the next row.
- 
- 



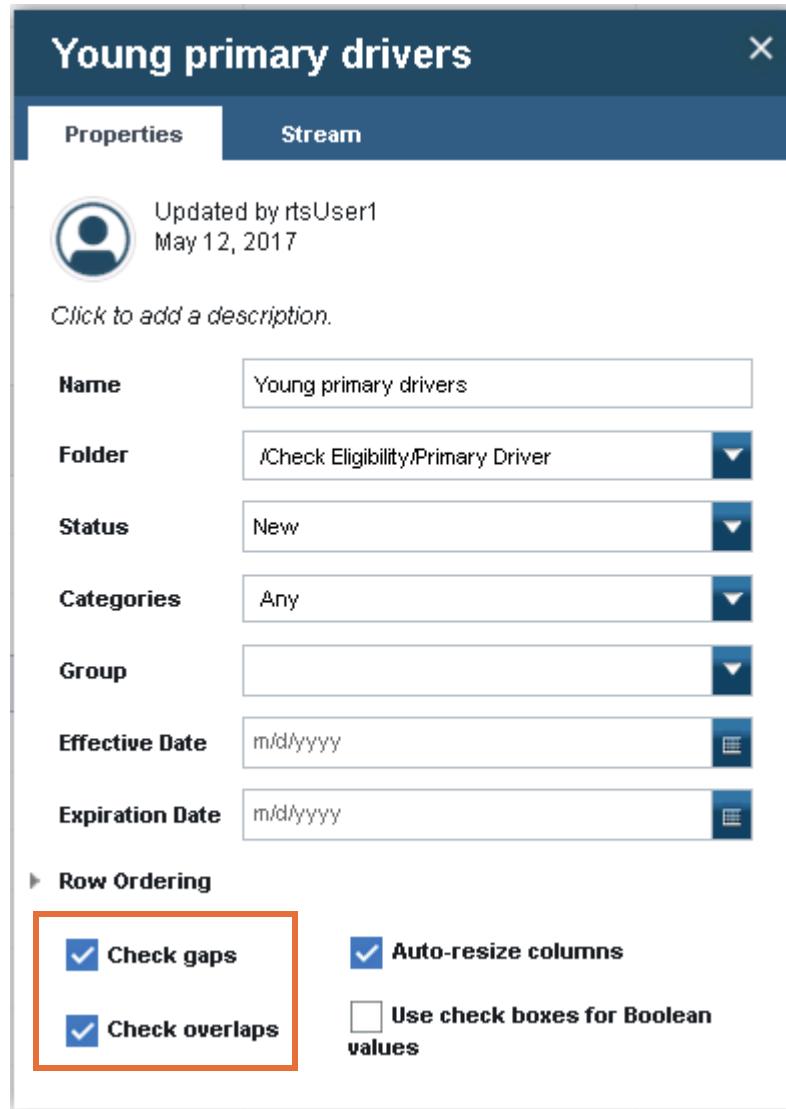
### Reminder

If you have gap errors on cells that are correctly defined, you can disable gap checking in the **Properties** tab of the Details window for the decision table.

You can open the Table Properties editor by clicking the **Details** icon on the decision table editor toolbar.



When the table Properties window is open, you can select or clear the **Check gaps** check box. The **Check gaps** check box is selected by default.



### Important

Save frequently to avoid losing work in the decision table if your Business console session times out.

## End of exercise

## Section 4. Solutions

This section shows the solution rule statements for the previous exercises.

### Section 1, "Implementing the mandatory constraints about the maximum insurable value of vehicles"

#### Rule 4.3.3.a. Solution for the “Luxury” rule

```
if
    the classification of 'the vehicle' is one of { High-end luxury , Ultra luxury
}
then
    refuse the application because "This insurance policy does not cover luxury
vehicles";
```

#### Rule 4.3.3.b. Solution for the “Basic value” rule

```
if
    the basic value of 'the vehicle' is more than 120000
then
    refuse the application because "Vehicle basic value too high" ;
```

### Section 2, "Implementing maximum collision liability amounts"

#### Solution for “Collision cover in min”

```
definitions
    set 'collision policy' to a collision insurance from 'the policy' ;
if
    the collision max amount cover desired of 'collision policy' is less than
2500
then
    set the collision max amount cover proposed of 'collision policy' to 2500;
```

#### Solution for “Collision cover in ]max”

```
definitions
    set 'collision policy' to a collision insurance from 'the policy' ;
if
    the collision max amount cover desired of 'collision policy' is more than
125000
then
    set the collision max amount cover proposed of 'collision policy' to 125000;
```

## Section 3, "Implementing a decision table to infer the risk for young primary drivers"

### Solution for “Young Primary Driver” table

The table includes six rows. Therefore, it has six rules. The rules are stated by row after the table.



#### Reminder

Remember that you can turn off gap checking for this table by opening the Table Properties editor.



	Driver age ▾	Car power ▾	Number at-fault ▾	License age ▾	Number ▾	Result	
						Status	Message
1	< 21	Class D	$\leq 1$		0	Process manually	Borderline
						Refused	Risk too high
3	< 21	Class E	$\leq 1$			Refused	Risk too high
						Refused	Risk too high
5	[21 25[	Class D, Class E		$\leq 36$	0	Process manually	Borderline
						Refused	Risk too high
6	[21 25[	Class D, Class E			$\geq 1$	Refused	Risk too high

**Row 1**

```

if
    all of the following conditions are true :
        - ( the age (in years) of the primary driver of 'the request' at the start
          date of 'the policy' is less than 21 )
        - ( the horse power of 'the vehicle' is Class D )
        - ( the number of at-fault accidents in the last five years of the primary
          driver of 'the request' is at most 1 )
        - ( the number license withdrawals of the primary driver of 'the request'
          equals 0 ) ,

then
    set response status to Process manually and add message "Borderline" ;

```

**Row 2**

```

if
    all of the following conditions are true :
        - ( the age (in years) of the primary driver of 'the request' at the start
          date of 'the policy' is less than 21 )
        - ( the horse power of 'the vehicle' is Class D )
        - ( the number of at-fault accidents in the last five years of the primary
          driver of 'the request' is at most 1 )
        - ( the number license withdrawals of the primary driver of 'the request' is
          at least 1 ) ,

then
    set response status to Refused and add message "Risk too high" ;

```

**Row 3**

```

if
    all of the following conditions are true :
        - ( the age (in years) of the primary driver of 'the request' at the start
          date of 'the policy' is less than 21 )
        - ( the horse power of 'the vehicle' is Class E )
        - ( the number of at-fault accidents in the last five years of the primary
          driver of 'the request' is at most 1 ) ,

then
    set response status to Refused and add message "Risk too high" ;

```

**Row 4**

```

if
  all of the following conditions are true :
    - ( the age (in years) of the primary driver of 'the request' at the start
      date of 'the policy' is less than 21 )
    - ( the number of at-fault accidents in the last five years of the primary
      driver of 'the request' is more than 1 ) ,

then
  set response status to Refused and add message "Risk too high" ;

```

**Row 5**

```

if
  all of the following conditions are true :
    - ( the age (in years) of the primary driver of 'the request' at the start
      date of 'the policy' is at least 21 and less than 25 )
    - ( the horse power of 'the vehicle' is one of { Class D , Class E } )
    - ( the license age in months of the primary driver of 'the request' is at
      most 36 )
    - ( the number license withdrawals of the primary driver of 'the request'
      equals 0 ) ,

then
  set response status to Process manually and add message "Borderline" ;

```

**Row 6**

```

if
  all of the following conditions are true :
    - ( the age (in years) of the primary driver of 'the request' at the start
      date of 'the policy' is at least 21 and less than 25 )
    - ( the horse power of 'the vehicle' is one of { Class D , Class E } )
    - ( the number license withdrawals of the primary driver of 'the request' is
      at least 1 ) ,

then
  set response status to Refused and add message "Risk too high" ;

```

## Exercise review and wrap-up

This exercise looked at how to transform business policies from a design document into formal rule statements.

---

# Exercise 15. Running tests and simulations in the Business console

## Estimated time

00:45

## Overview

This exercise and the following exercises focus on various Decision Center tools for rule and decision service management. In this exercise, you learn how to run tests and simulations in the Business console.

## Objectives

After completing this exercise, you should be able to:

- Test rules by using the Microsoft Excel format to store scenarios and their expected results
- Run simulations based on scenario data

## Introduction

This exercise provides you with hands-on experience with running tests and simulations, and working with scenario files in the Business console.

In this exercise, you run some test suites, and learn how to read the resulting reports. This exercise includes the following sections:

- [Section 1, "Preparing the lab environment"](#)
- [Section 2, "Running tests on the carinsurance-rules decision service"](#)
- [Section 3, "Running a simulation on the car insurance rules"](#)

## Requirements

This exercise requires some lab environment setup, including publishing the decision service to Decision Center.

This exercise uses Rule Designer and the Business console. All testing and simulation tasks are done in the Business console.

Populated Excel scenario files are provided for this exercise.

## Section 1. Preparing the lab environment

To work on this exercise, you must first set up the lab environment, which includes resetting the Decision Center database.

- \_\_\_ 1. *Optional.* Save the database in case you want to return to your previous work.
  - \_\_\_ a. Stop the sample server by going to **Start > Stop Sample Server**.
  - \_\_\_ b. Go to the following directory:  
C:\IBM\wlp\usr\servers\odm8900\data\h2
  - \_\_\_ c. Right-click the **rtsdb.h2.db** file and click **Copy**.
  - \_\_\_ d. Right-click the white space of the directory, and click **Paste**.
  - \_\_\_ e. Right-click the **rtsdb.h2 - Copy.db** file that you created, and click **Rename**.
  - \_\_\_ f. Rename the file so that you can identify it as the backup database file, such as:  
**rtsdb.h2-work.db**
  - \_\_\_ g. Restart the sample server (by going to **Start > Start Sample Server**).



### Information

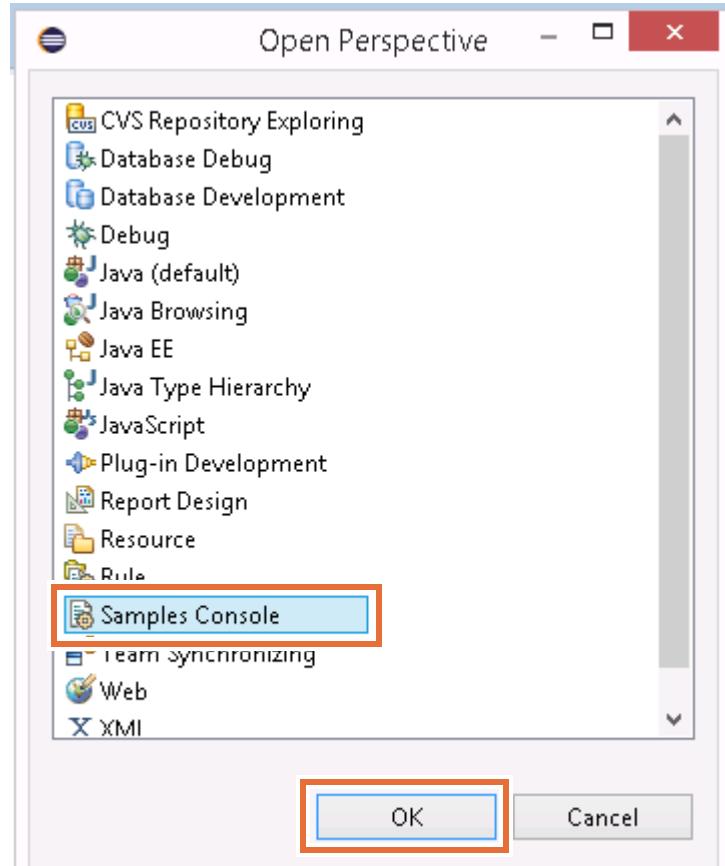
If you want to return to your previous work after you complete this exercise:

- Make sure that you stop the sample server.
- *Optional.* Save the database file for this exercise by making a copy and renaming it so that you can recognize it.
- Rename the backup file that you created in Step 1 to:  
**rtsdb.h2.db**
- Restart the server.

- \_\_\_ 2. Open a new Rule Designer workspace.
  - \_\_\_ a. If needed, start Rule Designer by going to **Start > Rule Designer**.  
If Rule Designer is already running, go to **File > Switch Workspace > Other**.
  - \_\_\_ b. When prompted for a workspace name, enter a name for this exercise, such as:  
C:\labfiles\workspaces\tests
  - \_\_\_ c. After the new workspace opens, close the **Welcome** tab.
- \_\_\_ 3. Open the Samples Console.
  - \_\_\_ a. Click the **Open Perspective** icon.



- \_\_ b. In the Open Perspective window, select **Samples Console**, and then click **OK**.



- \_\_ 4. In the Sample Server pane, click **Restore the sample server**.



The Samples Console switches to the Console view, which shows various messages as the sample server is stopped, restored, and restarted.

- \_\_ 5. Wait until you see the **BUILD SUCCESSFUL** message in the console.

```

Sample Server Console
[password] *****
GBRPS0029I: start.server is completed.
Completed: startserver
BUILD SUCCESSFUL

```

## 1.1. Importing the start project and publishing to Decision Center

- \_\_\_ 1. Import the start project that is provided for this exercise by using the Samples Console.
  - \_\_\_ a. On the **Samples and Tutorials** tab of the Samples Console, expand **Rule Designer > Training > Ex 15: Running tests and simulations in the Business console**.
  - \_\_\_ b. Under **start**, click **Import projects**.
  - \_\_\_ c. When the workspace finishes building, close the **Help** pane by clicking the **X**.
- \_\_\_ 2. Publish the `carinsurance` decision service to Decision Center.
  - \_\_\_ a. In the Rule Explorer pane, right-click **carinsurance-rules** and click **Decision Center > Connect**.
  - \_\_\_ b. In the Decision Center Configuration window, enter the following values:
    - **URL:** `http://localhost:9090/teamserver`  
If your computer lab environment uses a different port, use it here.
    - **User name:** `rtsAdmin`
    - **Password:** `rtsAdmin`
    - You can leave the **Data source** field blank.
  - \_\_\_ c. Click **Connect**.
  - \_\_\_ d. After the connection is established, click **Next**.
  - \_\_\_ e. In the Synchronization Settings window, click **Next**.
  - \_\_\_ f. In the Decision Service Dependent Projects window, click **Finish**.



### Troubleshooting

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

- \_\_\_ 3. Click **OK** when you see the Synchronize Complete window, which shows the following message:  
*Synchronizing: No changes found.*
- \_\_\_ 4. Click **No** when you are prompted to switch to the Team Synchronizing perspective.  
Because you published a decision service that is not in the Decision Center database, you should not have any changes to synchronize.

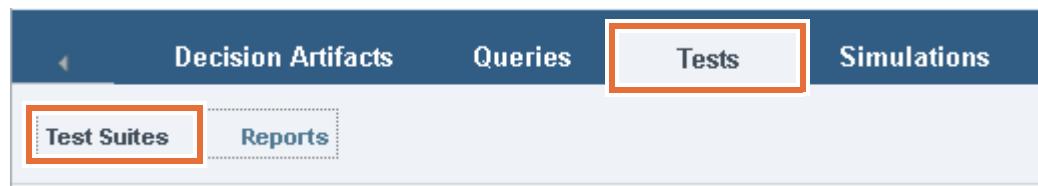
## Section 2. Running tests on the carinsurance-rules decision service

In this section, you create a scenario file to see the kinds of data that is required for a scenario. You also create and run a test suite by using a prepared scenario file that is available in the course lab files.

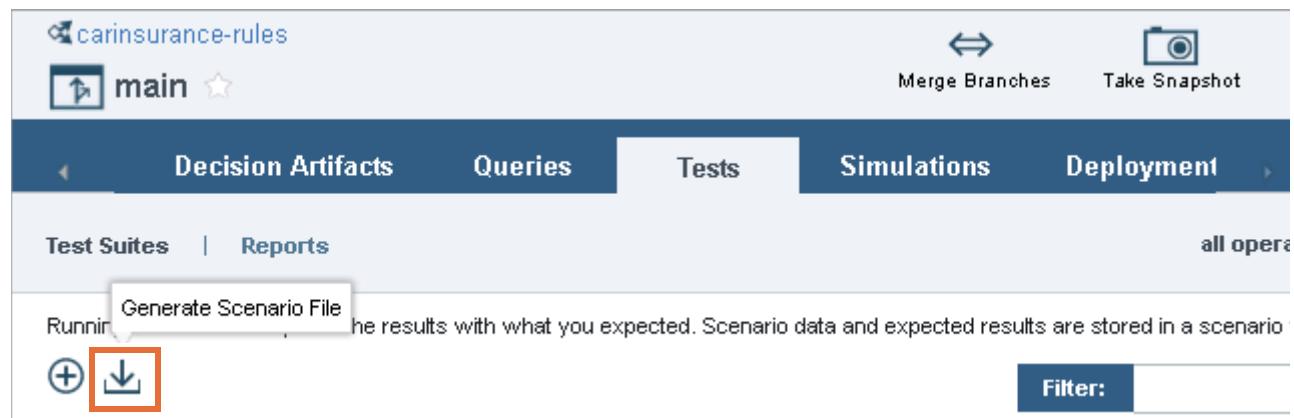
### 2.1. Creating a scenario file

The scenario file that you create is based on the car insurance scenario, and tests for the status of the insurance request (approved, refused, or process manually).

- 1. Open the Business console and sign in as rtsUser1.
- 2. Click the **Library** tab and open the **main** branch of the **carinsurance-rules** decision service by either:
  - Clicking the **carinsurance-rules** link, clicking the **Branches** tab, and then clicking **main**.
  - Or
  - Clicking the white space of the **carinsurance-rules** section to expand it, and then clicking **main**.
- 3. Click the **Tests** tab and make sure that you are on the Test Suites page.



- 4. Generate a scenario file called **Sample Scenario File**.
  - a. Click the **Generate Scenario File** icon (the down arrow).



- b. In the **Filename** field, type: **Sample Scenario File**

- \_\_ c. In the **Tests** section, expand **the response**.

Select the tests to include in the scenario file.

**\* Tests:**

Field	Operator
<input type="checkbox"/> the response	
<input type="checkbox"/> request	
<input type="checkbox"/> collision price before tax	equals
<input type="checkbox"/> messages	equals (unordered)
<input type="checkbox"/> status	equals
<input type="checkbox"/> third party price before tax	equals
<input type="checkbox"/> total price	equals
<input type="checkbox"/> total price before tax	equals



### Information

The **Tests** section is where you specify the kinds of results you want to see in the Expected Results sheet of the scenario file. You can test many aspects of a ruleset, but the Expected Results sheet contains only the tests that you specify when you generate the scenario file. In other words, if you do not select a test, you will not see results for that test.

Here, each option under **the response** represents a test that you can see results for.

---

For this scenario file, you want to see the results for the status of the response (Accepted, Refused, or ProcessManually).

- \_\_\_ d. Select the **status** test.

\* **Tests:**

Field	Operator
<input type="checkbox"/> the response	
<input type="checkbox"/> request	
<input type="checkbox"/> collision price before tax	equals
<input type="checkbox"/> messages	equals (unordered)
<input checked="" type="checkbox"/> status	equals

- \_\_\_ e. In the toolbar, click the **Download** icon.



- \_\_\_ f. In the Opening Scenario File window, select **Save File**, and save the file to a temporary location, such as the **Downloads** folder.

- \_\_\_ 5. Go to the folder that contains your scenario file and view it in Excel Viewer.  
\_\_\_ a. Look at the **Scenarios** sheet and the data that is required for each scenario.

1							
2		Create your scenarios...					
3			Click here to access the help sheet				
4							
5			the request				
6	Scenario ID	description	channel	drivers	policy	vehicle	
9	Scenario 1						
10	Scenarios	driver	third party insurance	option	vehicle	Expected Results	

- The input for the **channel** column cells indicates the source of the request: CarDealer, Broker, or CarOwner.



### Information

You cannot edit the scenario file in Excel Viewer. However, if you open the Sample Scenario File in Excel, double-clicking a **channel** column cell opens a menu that you can use to select the appropriate value.

- The input that you enter for the **drivers** column refers to scenario data that you define in the **driver** sheet.
  - The input that you enter for the **policy** column refers to scenario data that you define in the **third party insurance** sheet.
  - The input that you enter for the **vehicle** column refers to scenario data that you define in the **vehicle** sheet.
- \_\_\_ b. Look at the other tabs (**driver**, **third party insurance**, **option**, **vehicle**) to see the kinds of data that is required for each type of object.
- 



### Reminder

Notice that the data columns match the object attributes in the business object model.

- \_\_\_ c. Look at the **Expected Results** sheet, and notice the included test, **the status of the response equals**.
- 



### Information

You cannot edit `Sample Scenario File.xlsx` in Excel Viewer. However, if you open the scenario file in Excel, double-clicking a cell in **the status of the response equals** column opens a menu. You can use the menu to enter the appropriate value for this column (Accepted, Refused, ProcessManually).

- \_\_\_ d. After you review the various sheets in `Sample Scenario File.xlsx`, close the file.
- 



### Information

A copy of `Sample Scenario File.xlsx` is available in the `<labfilesDir>\scenarios` directory for your reference.

---



### Reminder

In the computer lab environment that is prepared for this course, `<labfilesDir>` is: `C:\labfiles`

- \_\_\_ 6. In Decision Center, click the **main** breadcrumb to return to the main page for the `carinsurance-rules` decision service.
-

## 2.2. Creating and running a test suite

In this section, you work in the `carinsurance-rules` decision service to create a test suite and run a test on the car insurance rules. You use a prepared scenario file to run the test.

- 1. Review the prepared scenario file, which is `eligibility.xlsx`.
  - a. Go to the `<labfilesDir>\scenarios` directory.
  - b. Open the `eligibility.xlsx` scenario file, and review it by using Excel Viewer.



### Information

The simulation spreadsheet consists of various eligibility scenarios that are based on six different drivers.

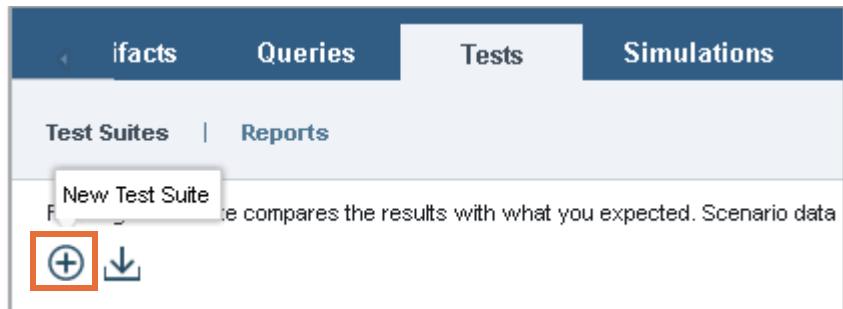
- c. Go to the **Scenarios** page of the spreadsheet, and read through the list of scenarios and their descriptions.



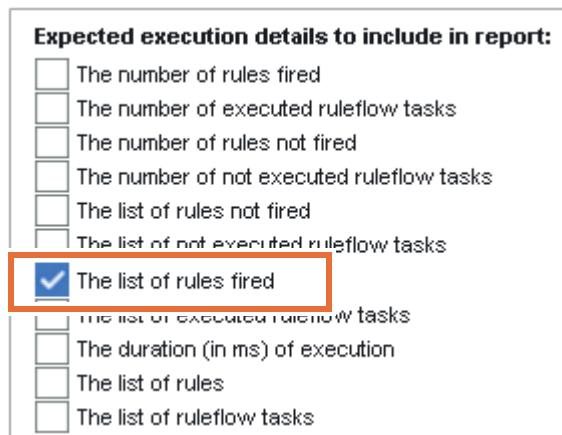
### Questions

Each scenario is designed to test a different rule. Do you see the connection between the values that are listed for each scenario and the tabs in the Excel file?

- d. Open the **driver** page and notice the attributes for each driver. For example, notice the birth dates and number of accidents.
- e. Open the vehicle page and review the attributes of each vehicle, such as the vehicle year and classification.
- f. Read through the listed values on the **Expected Results** page.
- g. When you are finished looking through the `eligibility.xlsx` file, close Excel Viewer.
- 2. Make sure that you are on the **Tests** tab of the **main** branch page for `carinsurance-rules`.
- 3. Create a test suite to check the `Driver` age rule by using the `eligibility.xlsx` file in `<labfilesDir>\scenarios`.
  - a. On the **Tests** tab, make sure that you are on the **Test Suites** subtab.
  - b. Click **New Test Suite** (the plus sign).



- \_\_\_ 4. In the New Test Suite window, in the **Name** field, enter: Eligibility Test Suite
- \_\_\_ 5. Upload the scenario file.
  - \_\_\_ a. Under **Scenarios > File to use**, click **Choose**.
  - \_\_\_ b. Browse to `<labfilesDir>\scenarios` and select `eligibility.xlsx`.
  - \_\_\_ c. Click **Open**.
- \_\_\_ 6. Define the test report.
  - \_\_\_ a. In the **Report name** field under the Report section, enter: Eligibility Report
  - \_\_\_ b. In the “Expected execution details to include in report” section, select **The list of rules fired**.



- \_\_\_ 7. Click **Save and Run**.



- \_\_\_ 8. When prompted, click **Create New Version**.
- \_\_\_ 9. Click **OK** on the Run Test Suite window to close it.

The **Reports** subtab opens on the **Tests** tab.

## 2.3. Viewing the test results

- \_\_\_ 1. When the test suite execution is complete, you should see a check mark in the **Status** column, which means the tests were successful.

- 2. Click the **Eligibility Report** link.

Name	Operation	Status
<input type="checkbox"/> Eligibility Report - 2017-0...	carinsurance-rulesOperati	<input checked="" type="checkbox"/>

The report shows actual results in comparison to the expected results. In this case, the scenarios succeeded.



- 3. In the **Results** section, take some time to explore the test results.
- a. Expand one of the scenarios to view its details. Each scenario has an **Execution Details** section and the results of the test.

The following screen capture shows **Eligibility Scenario 3**.

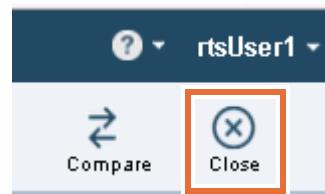
Status	Test	Result
▼ 1 / 1	Eligibility Scenario 3	Eligibility (Age of car is 41)
▶	Execution Details	
	✓ the status of the response equals	Refused
		Refused

- \_\_ b. In the scenario, expand the **Execution Details** section.

Because you selected **The list of rules fired** in the **Expected execution details to include in report** section of the test suite, you see the rules that fired during the test.

<b>Results</b>			
Status	Test	Result	
▼ 1 / 1	Eligibility Scenario 3	Eligibility (Age of car is 41)	
▼	Execution Details		
	The list of rules fired	Check Eligibility.Vehicle.Age Check Eligibility.Policy.Third-party liability in [min, max] Check Eligibility.Vehicle.Total insured value	
<input checked="" type="checkbox"/>	the status of the response equals	Refused	Refus

- \_\_ c. Continue reviewing other scenarios by looking at their execution details and results.  
 \_\_ d. When you are finished looking through the test report results, click **Close** to close the report.



## Section 3. Running a simulation on the car insurance rules

In this section, you set up and run a simple simulation that looks at the age of the driver and how it affects eligibility. You complete the following tasks:

- Define multiple metrics
- Define key performance indicators (KPIs)
- Upload a scenario spreadsheet as a data source
- Define the report format

### 3.1. Defining metrics

Simulation metrics take data that is generated from the rules. In this section, you define the following metrics.

- Driver age
- Eligible driver
- Ineligible driver

- 1. In the carinsurance-rules decision service, click the **Simulations** tab, and make sure that you are on the **Metrics** subtab.

The screenshot shows the top navigation bar with tabs: Decision Artifacts, Queries, Tests, Simulations (which is highlighted with a red box), and Deployment. Below this, a secondary navigation bar has tabs: Metrics (which is highlighted with a dashed red box), KPIs, Data, Report Formats, Simulations, Reports, and a 'all operations' dropdown. A large central area contains the text "Create metrics to define KPIs. Use model elements in metric expressions." with a plus sign icon and a "Filter:" button.

- 2. Create a metric to capture simulation data about driver age.

- a. Click the plus (+) icon.

The screenshot shows the top navigation bar with tabs: Decision Artifacts, Queries, Tests, Simulations, and Deployment. Below this, a secondary navigation bar has tabs: Metrics, KPIs, Data, Report Formats, Simulations, Reports, and a 'all operations' dropdown. A large central area contains the text "Create metrics to define KPIs. Use model elements in metric expressions." with a plus sign icon and a "Filter:" button. The plus sign icon is highlighted with a red box.

- b. On the New Metric page, define the metric name and type:

- **Name:** Driver age
- **Type:** Numeric

- \_\_ c. Enter the following phrase in the **Metric expression** field:

the age (in years) of the primary driver of 'the request' at the start date of the policy of 'the request'



### Note

You can use the completion menu to build the phrase by entering the, pressing the Space bar, and waiting for the completion menu to open in the editor. You can then select the appropriate completion menu options to build the phrase.

- \_\_ d. Click **Save**, and then click **Create New Version**.

carinsurance-rules-tests > main

New Metric

Save Cancel

Define a metric for use in KPI expressions.

**Name:** \* Driver age      **Operation:** \* carinsurance-rulesOperation

**Description:**

**Type:** Numeric      **Default value:**

**Metric expression:**

the age (in years) of the primary driver of 'the request' at the start date of the policy of 'the request'

- \_\_ 3. Create and save the Eligible driver and Ineligible driver metrics.

- \_\_ a. For each metric, click the **Create a metric** icon (plus sign), enter the information from the following table into the form, and save your work.

Name	Type	Metric expression
Eligible driver	Boolean	the status of 'the response' is Accepted
Ineligible driver	Boolean	the status of 'the response' is Refused



### Important

Make sure that you select the correct metric type from the **Type** menu.

## 3.2. Defining key performance indicators (KPIs)

### Defining the driver age KPIs

The KPIs that you define in this section refer to the `Driver age` metric. You define the following KPIs:

- Minimum driver age
- Maximum driver age
- Average driver age

- 1. Create a KPI to display the minimum driver age in a simulation report.
  - a. Click the **KPIs** subtab.
  - b. Click the **plus (+)** icon.
  - c. Enter the following information in the New KPI window:
    - **Name:** `Minimum driver age`
    - **KPI expression:** `minimum 'Driver age'`



#### Reminder

You must enter the name of the metric in the KPI expression exactly as it was defined. The metric names are case-sensitive.

- d. Click **Save**, and then click **Create New Version**.
- 2. Create and save the remaining KPIs about driver age by clicking **Create a KPI** (the plus [+]) icon for each KPI, entering the following information, and saving your work.

*Table 1. Driver age KPIs*

Name	KPI expression
Maximum driver age	<code>maximum 'Driver age'</code>
Average driver age	<code>average 'Driver age'</code>

## Defining the driver eligibility KPIs

In this section, you define the KPIs about driver eligibility for insurance. You create the following KPIs:

- Number of eligible drivers
- Number of ineligible drivers

- 1. Create and save the KPIs for eligible drivers and ineligible drivers by using the following information.

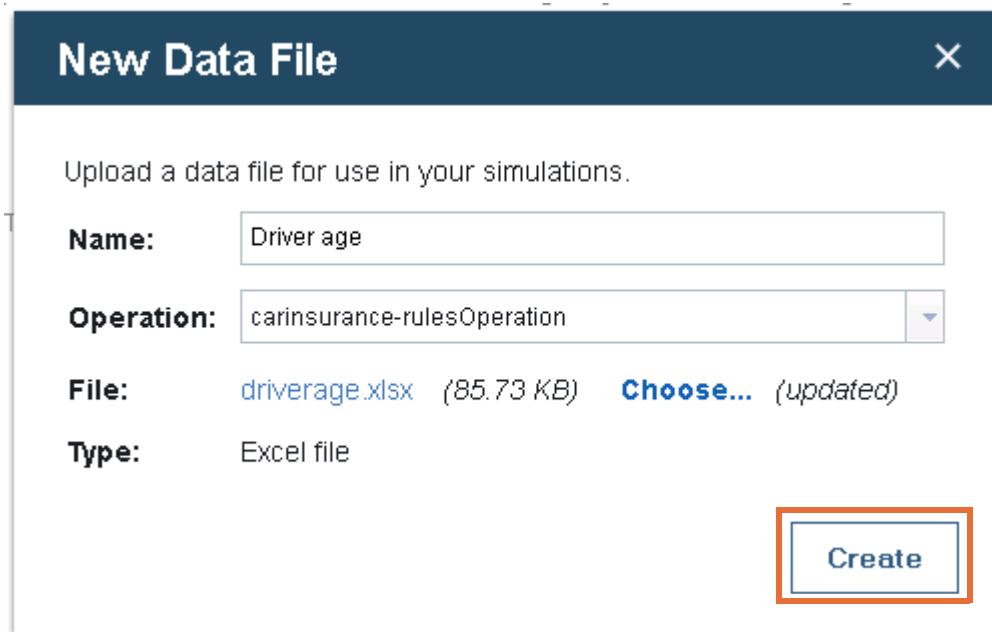
*Table 2. Driver eligibility KPIs*

Name	KPI expression
Number of eligible drivers	number of 'Eligible driver'
Number of ineligible drivers	number of 'Ineligible driver'

## 3.3. Defining the simulation data

In this section, you define the data that is used by the simulation. You use the provided `drivverage.xlsx` Excel scenario file to run this simulation.

- 1. Upload the simulation data by using the `drivverage.xlsx` file.
- a. Click the **Data** subtab.
  - b. Click the **plus (+)** icon.
  - c. In the New Data File window, enter the following information:
    - **Name:** Driver age
    - **Operation:** Keep it at **carinsurance-rulesOperation**
    - **File:** Click **Choose**, go to `<labfilesDir>\scenarios`, and select `drivverage.xlsx`
  - d. Click **Create**.



### 3.4. Defining the report format

In this section, you define the format of the simulation report. You add sections and KPIs to a template to display the simulation data.

- 1. Open the **New Report Formats** page to start defining the template for the simulation report.
  - a. Click the **Report Formats** subtab.
  - b. Click the **plus (+)** icon.
  - c. In the **Name** field, enter **Driver Age and Eligibility**
- 2. Add a section to the report template to display driver age data.
  - a. Click **Add section**.
  - b. In the section that opens, click the **New section** heading to select it, and enter: **Driver Age Information**
- 3. Add KPIs to the **Driver Age Information** section.
  - a. Drag the **Maximum driver age** KPI from the Key Performance Indicators section to the Driver Age Information section that is labeled **Drag and drop a KPI Here**.

The screenshot shows the 'New Report Formats' page. At the top, there's a breadcrumb navigation: carinsurance-rules-tests > main. Below that is a title bar with a mail icon and the text 'New Report Formats'.

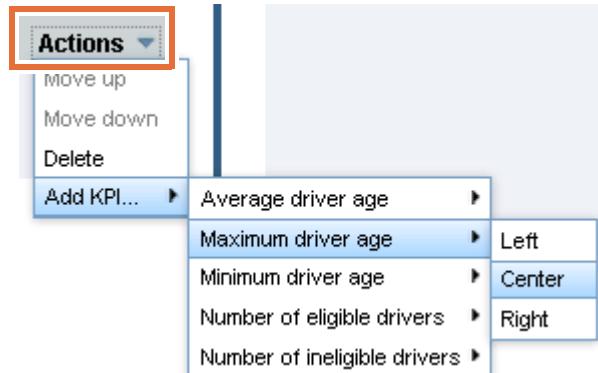
The main area has a header 'Simulation Model | Preview' and a sub-header 'A simulation model is a template where you define how the report will display the re...'. It contains fields for 'Name' (Driver Age and Eligibility) and 'Operation' (carinsurance-r...). There's also a 'Description:' field with an empty text area.

On the left, there's a button '+ Add section' and a section titled 'Driver Age Information' which is currently expanded. Inside this section, the 'Maximum driver age' KPI is highlighted with a red border, indicating it is being moved.

To the right, there's a sidebar titled 'Key Performance Indicators' with a 'Filter:' dropdown and a list of KPIs: Average driver age, Maximum driver age (which is highlighted with a red border), Minimum driver age, Number of eligible drivers, and Number of ineligible drivers. A note says 'Drag KPIs to sections.'

**Note**

Alternatively, you can click **Actions** in the Driver Age Information section header and select **Add KPI > Maximum driver age > Center**.



- \_\_\_ b. Click the **Maximum driver age** KPI in the Driver Age Information section, and click the **KPI Display Configuration** icon to open the KPI Display Configuration window.



- \_\_\_ c. In the **Options** field, replace the text **Maximum driver age** with:  
Age of oldest driver

### KPI Display Configuration

Display Type	Options
Text	<p>Text Template:</p> <p><b>Age of oldest driver</b>: \${value}</p>

**Important**

Do not delete the `:${value}` that follows the **Maximum driver age** text in the **Options** field. This text represents the numeric KPI value from the metric that is displayed in the simulation report.

- \_\_\_ d. Click **OK** to save your change.

**Note**

The metric name **Maximum driver age** does not change in the report template, but when you run the simulation, the report displays the updated heading of **Age of oldest driver**.

- \_\_\_ e. Add the **Minimum driver age** KPI to the Driver Age Information section.
- \_\_\_ f. Open the KPI Display Configuration options, replace the text **Minimum driver age** in the **Options** field with: **Age of youngest driver**

**Reminder**

Do not delete the `:$value` from the KPI display options.

- \_\_\_ g. Click **OK** to save your change.
- \_\_\_ h. Add the **Average driver age** KPI to the Driver Age Information section.
- \_\_\_ 4. Add a section to the report template to display information about request eligibility.
  - \_\_\_ a. Click **Add Section**, click **New section**, and rename the section to:  
Request Eligibility
  - \_\_\_ b. Add the **Number of eligible drivers** KPI to the Request Eligibility section.
  - \_\_\_ c. Add the **Number of ineligible drivers** KPI to the Request Eligibility section.
- \_\_\_ 5. Click **Save** to save the report format, and then click **Create New Version**.

### 3.5. Configuring and running the simulation

- \_\_\_ 1. Click the **Simulations** subtab.

The screenshot shows the top navigation bar with tabs: Queries, Tests, Simulations, Deployments, and Snippets. The 'Simulations' tab is highlighted. Below the navigation bar, there is a secondary navigation bar with tabs: Metrics, KPIs, Data, Report Formats, Simulations (which is highlighted with a red box), and Reports. A dropdown menu labeled 'carinsurance-rulesOperation' is open. The main content area has a subtitle 'Create and run simulation configurations.' and two buttons: a plus sign (+) and a play/pause icon. A 'Filter:' input field is also present.

- \_\_\_ 2. Click the **plus** (+) icon.

The New Simulation page opens. All of the fields are filled with the data that was defined for the simulation, including the decision operation, report format, and data input.

- \_\_\_ 3. Keep all of the default values for the simulation.

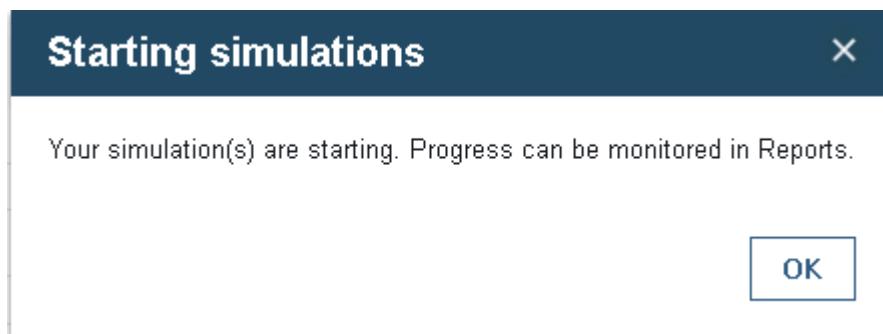
— 4. Save and run the simulation.

— a. Click **Save and Run**.



— b. Click **Create New Version**.

— c. When you see the “Starting simulations” window, click **OK** to close it.



When you close the “Starting simulations” window, the **Simulations** tab opens on the **Reports** subtab.

— 5. Wait for the simulation to complete, and view the simulation report.

— a. Click the **main - carinsurance-rulesOperation** link.

<input type="checkbox"/> Name	Operation	Status
<input type="checkbox"/> <a href="#">main - carinsurance-rulesOperation 2017-07-...</a>	carinsurance-rulesOperation	<input checked="" type="checkbox"/>

— b. Review the simulation results.

carinsurance-rules > main

## main - carinsurance-rulesOperation

**Driver Age Information**

Age of oldest driver: 87

Age of youngest driver: 16

Average driver age: 43

**Request Eligibility**

Number of eligible drivers: 5

Number of ineligible drivers: 1

In the report, you see age of the oldest driver, the age of the youngest driver, and the average driver age.



### Information

Because the `Driver age` rule currently considers drivers between the ages of 16 and 80 to be eligible, only one driver from the scenario data is considered ineligible. In this case, the ineligible driver is the oldest driver, whose age is over 80 years.

- \_\_\_ c. Click **Close** to close the report.



## End of exercise

## Exercise review and wrap-up

In this exercise, you ran tests and simulations in the Business console. You also viewed the results that were returned in the different test and simulation reports.

# Exercise 16. Working with management features in Decision Center

## Estimated time

00:30

## Overview

This exercise continues the focus on rule and decision service management. In this exercise, you explore some of the decision service management features of the Decision Center Enterprise console and Business console.

## Objectives

After completing this exercise, you should be able to:

- Import, export, and delete decision services by using the Enterprise console
- Work with ungoverned decision service branches in the Business console

## Introduction

In this exercise, you work with the `carinsurance-rules` decision service in both the Enterprise console and the Business console. You also see that changes that you make in one console, such as updates to decision service branches, are also available in the other console.

You log in to both Decision Center consoles as the administrative user `rtsAdmin`. The features that you work with in this exercise are available only to users who have administrator permissions.

The exercise includes these sections:

- [Section 1, "Preparing the lab environment"](#)
- [Section 2, "Importing a decision service project by using the Enterprise console"](#)
- [Section 3, "Working with decision service branches in the Business console"](#)
- [Section 4, "Exporting and erasing a decision service in the Enterprise console"](#)

## Requirements

This exercise requires some lab environment setup, including publishing the decision service to Decision Center.

This exercise uses Rule Designer, the Enterprise console, and the Business console. All of the decision service management and branch tasks are done in Decision Center.

## Section 1. Preparing the lab environment

To work on this exercise, you must reset the Decision Center database.

- \_\_\_ 1. *Optional.* Save the database in case you want to return to your previous work.
  - \_\_\_ a. Stop the sample server by going to **Start > Stop Sample Server**.
  - \_\_\_ b. Go to the following directory:  
C:\IBM\wlp\usr\servers\odm8900\data\h2
  - \_\_\_ c. Right-click the **rtsdb.h2.db** file and click **Copy**.
  - \_\_\_ d. Right-click the white space of the directory, and click **Paste**.
  - \_\_\_ e. Right-click the **rtsdb.h2 - Copy.db** file that you created, and click **Rename**.
  - \_\_\_ f. Rename the file so that you can identify it as the backup database file, such as:  
**rtsdb.h2-tests.db**
  - \_\_\_ g. Restart the sample server (by going to **Start > Start Sample Server**).



### Information

If you want to return to your previous work after you complete this exercise:

- Make sure that you stop the sample server.
- *Optional.* Save the database file for this exercise by making a copy and renaming it so that you can recognize it.
- Rename the backup file that you created in Step 1 to:  
**rtsdb.h2.db**
- Restart the server.

- \_\_\_ 2. Open a new Rule Designer workspace.

- \_\_\_ a. If needed, start Rule Designer by going to **Start > Rule Designer**.

If Rule Designer is already running, go to **File > Switch Workspace > Other**.

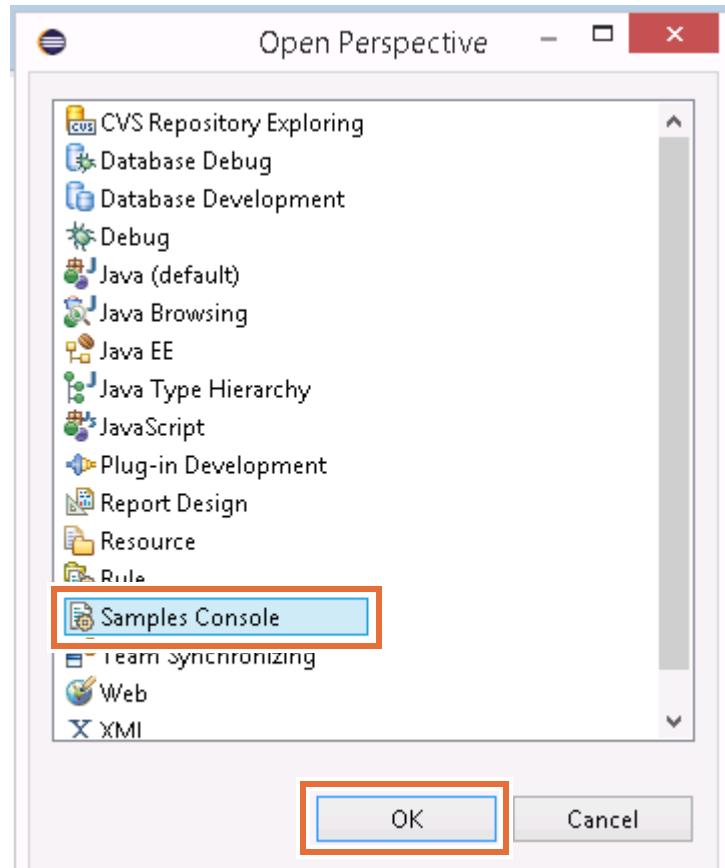
- \_\_\_ b. When prompted for a workspace name, enter a name for this exercise, such as:  
**C:\labfiles\workspaces\admin**
- \_\_\_ c. After the new workspace opens, close the **Welcome** tab.

- \_\_\_ 3. Open the Samples Console.

- \_\_\_ a. Click the **Open Perspective** icon.



- \_\_ b. In the Open Perspective window, select **Samples Console**, and then click **OK**.



- \_\_ 4. In the Sample Server pane, click **Restore the sample server**.



The Samples Console switches to the Console view, which shows various messages as the sample server is stopped, restored, and restarted.

- \_\_ 5. Wait until you see the **BUILD SUCCESSFUL** message in the console.

```

Sample Server Console
[password] *****
GBRPS0029I: start.server is completed.
Completed: startserver
BUILD SUCCESSFUL

```

## Section 2. Importing a decision service project by using the Enterprise console

In this section of the exercise, you learn how to import a decision service into Decision Center through the Enterprise console.

- \_\_\_ 1. Log in to the Enterprise console as `rtsAdmin`.
  - \_\_\_ a. Open the Enterprise console by completing any one of the following steps:
    - From the **Start** menu, clicking the **Decision Center Enterprise console** shortcut.
    - Or
    - Opening a Firefox window and entering the following URL:  
`http://localhost:9090/teamserver`
  - \_\_\_ b. In the **Username** and **Password** fields, enter: `rtsAdmin`
  - \_\_\_ c. Click **Sign In**.
- \_\_\_ 2. In the Enterprise console, click the **Configure** tab.



- \_\_\_ 3. Import the `carinsurance-rules` decision service from the `<labfilesDir>` directory.
  - \_\_\_ a. In the Administration section, click **Import Projects**.

**Administration**

[Installation Settings Wizard](#)  
Modify an existing installation of Decision Center

[Diagnostics](#)  
Run diagnostics to check the Decision Center system

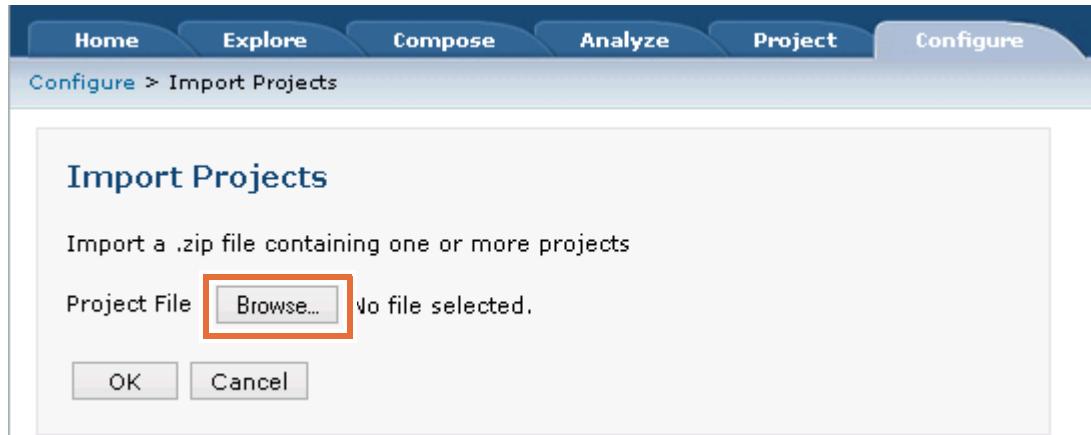
[Clean Decision Center Cache](#)  
Cleans the cache generated by the ruleset generation

[Import Projects](#)  
Import a .zip file containing one or more projects

[Export Current Project State](#)  
Export and download the current project for the selected branch or baseline

[Erase Current Decision Service](#)  
Erase the decision service, its branches, and its history. This operation cannot be undone

- \_\_ b. In the Import Projects page, click **Browse**.



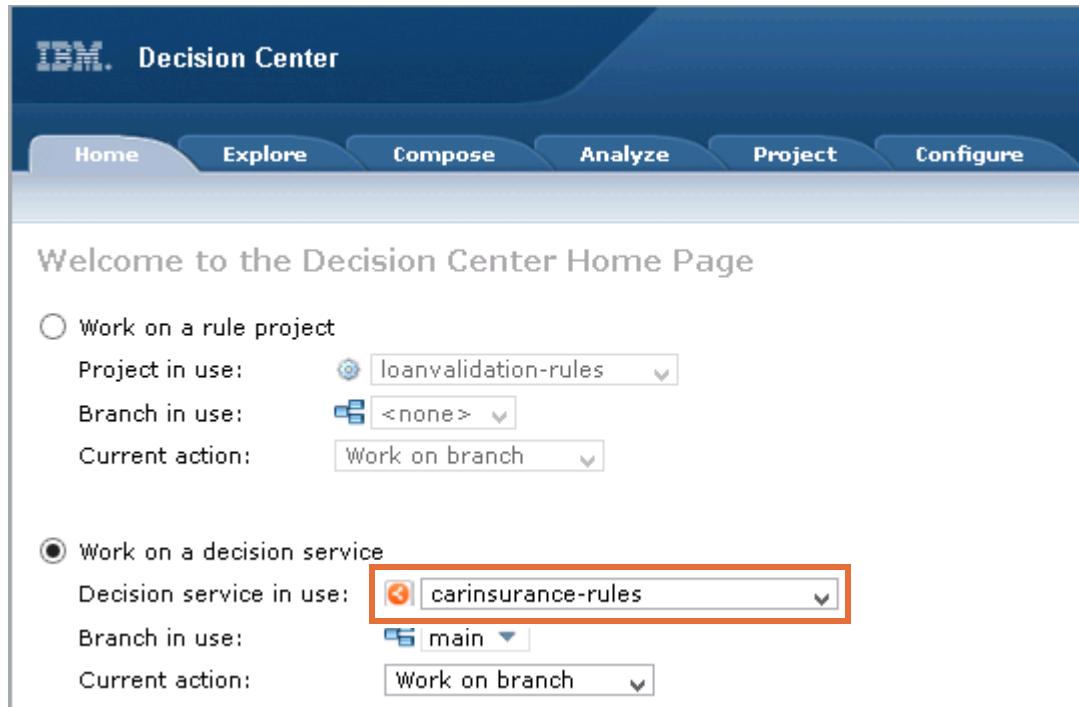
- \_\_ c. In the File Upload window, go to <labfilesDir>/Eclipse-decision services/Car insurance.



### Reminder

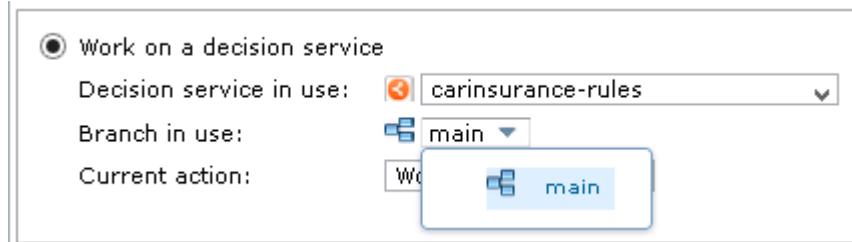
In the computer lab environment that is prepared for this course, <labfilesDir> is C:/labfiles.

- \_\_ d. Select carinsurance-rules.zip and click **Open**.  
 \_\_ e. Click **OK**.
4. After the decision service is imported, it is available on the **Home** tab.



- \_\_\_ 5. Explore the carinsurance-rules decision service.
  - \_\_\_ a. In the **Home** tab, make sure that **Work on a decision service** is selected, and that **carinsurance-rules** is selected from the **Decision service in use** menu.
  - \_\_\_ b. Click **main** from the **Branch in use** menu.

You see that the decision service has only one branch (**main**).



- \_\_\_ c. Click the **Explore** tab.
- \_\_\_ d. In the Smart Folders pane, expand the **Check Eligibility** and **Price Third-Party** folders.

A screenshot of the IBM Enterprise Console interface, specifically the 'Explore' tab. The left side features a 'Smart Folders' sidebar with various categories expanded, such as 'Business Rules', 'Check Eligibility' (which includes 'Drivers', 'Options', 'Policy', 'Primary driver', 'Vehicle', 'Price Collision', 'Price Options'), and 'Price Third-Party' (which includes 'Base Price' and 'Factors'). The main right-hand pane is titled 'Business Rules' and contains a table with columns 'Actions' and 'Name'. A message at the bottom of the pane states 'There are no business rules at this level'. Above the table, there are several actions buttons: 'New', 'Details', 'Edit', 'Delete', 'Copy', and 'Lock'.

- \_\_\_ e. Take a few minutes to explore some of the rules in the decision service. You might recall working on some of these rules in earlier exercises.
- \_\_\_ f. When you are finished exploring the rules, sign out of the Enterprise console, but do not close the Enterprise console window. You use the Enterprise console again later in this exercise.

## Section 3. Working with decision service branches in the Business console

In this part of the exercise, you learn how to work with branches of an ungoverned decision service by using the Decision Center Business console. Branches help you manage rule changes in an ungoverned decision service.



### Information

You work with governed decision services in [Exercise 18, "Working with the decision governance framework"](#).

- \_\_\_ 1. Sign in to the Decision Center Business console as `rtsAdmin`.
  - \_\_\_ a. Open the Decision Center Business console by completing either of the following steps:
    - Clicking **Start** and then clicking the **Decision Center Business console** shortcut.
    - Or
    - Entering the following URL in a web browser window:  
`http://localhost:9090/decisioncenter`
- \_\_\_ 2. In the **Username** and **Password** fields, enter: `rtsAdmin`
- \_\_\_ 3. Click **Log in**.

## Creating a branch

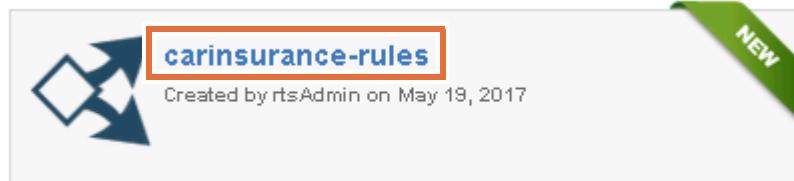


### Requirements

#### Scenario

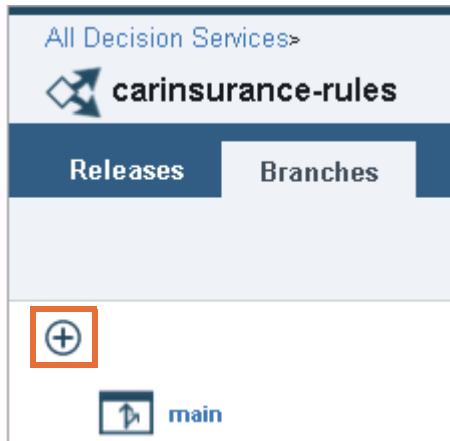
You must update a rule in the `carinsurance-rules` decision service, and decide to create a branch to help manage decision service changes.

- \_\_\_ 1. Click the **Library** tab.
- \_\_\_ 2. In the list of decision services, click the **carinsurance-rules** link.



- \_\_\_ 3. Click the **Branches** tab.

- \_\_\_ 4. Create a branch for carinsurance-rules.
- \_\_\_ a. Click the **New Branch** (plus [+]) sign icon.



- \_\_\_ b. In the “Create a Branch” window, in the **Give your branch a name** field, enter: changes
- \_\_\_ c. Keep **main** selected for the **Base it on this parent branch** menu.
- \_\_\_ d. Click **Create**.

**Create a Branch**

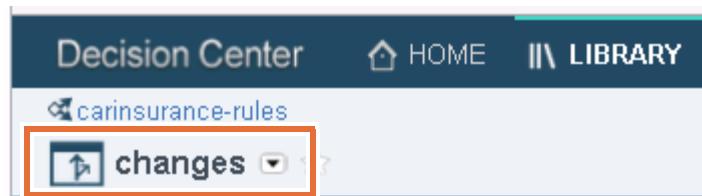
Give your branch a name:  
changes

Base it on this parent branch:  
main

Specify goals for this branch:

**Create**

- \_\_\_ 5. You should now be in the **changes** branch of the **carinsurance-rules** decision service.





## Information

You can switch between decision service branches by clicking the menu next to the branch name, and selecting the branch that you want to work in.



For example, if you are in the **main** branch and want to switch to the **changes** branch, you can click the menu and select **changes**.

---

## Updating a rule and merging branches

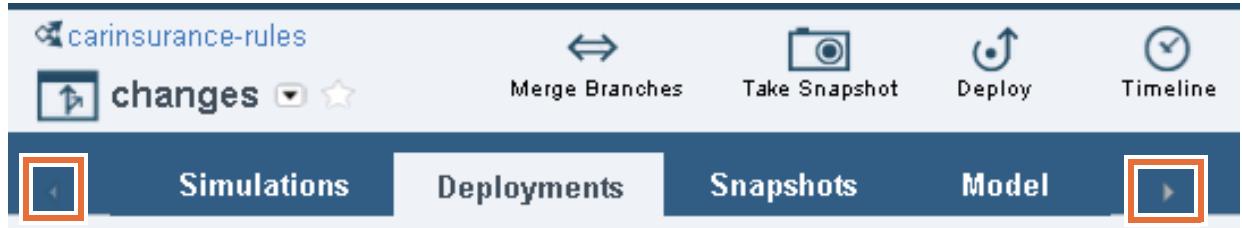
In this section, you edit a rule and merge the **changes** and **main** branches.

- 1. In the **changes** branch, go to the **Decision Artifacts** tab.



## Reminder

You might need to click the left and right arrows to access other tabs in the **changes** branch tab bar.



- 2. Change the minimum age of the **Check Eligibility > Drivers > Driver age** rule to 18.
  - a. In the left explorer pane, expand **carinsurance-rules > Check Eligibility** and click the **Drivers** folder to see the list of rules.
  - b. In the main Drivers pane, hover the mouse pointer over the **Driver age** rule and click **Edit this rule** (the pencil icon).
  - c. In the condition statement, change 16 to: 18  
The rule statement should read:  

```
if
    it is not true that the age (in years) of the driver at the start date
    of 'the policy' is between 18 and 80
```
  - d. Click **Save**, and then click **Create New Version**.

- 3. Click the **changes** breadcrumb to return to the **changes** branch from the **Driver age rule** view.

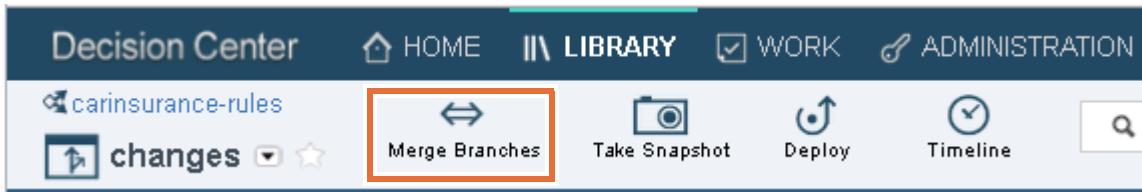


## Requirements

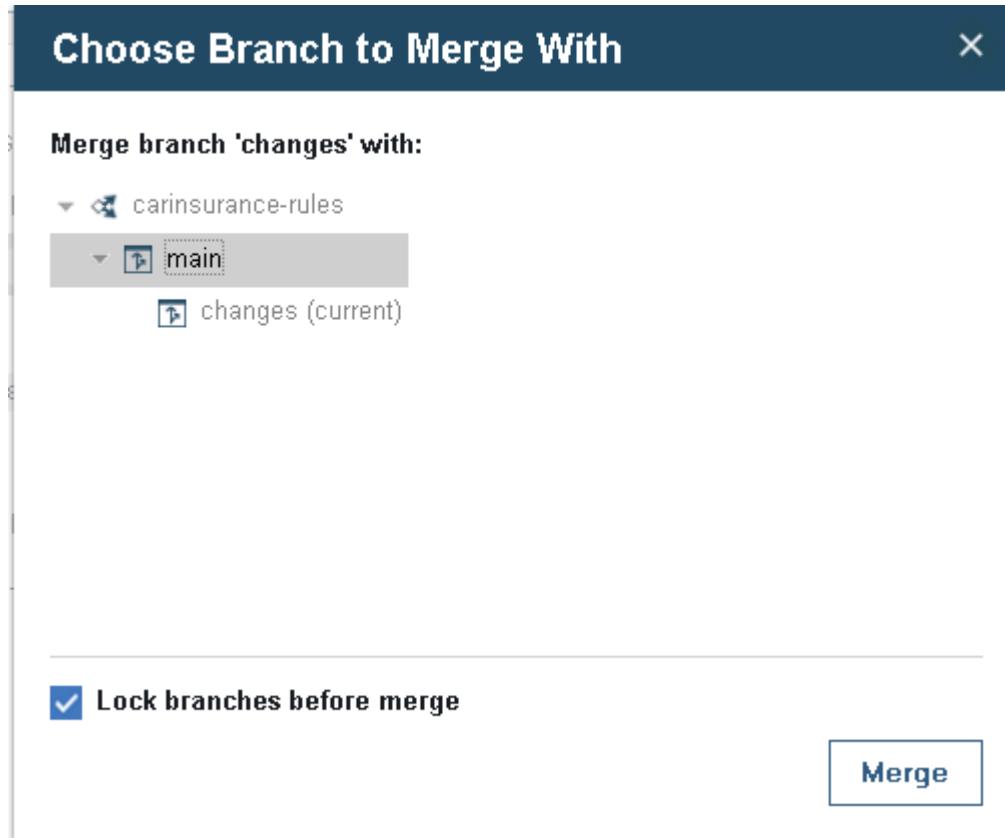
### Scenario

Now that you updated the **Driver age rule**, you merge the change that you made in the **changes** branch into the **main** decision service branch.

- 4. In the **changes** branch toolbar, click **Merge Branches**.



- 5. In the “Choose Branch to Merge With” window, click **main**, and then click **Merge**.



The “Merge changes with main” page opens. You can see various merge-related options, and a table that lists the differences between the branches and any actions that will be applied during the merge.

For example, the table notes that one element was modified, and the suggested action is to update the **main** branch with the modified element from the **changes** branch.

The screenshot shows the 'carinsurance-rules' application interface with the path 'carinsurance-rules > changes'. The main title is 'Merge changes with main'. Below it, there are merge direction options: 'Both directions' (selected), 'Only to changes', 'Only to main', 'Expand all', 'Collapse all', 'Reset all actions', and 'Ignore all changes'. A note below says 'Review and select the action to perform to merge each difference between the two branches.' A table lists differences between 'changes' and 'main' branches:

Name	changes	main	Action
carinsurance-rul...	1 element modified	-	update 1 element in main
carinsurance-bom	-	-	-

The merge-related options include:

- **Merge direction:** These options determine which changes are merged into the two branches.
  - o **Both directions:** All updates from the **main** branch are merged into the **changes** branch, and all updates from the **changes** branch are merged into the **main** branch.
  - o **Only to changes:** All updates from only the **main** branch are merged into the **changes** branch.
  - o **Only to main:** all updates from only the **changes** branch are merged into the **main** branch.
- **Expand all:** Expands all items in the table.
- **Collapse all:** Collapses all items in the table.
- **Reset all actions:** Resets the actions in the table to their default setting.
- **Ignore all changes:** Sets the action for all of the branch differences that are listed in the table to do not modify.



### Note

You can set merge actions, including **do not modify**, on an individual basis in the table **Action** column.

- 
6. Click **Expand all** to see the Driver age rule listed.

The table notes that the **Driver age** rule was modified in the **changes** branch, and the recommended action is to update the **main** branch with the modified rule.

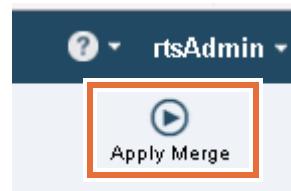
Name	changes	main	Action
carinsurance-rules			
Check Eligibility			
Drivers			
Driver age	modified	-	update in main
carinsurance-bom	-	-	-

7. In the **Action** column, click **update in main** to see the options, which include:

- **do not modify**
- **update in main**
- **update in changes**

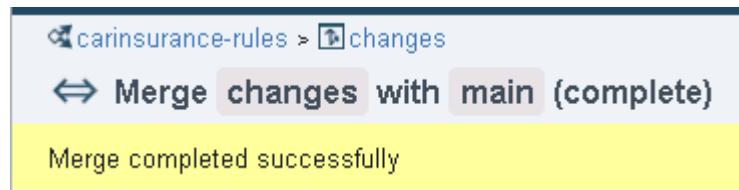
Name	changes	main	Action
carinsurance-rules			
Check Eligibility			
Drivers			
Driver age	modified	-	<div style="border: 1px solid #ccc; padding: 2px;">update in main</div> <div style="border: 1px solid #ccc; padding: 2px;">do not modify</div> <div style="border: 1px solid #ccc; padding: 2px; background-color: #0070C0; color: white; font-weight: bold;">update in main</div> <div style="border: 1px solid #ccc; padding: 2px;">update in changes</div>
carinsurance-bom	-	-	

8. Keep **update in main** as the action, and in the “Merge changes with main” toolbar, click **Apply Merge**.



9. In the Merge Branches window, click **Apply Merge**.

- 10. You see the Merge completed successfully message.



- 11. Switch to the main branch.
- Click the carinsurance-rules breadcrumb.
  - In the Branches tab, click main.
- 12. In the Decision Artifacts explorer pane, make sure that the Check Eligibility > Drivers folder is selected, and in the main Drivers pane, click Driver age.

You see that the minimum age modification (from 16 to 18) that you made in the changes branch was merged into the main branch.

A screenshot of the Business console. The top navigation bar shows the project name "carinsurance-rules" and a "main" branch selected, indicated by a red box. The main area has tabs for "Decision Artifacts", "Queries", "Tests", and "Simulations". The "Decision Artifacts" tab is active. The left sidebar shows a tree view of artifacts: "carinsurance-rules" has "Calculate Global Pr...", "Check Eligibility" (selected), "Drivers" (7 items, highlighted in grey), "Options" (6), "Policy" (8), and "Primary driver" (2). The right pane shows the "Drivers > Driver age" definition. A red box highlights the condition part of the rule: "if it is not true that the age (in years) of driver at the start date of 'the policy' is between 18 and 80 then refuse the application because 'At least one driver does not meet the age constraints';".

- 13. Log out of the Business console.

## Section 4. Exporting and erasing a decision service in the Enterprise console

In this section, you learn how to export a decision service and its projects to a compressed file. You also erase a decision service from the Decision Center repository.

### Logging in to the Enterprise console and viewing the Business console changes

- 1. Return to the Enterprise console window and log in as `rtsAdmin`.

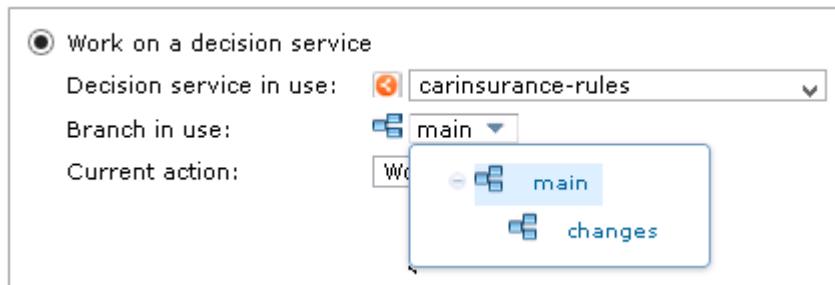
If needed, you can reopen the Enterprise console through the **Start** menu shortcut or by entering the following URL in a web browser window:

`http://localhost:9090/teamserver`

- 2. Click the **Home** tab, and make sure that **Work on a decision service** is selected.
- 3. Make sure that **carinsurance-rules** is selected from the **Decision service in use** menu.
- 4. View the changes that you worked on in the Business console.

- a. In the “Work on a decision service” section, click **main** from the **Branch in use** menu.

You can see that the **changes** branch is listed as a branch that you can work on.



- b. Keep **main** as the branch.
- c. Click the **Explore** tab and in the Smart Folders pane, expand **Check Eligibility**.
- d. Click the **Drivers** folder to load the list of rules.
- e. In the **Actions** column of the **Driver age** row, click **Preview** (the magnifying glass icon).

	Actions	Name	Status
<input type="checkbox"/>		Driver age	New
<input type="checkbox"/>		Number of at-fault accidents	New
<input type="checkbox"/>		Number of drivers	New

- f. In the Rule Preview section, you can see the modification to the minimum driver age (from 16 to 18) that you made in the Business console reflected in the rule.

The screenshot shows the 'Rule Preview' interface. At the top, there are two buttons: 'Rule Preview' (highlighted with a blue border) and 'Edit'. Below these are two sections: 'Name' (Driver age) and 'Status' (New). Under 'definitions', there is a conditional rule:  
**if** *it is not true that* the age (in years) of **driver** at the start date of '**the policy**' is between **18** and **80**  
**then** refuse the application because "**At least one driver does not meet the age constraints**" ;

## Exporting a decision service

When you export a decision service, the Enterprise console generates a compressed file that contains the artifacts and dependent projects of the decision service.

- 1. Click the **Configure** tab, and in the Administration section, click **Export Current Project State**.

The screenshot shows the 'Administration' section of the Enterprise console. It includes links for 'Installation Settings Wizard', 'Diagnostics', 'Clean Decision Center Cache', 'Import Projects', and 'Export Current Project State'. The 'Export Current Project State' link is highlighted with a red border. Below it is a note: 'Export and download the current project for the selected branch or baseline'. At the bottom of the list is 'Erase Current Decision Service'.

- 2. In the Export Current Project State page, click **Yes** to confirm the export.

The Enterprise console generates a compressed file that you can download.

- \_\_\_ 3. In the Export Current Project State page, click the **carinsurance-rules.zip** link to download the compressed file.

**Export Current Project State**

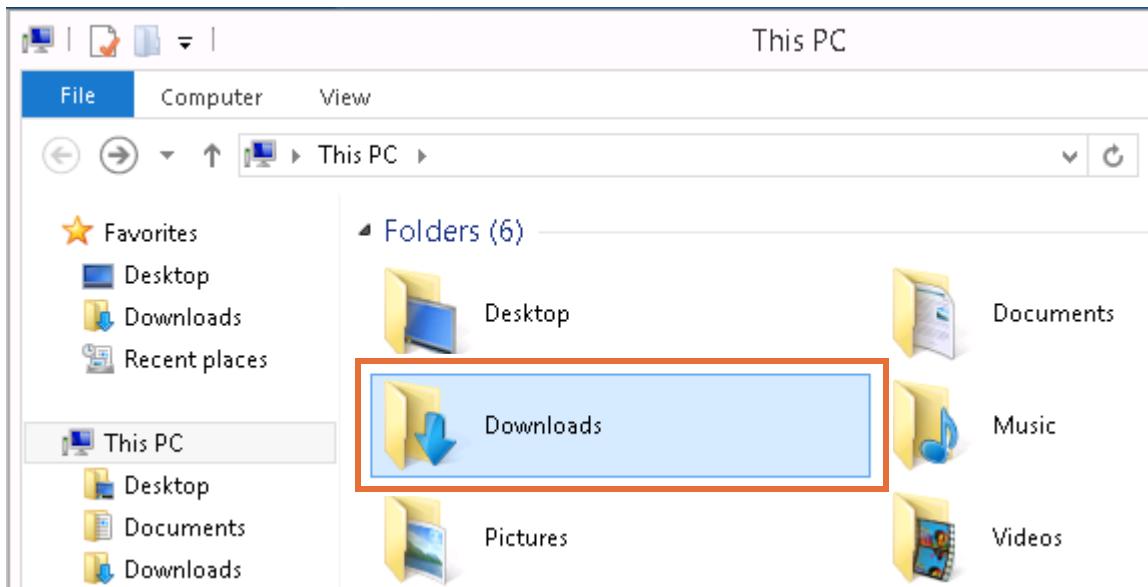
Click here to download the current project file:[[carinsurance-rules.zip](#)]

Back

- \_\_\_ 4. In the “Opening carinsurance-rules.zip” window, select **Save File** and click **OK**.
- \_\_\_ 5. View the exported file and its contents.
  - \_\_\_ a. In the Window Explorer, go to the Downloads folder by clicking **Start**, and then clicking **This PC**.



- \_\_\_ b. In the Folders section, double-click **Downloads**.



- \_\_\_ c. In the **Downloads** folder, double-click the **carinsurance-rules.zip** file to open it in Windows Explorer.
- \_\_\_ d. Take a few minutes to look through the project folders and explore the various decision service artifacts and projects.

- \_\_\_ e. When you are finished reviewing the decision service, close the carinsurances-rules.zip file.
- \_\_\_ f. Return to the Enterprise console window, and click **Back** to return to the main **Configure** tab page.

**Export Current Project State**

Click here to download the current project file:[[carinsurance-rules.zip](#)]

**Back**

## Erasing a decision service

When you erase a decision service, you delete it from the Decision Center repository. This action is final.

- \_\_\_ 1. Confirm that you are still using the carinsurance-rules decision service by looking at decision service name in the tab bar.

**Configure**

carinsurance-rules > main **Project:** carinsurance-rules

- \_\_\_ 2. In the Administration section, click **Erase Current Decision Service**.
- \_\_\_ 3. In the Confirm Decision Service Delete page, click **Yes** to confirm the deletion.
- \_\_\_ 4. When the deletion is complete, the Enterprise console returns to the **Home** tab.
- \_\_\_ 5. In the “Work on a decision service” section, click the **Decision service in use** menu and confirm that carinsurance-rules is no longer listed.

Work on a decision service

Decision service in use: **bomdomainpopulate-rules**

Branch in use:

Current action:

- \_\_\_ 6. Sign out of the Enterprise console.

## End of exercise

## Exercise review and wrap-up

In the first part of the exercise, you used the Enterprise console to import a decision service into Decision Center. In the second part of the exercise, you used the Business console to create and merge ungoverned decision service branches. Finally, in the last part of the exercise, you used the Enterprise console to export a decision service to a compressed file, and to erase a decision service from Decision Center.

# Exercise 17. Managing user access in Decision Center

## Estimated time

01:30

## Overview

This exercise continues the focus on rule and decision service management. In this exercise, you work with users in the Business console.

## Objectives

After completing this exercise, you should be able to:

- Define custom groups and users on the application server
- Manage groups and users from an LDAP repository

## Introduction

In this exercise, you learn how to manage security and user access for business users in Decision Center, including authorizing users to access Decision Center. While you might not be involved with creating and configuring users and passwords in your organization, this exercise covers these tasks so that you can see how user authentication is handled in ODM.

The workflow in this exercise consists of three main steps:

1. You learn how to manually configure users and groups in the WebSphere Liberty Profile application server.

In this part of the exercise, you work with the WebSphere Liberty Profile `server.xml` file to add a custom user to Decision Center.

2. You work with an LDAP server to set up custom groups and roles on the application server.

In this part of the exercise, you work with Apache Directory Server to create an LDAP server with custom groups and roles. The LDAP server authenticates users by making sure that the user name and password that are entered in the Decision Center login page are correct.

3. You learn how to manage access to the Decision Center Business console for the LDAP users.

In this part of the exercise, you work with the Decision Center Business console to import the users from the LDAP server. You also use the Business console user administration features to authorize the access of the LDAP users to Decision Center tasks.

This exercise includes these sections:

- [Section 1, "Viewing permissions for default groups"](#)
- [Section 2, "Configuring security on WebSphere Liberty Profile"](#)
- [Section 3, "Setting up new users on an LDAP server"](#)
- [Section 4, "Adding users and groups in LDAP"](#)
- [Section 5, "Delegating authorization to Decision Center"](#)
- [Section 6, "Administering groups and users in the Business console"](#)

## Requirements

This exercise requires you to work in Decision Center and with the WebSphere Liberty Profile `server.xml` file. You also work with Apache Directory Server to set up an LDAP server with users and user groups.

## Section 1. Viewing permissions for default groups

In this section, you see how permissions affect the Decision Center tools that are available to the rule author and administrator user roles.

- 1. If the sample server is not already started, click **Start** and then click the **Start Sample Server** shortcut.
- 2. Open Decision Center Business console by clicking **Start** and then clicking the **Decision Center Business console** shortcut, or by entering the following URL in a browser:

`http://localhost:9090/decisioncenter`

The following table lists some of the user groups that are defined for Decision Center. You can sign in as a member of each of the following groups.

Role	User name	Password
Rule author	rtsUser1	rtsUser1
Configuration Manager	rtsConfigManager	rtsConfigManager
Installer	rtsInstaller	rtsInstaller
Administrator	rtsAdmin	rtsAdmin

- 3. Sign in to Business console with rule author permissions to see which tabs are available to business users.
  - **User name:** rtsUser1
  - **Password:** rtsUser1
- 4. Notice that the **Administration** tab is not available.



- 5. Sign out and sign in again with configuration permissions.
  - **User name:** rtsAdmin
  - **Password:** rtsAdmin
- 6. Notice that the **Administration** tab is now available.



- 7. Sign out of the Business console.

## Section 2. Configuring security on WebSphere Liberty Profile

In this section, you view the user and group settings in the WebSphere Liberty Profile `server.xml` file, which contains the server configuration settings. You also create a user by updating the user definitions in the WebSphere Liberty Profile basic user registry.

### 2.1. Exploring the `server.xml` file

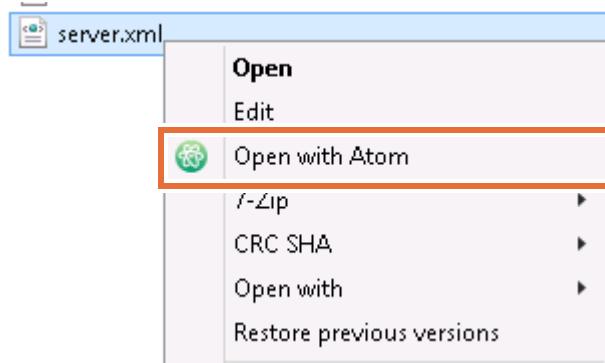
- \_\_\_ 1. Stop the sample server.
  - \_\_\_ a. Click **Start**, and then click the **Stop Sample Server** shortcut.
  - \_\_\_ b. When you see the `Server odm8900 stopped` and `BUILD SUCCESSFUL` messages, press any key to close the Stop Sample Server window.
- \_\_\_ 2. In Windows Explorer, go to the `C:\IBM\wlp\usr\servers\odm8900` directory.



#### Important

Going forward, these exercise instructions refer to the `C:\IBM\wlp\usr\servers\odm8900` directory path as: `<serverDir>`

- \_\_\_ 3. Make a backup of the `server.xml` file.  
If needed, you can use the backup file to restore the server settings.
  - \_\_\_ a. Copy and paste the `server.xml` file by selecting the file, pressing **Ctrl+C**, and then pressing **Ctrl+V**.
  - \_\_\_ b. Rename the file so that you can identify it as the backup file, such as:  
`server-backup.xml`
- \_\_\_ 4. Open the WebSphere Liberty Profile `server.xml` file.
  - \_\_\_ a. Right-click the `server.xml` file and click **Open with Atom**.



The `server.xml` file opens, and you can view its contents.

**Note**

Depending on the theme that you use for Atom, the Atom interface that you use might look different from the screen captures in this exercise.

To see the themes that are available, go to **File > Settings**, and in the **Settings** menu, click **Themes**.

5. In the `server.xml` file, go to the “Web application security” section.

- \_\_\_ a. In Atom, go to **Find > Find in Buffer**.
- \_\_\_ b. In the **Find in current buffer** field, enter: `web application security`
- \_\_\_ c. Click **Find**.



In the file, you should see the commented header:

```
<! -- Web application security -->
```

The screenshot shows the `server.xml` file in Atom. The code editor displays the XML configuration. A specific section is highlighted with a red box, showing the commented header `<! -- Web application security -->`. The XML content below includes basic registry definitions and user entries for `resAdmin`, `resDeploy`, and `resMonitor`.

```

<!-- Web application security -->
<basicRegistry id= "basic" realm= "customRealm">
    <user name= "resAdmin" password= "resAdmin" />
    <user name= "resDeploy" password= "resDeploy" />
    <user name= "resMonitor" password= "resMonitor" />

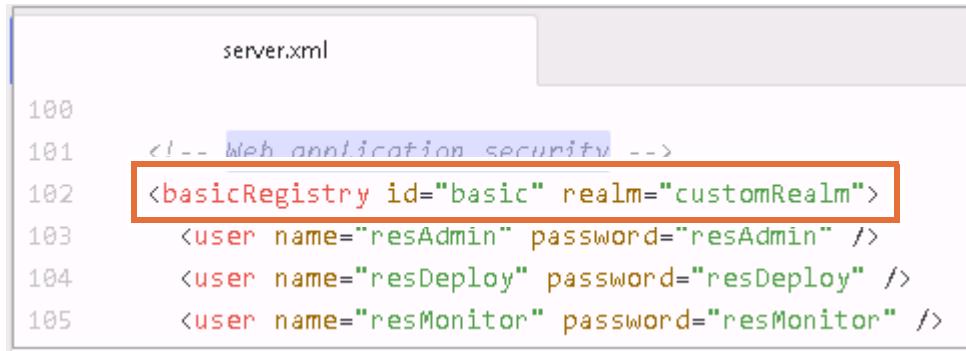
```

**Information**

This section contains the sample server user and group definitions. User authentication can occur either through the web application server (for example, through the basic user registry section, which is described in the next step), or it can occur through a separate registry, such as LDAP.

You work on creating and adding users through LDAP later in this exercise.

- \_\_\_ 6. Explore the user and group definitions in the “Web application security” section.
- \_\_\_ a. Notice the `<basicRegistry>` tag.



```

server.xml

100
101    <!-- Web application security -->
102    <basicRegistry id="basic" realm="customRealm">
103        <user name="resAdmin" password="resAdmin" />
104        <user name="resDeploy" password="resDeploy" />
105        <user name="resMonitor" password="resMonitor" />

```

The `<basicRegistry>` section is where you can create the `<user>` and `<group>` definitions for the server.

## Information

Individual users must be mapped to a user group. Group membership determines the permissions that a user has.

- 
- The first set of user and group definitions are for Rule Execution Server (RES).
  - \_\_\_ b. Scroll to the Decision Center user and group definitions, which are after the Rule Execution Server group definitions.

## Information

Some of the Decision Center user and group names are prefixed by `rts`. RTS stands for Rule Team Server, which was the original product name for Decision Center.

- 
- \_\_\_ c. Notice the Decision Center section starts with a list of the Decision Center user name and password definitions, including `rtsAdmin`, `rtsUser1`, `Bea`, and `Paul`.



```

<!-- Web application security -->
<basicRegistry id="basic" realm="customRealm">
    <user name="rtsAdmin" password="rtsAdmin" />
    <user name="rtsConfig" password="rtsConfig" />
    <user name="rtsUser1" password="rtsUser1" />
    <user name="Bea" password="Bea" />
    <user name="Paul" password="Paul" />
    <user name="Val" password="Val" />
    <user name="Eli" password="Eli" />
    <user name="rmtuser" password="rmtuser" />

```

- \_\_\_ d. Scroll to the group definitions for Decision Center, which are listed after the user definitions.
- \_\_\_ e. Notice that each opening `<group>` tag has a `name` attribute.

- \_\_\_ f. Also, notice that each group definition includes the members who belong to the group. Each `<member>` tag includes a `name` attribute that is used to indicate the user names of the users who are in the group.

For example, the `rtsAdministrator` group includes the `rtsAdmin` and `Paul` users as members.

```
<group name="rtsAdministrator">
  <member name="rtsAdmin" />
  <member name="Paul" />
</group>
```

## 2.2. Creating custom users through the basic user registry

In this section, you add a custom user to Decision Center.



### Note

You can copy and paste the `myUser` definitions from `<labfilesDir>\users\myUser.txt`.

The directory for `<labfilesDir>` in the computer lab environment that is created for this course is:

`C:\labfiles`

- \_\_\_ 1. Add a definition for the user `myUser` to the `server.xml` file.

- \_\_\_ a. In the `server.xml` file, place the cursor after the `<user name="rmtuser" password="rmtuser"/>` tag.

```
<user name="rmtuser" password="rmtuser" /> |
```

- \_\_\_ b. Press Enter to create a new line.

- \_\_\_ c. Enter the following line:

```
<user name="myUser" password="myUser" />
```

```
<user name="Eli" password="Eli" />
<user name="rmtuser" password="rmtuser" />
<user name="myUser" password="myUser" /> |
```

- \_\_ 2. Add `myUser` as a member of the `rtsUser` group.
  - \_\_ a. In the `server.xml` file, find the `rtsUser` group definition, which has the opening tag `<group name="rtsUser">`.

```
<group name="rtsUser">
  <member name="rtsUser1" />
  <member name="Bea" />
  <member name="Val" />
  <member name="Eli" />
  <member name="rmtuser" />
</group>
```

- \_\_ b. Place the cursor after the `<member name="rmtuser" />` tag, and press Enter to create a new line.
- \_\_ c. Enter the following line:

`<member name="myUser" />`

```
<member name="Eli" />
<member name="rmtuser" />
<member name="myUser" />|
```

## Important

Make sure that the `rtsUser` group member definition for `myUser` is added before the closing `</group>` tag.

- \_\_ d. Save your changes to the `server.xml` file (Ctrl+S).
- \_\_ e. Close the `server.xml` file. However, do not close Atom, as you use it later in this exercise.
- \_\_ f. Keep the `<serverDir>` window open. You use `<serverDir>` later in this exercise.
- \_\_ 3. Restart the sample server by clicking **Start**, and then clicking the **Start Sample Server** shortcut.
- \_\_ 4. When the server is restarted, press any key to close the Start Sample Server window.

## Information

When you change security options on the application server, make sure that you restart WebSphere Liberty Profile by stopping the server and restarting it.

## 2.3. Signing in to Decision Center with your custom user

- 1. If you need to restart the Decision Center Business console, double-click the Decision Center Business console desktop shortcut or enter the following URL in the web browser:

<http://localhost:9090/decisioncenter>



### Important

Make sure that you are using the correct port number for the sample server.

- 2. Sign in to the Business console with your new user login:
  - **Username:** myUser
  - **Password:** myUser
- 3. Notice that the available tabs are consistent with the `rtsUser` privileges. The **Administration** tab is not available.
- 4. Sign out of the Business console.

## Section 3. Setting up new users on an LDAP server

### 3.1. Setting up an LDAP connection in Apache Directory Studio

- \_\_\_ 1. Make sure that the LDAP server instance is running.
- \_\_\_ a. From the **Start** menu, click **Administrative Tools**.



- \_\_\_ b. In the list of tools, double-click **Services**.
- \_\_\_ c. In the Services window, find **ApacheDS - default** in the list, and make sure that it is running.

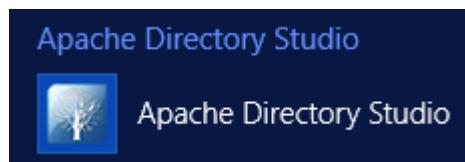
Services (Local)			
ApacheDS - default	Name	Description	Status
<a href="#">Stop</a> the service <a href="#">Restart</a> the service	ActiveX Installer (AxInstSV)	Provides Us...	
	Adobe Acrobat Update Serv...	Adobe Acro...	Running
	ApacheDS - default	ApacheDS d...	Running
	App Readiness	Gets apps re...	
	Application Experience	Processes a...	
	Application Identity	Determines ...	
	Application Information	Facilitates t...	



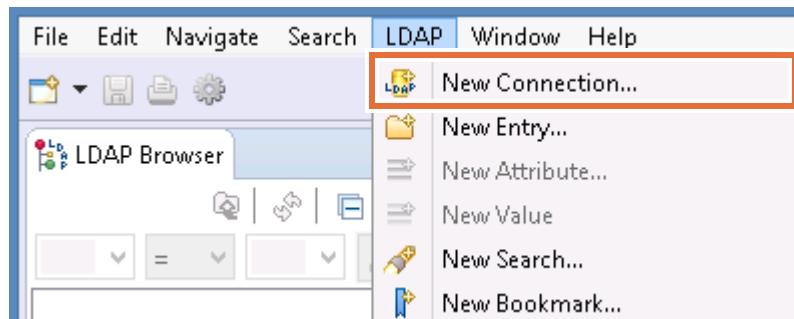
#### Important

Do not close the Services window, as you restart the service later in this exercise.

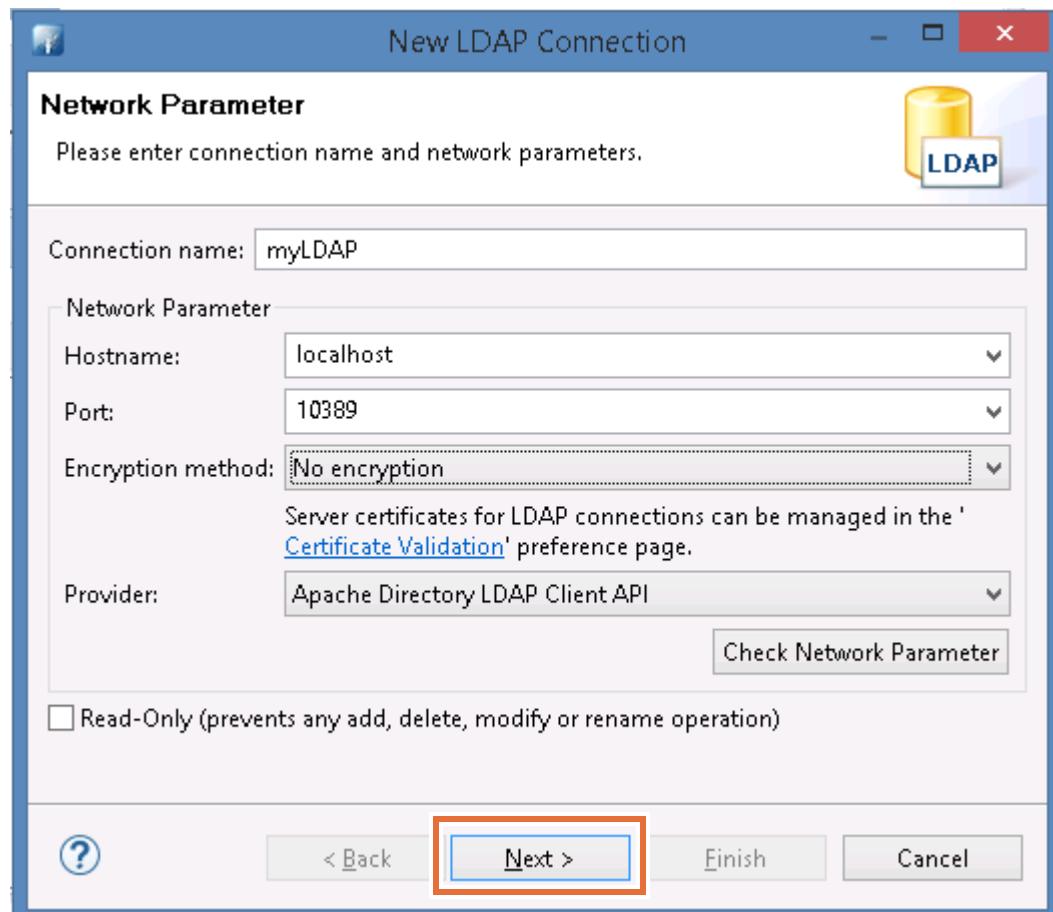
- 
- \_\_\_ 2. Start Apache Directory Studio.
  - \_\_\_ a. Click **Start**, and then click the down arrow icon to open the Apps list.
  - \_\_\_ b. In the Apache Directory Studio section, click **Apache Directory Studio**.



- \_\_\_ 3. From the **LDAP** menu, click **New Connection**.

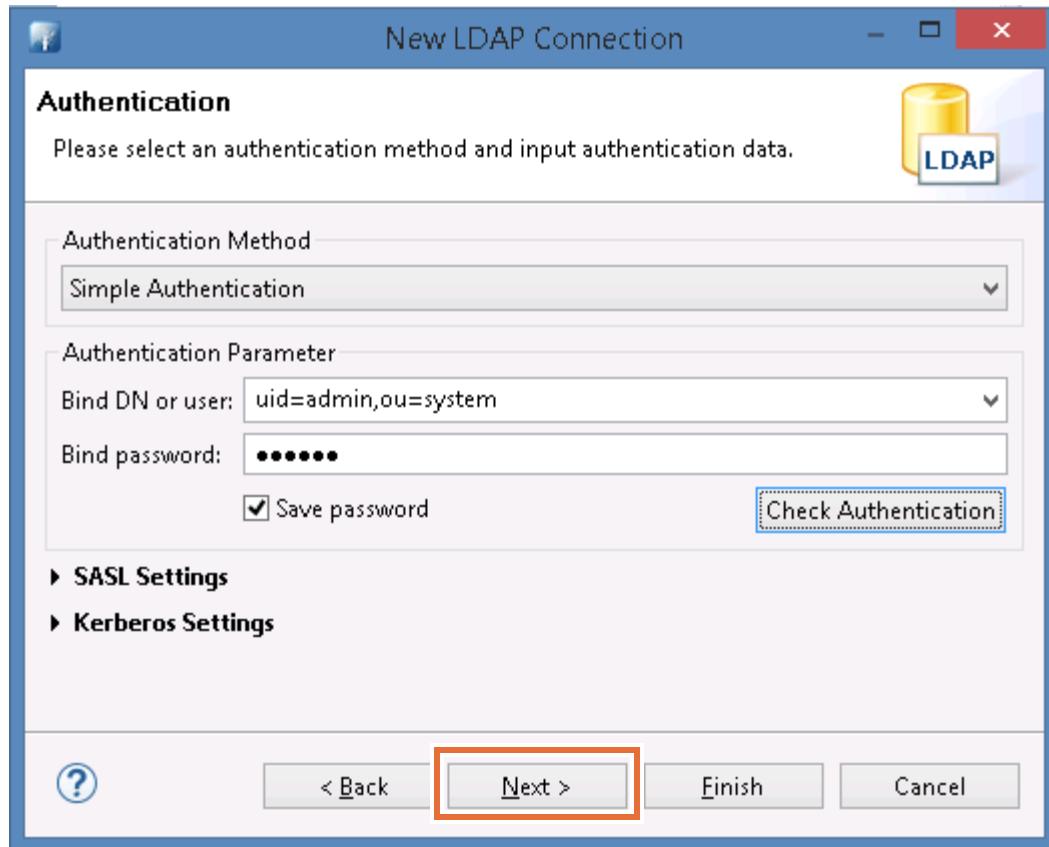


- \_\_\_ 4. In the New LDAP Connection wizard, enter a connection name and network properties.
- \_\_\_ a. For the **Connection name** field, type: `myLDAP`
  - \_\_\_ b. For the **Hostname** field, type: `localhost`
  - \_\_\_ c. For the **Port** field, type: `10389`
  - \_\_\_ d. Click **Next**.



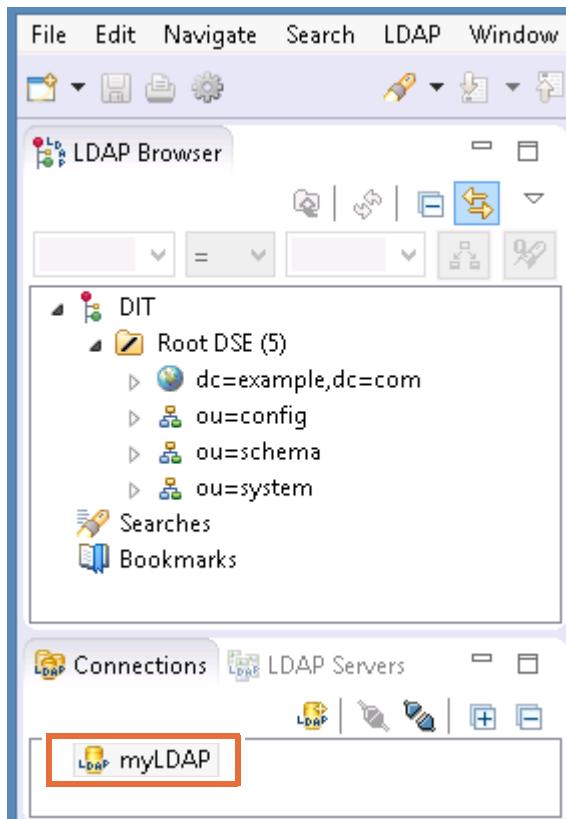
- \_\_\_ 5. On the Authentication page, enter the Bind distinguished name (DN) and password.
- \_\_\_ a. In the **Bind DN or user** field, type: `uid=admin,ou=system`
  - \_\_\_ b. In the **Bind password** field, type: `secret`

- \_\_ c. *Optional.* Click **Check Authentication** to confirm the settings. A Check Authentication window opens, and shows the progress of the check. When you see the message **The authentication was successful**, click **OK** to close the window.
- \_\_ d. Click **Next**.

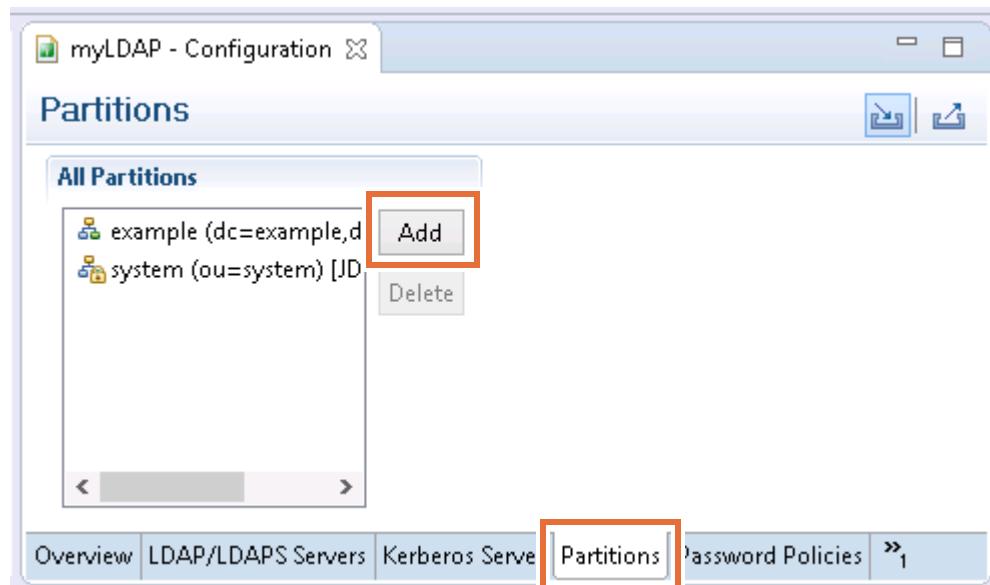


- \_\_ 6. Complete the connection setup.
  - \_\_ a. On the Browser Options page, leave the options at their default settings page, and click **Next** again.
  - \_\_ b. On the Edit Options page, click **Finish**.

When the connection is established, you see it open in the **Connections** view.

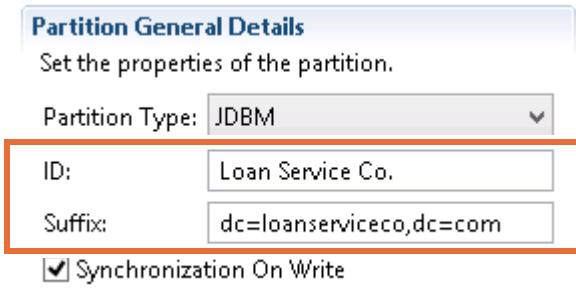


- 7. Create a base distinguished name (DN).
  - a. In the Connections pane, right-click the **myLDAP** connection, and click **Open Configuration**.
  - b. In the configuration editor, click the **Partitions** tab and click **Add**.

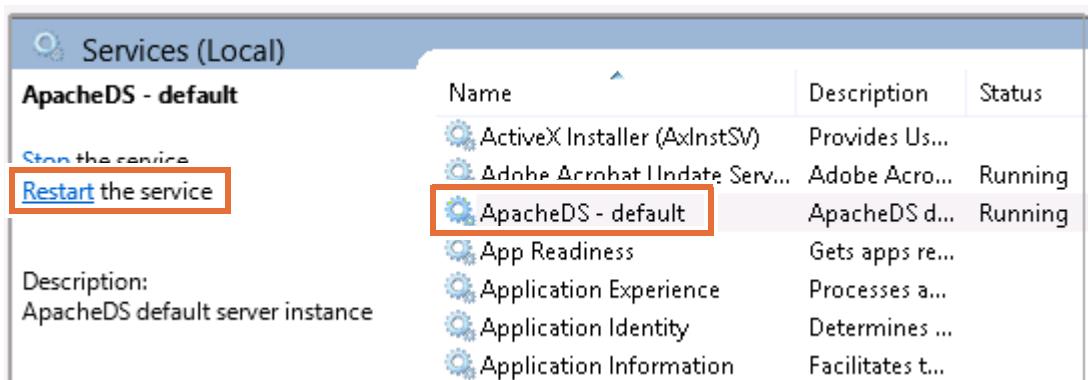


- c. The **Partition General Details** section opens.
- d. In the **ID** field, replace **partition1** with the name: **Loan Service Co.**

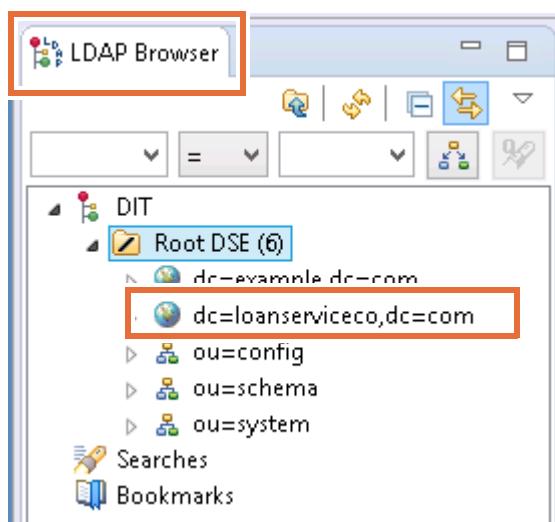
- \_\_ e. In the **Suffix** field, replace `partition1` in `dc=partition1,dc=com` with: `loanserviceco`



- \_\_ 8. Save the configuration changes (Ctrl+S).
- \_\_ 9. Restart the LDAP server to get the configuration changes in effect.
  - \_\_ a. Go back to the Services window.  
If needed, reopen the Services list by clicking the **Start** menu, clicking **Administrative Tools**, and in the list of tools, double-clicking **Services**.
  - \_\_ b. In the list of services, select **ApacheDS - default** and click **Restart**.



- \_\_ 10. Go back to Apache Directory Studio.
- \_\_ 11. In the **LDAP Browser** pane, right-click the **Root DSE** folder and click **Reload Entry** to see the new base DN.



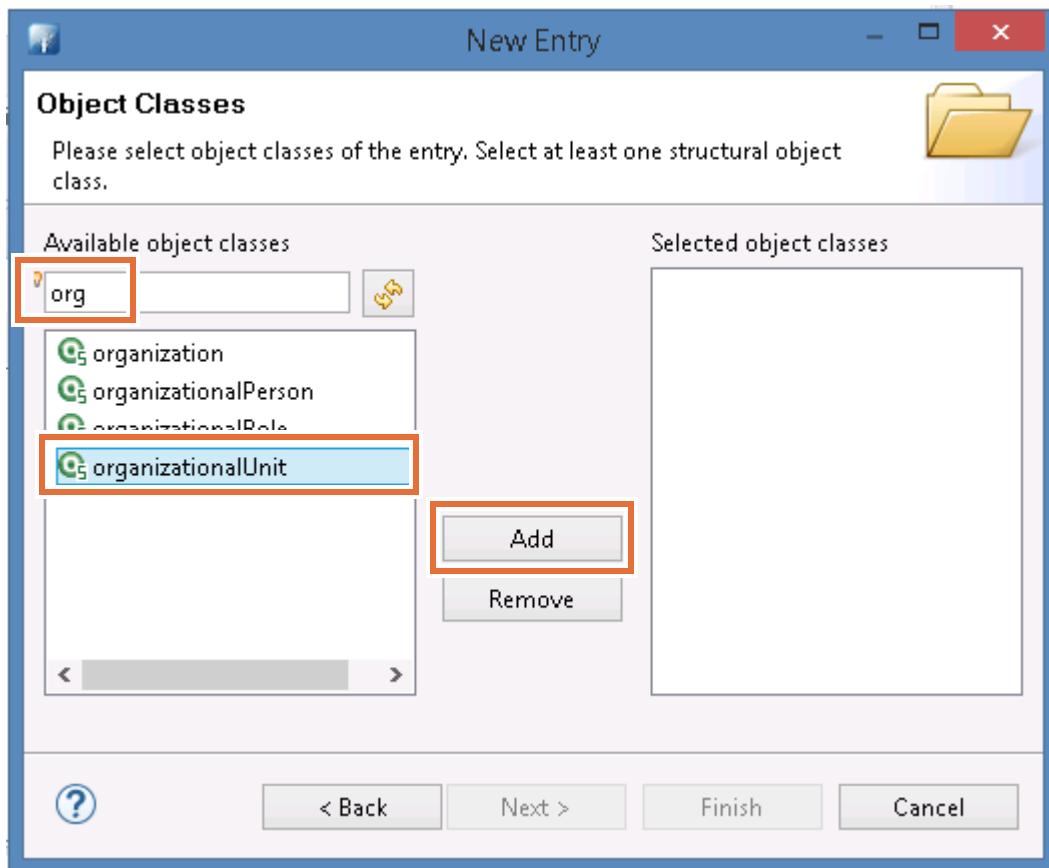
## Section 4. Adding users and groups in LDAP

Before you create the actual users, you first create an organizational unit to contain all the users.

### 4.1. Creating an organizational unit

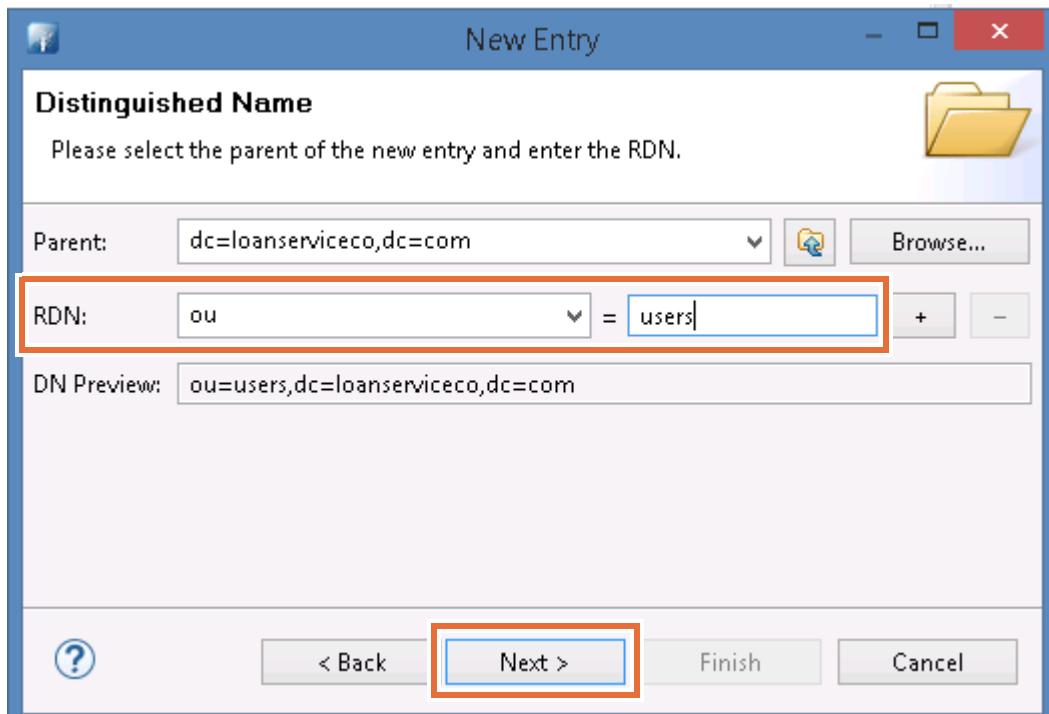
In this section, you create an organizational unit, which is a way of classifying objects (such as users or groups) in a directory.

- 1. In the LDAP Browser pane, right-click the **dc=loanserviceco,dc=com** entry and click **New > New Entry**.
- 2. In the New Entry wizard, select **Create entry from scratch** and click **Next**.
- 3. On the Object Classes page, in the **Available object classes** filter field, type: **org**
- 4. Select **organizationalUnit** in the list, and click **Add**.



- 5. Click **Next**.
- 6. On the Distinguished Name page, enter the RDN settings.
  - a. In the **RDN** field, select **ou** from the list.
  - b. In the value ("=") field, type: **users**

- c. Click **Next**.



- 7. On the Attributes page, review the attributes, and then click **Finish**.

## 4.2. Creating users

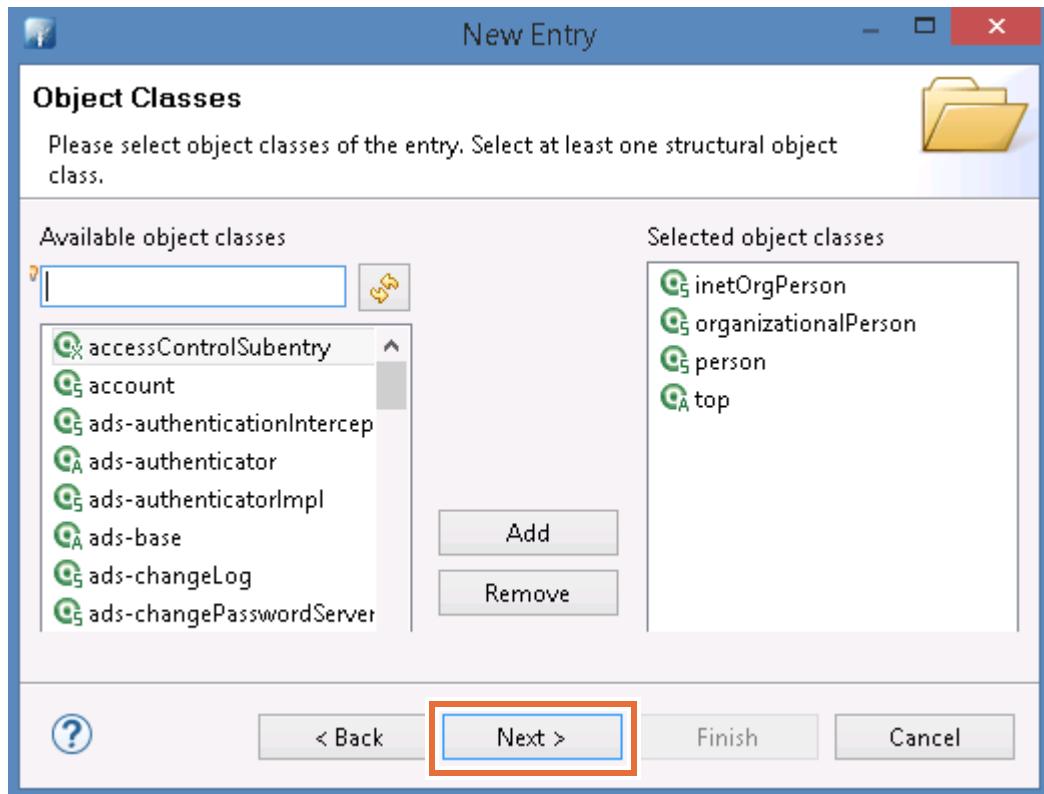
In this section, you create two users: Jane and John.

- 1. In the LDAP Browser pane, right-click the **ou=users** entry and click **New > New Entry** to open the New Entry wizard.

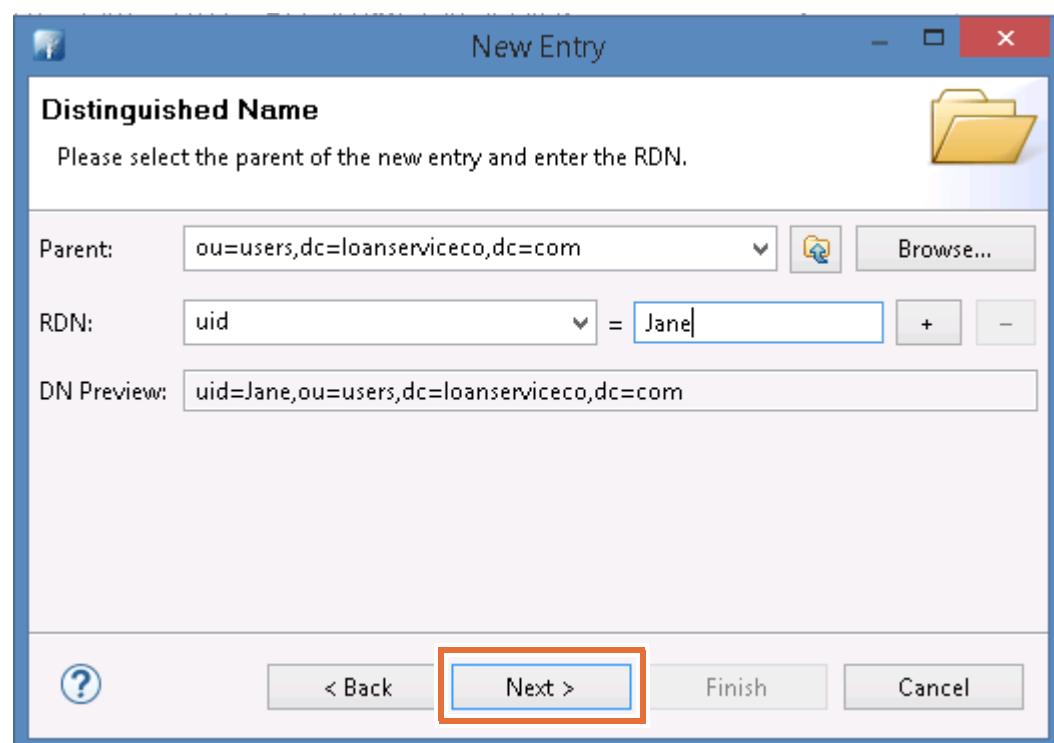
Attribute Description	Value
<b>objectClass</b>	<i>organizationalUnit (str)</i>
<b>objectClass</b>	<i>top (abstract)</i>
<b>ou</b>	<i>users</i>

- 2. On the Entry Creation Method page, select **Create entry from scratch**, and click **Next**.
- 3. On the Object Classes page, add the *inet* object class.
- a. In the **Available object classes** filter field, type: *inet*
- b. Select **inetOrgPerson** in the list and click **Add**.

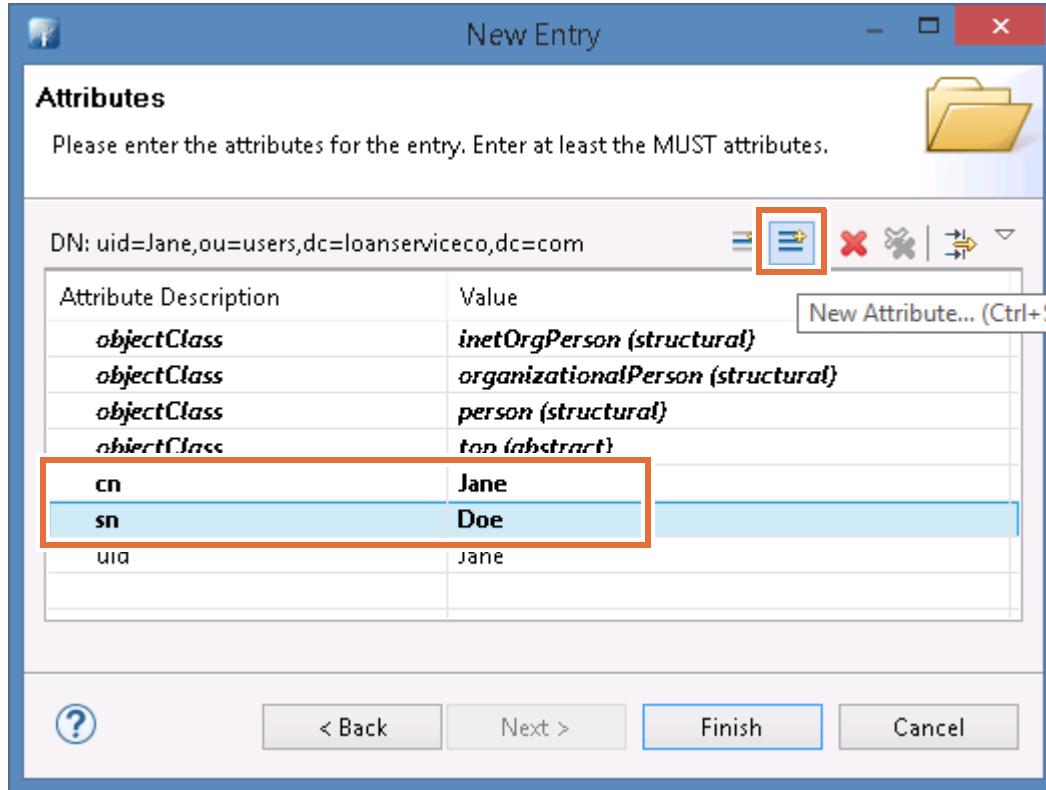
- \_\_ c. Click **Next**.



- \_\_ 4. On the Distinguished Name page, enter the user ID for Jane.
- \_\_ a. In the **RDN** field, select **uid** from the list.
- \_\_ b. In the value ("=") field, type: Jane
- \_\_ c. Click **Next**.

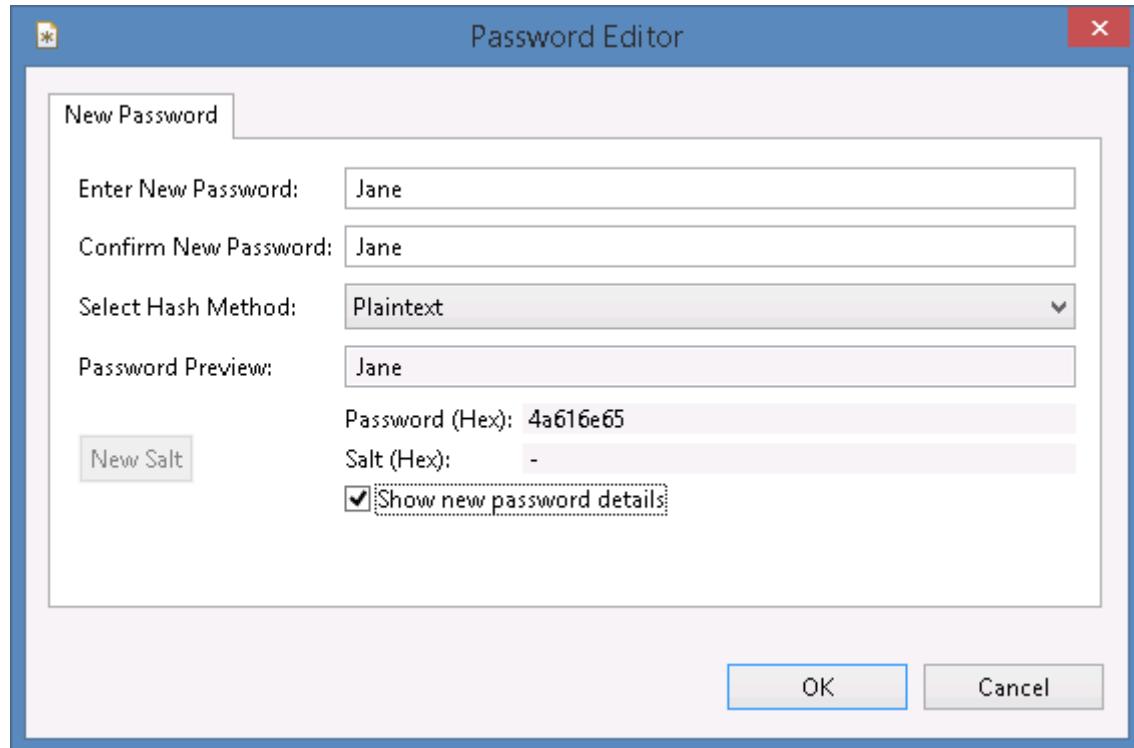


- \_\_\_ 5. On the Attributes page, define the login name and password for Jane.
- \_\_\_ a. In the **Value** column field for **cn**, type: Jane
  - \_\_\_ b. Double-click the **Value** column field for **sn**, and type: Doe
  - \_\_\_ c. In the toolbar, click the **New Attribute** icon.



- \_\_\_ d. For the **Attribute type** field, choose **userPassword** and click **Finish**.
- \_\_\_ 6. In the Password Editor, set the password to: Jane
- \_\_\_ a. In the **Enter New Password** field, type: Jane
  - \_\_\_ b. In the **Confirm New Password** field, type: Jane
  - \_\_\_ c. In the **Select Hash Method**, select **Plaintext**.

- \_\_ d. Select **Show new password details** to see the password in clear text.



- \_\_ e. Click **OK**.
- \_\_ 7. Click **Finish** to close the New Entry wizard.  
The user Jane is created in the LDAP directory.
- \_\_ 8. Repeat the steps in this section to create another user with these properties:
- uid=John
  - cn=John
  - sn=Doe
  - userPassword=John



### Troubleshooting

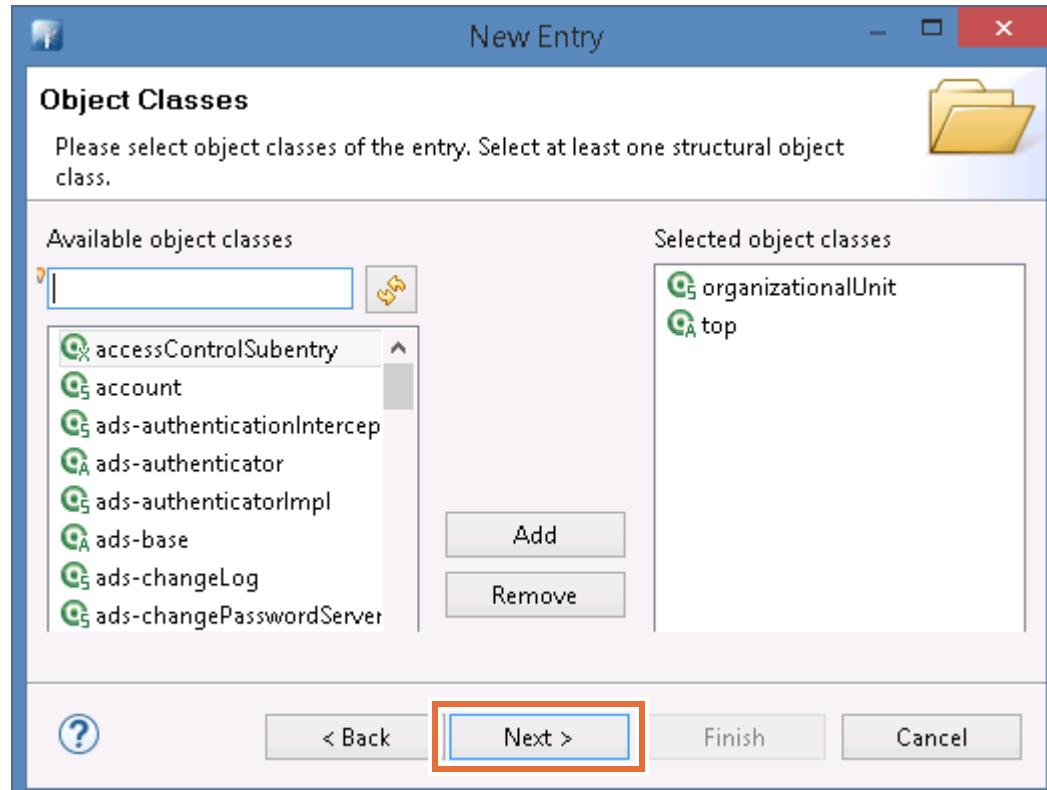
You might need to right-click **ou=users** and click **Reload entry** to see your two new users listed.

## 4.3. Adding groups

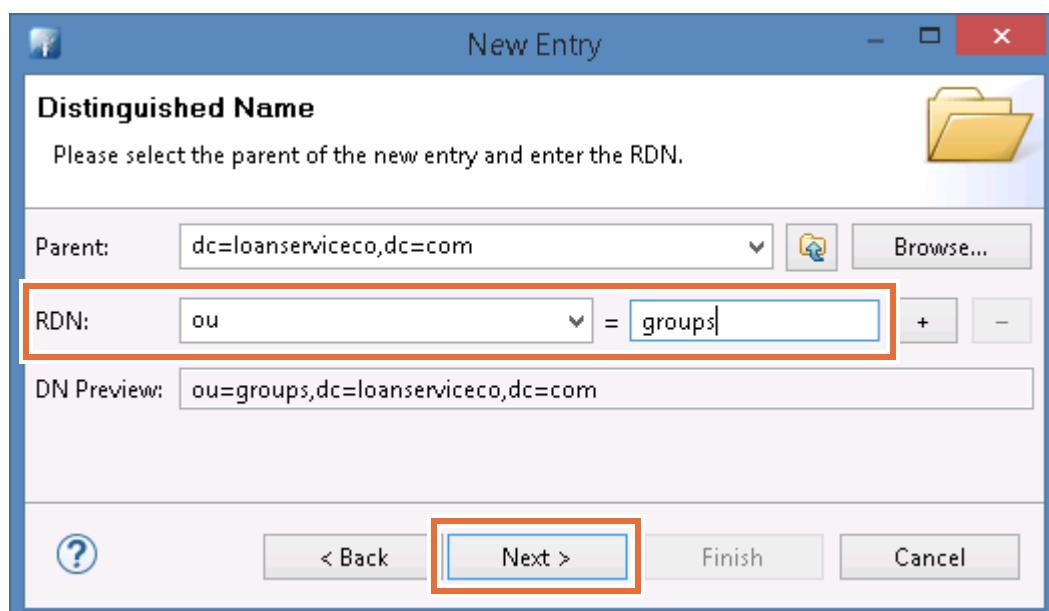
First, you create an organizational unit to hold your groups.

- \_\_ 1. Right-click the **dc=loanserviceeco,dc=com** entry, click **New > New Entry**.
- \_\_ 2. In the New Entry wizard, choose **Create entry from scratch** and click **Next**.
- \_\_ 3. On the Object Classes page, add **organizationalUnit** to the **Selected object classes** field.
  - \_\_ a. In the **Available object classes** filter field, type: **org**

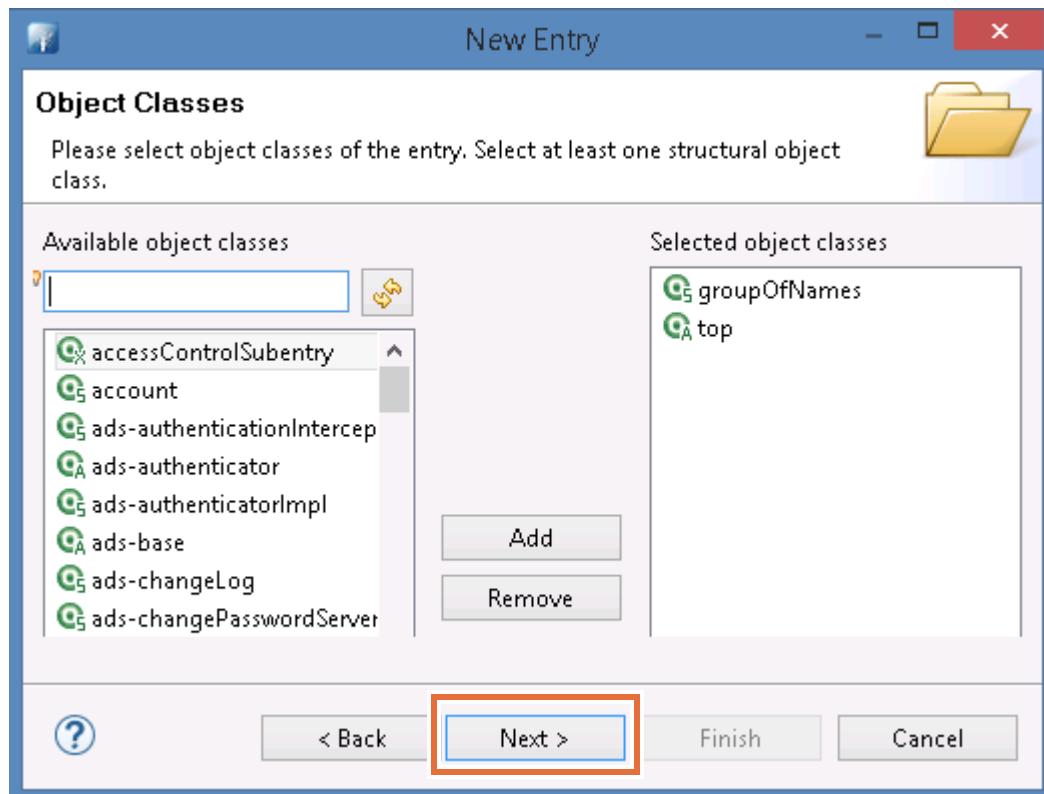
- \_\_ b. Select **organizationalUnit** from the list and click **Add** to move it to the **Selected object classes** field.
- \_\_ c. Click **Next**.



- \_\_ 4. On the Distinguished Name page, define the organizational unit name.
- \_\_ a. In the **RDN** field, select **ou** from the list.
- \_\_ b. In the value ("=") field, type: **groups**
- \_\_ c. Click **Next**.

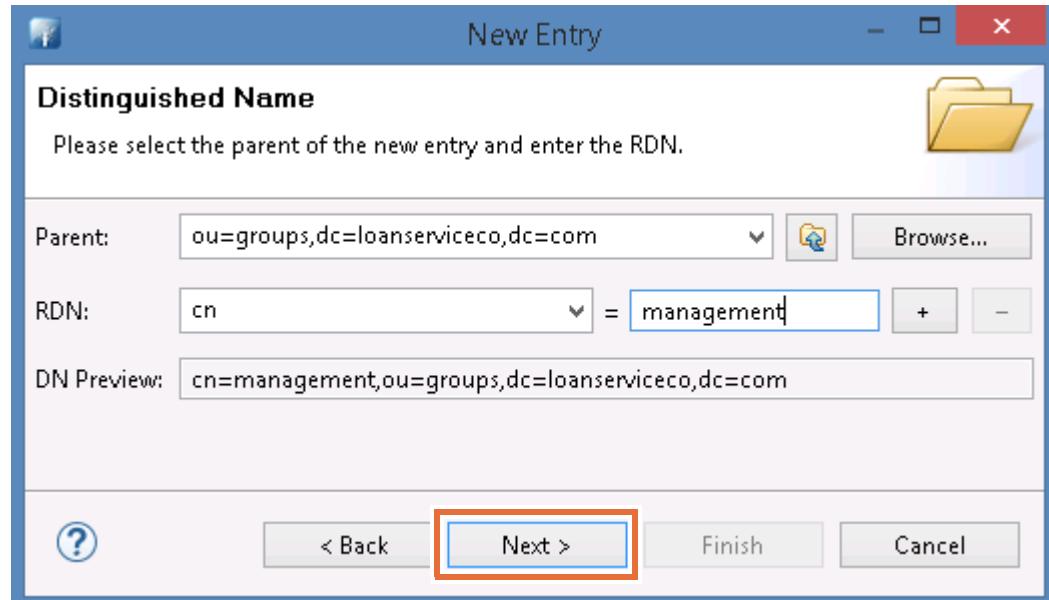


- \_\_\_ 5. On the Attributes page, review the attribute list and click **Finish**.
- \_\_\_ 6. Create a **management** group with **Jane Doe** as a member.
  - \_\_\_ a. In the LDAP Browser pane, right-click **ou=groups** entry, and click **New > New Entry**.
  - \_\_\_ b. In the New Entry wizard, select **Create entry from scratch** and click **Next**.
  - \_\_\_ c. On the Object Classes page:
    - In the **Available object classes** filter field, type: **group**
    - Select **groupOfNames** from the list, and click **Add**.
    - Click **Next**.

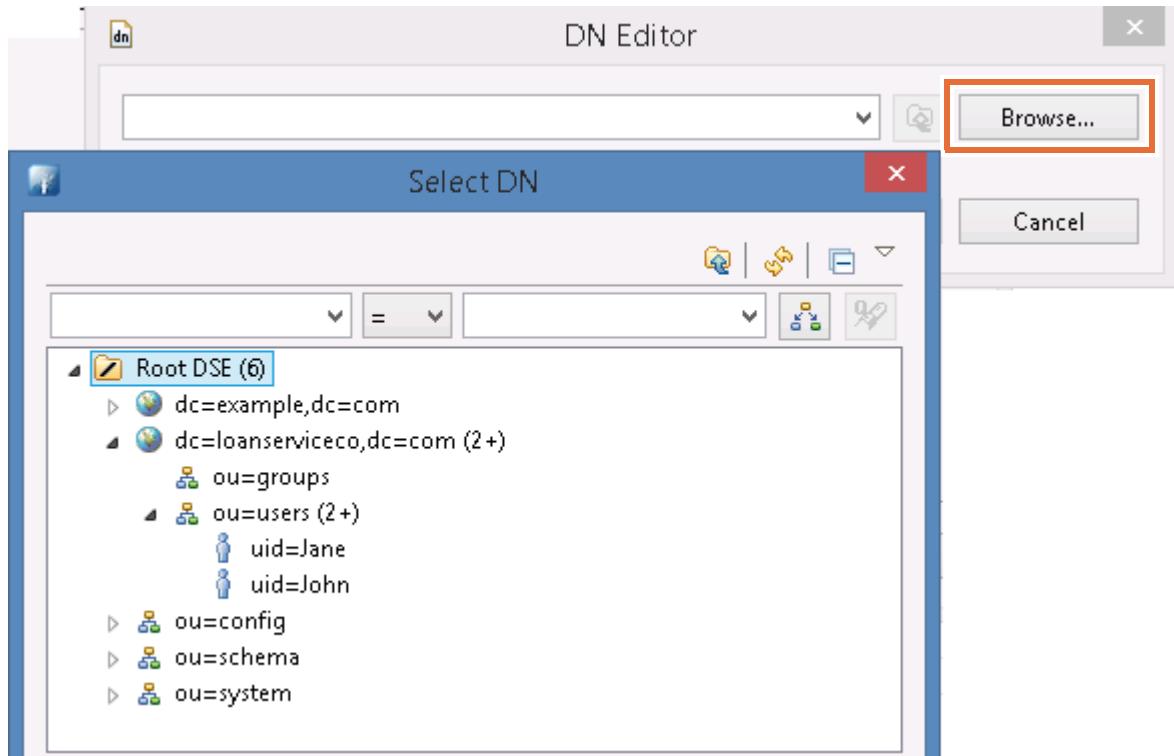


- \_\_\_ d. On the Distinguished Name page:
  - In the **RDN** field, select **cn** from the list.
  - In the value ("=") field, type: **management**

- Click **Next**.

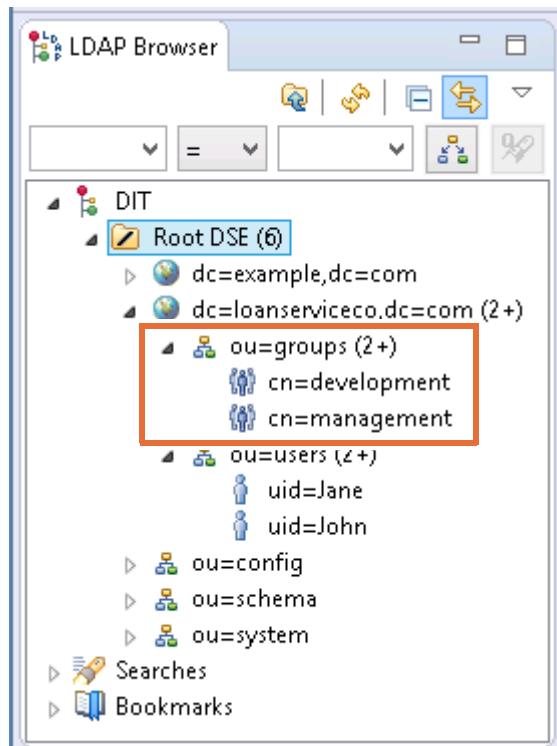


- e. When the DN Editor window opens, click **Browse** and expand **dc=loanserviceco,dc=com > ou=users**



- f. Select **uid=Jane** and click **OK**.
- g. Click **OK** to close the DN Editor.
- h. Back in the New Entry wizard, click **Finish**.

- 7. Repeat [Step 6](#) to create a **development** group (`cn=development`) with **John Doe** as a member.



## Section 5. Delegating authorization to Decision Center

To authorize all authenticated users from LDAP and delegate authorization to Decision Center, LDAP configuration information must be added to the `server.xml` file. In this section, you install a prepared server file that includes the required LDAP configuration. You also review the LDAP information in the file.

### 5.1. Installing the prepared server.xml file

In this section, you copy and overwrite the prepared server file into `<serverDir>`. This version of the `server.xml` file has the LDAP configuration information included in it.

- \_\_\_ 1. Shut down the sample server by clicking **Start**, and then clicking **Stop Sample Server**.
- \_\_\_ 2. In `<serverDir>`, rename the `server.xml` file to: `server-backup2.xml`



#### Reminder

The directory path for `<serverDir>` is: `C:\IBM\wlp\usr\servers\odm8900`

- \_\_\_ 3. Go to `<labfilesDir>\users`.
- \_\_\_ 4. Click `server-ldapConfig.xml` and press **Ctrl+C** (or right-click `server-ldapConfig.xml` and click **Copy**).
- \_\_\_ 5. Go back to `<serverDir>` and press **Ctrl+V** (or right-click in the white space of the folder window, and click **Paste**).
- \_\_\_ 6. Rename the `server-ldapConfig.xml` file to: `server.xml`

### 5.2. Viewing the LDAP configuration information

In this section, you review the LDAP configuration information.

- \_\_\_ 1. Open the `server.xml` file with Atom.
- \_\_\_ 2. View the LDAP registry information.
  - \_\_\_ a. In Atom, go to **Find > Find in Buffer**.
  - \_\_\_ b. In the **Find in current buffer** field, enter: `ldapRegistry`



- \_\_\_ c. Click **Find**.

- d. The first search result should be in the <featureManager> section, where it is enclosed in a pair of tags:

```
<feature>ldapRegistry-3.0</feature>
```

The ldapRegistry-3.0 feature adds LDAP functionality to the sample server, and was added to the <featureManager> section.

- e. Click **Find** again.

- f. The second search result should be the <ldapRegistry> section.

```
<ldapRegistry
    id="myLDAP"
    realm="myLDAP"
    host="localhost" port="10389"
    baseDN="dc=loanserviceco,dc=com"
    bindDN="uid=admin,ou=system"
    bindPassword="{xor}LDo8LTor"
    ldapType="Custom">
    <customFilters
        userFilter="(&(uid=%v)(objectclass=inetOrgPerson))"
        groupFilter="(&(cn=%v)(objectclass=groupOfNames))"
        userIdMap="*:uid"
        groupIdMap="*:cn"
        groupMemberIdMap="groupOfNames:member">
    </customFilters>
</ldapRegistry>
```

This section contains the information that the server uses to connect to the LDAP registry that you set up in the previous section, such as:

- The LDAP registry ID
- The host and port
- The base distinguished name (DN)
- The bind DN and bind password



## Information

The bind password, which is secret, was encoded for inclusion in the `server.xml` file.

- The object classes for the user and group organizational units
3. Scroll up in the `server.xml` file to see the <basicRegistry> section.

The `server.xml` file also includes all of the user and group definitions from the basic registry, such as `rtsAdmin`. An administrative user who has both administration and

installation permissions is required to complete the process of importing LDAP users into Decision Center.

- \_\_\_ 4. View the security role of `rtsUser` in the Decision Center consoles.

Another step in configuring the `server.xml` file to use LDAP is to give all authenticated users the `rtsUser` role. The prepared server file includes the required security role definition.

- \_\_\_ a. In the **Find in current buffer** field, delete `ldapRegistry` and type: `all_auth`
  - \_\_\_ b. Click **Find**.
  - \_\_\_ c. The first search result should be in the section that is labeled with the comment:  
`<!-- ODM Enterprise Console (Teamserver) -->`
  - \_\_\_ d. Notice that the `<security-role>` definition for `rtsUser` is now:  
`<special-subject type="ALL_AUTHENTICATED_USERS" />`  

`<security-role name="rtsUser">
 <special-subject type="ALL_AUTHENTICATED_USERS" />
</security-role>`
  - \_\_\_ e. Click **Find** again to see the `<security-role name="rtsUser">` definition for the `<!-- Business console -->` section.  

It should be the same as the `rtsUser` definition in the Enterprise console.
- \_\_\_ 5. Close the `server.xml` file.

### 5.3. Checking Decision Center authentication

- \_\_\_ 1. Start the sample server.
- \_\_\_ 2. When the server start is complete, start the Decision Center Business console.
- \_\_\_ 3. Log in to the Business console by entering `Jane` in both the **Username** and **Password** fields, and clicking **Log in**.

Jane can successfully log in to the console and see the **Home**, **Library**, and **Work** tabs, but not the **Administration** tab. Currently, Jane has the `rtsUser` role. However, only users with the `rtsAdministrator` role can access the **Administration** tab.

- \_\_\_ 4. Click the **Library** tab, and then click **Loan Validation Service**.

- 5. On the **Releases** tab, click **New Releases** (the plus [+] icon) to create a release.

The screenshot shows the 'Decision Center' interface with the 'LIBRARY' tab selected. Under 'All Decision Services', the 'Loan Validation Service' is selected. The 'Releases' tab is active. In the 'New Release' section, there is a red box around the blue '+' icon. Below it, a card displays 'Initial Release' with a large '2', 'Created by Paul on Jan 23, 2017', 'Owner: Paul', and 'Due Date: Jan 23, 2017'. The status 'Complete' is shown in green.

- 6. In the “Create a Release” window, click the **Owner** list to see who is included.

The screenshot shows the 'Create a Release' dialog box. It asks 'Specify who will participate in this release'. The 'Owner:' field has a dropdown menu open with the following options: Abu, Bea, Paul, and rmtuser. A 'Create' button is at the bottom right.

Jane and John are not listed. With their current permissions, they cannot participate in decision service governance.



### Note

The decision governance framework is covered in [Exercise 18, "Working with the decision governance framework"](#).

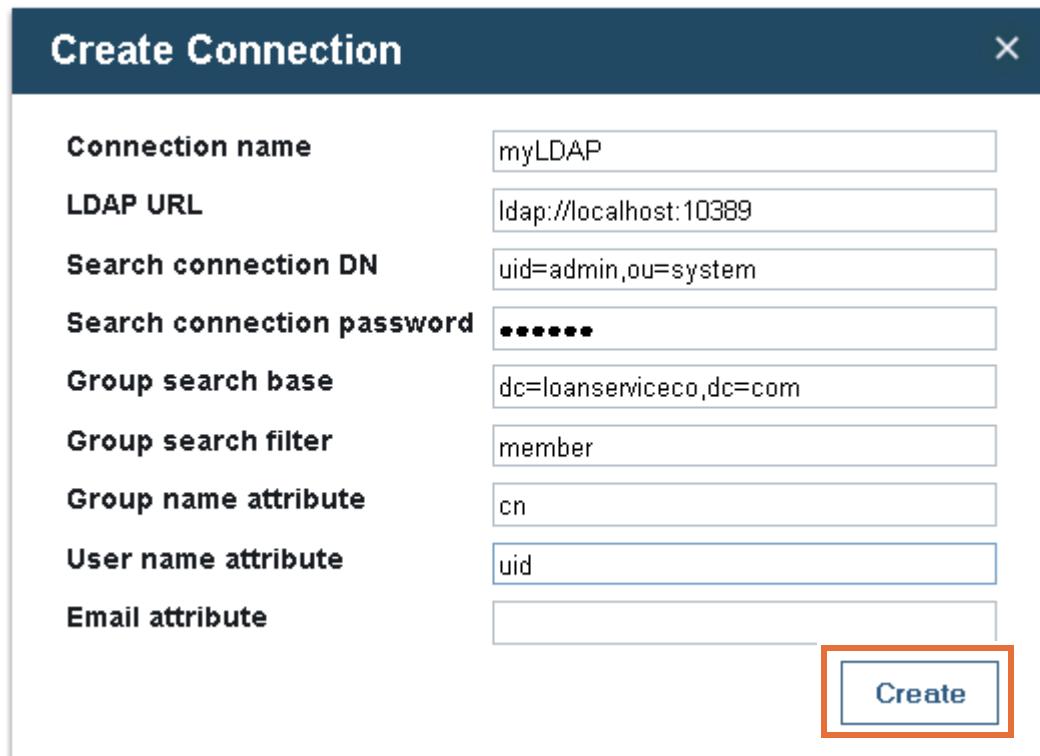
- 7. Close the “Create a Release” window and log out of the Business console.

## Section 6. Administering groups and users in the Business console

In this section, you add an LDAP connection into Decision Center and import its groups and users by using the Administration feature. After assigning the proper roles to the groups, the users can participate in the governance framework.

### 6.1. Adding an LDAP connection to the Business console

- 1. Log in to the Decision Center Business console as: rtsAdmin
- 2. Click the **Administration** tab.
- 3. On the **Connection Settings** subtab, click **New Connection** (the plus [+] sign).
- 4. In the Create Connection window, complete the LDAP connection information with these details:
  - **Connection name:** myLDAP
  - **LDAP URL:** ldap://localhost:10389
  - **Search connection DN:** uid=admin,ou=system
  - **Search connection password:** secret
  - **Group search base:** dc=loanserviceeco,dc=com
  - **Group search filter:** member
  - **Group name attribute:** cn
  - **User name attribute:** uid
- 5. Click **Create**.



The screenshot shows the 'Create Connection' dialog box. It contains fields for various LDAP connection parameters. The 'Create' button at the bottom right is highlighted with a red border.

Parameter	Value
<b>Connection name</b>	myLDAP
<b>LDAP URL</b>	ldap://localhost:10389
<b>Search connection DN</b>	uid=admin,ou=system
<b>Search connection password</b>	*****
<b>Group search base</b>	dc=loanserviceeco,dc=com
<b>Group search filter</b>	member
<b>Group name attribute</b>	cn
<b>User name attribute</b>	uid
<b>Email attribute</b>	

You see the connection in the table.

Connection Name	LDAP URL	Status
<input type="checkbox"/> myLDAP	ldap://localhost:10389	Disabled



### Troubleshooting

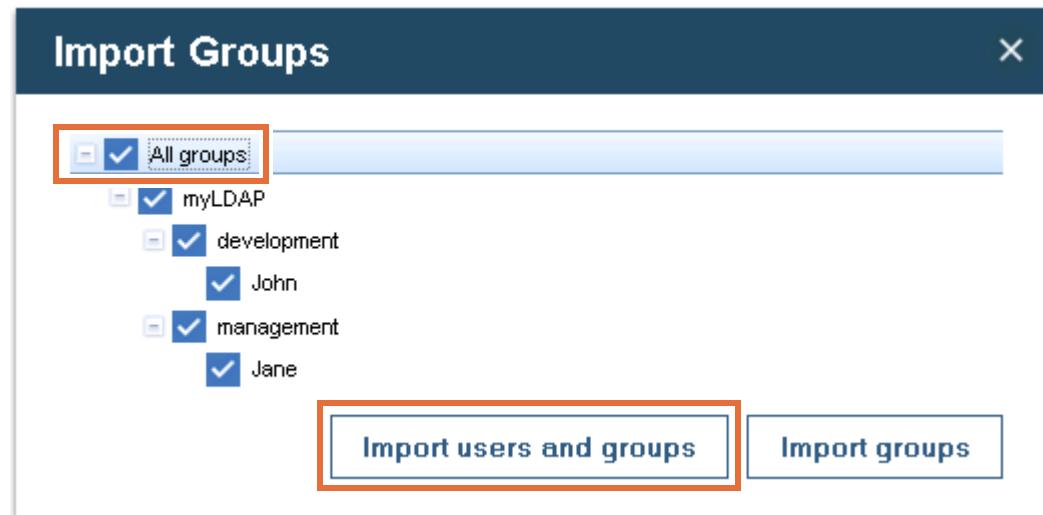
The **Status** column might read `Disabled` when you first create the connection. The status message should update to `Enabled` after you import groups and users.

## 6.2. Importing LDAP groups and users

- 1. Go to the **Groups** tab.

- 2. On the **Groups** tab, click the **Import Groups From Ldap** icon.

3. In the Import Groups window, select All groups and click Import users and groups.



Two selected groups, development and management, are now imported.

Group	Members	Permissions
development	1 member	None
management	1 member	None

4. Click the **Users** tab to see that the Jane and John users were also imported with the groups.

User	Groups	Last Logged In
Abu	rtsUser	-
Bea	rtsUser	May 3, 2017, 3:14:58 PM
Jane	management	-
John	development	-
Paul	rtsAdministrator	May 3, 2017, 4:11:29 PM

### 6.3. Assigning permissions to groups

In this section, you assign Decision Center user permissions to the groups that you imported.

- 1. Click the **Groups** tab.
- 2. *Optional.* Sort the group list again so that the most recent groups are listed first by clicking the **Last Updated** column heading.
- 3. Hover the mouse pointer over the **development** group row and click the **Edit** icon.

Group	Members	Permissions
development	1 member	None
management	1 member	None

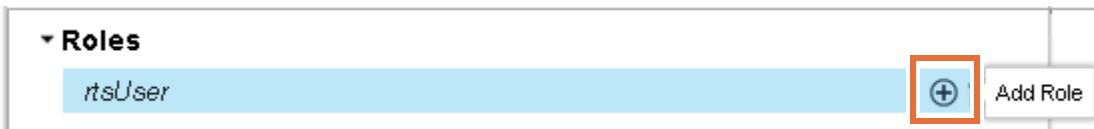
- 4. Edit the development group properties.
  - a. In the **Roles** section, make sure that the **rtsUser** role is selected.
  - b. In the **Permissions** section, click **None** to activate the menu, and then click **Full Authoring**.

- c. Click **Done**.

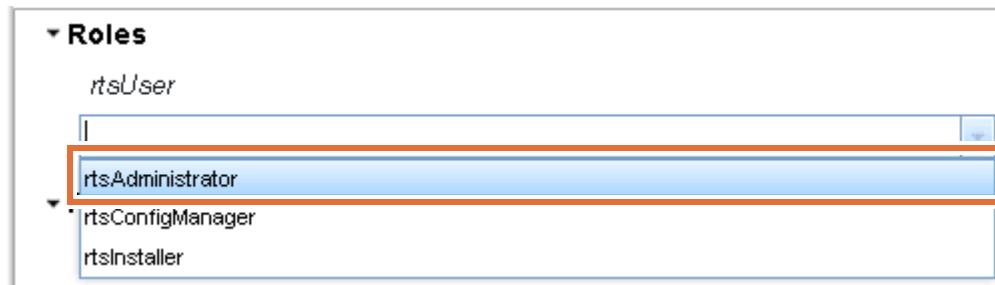
- \_\_\_ d. The **Permissions** column for the development group now reads: Full Authoring

Group	Members	Permissions
development	1 member	Full Authoring

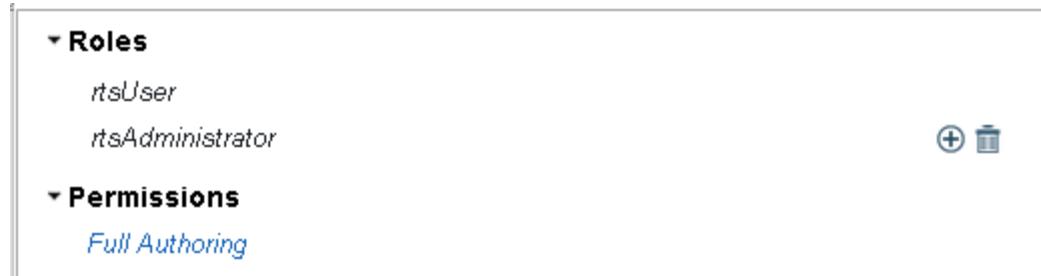
- \_\_\_ 5. Edit the management group by adding the `rtsAdministrator` role and assigning full authoring permissions.
- \_\_\_ a. Hover the mouse pointer over the **management** group name, and click **Edit** (the pencil icon).
  - \_\_\_ b. In the Edit Group window, in the Roles section, hover the mouse pointer over `rtsUser` and click **Add Role** (the plus [+]) icon).



- \_\_\_ c. Click the menu, and select **rtsAdministrator**.



- \_\_\_ d. In the Permissions section, click **None**, and then select **Full Authoring**.



- \_\_\_ e. Click **Done**.

- \_\_\_ 6. Log out of the Business console as `rtsAdmin`.

## 6.4. Verifying the role assignments and permissions

- \_\_\_ 1. Log in to the Business console as Jane.
- \_\_\_ a. Enter `Jane` in the **Username** and **Password** fields.
  - \_\_\_ b. Click **Log in**.

Jane belongs to the management group, which you assigned the `rtsAdministrator` role. Jane can now work on the **Administration** tab.

- \_\_\_ 2. Click the **Library** tab and click **Loan Validation Service**.
- \_\_\_ 3. Create a release for Loan Validation Service that is called Winter Release.
  - \_\_\_ a. On the **Releases** tab, click the **New Release** (the plus [+]) icon.
  - \_\_\_ b. In the “Create a Release” window, in the **Give your release a name** field, enter: Winter Release

The Owner is set to **Jane** by default.
- \_\_\_ c. Click the **Owner** menu to see that both Jane and John are listed among the users.

Specify who will participate in this release:

Owner:	<input type="text" value="Jane"/>
Approvers:	<input type="text" value="Abu"/> <input type="text" value="Bea"/> <input style="outline: 2px solid orange;" type="text" value="Jane"/> <input type="text" value="John"/>

- \_\_\_ d. Click **Create**.
- \_\_\_ 4. On the Winter Release **Activities** tab, create a change activity that is called **Minimum Age Update**.
  - \_\_\_ a. Click **New Activity** (the plus [+]) icon), and then select **New Change Activity**.



The screenshot shows the IBM Decision Center interface with the 'Activities' tab selected. At the top, there are tabs for 'Activities', 'Decision Artifacts', 'Queries', 'Tests', and 'Simulation'. Below the tabs, there are filters for 'All', 'Due Date', 'Name', 'Type', and a 'Filter:' button. A large red box highlights the 'New Activity' button, which is a plus sign icon with a dropdown arrow. Underneath it, another red box highlights the 'New Change Activity' option in a dropdown menu. To the right, a message says 'There are no activities.'.

- \_\_\_ b. Set the activity name to: **Minimum Age Update**
- \_\_\_ c. For the **Owner** and **Approver** fields, select **Jane**.
- \_\_\_ d. For the **Author** field, select **John**.

- \_\_ e. Click **Create**.

Specify who will participate in this activity:

Owner:	Jane
Approvers:	Jane
Authors:	John

**Create**

- \_\_ 5. Log out of the Business console as Jane.
- \_\_ 6. Log in to Business console as John with `John` for the user name and password.
- \_\_ 7. Click the **Work** tab.

Decision Center    HOME    LIBRARY    **WORK**

The following items require your attention.

All	Due Date	Release	Type
▼ Due Later (1)			
<span style="color: blue;">★</span> <b>Minimum Age Update</b> In Loan Validation Service > Winter Release			

- \_\_ 8. The **Minimum Age Update** change activity is listed in John's work items.  
John can now participate in governance work.
- \_\_ 9. Log out of the Business console and close it.
- \_\_ 10. Close Apache Directory Studio and confirm that you want to exit.

## End of exercise

## Exercise review and wrap-up

This exercise demonstrated how to work with an LDAP server and how to set up groups and users on the application server. You also saw how to manage user access in Decision Center.

# Exercise 18. Working with the decision governance framework

## Estimated time

01:30

## Overview

This exercise continues the focus on rule and decision service management. In this exercise, you work with the decision governance framework in Decision Center by simulating a collaborative workflow that includes working with releases, change activities, and other tasks in a governed decision service.

## Objectives

After completing this exercise, you should be able to:

- Work with releases
- Work with change and validation activities
- Deploy decision service releases

## Introduction

In this exercise, you take on different roles to perform a series of actions in the decision governance framework. Some of these actions are manual, and some are automatic.

You start by preparing the lab environment for the decision service, importing the decision service into Rule Designer, and publishing it to Decision Center.

Next, you use the decision governance framework in the Business console by working as two different users: Paul and Bea. You previously worked as Paul and Bea in [Exercise 6, "Exploring the Decision Center Business console"](#).

Paul and Bea do different role-based tasks in the decision governance framework:

- Paul is a manager who creates releases and reviews changes to rules. He also approves and deploys releases. In this exercise, he also works on validation activities.
- Bea is a rule author who implements the changes.

By completing tasks in the governance framework workflow, you can see how the decision governance framework manages changes to decision service releases.

The exercise includes these sections:

- [Section 1, "Preparing the lab environment"](#)
- [Section 2, "Working with the decision service"](#)

- [Section 3, "Creating a change activity"](#)
- [Section 4, "Creating a validation activity"](#)
- [Section 5, "Updating the rule in the change activity"](#)
- [Section 6, "Reviewing and approving the change activity"](#)
- [Section 7, "Working with the validation activity"](#)
- [Section 8, "Approving the validation activity, completing the release, and deploying the release"](#)

## Requirements

This exercise requires some lab environment setup, including publishing the decision service to Decision Center.

This exercise uses Rule Designer and the Business console. All decision governance framework tasks are done in the Business console.

A populated Excel scenario file is provided for this exercise.

## Section 1. Preparing the lab environment

In this section, you use Rule Designer to reset the database. You then import and publish the carinsurance-rules decision service.

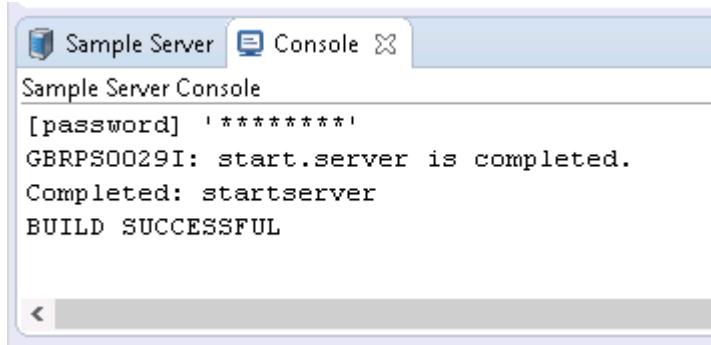
### 1.1. Resetting the database

- 1. *Optional.* Save the database in case you want to return to your previous work.
  - a. Stop the sample server by going to **Start > Stop Sample Server**.
  - b. Go to the following directory:  
C:\IBM\wlp\usr\servers\odm8900\data\h2
  - c. Right-click the **rtsdb.h2.db** file and click **Copy**.
  - d. Right-click the white space of the directory, and click **Paste**.
  - e. Right-click the **rtsdb.h2 - Copy.db** file that you created, and click **Rename**.
  - f. Rename the file so that you can identify it as the backup database file, such as:  
**rtsdb.h2-admin.db**
  - g. Restart the sample server (by going to **Start > Start Sample Server**).
- 2. Open a new Rule Designer workspace.
  - a. If needed, start Rule Designer by going to **Start > Rule Designer**.  
If Rule Designer is already running, go to **File > Switch Workspace > Other**.
  - b. When prompted for a workspace name, enter a name for this exercise, such as:  
C:\labfiles\workspaces\governance
  - c. After the new workspace opens, close the **Welcome** tab.
- 3. Open the Samples Console by clicking the **Open Perspective** icon, selecting **Samples Console**, and then clicking **OK**.
- 4. In the Sample Server pane, click **Restore the sample server**.



The Samples Console switches to the Console view, which shows various messages as the sample server is stopped, restored, and restarted.

- \_\_\_ 5. Wait until you see the `BUILD SUCCESSFUL` message in the console.



The screenshot shows a window titled "Sample Server Console". The title bar has tabs for "Sample Server" and "Console". The main area displays the following text:  
[password] \*\*\*\*\*  
GBRPS0029I: start.server is completed.  
Completed: startserver  
BUILD SUCCESSFUL

## 1.2. Importing the carinsurance-rules decision service

- \_\_\_ 1. Import the start project that is provided for this exercise by using the Samples Console.
- \_\_\_ a. In the **Samples and Tutorials** tab of the Samples Console, expand **Rule Designer > Training > Ex 18: Working with the decision governance framework**.
- \_\_\_ b. Under **start**, click **Import projects**.
- \_\_\_ c. When the workspace finishes building, close the **Help** pane by clicking the **X**.

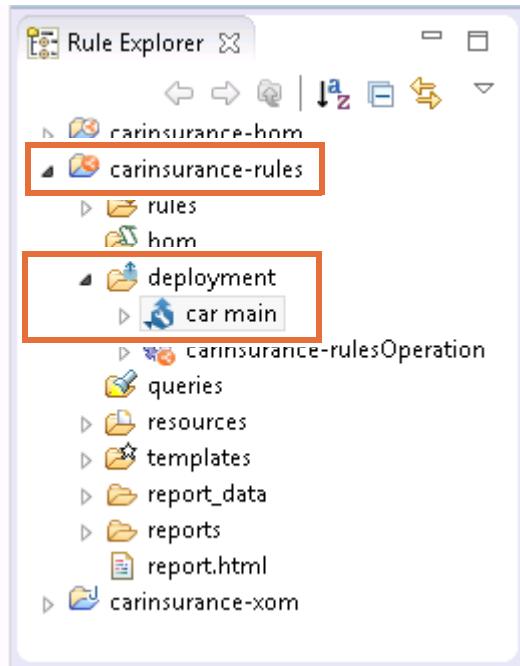


### Information

In the next steps, you add server settings for the deployment configuration, which is typically a development task. However, by doing these steps, you can see where the decision service XOM is located, which is the **resdb** server in this exercise.

- 
- \_\_\_ 2. Open the `car main` deployment configuration.
- \_\_\_ a. In the Rule Explorer, expand **carinsurance-rules > deployment**.

- \_\_ b. Double-click **car main**.



The Deployment Overview window for the **car main** deployment configuration opens.

- \_\_ 3. Click the **Target Servers** tab to see the deployment location settings.

The Deployment Overview window for the **car main** deployment configuration is shown. The **Target Servers** tab is selected and highlighted with a red box at the bottom of the screen.

**General**

Name: car main  
Type:  Production  Nonproduction Deploy the XOM:  Yes  No

Description:

**Decision Operations**  
Define decision operations deployment:  
[carinsurance-rulesOp](#)

**RuleApp**  
Configure the RuleApp to be created upon deployment.

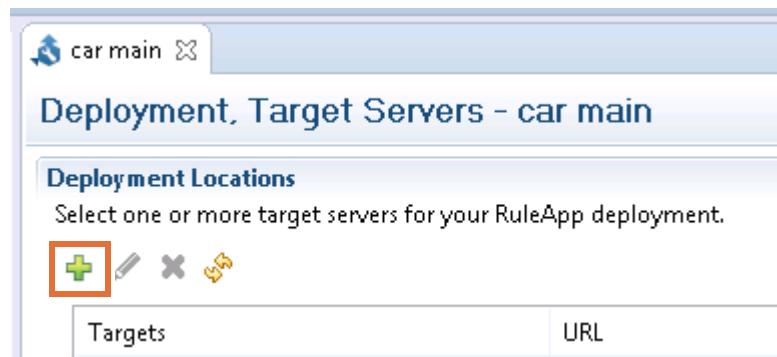
RuleApp Name: **car\_main**  
RuleApp Base Version: **1.0**

**Target Servers**  
Define target servers for RuleApp deployment:  
[Define target servers](#)

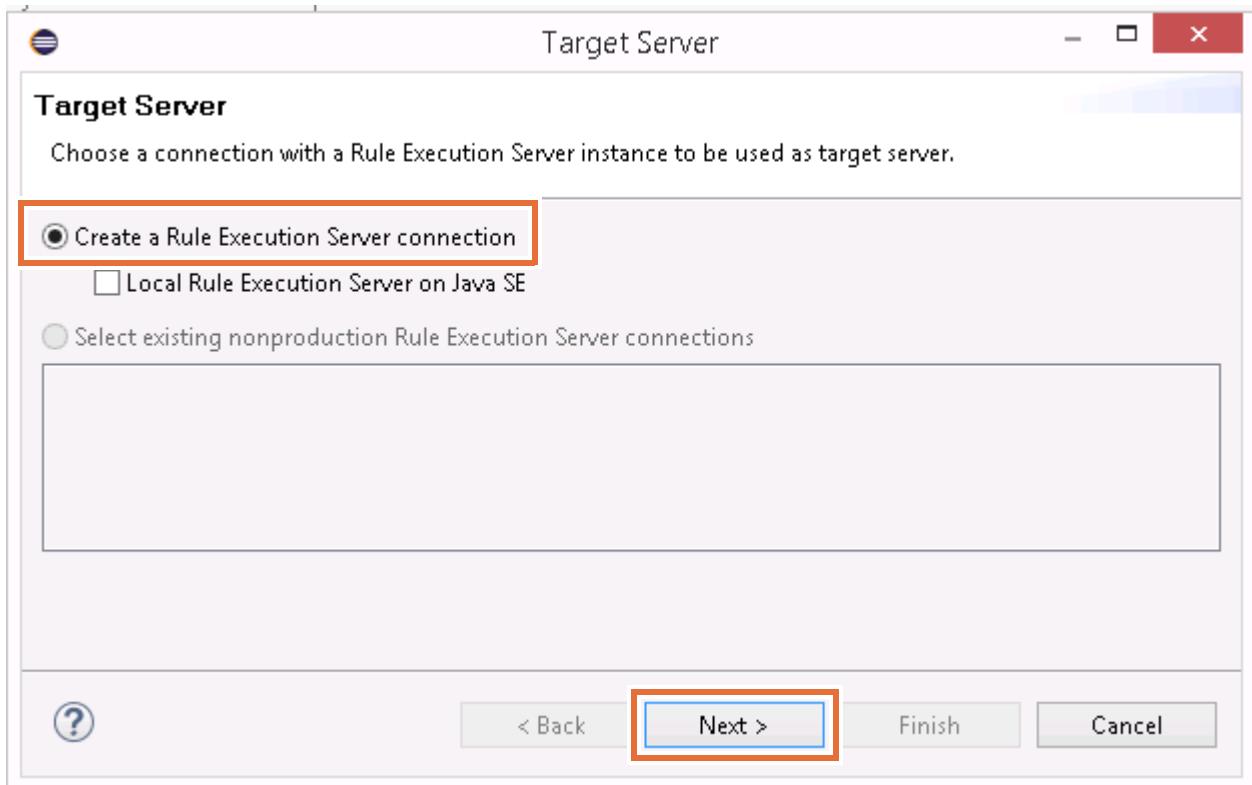
Overview Decision Operation **Target Servers** Source

— 4. Update the server settings for the `car main` deployment configuration.

— a. Click **Add target server** (the plus sign).

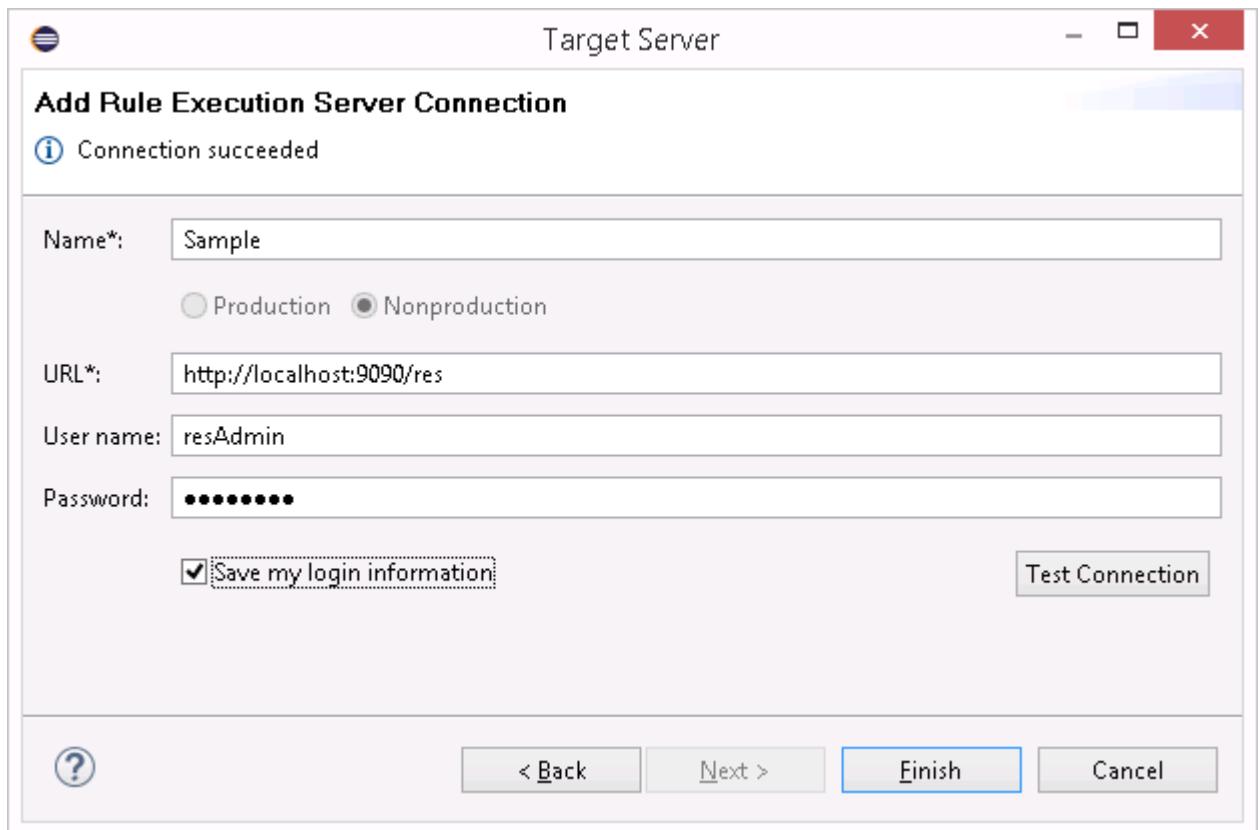


— b. In the Target Server window, make sure that **Create a Rule Execution Server connection** is selected, and click **Next**.



— c. In the Add Rule Execution Server Connection window, enter the following values:

- **Name:** Sample
- **URL:** `http://localhost:9090/res`  
If your lab environment uses a different port for `localhost`, use it here.
- **User name:** resAdmin
- **Password:** resAdmin
- Select **Save my login information**

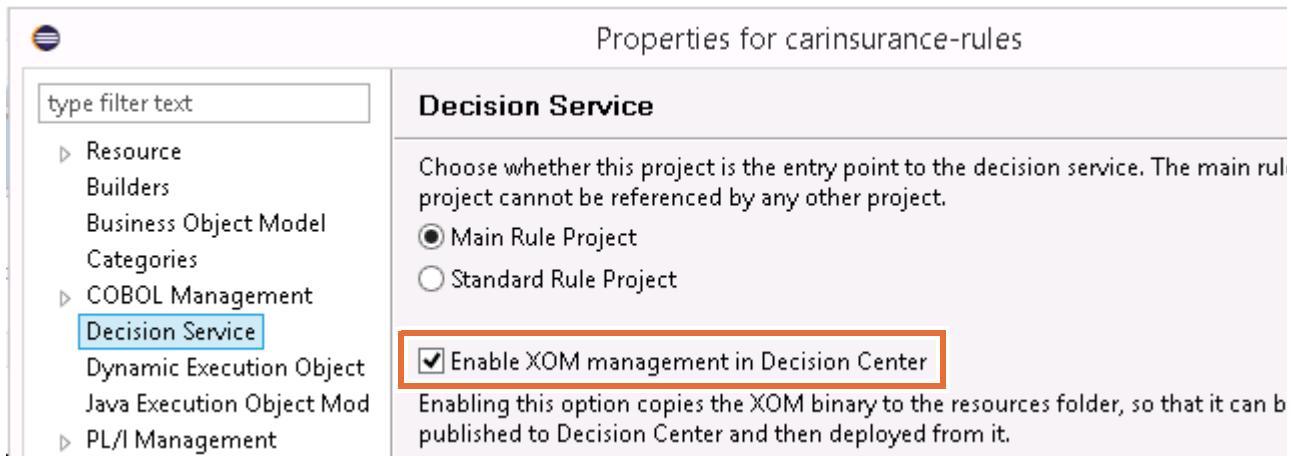


- \_\_\_ d. Click **Test Connection**, and when the connection is established, click **Finish**.
- \_\_\_ 5. Save your work.

### 1.3. Publishing from Rule Designer to Decision Center

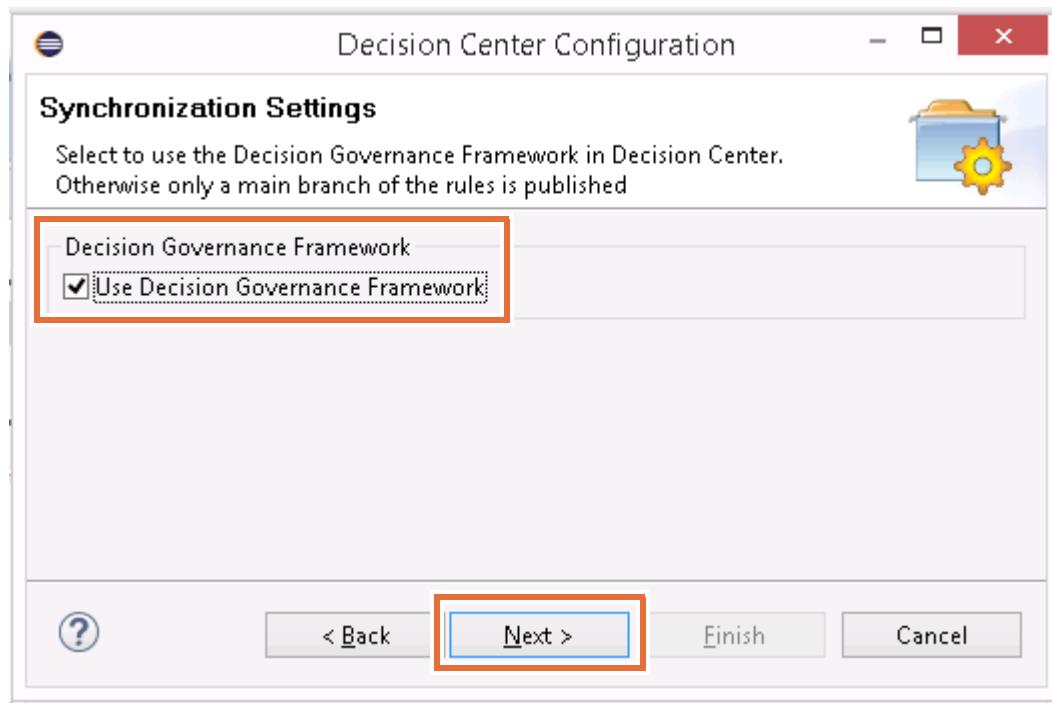
- \_\_\_ 1. Confirm that XOM management for this decision service is enabled.  
With XOM management, execution artifacts are accessible to the rule projects after they are published to Decision Center.
- \_\_\_ a. In the Rule Explorer pane, right-click **carinsurance-rules** and click **Properties**.

- \_\_\_ b. In the Properties window, select **Decision Service** from the list and make sure that **Enable XOM management in Decision Center** is selected.



- \_\_\_ c. Click **OK** to close the Properties window.
- \_\_\_ d. Expand **carinsurance-rules > resources > xom-libraries**, and notice the **carinsurance-xom.zip** file, which contains the execution artifacts.
2. Publish the **carinsurance** decision service to Decision Center.
- \_\_\_ a. In the Rule Explorer pane, right-click **carinsurance-rules** and click **Decision Center > Connect**.
- \_\_\_ b. In the Decision Center Configuration window, enter the following values:
- **URL:** `http://localhost:9090/teamserver`  
If your computer lab environment uses a different port, use it here.
  - **User name:** `rtsAdmin`
  - **Password:** `rtsAdmin`
  - You can leave the **Data source** field blank.
- \_\_\_ c. Click **Connect**.
- \_\_\_ d. After the connection is established, click **Next**.

- e. In the Synchronization Settings window, select **Use Decision Governance Framework**, and click **Next**.



- f. In the Decision Service Dependent Projects window, click **Finish**.



### Note

All dependent projects must be published to Decision Center to create an initial release. In subsequent synchronizations, you can publish individual projects.



### Troubleshooting

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

- 3. Click **OK** when you see the Synchronize Complete window, which shows the following message:  
Synchronizing: No changes found.
- 4. Click **No** when you are prompted to switch to the Team Synchronizing perspective.  
Because you published a decision service that is not in Decision Center, you should not have synchronization errors.
- 5. Close Rule Designer, and confirm that you want to exit.

## Section 2. Working with the decision service

In this section, you log in to the Business console as Paul. You then work with the decision service by creating a release, a change activity, and a validation activity.



### Requirements

Paul received a request from the underwriting department to change the car insurance eligibility rules. The rule was initially defined to have a minimum age of 16, and the minimum age must change to 18.

Since it is the start of a new release cycle, Paul also creates a release to manage changes to the decision service.

### 2.1. Logging in to the Business console as Paul

- 1. Start the Business console and log in as Paul.
  - a. Open the Business console by clicking **Start**, and then clicking the **Decision Center Business console** shortcut.



### Information

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

`http://localhost:9090/decisioncenter`

If your computer lab environment uses a different port, use it here.



### Note

If you see the following Privacy message about cookies, 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**

- b. Enter **Paul** in the **Username** and **Password** fields, and click **Log in**.

## 2.2. Exploring the carinsurance decision service and creating a release

In this part of the exercise, you look at different parts of the decision service to see how it is structured. You also create a release to manage changes to the decision service.

- 1. Click the **Library** tab to open the list of decision services.

The screenshot shows the Decision Center interface with the 'LIBRARY' tab selected. The main area displays a list of decision services:

- carinsurance-rules** (Created by rtsAdmin on May 24, 2017)
- bomdomainpopulate-rules** (Created by Paul on Jan 23, 2017)
- Loan Validation Service** (Created by Paul on Jan 23, 2017)

Each service entry includes a small icon and a green 'NEW' badge in the top right corner.

- 2. Open the carinsurance-rules decision service by clicking the **carinsurance-rules** decision service link.

You see two tabs: one that is called **Releases**, and one that is called **Branches**.

The screenshot shows the Decision Center interface. At the top, there are tabs for HOME, LIBRARY (selected), WORK, and ADMINISTRATION. Below that, it shows 'All Decision Services > carinsurance-rules'. Under the 'carinsurance-rules' service, there are two tabs: 'Releases' (selected) and 'Branches'. There is a search bar with filters for 'All', 'Due Date', and 'Name', and a 'Filter:' input field. Below the tabs, there is a large '+' button. A card for the 'Initial Release' is shown, indicating it is 'Complete'. The card also shows it was created by 'rtsAdmin' on 'May 24, 2017' and has a 'Due Date: May 24, 2017'.

- On the **Releases** tab, you see the Initial Release for the `carinsurance-rules` decision service.
- On the **Branches** tab, you can access the main branch. However, managing rules through branches is not covered in this exercise.



### Information

A release is a branch of the rules that are contained in the decision service, but with special characteristics. Recall that releases cannot be edited directly: changes to a decision service can happen only within the context of change activities.

- 3. Create a release that is called **Training Release** to manage the change to the decision service, and assign Paul as the approver for the release.
- a. On the **Releases** tab, click **New Release** (the plus [+] sign icon).

The screenshot shows the 'Releases' tab selected in a software interface. At the top, there are tabs for 'Releases' and 'Branches'. Below the tabs are filters for 'Due Date' and 'Name', and a 'Filter:' input field. A large orange box highlights the 'New Release' button, which is a blue square with a white plus sign (+). To the right of the button, the text 'New Release' is visible. Below the button, there is a list of releases. The first release listed is 'Initial Release', which is marked as 'Complete'. It has a large blue number '2' next to it, indicating two items. Below the release name, it says 'Created by rtsAdmin on May 24, 2017'. Underneath the release list, it shows 'Owner: rtsAdmin' and 'Due Date: May 24, 2017'.

- b. In the **Give your release a name** field, enter: **Training Release**
- c. Leave the value in the **Base it on this completed release** field at **Initial Release**.
- d. (*Optional*) You can enter a goal for the release for documentation purposes.
- e. (*Optional*) You can enter a different due date for the release.

The default value is two weeks from the date that you create the release.

The screenshot shows the 'Create a Release' dialog box. At the top, it says 'Create a Release' and has a close button ('X'). The first section asks 'Give your release a name:' with a text input field containing 'Training Release'. The second section asks 'Base it on this completed release:' with a dropdown menu showing 'Initial Release'. The third section asks 'Specify goals for this release:' with a rich text editor toolbar above an empty text area. The fourth section asks 'Specify a due date:' with a date input field showing 'Jun 7, 2017' and a calendar icon.

- f. Scroll down in the “Create a Release” window, and click the menu for the **Approvers** field, and select **Paul** to be the release approver.

Specify who will participate in this release:

Owner: Paul

Approvers:

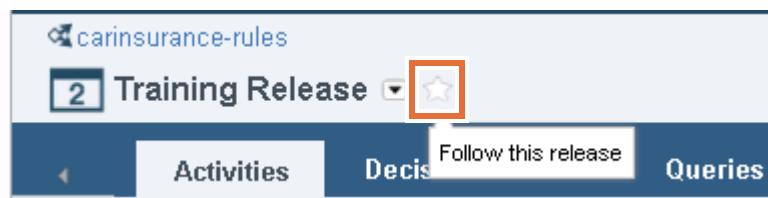
- Abu
- Bea
- Paul**
- rmtuser

Create

- g. Click **Create**.

After the release is created, it opens in the Release view.

- 4. Follow Training Release so that you are notified of any changes to the release.
- a. In the Training Release title bar, click **Follow this release** (the star icon).



The star turns yellow, which indicates that you are now following the release.



### Reminder

When you follow a decision service, its releases, or its decision artifacts, you are notified of any changes or updates. You can toggle following and unfollowing by clicking the icon.

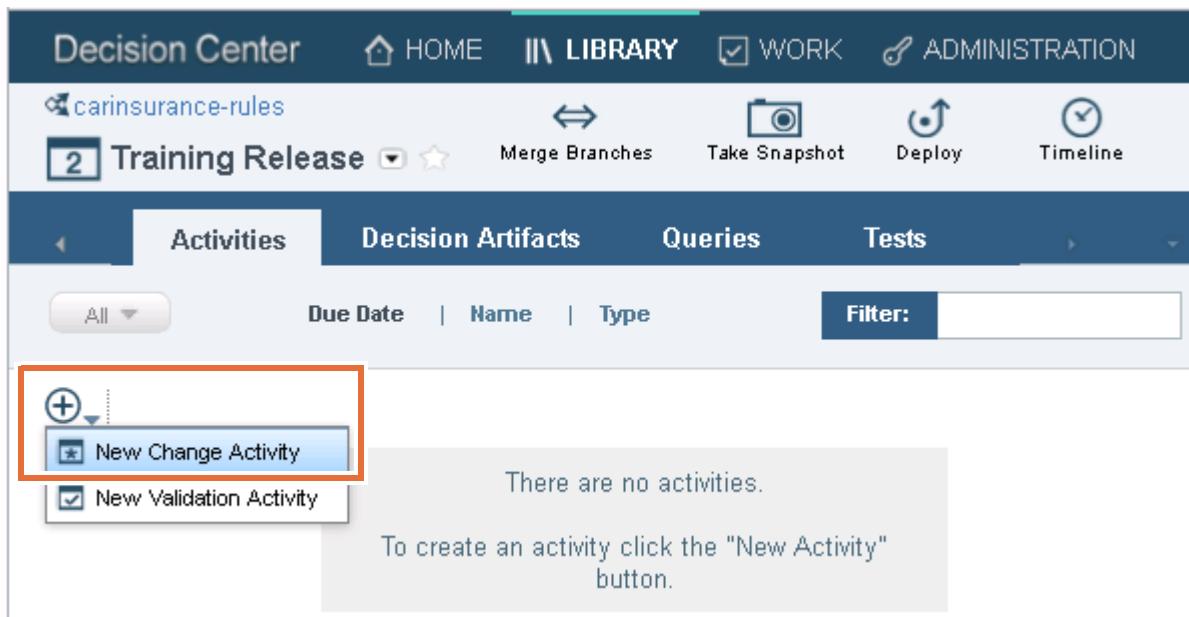
## Section 3. Creating a change activity

In this part of the exercise, you create a change activity.

### 3.1. Creating a change activity for the Driver age rule

Within a release, a change activity contains and manages the creation and modification of rules. A release can contain multiple change activities.

- 1. Create a change activity in the Training Release.
  - a. In the Training Release, make sure that you are on the **Activities** tab.
  - b. In the main Activities pane, click **New Activity** (the plus [+] sign icon) > **Change Activity**.



- 2. Specify the name, goals, approver, and author for the new change activity.
  - a. For the name of the activity, enter: `Update minimum driver age`
  - b. Enter the goals for the change activity:  
`Change the minimum age of the driver from 16 to 18.`
  - c. (Optional) Specify a due date.  
The default due date is the same due date as the release.
  - d. Leave `Paul` as the owner.  
As the user who created the change activity, Paul is indicated as the change activity owner by default.
  - e. Select `Paul` as the approver.
  - f. Select `Bea` as the author.

\_\_ g. Click **Create**.

**Create a Change Activity**

Give your activity a name:  
Update minimum driver age

Specify goals for this new activity:  
**B I U | ≡ ≡ ≡ ≡ | A▼ | ≡ ≡ ≡ ≡ | ↕▼ ↕▼**  
Change the minimum age of the driver from 16 to 18.

Specify a due date:  
Jun 7, 2017

Specify who will participate in this activity:

Owner:

Approvers:

Authors:

**Create**

On the **Change Activity** tab, the status is set to **In Progress**. The status for Bea, who is the rule author, is set to **Working**.

The screenshot shows the 'Change Activity' tab of a software interface. At the top, there are two tabs: 'Change Activity' (selected) and 'Stream'. Below the tabs, there's a profile picture of a man and the text 'Created by Paul May 24, 2017'. Under the 'Goals' section, there's a goal: 'Change the minimum age of the driver from 16 to 18.' In the 'Release' section, it says 'Training Release'. The 'Decision Serv...' section shows 'carinsurance-rules'. Under 'Due Date', it lists 'Jun 7, 2017'. The 'Status' section is highlighted with a red border and shows 'In Progress'. The 'Owner' section shows 'Paul'. Under the 'Approvers' section, 'Paul' is listed with 'Not Reviewed'. The 'Authors' section is also highlighted with a red border and shows 'Bea' with the status 'Working'.

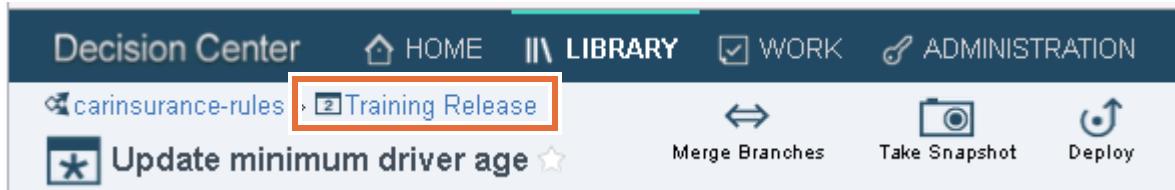
Status	In Progress
Owner	Paul
Approvers	Paul Not Reviewed
Authors	Bea Working

A snapshot is also automatically created to capture the initial state of the change activity.

## Section 4. Creating a validation activity

In this task, you create a validation activity to validate the eligibility package.

- 1. Create a validation activity for the Training release.
- a. Click the **Training Release** breadcrumb, and make sure that you are on the **Activities** tab.



- b. In the main Activities pane, click **New Activity** (the plus sign) > **Validation Activity**.



- 2. Specify the name, goals, due date, approver, and tester for the validation activity.

- a. For the name of the activity, enter: Validate eligibility
- b. Enter the goals for the validation activity:  
Validate the rules in the eligibility package.
- c. *Optional.* Specify a due date.  
The default date is the same as the release due date.
- d. Leave Paul as the owner.  
As the creator of the validation activity, Paul is listed as the owner of the validation activity by default.
- e. Select Paul as the approver.
- f. Select Paul as the tester.

**Note**

In this simplified scenario, Paul is both the approver and the tester. However, in your organization, the tester and the approver might be different users.

- 
- \_\_\_ g. Click **Create**.  
On the **Validation Activity** tab, the validation activity status is set to **In Progress**. The status for Paul, who is the assigned tester, is set to **Working**.
  - \_\_\_ 3. Log out as Paul.
    - \_\_\_ a. Click **Paul > Log out**.

## Section 5. Updating the rule in the change activity

In this task, you log in as Bea to change the rule within the change activity that Paul created.

### 5.1. Logging in as Bea and finding change activity work

- 1. Log in to the Business console as Bea.
- a. Enter Bea into the **Username** and **Password** fields.
- b. Click **Log in**.
- 2. Find the change activity work that Paul assigned.
- a. Click **Work**.



- b. On the **Work** tab, in the **Due Later** section, click **Update minimum driver age**.

#### Information

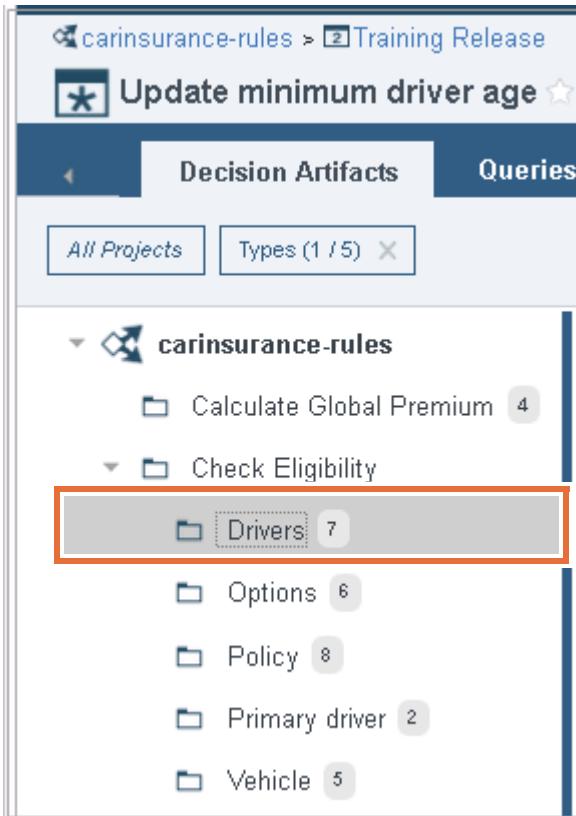
You can ignore the Adjust Scoring change activity.

Change Activity	Due Date	Owner
Update minimum driver age	Jun 7, 2017	Paul
Adjust Scoring	Nov 23, 2017	Paul

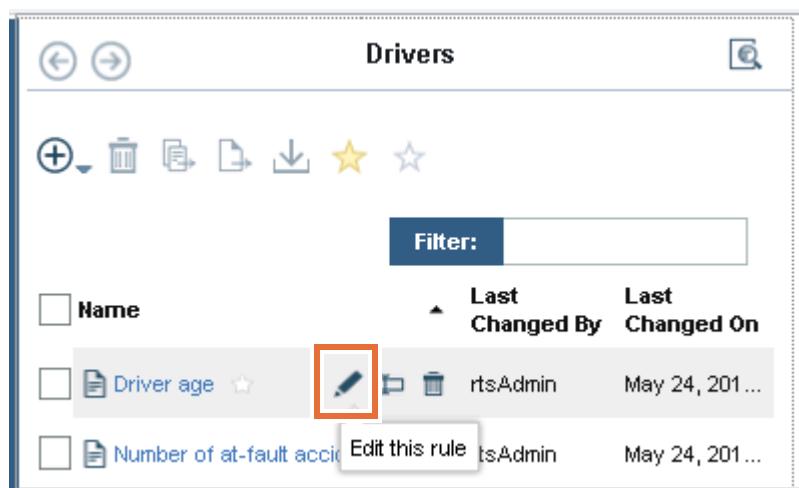
The change activity view opens, and lists the rules that are in the Training Release. You can now edit these rules.

## 5.2. Updating the rule

- 1. Update the rule to reflect the change in age from 16 to 18.
  - a. Make sure that you are on the **Decision Artifacts** tab.
  - b. In the left explorer pane, expand **Check Eligibility**, and click the **Drivers** folder.



- c. In the main Drivers pane, hover the mouse pointer over the `Driver age` rule and click **Edit this rule** (the pencil icon).



- d. In the condition statement, change the minimum age from **16** to **18**, and click **Save**.

The screenshot shows a page titled 'Training Release > Update minimum driver age'. At the top right are three buttons: 'Details', 'Save' (which is highlighted with a red box), and 'Cancel'. Below the buttons, there is a message in green text: 'er in the drivers of 'the request';'. A red box highlights a validation error message: 'The age (in years) of driver at the start date of 'the policy' is between 18 and 80'. Another message in green text follows: 'on because "At least one driver does not meet the age constraints"';'.

- \_\_\_ e. In the Create New Version window, enter a description of the change, and click **Create New Version**:

Changed minimum age from 16 to 18.



- \_\_\_ 2. Click the **Update minimum driver age** breadcrumb to return to the main change activity page.

The screenshot shows a navigation bar with tabs 'Decision Center', 'HOME', 'LIBRARY' (selected), and 'WORK'. Below the tabs, the breadcrumb path is 'carinsurance-rules > Training Release > Update minimum driver age'. The last item in the path is highlighted with a red box. Below the breadcrumb is the file name 'Driver age (v4.0)' with a star and info icon.

### 5.3. Changing the working status of the change activity

After you update the rule and validate the change, you can mark your work as done.

- 1. Change the status of the change activity to **Finish working**.
- a. On the **Change Activity** tab, in the Authors section, click **Working**.

The screenshot shows the 'Change Activity' tab with the 'Stream' tab selected. In the 'Authors' section, under the 'Status' column, the word 'Working' is highlighted with a red box. Other status options like 'Not Reviewed' and 'Finished' are also visible in the dropdown menu.

Author	Status
Bea	Working

- b. Select **Finish working**.

The screenshot shows the 'Change Activity' tab with the 'Stream' tab selected. In the 'Authors' section, under the 'Status' column, the word 'Working' is still present, but the status has been updated to 'Finished' in the dropdown menu, which is highlighted with a red box.

Author	Status
Bea	Finished

The status for Bea changes to **Finished**.

- 2. Sign out of the Business console as Bea.

## Section 6. Reviewing and approving the change activity

In this task, you log in as Paul to review and approve the change that Bea made to the rule.

### 6.1. Reviewing the change activity

- \_\_\_ 1. Sign in to the Business console as Paul.
- \_\_\_ 2. Click the **Home** tab.
- \_\_\_ 3. Review the change activity by using the Stream feature.
- \_\_\_ a. On the **Home** tab, click **Stream**.

Because you are following the release, you see can Bea's change activity updates.

- \_\_\_ b. In Bea's entry, click **Update minimum driver age**.
- \_\_\_ c. Make sure that you are on the **Decision Artifacts** tab.
- \_\_\_ d. If needed, in the Explorer, make sure that the **Check Eligibility > Drivers** folder is selected, and that the `Driver age` rule is open in the main pane.

You see that the age was changed correctly to 18.

### 6.2. Approving the change activity

In this part of the exercise, you approve the change that Bea made to the **Update minimum driver age** change activity.

- \_\_\_ 1. On the **Change Activity** tab, click the **In Progress** status, and change it to **Proceed to approval**.

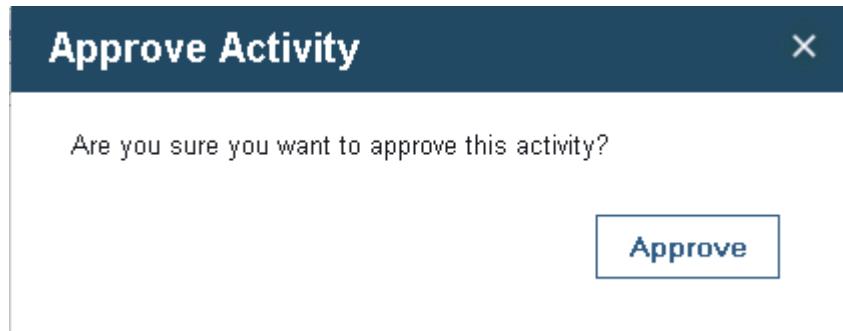
The activity status now reads: **Ready for Approval**

In this simplified example, Paul is both the reviewer and the approver, so you can now approve the change.

- 2. On the **Change Activity** tab under Approvers, click the **Not Reviewed** status, and change it to **Approve changes**.



- 3. In the Approve Activity window, click **Approve**.



The change activity now has a status of **Complete**.

A snapshot of the release is also automatically created. The rule that was modified during the change activity is merged into the release.

## Section 7. Working with the validation activity

### 7.1. Creating and running a test suite

In this task, you complete validation activity. You create and run a test suite for the eligibility package.

- 1. Open the validation activity.

  - a. Click **Work**.
  - b. In the “Due Later” section, click **Validate eligibility**.



#### Note

You can ignore the Spring Validation activity.

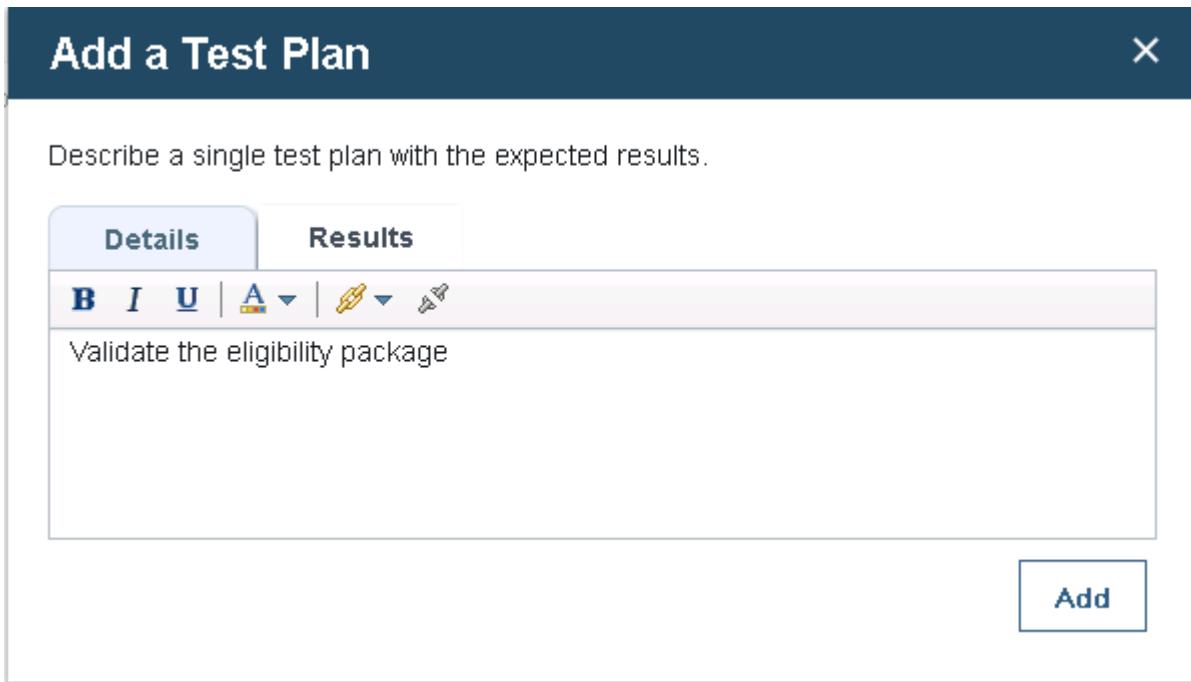
The validation activity opens to the Test Plans page.

- 2. Create a test plan for the validation activity.

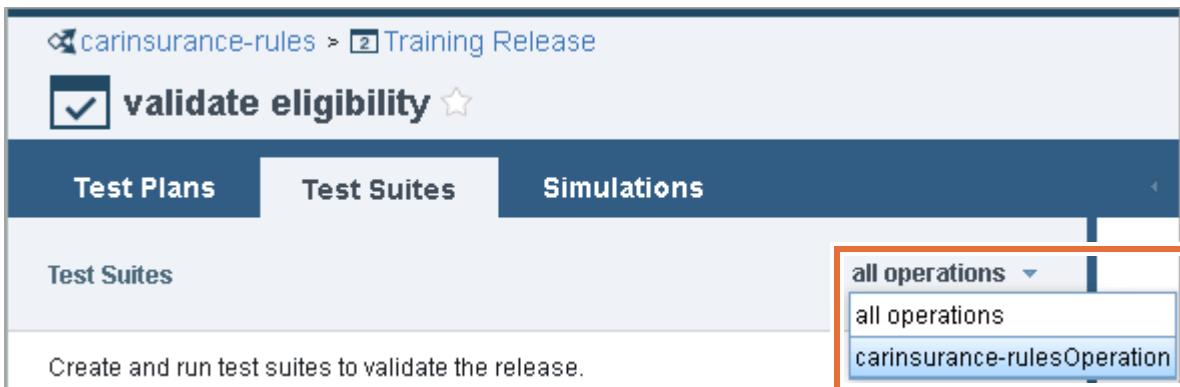
  - a. Make sure that you are on the **Test Plans** tab.

- b. Click **Add a Test Plan** (the plus [+]-sign icon).

- \_\_ c. In the “Add a Test Plan” window, enter the following message, and click **Add**:  
Validate the eligibility package



- \_\_ 3. Create a test suite to check the changes to the eligibility package.
- \_\_ a. Click **Test Suites**.
  - \_\_ b. Click **all operations** and select **carinsurance-rules Operation**.



- \_\_ c. Click **New Test Suite** (the plus [+] sign icon).
  - \_\_ d. In the New Test Suite window, enter **Eligibility Test Suite** in the **Name** field.
- \_\_ 4. Upload the **eligibility.xlsx** scenario file to the test suite.
- \_\_ a. Under **Scenarios > File to use**, click **Choose**.
  - \_\_ b. Browse to `<labfilesDir>\scenarios`.



## Reminder

In the lab environment that is prepared for this course, `<labfilesDir>` is `C:\labfiles`.

- \_\_\_ c. Select the `eligibility.xlsx` file, and click **Open**.
- \_\_\_ 5. Complete and run the test suite.
  - \_\_\_ a. In the **Report name** field, enter: `Eligibility Test Report`
  - \_\_\_ b. In the “Expected execution details to include in report” section, select **The list of rules fired**.

**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 not executed ruleflow tasks
- The list of rules fired
- The list of executed ruleflow tasks
- The duration (in ms) of execution
- The list of rules
- The list of ruleflow tasks

- \_\_\_ c. Click **Save and Run**, and then click **Create New Version**.
- \_\_\_ 6. Wait for the tests to complete, and then notice the exclamation point (!) icon in the **Status** column, which indicates an unexpected result.
- \_\_\_ 7. Hover the mouse pointer over the **Status** icon, and you see a message that the success rate is 90%.

This success percentage means that 9 out of the 10 scenarios executed as expected.

Create and run test suites to validate the release.				
<span style="font-size: 1.5em;">+</span> <span style="font-size: 1.5em;">trash</span> <span style="font-size: 1.5em;">down</span> <span style="font-size: 1.5em;">file</span> <span style="font-size: 1.5em;">play</span> <span style="float: right;">Filter: <input type="text"/></span>				
Name	Operation	Status	Recent Report	Last Changed By
<input type="checkbox"/> Eligibility...	carinsurance-rulesOper	!	Eligibility Test Report ...	Paul

Success rate: 90% (9/10)

- \_\_\_ 8. Review the results by clicking the **Eligibility Test Report** link, and in the Results section, looking through the different scenario details.



## Questions

Were the correct rules fired as expected for each scenario?

Why did one of the tests fail? Does this failure indicate a problem with the scenario or with the rule?

\_\_\_ 9. Review the failed test, which is **Eligibility Scenario 9**.

\_\_\_ a. Expand **Eligibility Scenario 9 > Execution details**.

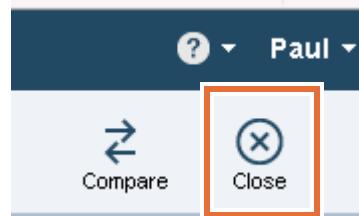
<b>0 / 1</b>	Eligibility Scenario 9	Eligibility (young primary driver)
▼	<b>i</b> Execution Details	
	The list of rules fired	Check Eligibility.Drivers.Driver age Check Eligibility.Policy.Third-party liability in [min, max] Check Eligibility.Primary driver.Young Primary Drivers 1 Check Eligibility.Vehicle.Total insured value
<b>x</b>	the status of the response equals	ProcessManually <b>Refused</b>

- \_\_\_ b. Notice the list of rules that were fired during the test, which include the `Driver age` rule that you updated in the change activity.
- \_\_\_ c. Notice that the expected result for the status of the response (`ProcessManually`) does not match the actual result of the test (`Refused`).

## Answer

The test shows that the scenario needs to change. The expected result for **Eligibility Scenario 9** needs to be updated to `Refused` due to the changes to the `Driver age` rule. Because the minimum age of a driver changed from 16 to 18, all drivers under the age of 18 should have a result of `Refused`.

\_\_\_ 10. Close the report.



- \_\_\_ 11. Update the test plan with the information that the unexpected result for Scenario 9 is not an error in the rules.
- \_\_\_ a. Click **Test Plans**.

- \_\_ b. Hover the mouse pointer over the test plan, and click **Click to edit this test plan** (the pencil icon).

Details	Results
Validate the eligibility package	

**Click to edit this test plan**

- \_\_ c. Click **Results**, and enter:

The result for Scenario 9 shows that the expected results for this scenario must be updated. The minimum age in the Driver age rule was changed from 16 to 18.

- \_\_ d. Click **Done**.

## 7.2. Completing the validation activity

- \_\_ 1. Change the status of the validation activity to **Finished**.
- \_\_ a. In the **Validation Activity** tab in the right pane, go to the Testers section.
  - \_\_ b. Click **Working**, and select **Finish working**.

The status changes to **Finished**.

## Section 8. Approving the validation activity, completing the release, and deploying the release

In this section, you work as Paul to approve the validation activity, complete the release, and deploy the release.



### Note

The steps in this part of the exercise are simplified because Paul is acting as the approver for both the validation activity and for the release.

### 8.1. Viewing the results of the test plan and completing the validation activity

In this part of the exercise, you approve the validation activity.

- 1. Click the **Home** tab, and then click **Stream**.

You can see that the Stream recorded that you finished working on the validation activity.

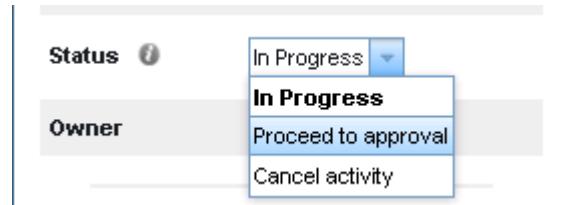
The screenshot shows the Stream interface with the following entries:

- Post a new comment here**
- Expand All | Collapse All**
- Today** (under a collapsed section indicator)
- You finished working on Validate eligibility validation activity in carinsurance-rules > Training Release release** (by Paul, 5:23:39 PM) with a [Comment](#) link
- You updated a test plan in Validate eligibility validation activity in carinsurance-rules > Training Release release** (by Paul, 5:17:39 PM) with a [Comment](#) link
- You added a test plan in Validate eligibility validation activity in carinsurance-rules > Training Release release** (by Paul, 4:31:34 PM) with a [Comment](#) link

- 2. Click **Validate eligibility** in the Stream, or go to **Work > Due Later > Validate eligibility**.

\_\_\_ 3. Complete the validation activity.

- \_\_\_ a. On the **Validation Activity** tab, click the **In Progress** status, and select **Proceed to approval**.

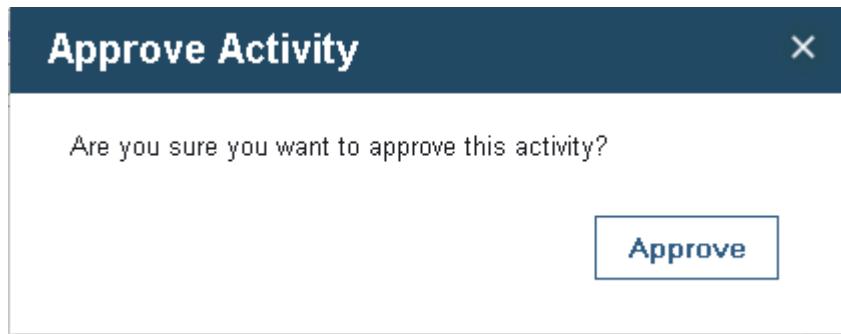


The status is now **Ready for Approval**.

- \_\_\_ b. In the Approvers section, click **Not Reviewed** and select **Approve changes**.



- \_\_\_ c. In the Approve Activity window, click **Approve**.



The validation activity status is now **Complete**.

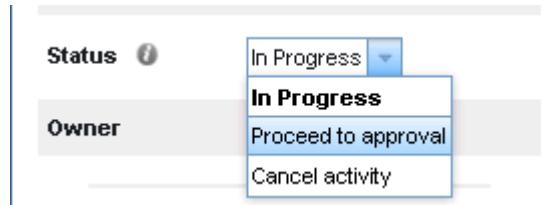
## 8.2. Completing the release

As the approver for the release, Paul can approve the release when all change and validation activities are finished.

Because the change and validation activities for the Training Release are done, you can now complete the release. After the release is approved and completed, it can be deployed.

- \_\_\_ 1. Click the **Training Release** breadcrumb.
- \_\_\_ 2. On the **Release** tab, go to the **Status** menu.

- 3. Click **In Progress**, and select **Proceed to approval**.



The release status is now **Ready for Approval**.

- 4. In the Approvers section, change **Not Reviewed** to **Approve changes**.



- 5. In the Approve Release window, click **Approve**.

The release status is now **Complete**.

### 8.3. Deploying the Training Release

Now that the release is completed, you can deploy it.

- 1. On the Training Release page, click the **Deployments** tab.



#### Reminder

Use the left and right navigation arrows to access the different tabs.



For this exercise, you use the `car main` deployment configuration.

- 2. Click `car main` to open it and view the deployment details.

The deployment configuration includes the following tabs:

- The **General** tab, which lists information about the deployment configuration, such as the name, type, RuleApp name, and RuleApp base version number.
- The **Operations** tab, which lists the decision operations to deploy. Here, **carinsurance-rulesOperation** is selected for deployment.
- The **Targets** tab, which lists the server where the rules can be deployed.
- The **Ruleset Properties** tab, which you can use to define the version policy for each deployment.

- The **Groups** tab, which the administrator uses to choose which groups can see and use this deployment configuration.
  - The **Deployment Snapshot** tab, which defines whether a snapshot is taken at the moment of deployment.
- \_\_\_ 3. Deploy the Training Release.
- \_\_\_ a. On the toolbar, click **Deploy**.



- \_\_\_ b. In the Deploy Training Release window, review the configuration details, and click **Deploy**.
- \_\_\_ c. Click **OK** to close the “Deployment status” window.

The Business console opens to the Reports page of the **Deployments** tab.

This table lists the deployment reports and the current deployments for this branch

	Name	Status	Run By
<input type="checkbox"/>	<a href="#">Report 2017-05-...</a>	<input checked="" type="checkbox"/>	Paul

- \_\_\_ d. Wait for the deployment to complete. The Status column should show a check mark, which indicates a successful deployment.

<input type="checkbox"/> Name	Status	Run By
<input type="checkbox"/> <a href="#">Report 2017-05-...</a>	<input checked="" type="checkbox"/>	Paul

- \_\_\_ 4. View the summary report of your deployment by clicking the report link.

Notice that a deployment snapshot was created. You can click the link to open it.

The **Redeploy** icon next to the deployment snapshot link can be used to redeploy the content of this deployment from the report. This feature is useful if you need to redeploy, or to deploy to a nonproduction environment to align with the production.

The screenshot shows a deployment summary for the 'carinsurance-rules' application. The top navigation bar indicates the path: carinsurance-rules > Training Release. The main title is 'Report 2017-05-24\_5-48-38'. Below this, there's a 'Summary' section with several key details:

- Deployment Content:** Training Release In carinsurance-rules. To the right is a blue circle with a white checkmark and the word "Success".
- Configuration name:** car main. To the right is a blue circle with a white checkmark and the word "Success".
- Operations Deployed:** carinsurance-rulesOperation. To the right is a blue circle with a white checkmark and the word "Success".
- Target Servers:** Sample (radio button selected).
- Deployment snapshot:** car\_main\_2017-05-25T02\_48\_24Z (with a redeploy icon).

- \_\_\_ e. Notice the **XOM Deployment** section, which describes how the XOM was handled during deployment.
- \_\_\_ f. Close the report.
- \_\_\_ 5. Log out of the Business console and close the web browser.

## End of exercise

## Exercise review and wrap-up

In this exercise, you worked as different users to do different tasks within a decision governance framework workflow. As Paul, you created a release with a change activity and a validation activity. As Bea, you implemented the change to the decision service release. Finally, as Paul, you completed validation activity tasks, completed the release, and deployed the release.



IBM Training



© Copyright International Business Machines Corporation 2017.